IoT Hub

Console Guide

Product Documentation
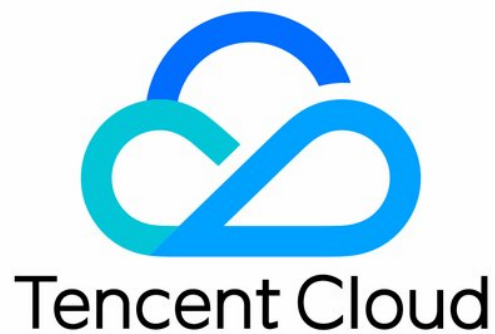
Copyright Notice

Trademark Notice

Tencent Cloud

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Console Guide

Product Management

Device Connection Preparations

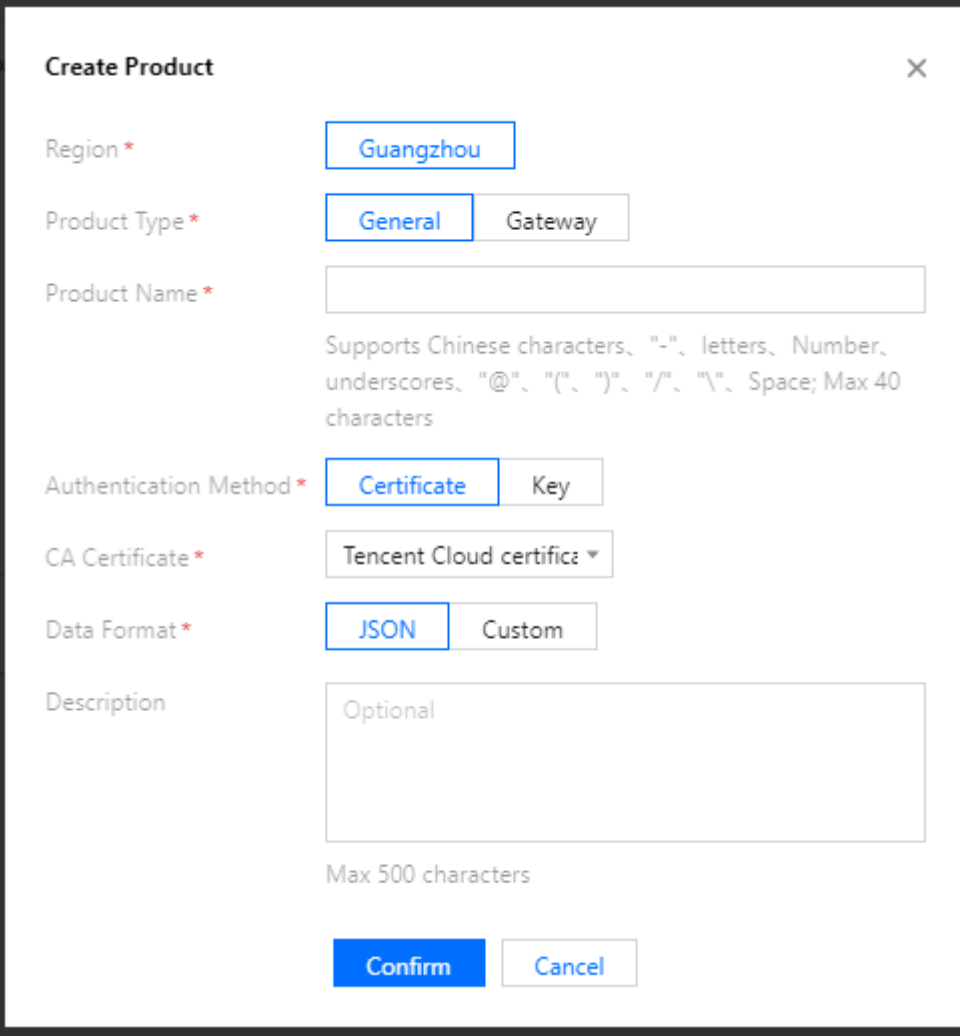Last updated：2021-09-08 12:23:10

Overview

Before connecting a device to IoT Hub, you need to create a virtual product and device in the IoT Hub console and match them with the real device. IoT Hub will assign each device a unique authentication identifier for connection. This document describes how to make preparations for connection to the platform.

Directions

Creating product

1. Log in to the IoT Hub console and click **Products** on the left sidebar.
2. On the product list page, click **Create Product** and enter the following information as needed.
   - **Region**: it is **Guangzhou** by default.
   - **Product Type**:
     - General: it communicates over 2G/3G/4G/Wi-Fi or wired connection. You can further develop its communication module for creation.
     - Gateway: it is used to proxy communication between subdevices and the backend.
   - **Product Name**: it can contain 1–32 letters, digits, and underscores and can be modified.
   - **Authentication Method**:
     - Certificate: when you create a device, the platform will generate a certificate and a private key for authentication before the device can communicate with IoT Hub.
     - Key: when creating a device, you can use a custom PSK or the random PSK generated by the platform for the device.
     - TID: when creating a device, you can purchase the IoT TID authentication service to authenticate the device.
   - **Data Format**:
     - JSON: you can match the rules and extract the content based on the data.
     - Custom: no data parsing is performed.

- **Product Description**: it can contain up to 500 characters.

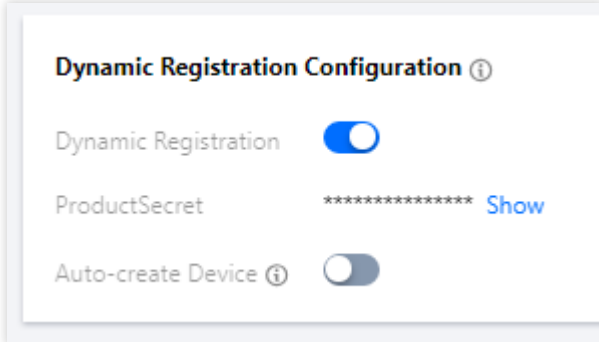3. Click **Confirm** (a product is a collection of devices of a certain type. You can manage all devices through the product).

4. After the product is created, you can enable dynamic registration to generate a product key (ProductSecret).

Note：

Unified product information can be burned for all devices under it in the production process, i.e., product ID (ProductId) and product key (ProductSecret). After the devices are shipped, the device identity information can be obtained through dynamic registration and then saved, and then obtained three or four pieces of information can be used for device authentication. If you enable dynamic registration and select automatic device creation, device names can be generated automatically, which are generally IMEIs or MAC addresses but must be unique under the same product ID (ProductId). For more information on the dynamic registration feature, please see SDK for C Connection Description.

Deleting product

1. After a product is created, you can view its basic information on the **Product Settings** page.
2. When you no longer need the product, you can click **Delete** on the right of the **Products** page.

Creating device

After creating a product, you can add one device or batch add devices under it:

Adding one device

1. Click the **Devices** tab of the corresponding product.
2. Click **Add Device** and enter the relevant device information.
   - Device Name: it can contain up to 48 letters, digits, underscores, and hyphens.
   - Remarks: it can contain up to 16 letters, digits, underscores, and hyphens.

> Note：
>
> - The device name cannot be modified once confirmed.
> - The device name must be unique under the same product.
> - The key option appears only when the authentication method is key authentication. A custom key must be a Base64-encoded string. You can enter an ordinary string in the input box and then click **Convert to Base64** to Base64-encode it.
> - Specific parameters that need to be entered during device creation vary by product type.

If the selected authentication method is certificate authentication, after the device is created, the device private key will be the unique identifier used by it to connect to the IoT Hub backend, which does not store the device private key. Therefore, please keep it secure and confidential.
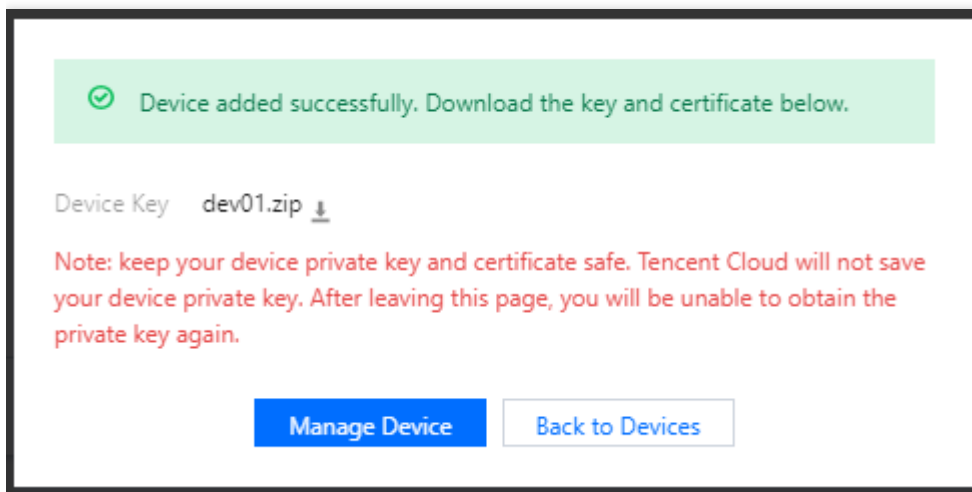
If the selected authentication method is IoT TID authentication, you can configure an allowlist to specify whether to support device precreation.
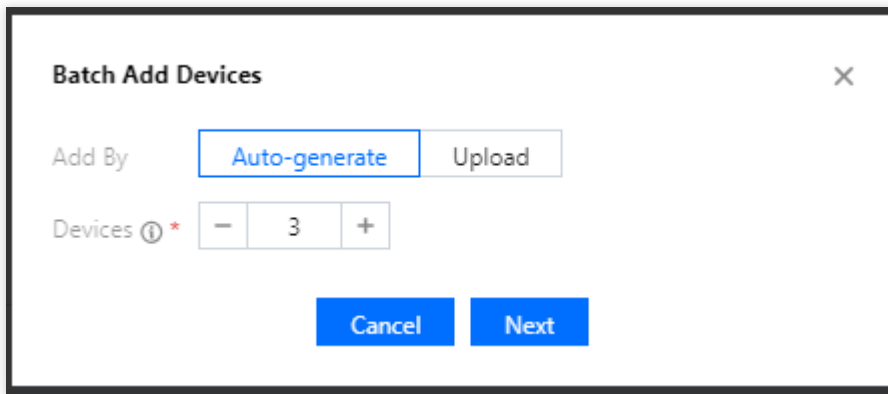
> Note：
>
> After allowlist authentication is enabled, the device must carry the `DeviceName` in the precreation allowlist for IoT TID authentication; otherwise, the authentication will fail. If the allowlist is not enabled, the platform will automatically create a device according to the `DeviceName` carried for device authentication but will not create the device again if it has already been created. Allowlist authentication is disabled by default.

Batch adding
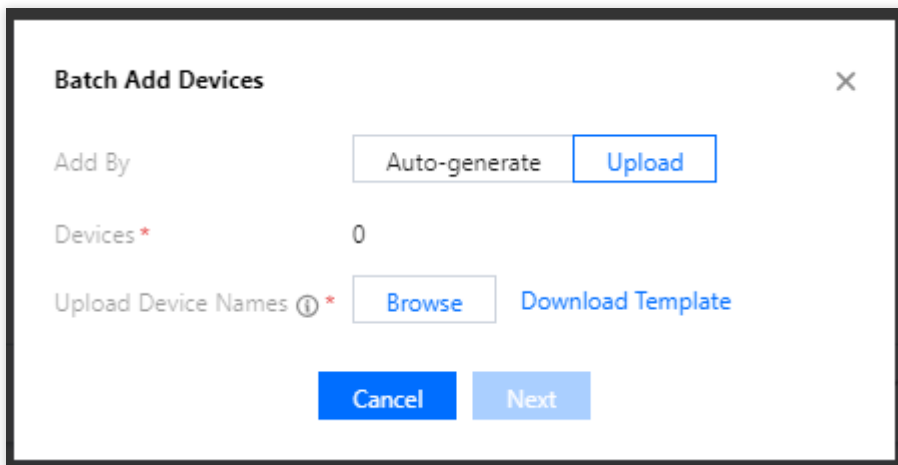
1. Click **Batch Add** on the **Devices** tab.
2. There are two batch adding methods: **Auto-generate** and **Upload**.

- Auto-generate: set the desired number of devices, and the console will generate the same number of devices. In this case, a device name is a combination of 18 uppercase and lowercase letters and digits.

- Upload: upload a CSV file of device names, and IoT Hub will create devices according to the device names in the CSV file.

3. After the devices are added successfully, you can view the execution status on the **Result** page and download the device certificate/key file. You can also view and download the corresponding information and file in **Batch Management**.

Note:

- If the selected authentication method is key authentication, the file available on the **Result** and **Batch Management** pages for download is a CSV file, which contains the device names, device keys, error codes, and error messages.
- If the selected authentication method is certificate authentication, the file available on the **Result** and **Batch Management** pages for download is a ZIP package file, which contains folders of the same number as that of devices as well as a CSV file. The name of each folder is the corresponding device name, and the content

of the folder is the certificate and private key files. The CSV file contains the device names, error codes, error messages, and the relative paths of the device certificate and private key files.

Device details

Certificate-authenticated device

Device details include all the content of the device: device name, device remarks, connection status, version information, device certificate (click to download), device private key (click to download), device log configuration, etc.

Key-authenticated device

Device details include all the content of the device: device name, device remarks, connection status, version information, device key (click to view), device log configuration, etc.

IoT TID-authenticated device

Device details include all the content of the device: device name, device remarks, connection status, version information, device log configuration, etc.

Gateway Product Connection

Last updated：2021-09-08 12:08:10

This document describes how to manage subdevices.

Prerequisites

You have created a gateway product and a device.

Directions

Binding subdevice

> Note：
>
> A subdevice is a device that can connect to IoT Hub only through a gateway device.

1. Log in to the IoT Hub console and click **Products** on the left sidebar to enter the product list page.
2. Click the selected **product name** to enter the product details page. Click **Devices** and select **Subdevice** to enter the subdevice page.

3. Click **Add Subdevice**, select a product, select the subdevice to be bound to it, and click **Add**.
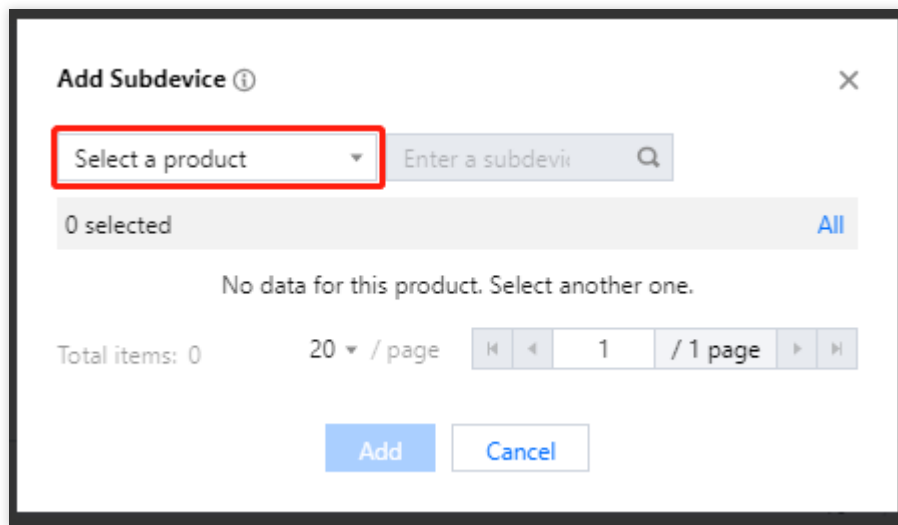
Unbinding subdevice

Select the corresponding product on the subdevice page, select the subdevice bound to it, and click **Unbind**.

Device Grouping

Last updated：2021-08-30 17:02:41

Overview

IoT Hub provides the multi-level device grouping feature to meet your needs for managing devices under different products by group in different business scenarios. You can:

- Add, delete, modify, and query groups.
- Add devices under different products or the same product to a group.
- Add subgroups under a group or subgroup.
- Query the list of devices in a group.
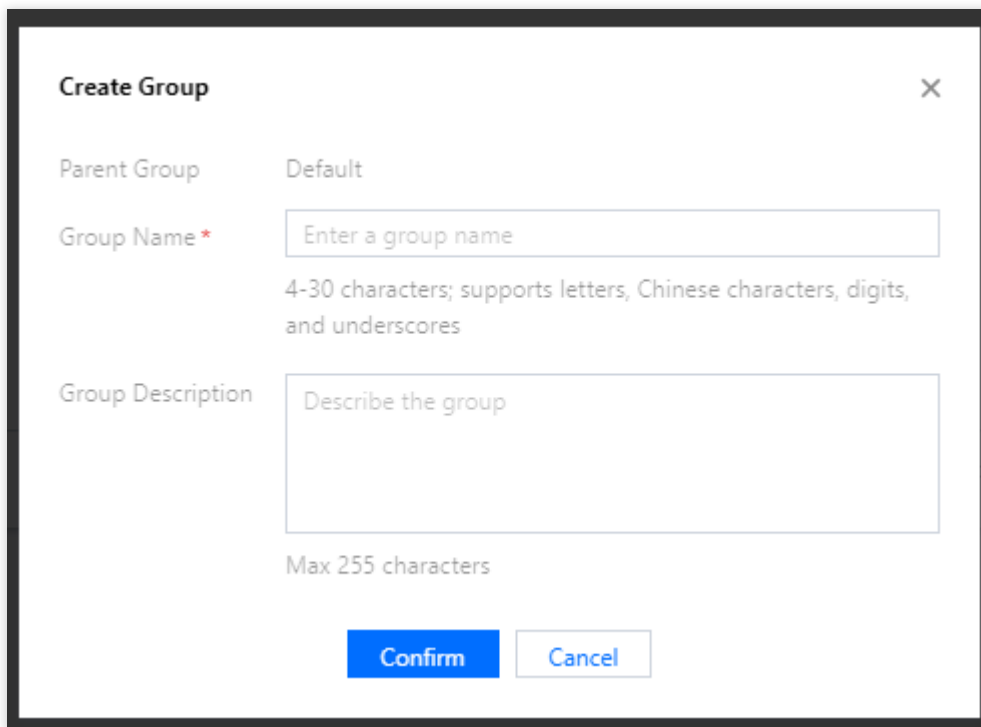
This document describes how to group devices in the console.

Directions

Creating group

1. Log in to the IoT Hub console and click **Groups** on the left sidebar.
2. On the group management page, click **Create Group**.
3. In the group adding pop-up window, enter relevant information. When creating a group, you need to select the group to which it belongs. A level-1 group always belongs to the default project, a level-2 group can belong to a level-1 group, a level-3 group can belong to a level-2 group, and so on. You can customize the group name and description.
   - Parent Group: there is no limit on the number of group levels. One group can have up to 100 subgroups.
   - Group Name: it can contain 4–30 letters, digits, and underscores.

- Group Description: it can contain up to 255 characters.

4. Click **Confirm**.

Viewing group

1. Log in to the IoT Hub console.
2. Click **Groups** on the left sidebar to view the group list, where you can manage, edit, and delete groups.
3. View the basic information of a group displayed in the console, including:
   - Group Name: it is customizable.
   - Group ID (groupID): it is randomly assigned by the backend and is the unique group identifier.
   - Group Description: it is customizable and describes the group.
   - Devices: it indicates the total number of devices in the group.
   - Subgroups: it indicates the total number of subgroups in the group.
   - Creation Time: it indicates the time the group was created in the console or through APIs, accurate to the second.

Managing group

1. Log in to the IoT Hub console.
2. Click the group name to enter the group management page, where you can manage the device list.
3. To add one or more devices to the group, click **Add Device** under the device list. You can search for devices by product name or device name and select different devices for batch adding.

4. The device list displays the information of the devices in the group (e.g., floor1), including device name, status, product, and last connection time and operation.

   ○ Device Name: `devicename` .

   ○ Status: online/offline status of the device.

   ○ Product: name of the product to which the device belongs.

   ○ Login Time: last connection time of the device, accurate to the second. "-" will be displayed if the device has never been connected.

   ○ Click **Manage** in the **Operation** column to enter the device details page of the corresponding device.

   ○ Click **Remove** in the **Operation** column to remove the device from the group. **This operation will not delete the device**.

5. You can add subgroups to or remove subgroups from any group. Click the group name to enter the group management page, where you can manage the subgroup list.

   ○ Just like under parent groups, you can create subgroups under subgroups and add devices to them.

○ As regards the group level, if the group is a level-1 group, only the name of the level-1 group is displayed; if it is a level-n group, the names of the n levels of groups are displayed, such as `group1` and `group1/group2.../groupn` .
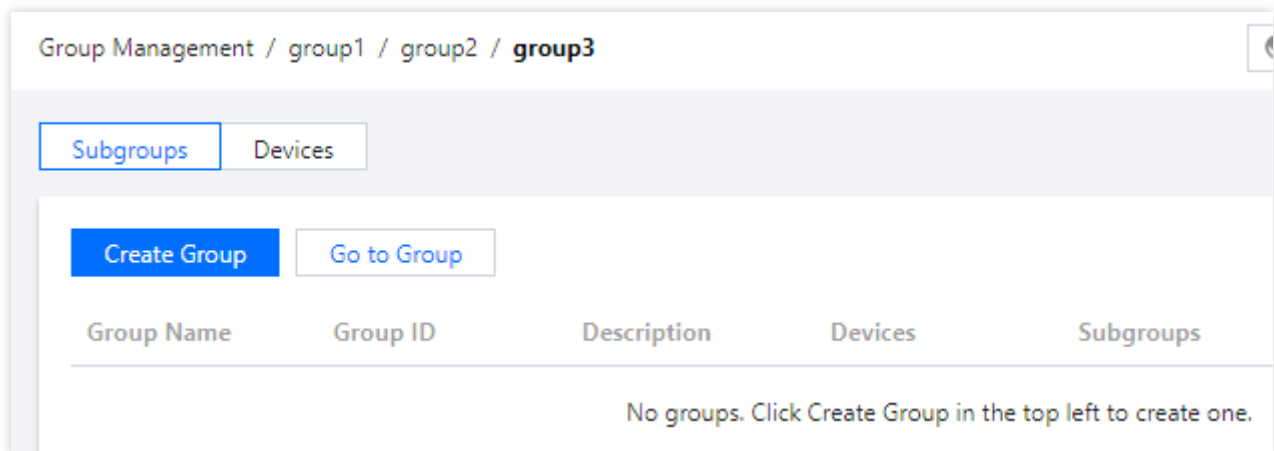
Product Types

Last updated：2021-08-20 16:11:59

When creating a product in the IoT Hub console, you can select general product or gateway product as the product type. Their differences lie in the communication method and whether the communication module can be further developed.

General Product

A general product communicates over 2G/3G/4G/Wi-Fi or wired connection. You can further develop its communication module to implement socket communication.

Gateway Product

A gateway product communicates over 2G/3G/4G/Wi-Fi or wired connection. In addition to the basic features of general products, it can also be bound to products that cannot directly access the internet and send messages on their behalf, such as LoRa products.

LoRa Product

A LoRa product communicates over LoRa and cannot directly access the internet. It uses the standard LoRaWAN protocol and relies on a gateway to send messages on its behalf.

CLAA Product

A CLAA product communicates over LoRa and cannot directly access the internet. It uses the CLAA protocol (with a built-in CLAA module) and relies on a gateway to send messages on its behalf.

Device Shadow

Last updated：2021-09-08 12:19:44

Initial Status

After a product and a device are created, the device is in **Inactive** status, and the device shadow is empty by default.

Reporting Status by Device

After the device reports its status to IoT Hub, its latest device shadow status will be displayed in the console.

Updating Device Shadow Status by Application

You can modify the virtual device (device shadow) in the console. After the change is saved, the device will receive the update to the virtual device.

1. Log in to the IoT Hub console and click **Products** on the left sidebar.
2. On the product list page, click the product name to enter the product details page.
3. Select **Devices**, click **Device Shadow** on the right, and click **Modify** in the top-right corner.
4. Add JSON data to the device shadow as prompted on the right. The `reported` field can be empty, while the `desired` field cannot.

5. Click **Confirm** to complete the modification.

Topic Management

Last updated：2022-11-16 18:27:47

Overview

After creating a product, you can configure the topics that its devices can publish or subscribe to. The device topic list is inherited from the product topic list. You can add, delete, or modify the items in the topic list only at the product level.

Directions

Adding a custom topic

1. Log in to the IoT Hub console and click **View details** on the upper right of the Overview module.
2. Click **Products** on the left sidebar.
3. Click a product name, and select **Topics**.
4. Click **Add Custom Topic** and set the operation name and permission.
   - Operation name: The name can contain 1 - 255 letters, numbers, and underscores, where different levels are separated by `/`, and `+` indicates a level. The name should be in the format of `/+/` rather than `/+aaa/`.
   - Permission: **Subscribe**, **Publish**, and **Subscribe and publish** are available for selection, and the selected permission can be changed subsequently.

Editing a custom topic

Click **Edit** in the topic permission list to modify the name of the corresponding topic permission and the permission itself.

Deleting a topic permission

Click **Delete** in the topic permission list to delete the corresponding topic permission.

Permission List

Last updated：2021-09-08 12:29:06

Overview

After a product is created, you can configure topics that devices can publish or subscribe to on the permission list page. A device inherits permissions from the product to which it belongs, and permissions can be added, deleted, or modified only at the product level.

Directions

Adding topic permission

1. Log in to the IoT Hub console and click **Products** on the left sidebar.
2. On the product list page, click the product name and select **Permissions**.
3. On the permission list page, click **Add Topic Permission**, enter the topic name, and assign an operation permission to the device.
   - Operation Name: the name can contain 1–64 letters, digits, and underscores, where different levels are separated by `/` , and `+` indicates a level. The name should be in the format of `/+/name` rather than `/+aaa/` .
   - Permission: you can select **Subscribe**, **Publish**, or **Subscribe and Publish**, and the selected permission can be changed subsequently.

Editing topic permission

Click **Edit** in the topic permission list to change the name of the corresponding topic permission and the permission itself.

Deleting topic permission

Click **Delete** in the topic permission list to delete the corresponding topic permission.

Cloud Logs

Last updated：2022-11-16 18:27:47

The cloud log module of IoT Hub provides a comprehensive, stable, and reliable log service that logs information in multiple dimensions such as device behavior, message content, and device exception. You can search for key device logs by time, log type, result, device name, `RequestID`, and keyword to identify and troubleshoot business issues easily.

Directions

1. Log in to the IoT Hub console and click **View details** on the upper right of the Overview module.
2. Click **Products** on the left sidebar to enter the Products page.
3. Click the name of the target product to enter the product details page.
4. Click **Cloud logs** to enter the log management page.

Running Logs

Running logs are divided into behavior logs, content logs, and device logs. You can search for logs by time and keyword.

- **Search by time**

  You can select the time range during which to view logs on the time panel, including the last 15 minutes, last 60 minutes, last 4 hours, last 24 hours, and a custom time range. **Cloud logs can be stored for up to 7 days**.

- **Search by keyword**

  In the log search box, you can filter the logs to be queried. When you need to filter target logs by multiple conditions, you can press Enter to separate different tags.

Behavior logs

You can search for logs of all communication behaviors between a device and the cloud, including device connection, disconnection, publishing, and subscribing.

Content logs

You can search for detailed logs of communication content between a device and the cloud.

Device logs

Device logs display device information at four levels: ERROR, WARN, INFO, and DEBUG. You can enable/configure device log collection on the device details page.

Loading more

A cloud log loads 100 data entries each time. You can click **Load more** at the bottom of the page to view more log information.

Log Dump

By configuring CLS, you can dump cloud log data to CLS for permanent storage and log analysis.

> Note：
>
> Access to the following services requires authorization and fees may be incurred.

During the initial configuration, click **Configure** > **Authorize** to enter the role page in CAM and click **Authorize**.

Status Monitoring

Last updated：2021-09-08 12:34:44

IoT Hub provides monitoring capabilities for the device connection, message sending/receiving, device shadow, rule engine, and OTA features. You can view the monitoring data in the last 30 days on the Status Monitoring page in the IoT Hub console, which helps you identify and troubleshoot business issues.

Status Monitoring Types

Currently, IoT Hub provides the following five types of monitoring statistics.

Device connection statistics

Device connection statistics are used to monitor device connection.
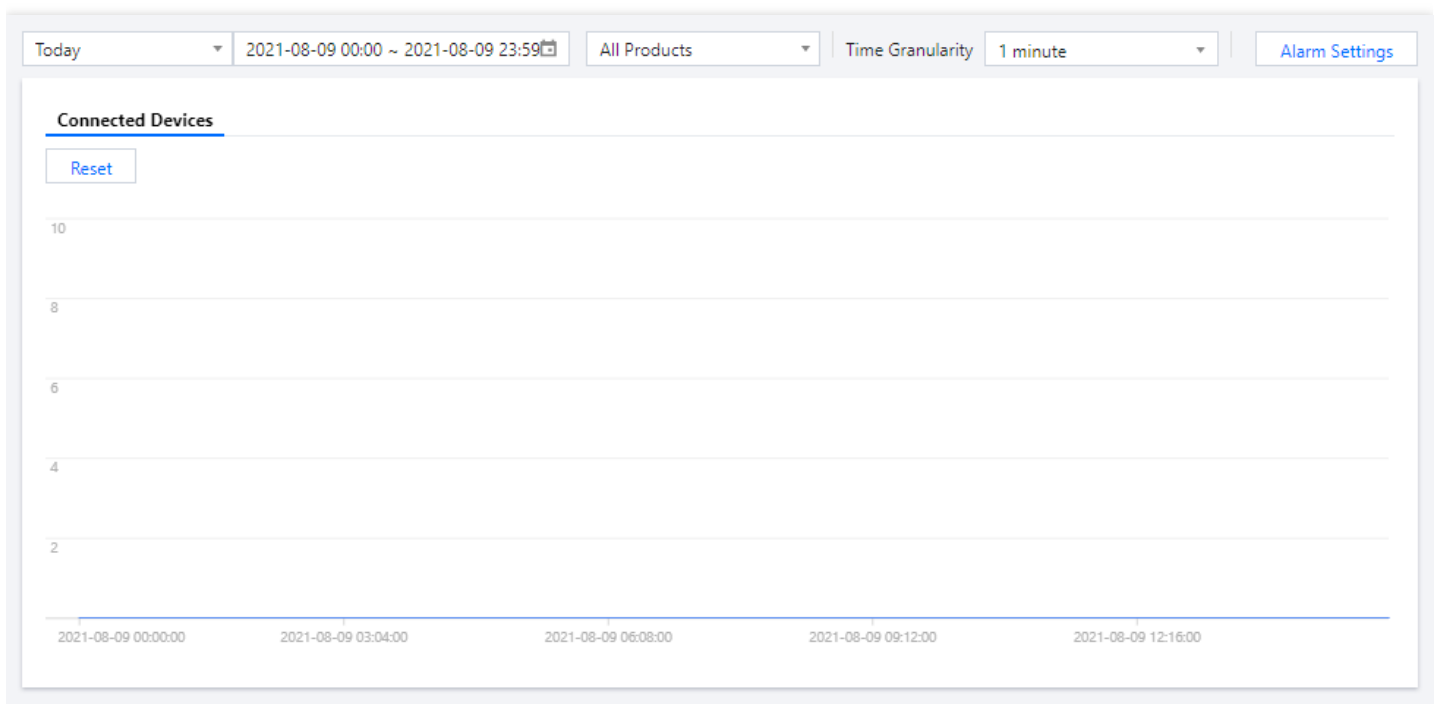
Device message statistics

Device message statistics include the numbers of sent and received messages, failed messages, and billable messages.
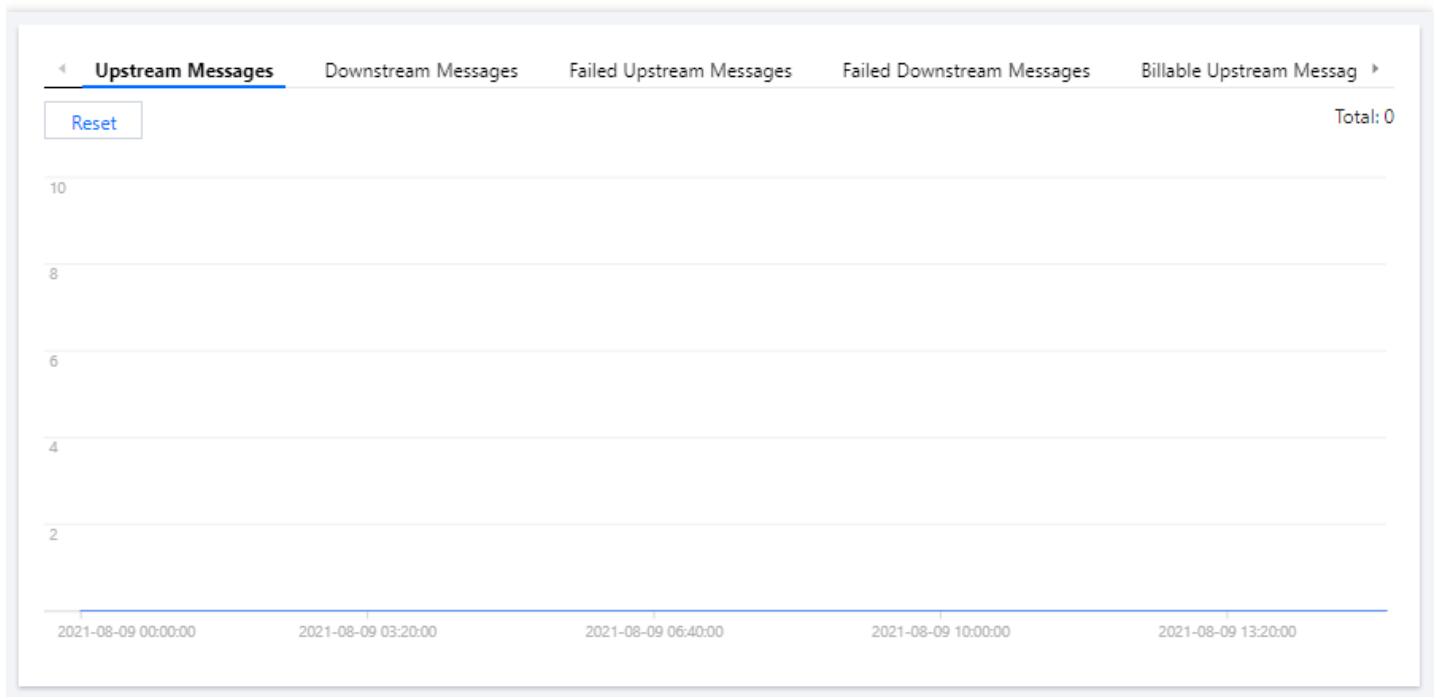
- Upstream Messages: number of messages published by the device or by the application through TencentCloud API.
- Downstream Messages: number of messages transferred to the device by IoT Hub (messages subscribed to by the device).
- Failed Upstream Messages: number of messages published by the device or by the application through TencentCloud API which failed to be published due to invalid format (such as excessive length), frequency limit, traffic limit, or lack of permission.
- Failed Downstream Messages: number of messages transferred to the device by IoT Hub (messages subscribed to by the device) which failed to be received due to invalid format (such as excessive length), frequency limit, or traffic limit.
- Billable Upstream Messages: number of messages published by the device or by the application through TencentCloud API which are converted according to the billing standards (message content length divided by 512 bytes and rounded up).
- Billable Downstream Messages: number of messages transferred to the device by IoT Hub (messages subscribed to by the device) which are converted according to the billing standards (message content length divided by 512 bytes and rounded up).

Device shadow update statistics

Device shadow update statistics include the numbers of active device shadow updates and failed updates.
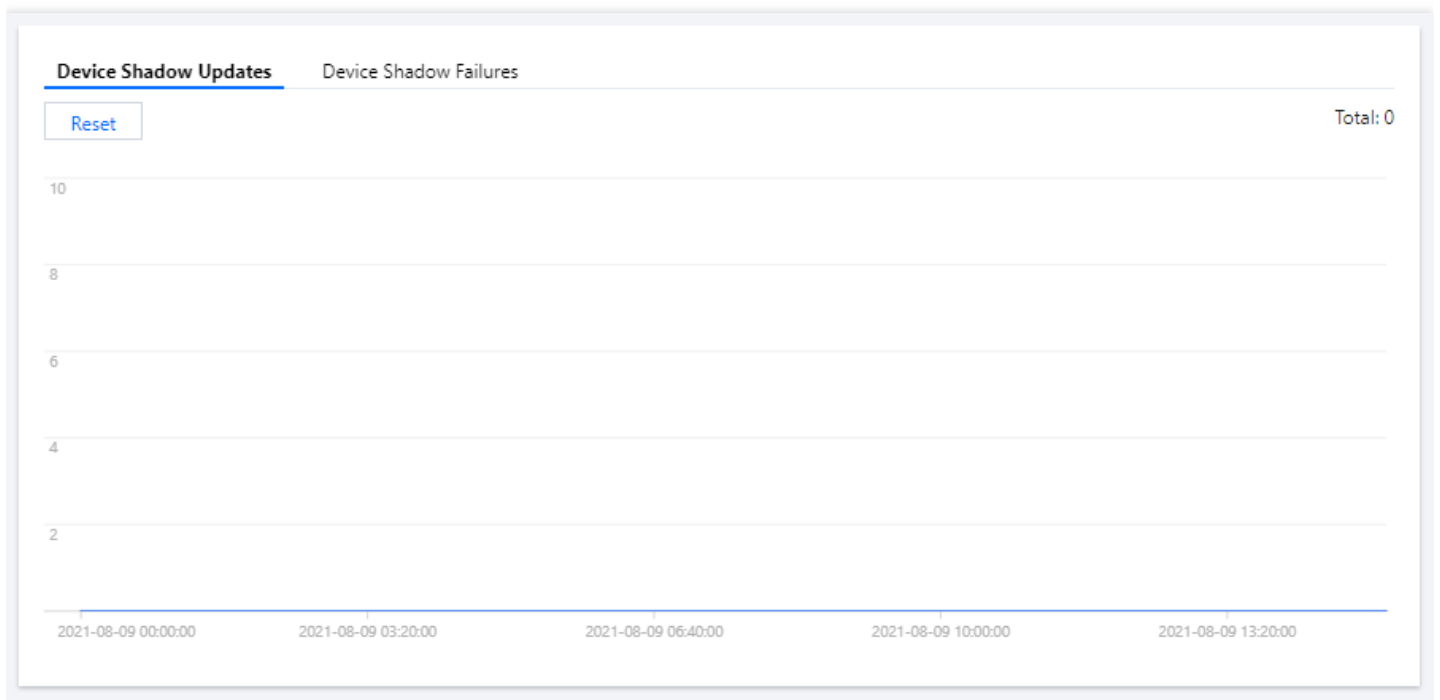
Rule engine statistics

Rule engine statistics include the numbers of times messages hit the rule engine and messages are forwarded to other services.
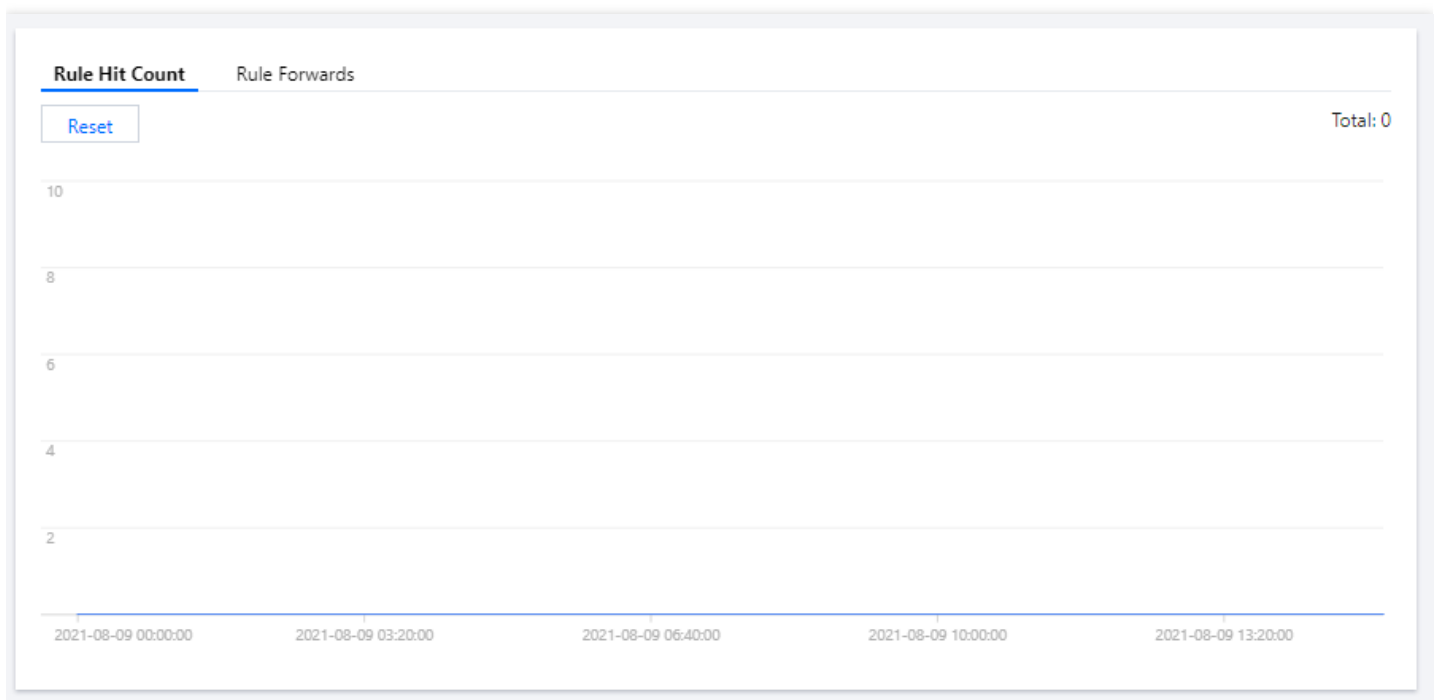
Firmware statistics

Firmware statistics include device version distribution and firmware version capacity statistics.
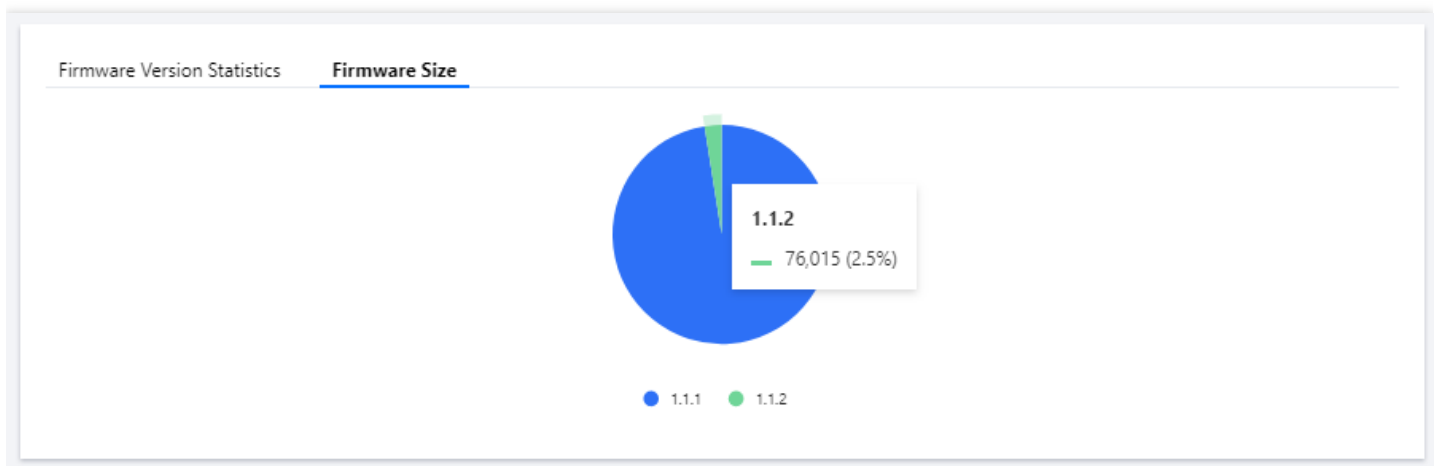
Product-Level Monitoring

Status monitoring supports not only global monitoring but also product-level monitoring. You can search for desired product data by product name.
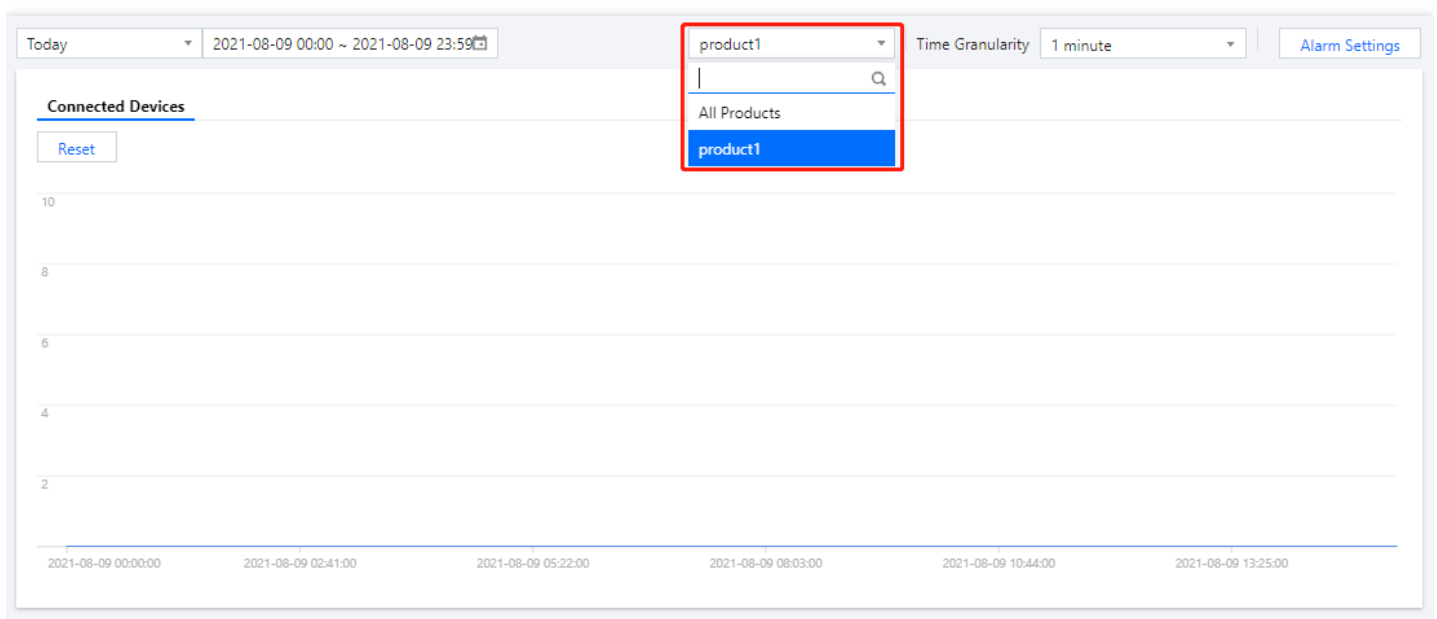
Note:

You can view the firmware statistics only after specifying a product.

Search by Time

For status monitoring, you can select a time range to view data, including the last day, last 7 days, last 15 days, last 30 days, and custom time range.
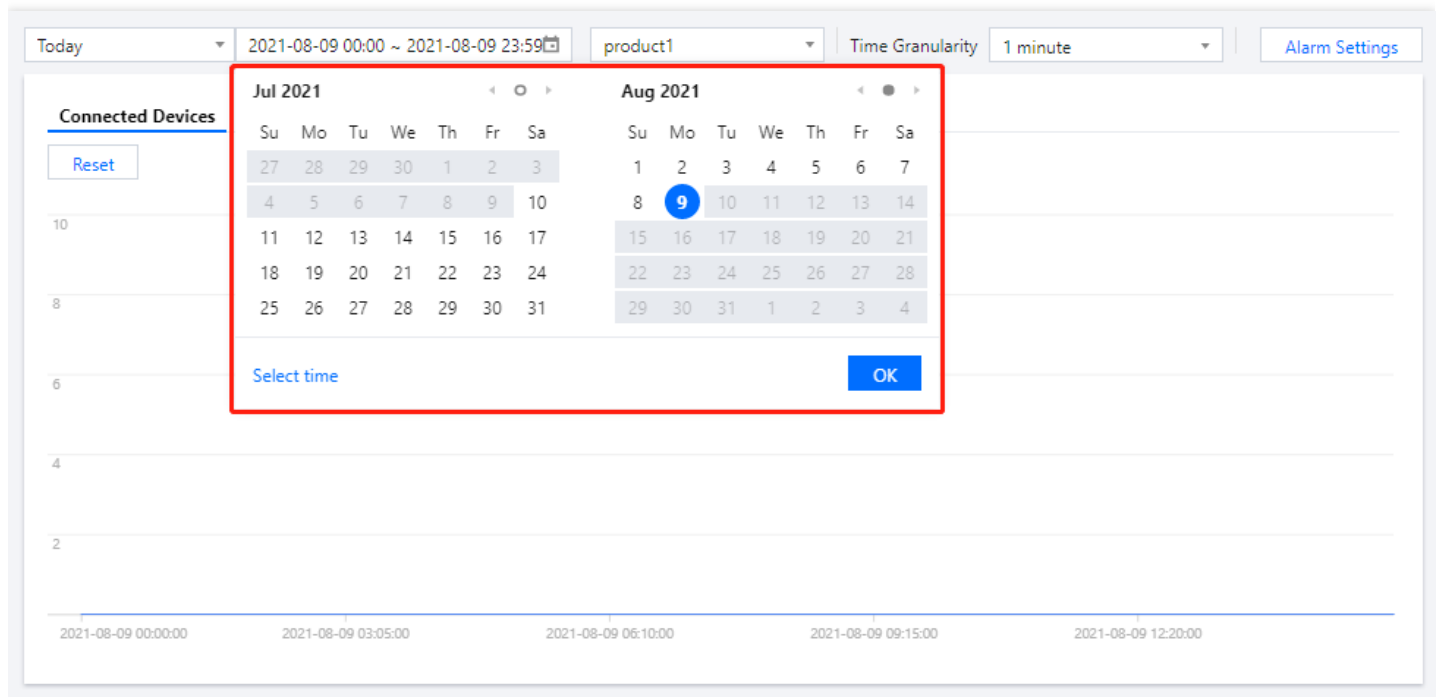
Data Point at Selected Time Granularity

In status monitoring, each displayed point is counted based on the selected time granularity. For the number of connected devices, the value displayed by a single point is the peak value over the selected time granularity. For other

types, the value displayed by a single point is the sum calculated for the selected time granularity.
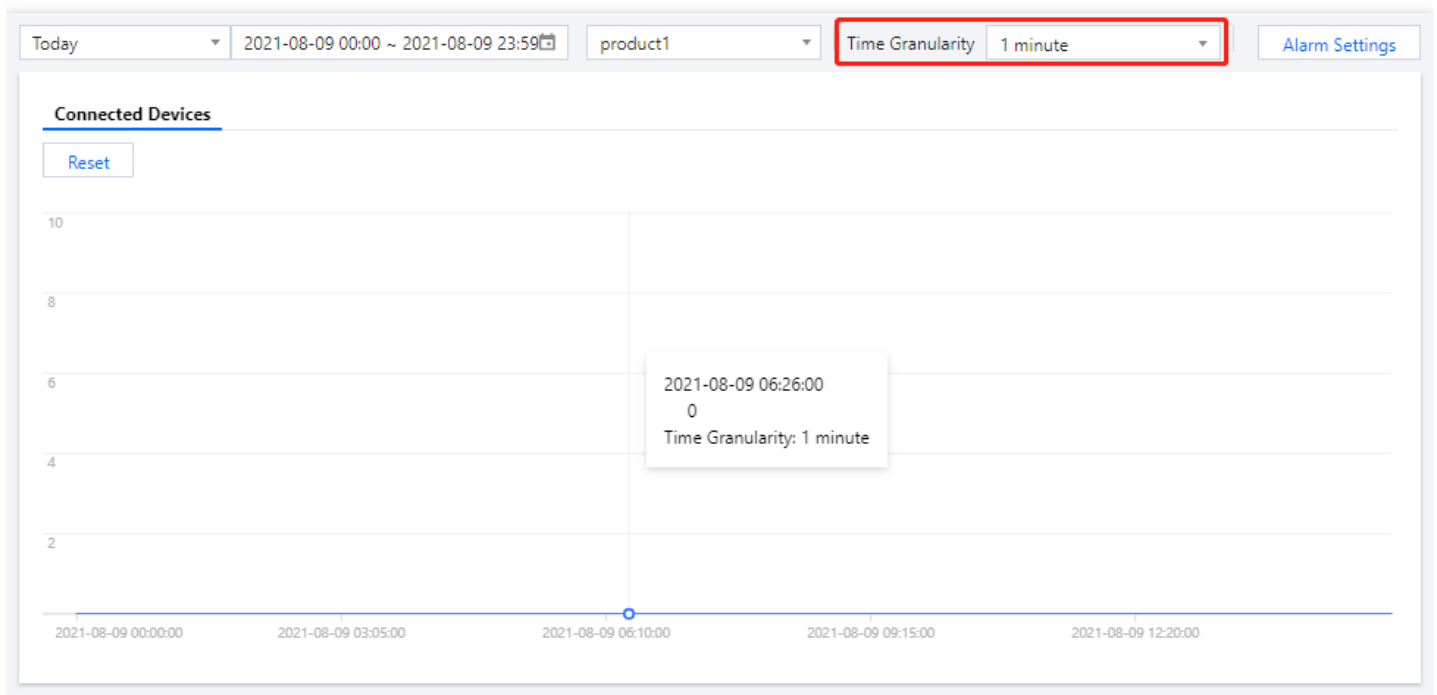
Zoom and Aggregation

The monitoring sequence diagram also supports zoom and aggregation. You can select a start time on it, hold down the mouse button, and drag the cursor to the position of the desired end time to view the monitoring data in the time period.
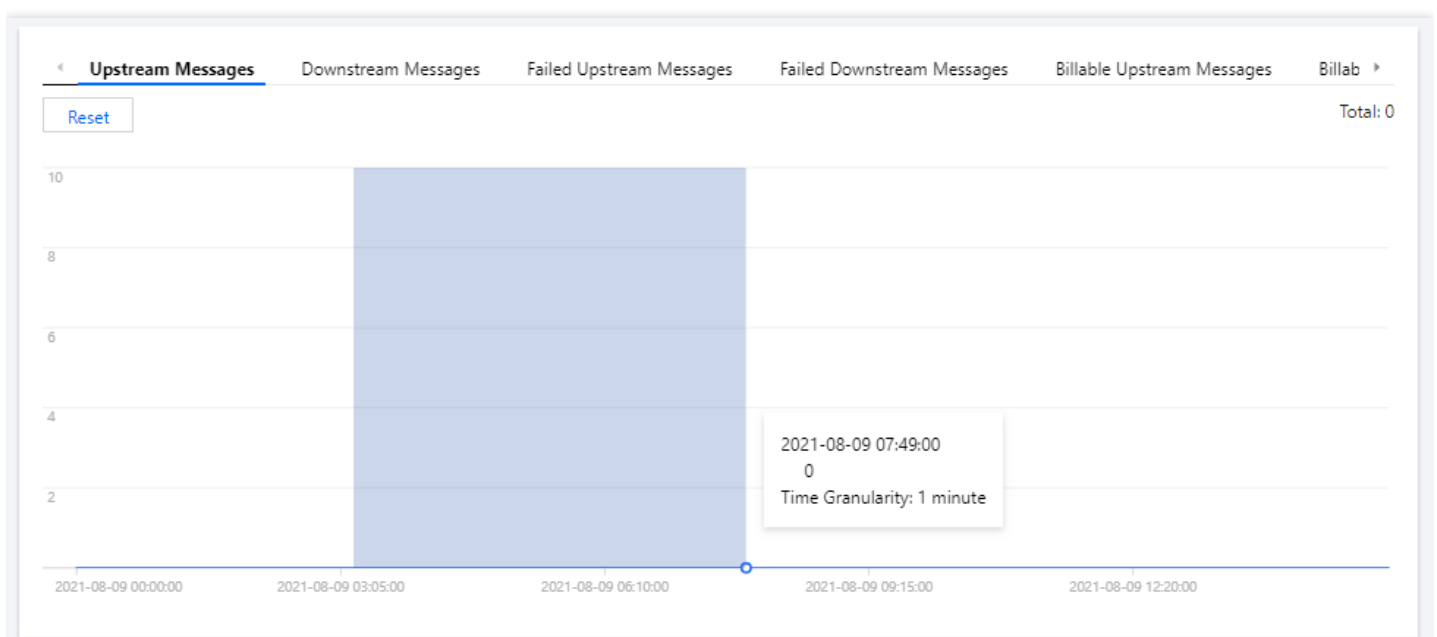
The aggregation feature sums the total number in the corresponding data type on the displayed timeline by default and displays it in the top-right corner of the statistics page.
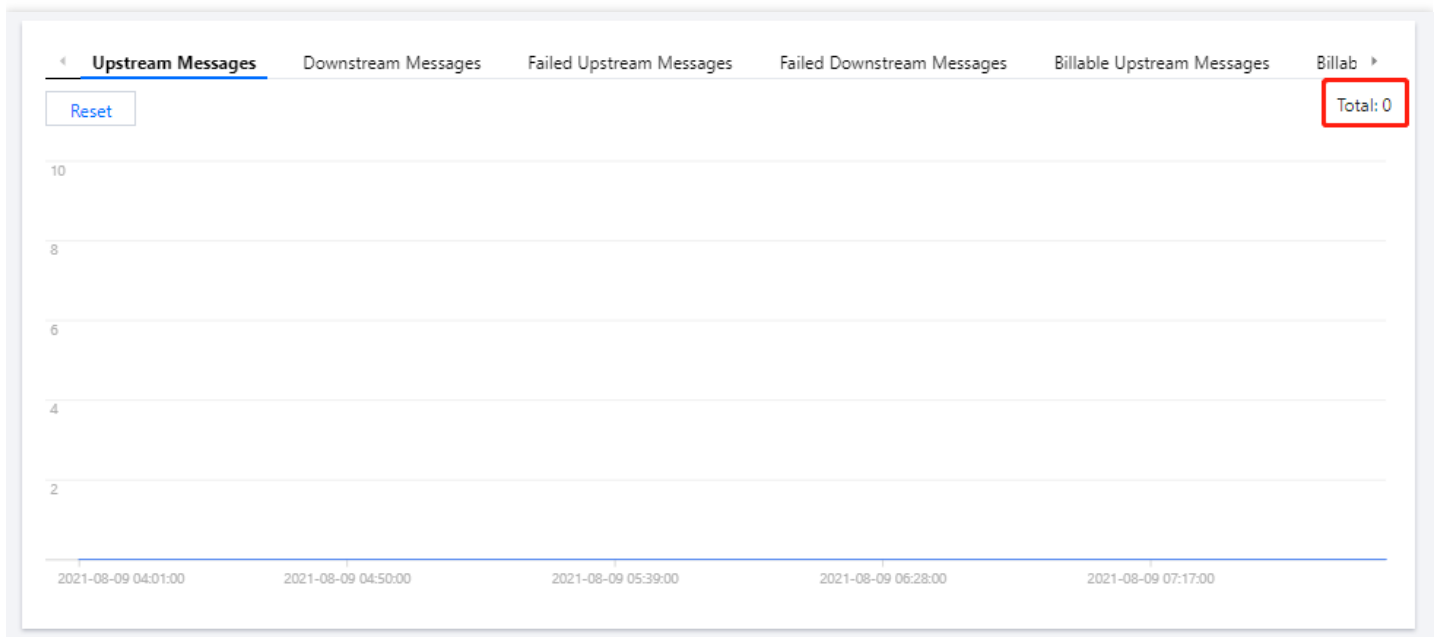
You can click **Reset** to reset the page zoom.

Monitoring and Alarming

Status monitoring allows you to configure alarms in Cloud Monitor in order to monitor the numbers of connected devices, upstream and downstream messages, and failed upstream and downstream messages. When an alarm is triggered, alarm messages will be sent to the specified recipients by SMS, email, or WeChat, so that they can promptly take actions accordingly.

Creating alarm policy

IoT Hub supports setting alarms for the following metrics:

- Connected Devices
- Upstream Messages
- Downstream Messages
- Failed Upstream Messages
- Failed Downstream Messages
- Device Shadow Updates
- Device Shadow Failures
- Rule Hit Count
- Rule Forwards

Directions

1. Log in to the IoT Hub console, enter the Status Monitoring page, and click **Alarm Settings** in the top-right corner of the page to enter the alarm settings page.
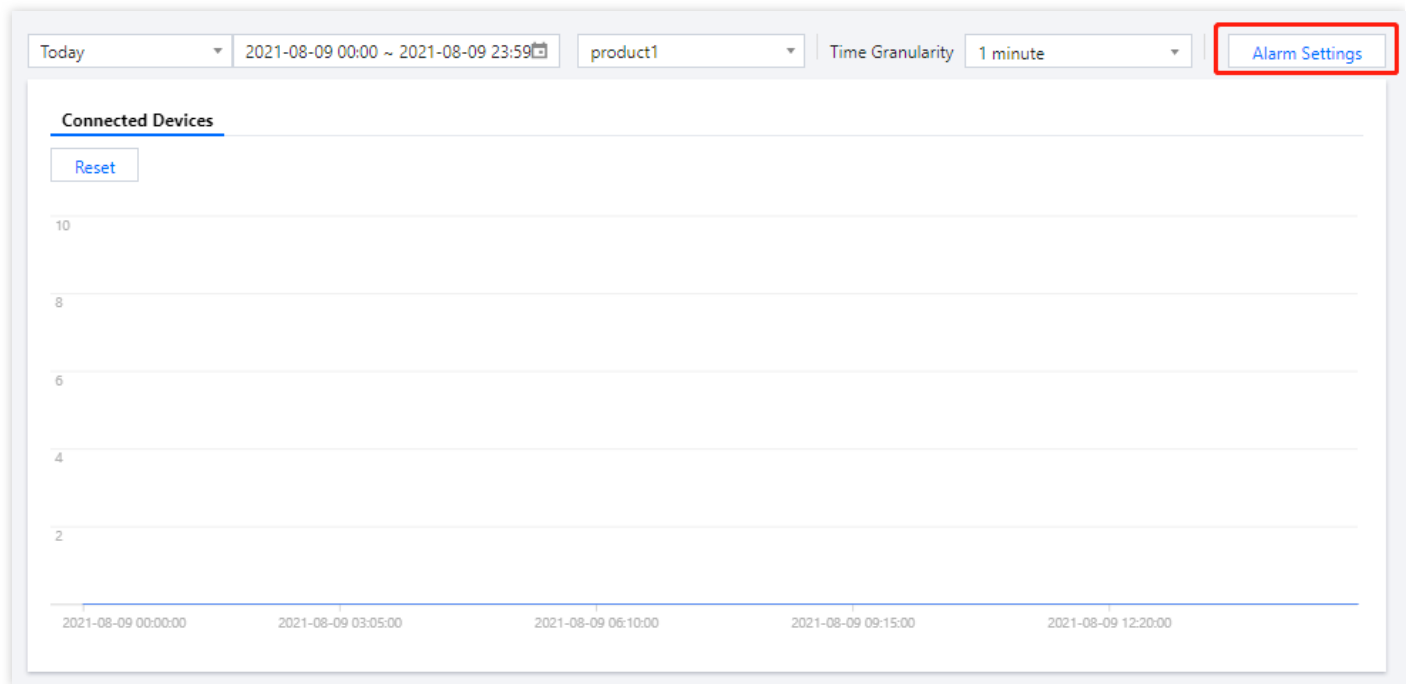
2. On the alarm policy page, click **Create** to enter the policy creation page and set the alarm policy parameters.

- Policy Name: it can contain letters and underscores.
- Remarks: it describes the alarm policy and can contain letters and underscores.
- Policy Type: it is the corresponding alarm policy type of the product. **IoT Hub** > **Status Monitoring** is selected here.
- Project: it is the project of the alarm policy. The default project is selected here.
- Alarm Object: it can be **All objects**, **Certain objects**, or **Instance group**.
  - **All objects** means all product objects in IoT Hub. When any product in IoT Hub hits the alarm condition, an alarm notification will be sent.

- **Certain objects** means the selected product objects. When a selected product object hits the alarm condition, an alarm notification will be sent.
      - **Instance group** means the product objects included in the instance group. When an object in the instance group hits the alarm condition, an alarm notification will be sent.
  - Trigger Condition: you can either select a trigger condition template or configure a trigger condition. For more information, please see Creating Alarm Policy. Trigger conditions that can be set include the following:
      - You can set the alarm to be triggered when meeting any condition or all conditions.
      - You can set the categories for monitoring according to the status types supported by IoT Hub.
      - You can set the statistical period and number of consecutive periods for the policy.
      - You can set the comparison relationship in the alarm policy.
      - You can set an alarm threshold.
      - You can set a notification policy. This way, an alarm notification will be sent repeatedly at a specified frequency when an alarm is triggered. Frequency options: do not repeat, once every 5 minutes, once every 10 minutes, at an exponentially increasing interval, and other frequency options. An exponentially increasing interval means that a notification is sent when an alarm is triggered the first time, second time, fourth time, eighth time, and so on. In other words, the alarm notification will be sent less and less frequently as time goes on to reduce the disturbance caused by repeated notifications.

> Note：
>
> The default logic for repeated alarm notifications is as follows:
>
> - The alarm notification will be sent to you at the configured frequency for 24 hours after an alarm is triggered.
> - Following 24 hours after an alarm is triggered, the alarm notification will be sent once every day by default.
> - The alarm notification will be sent for the last time 72 hours after the alarm is triggered and then will no longer be sent.

Sample trigger condition: set the following: the alarm is triggered when meeting all conditions; the monitored category is the number of connected devices; the statistical period is 1 minute for 5 consecutive periods; the comparison relationship is "greater than"; the alarm threshold is 100; and the notification policy is to send notifications once every day. Then, the number of connected devices will be determined once every minute, and an alarm will be triggered when the number of connected devices in the product object is found to be greater than 100 for 6 consecutive times.

- Alarm Channel: you can set the recipients and the time periods, channels, and languages for receiving alarm notifications.
- Advanced Feature: after it is enabled, an auto scaling policy can be triggered when the alarm conditions are met.
- API Callback: alarm messages can be pushed to this address when the alarm conditions are met.

3. Click **Save**.

Rule Engine
Overview

Last updated：2021-08-20 16:11:59

Use

When communication is performed based on a topic, you can use the rule engine to process the data in the topic and then forward it to other Tencent Cloud services or your business backend services. You don't need to purchase a server or deploy a distributed architecture; instead, you can implement full-stack services for message acquisition, computing, and storage simply by configuring the rule engine in the console. The following forwarding types are supported:

- Data forwarding to another topic
- Data forwarding to a third-party service
- Data forwarding to CKafka
- Data forwarding to a CMQ topic
- Data forwarding to a CMQ queue
- Data forwarding to CTSDB
- Data forwarding to TencentDB for MySQL
- Data forwarding to TencentDB for MongoDB

Creating Rule

1. Log in to the IoT Hub console and select **Rule Engine** on the left sidebar.
2. Go to the rule engine page, click **Create Rule**, enter the rule name, and click **Confirm**.
   - Rule Name: it can contain up to 32 letters, digits, and underscores (the name cannot be modified once confirmed).

○ Rule Description: it can contain 0–256 characters and can be modified.

3. After the rule is created successfully, you will be automatically redirected to the rule details page.

Basic Information

Rule Name   rule1

Status   Disabled

Rule Description

Filter Data ⑦

Field

Topic $(productid)/$(devicename)/event

Condition

Current SQL   SELECT FROM '$(productid)/$(devicename)/event'

Action

Add Action

Forward Error Actions

Add Action

Then, you can write different forwarding rules.

Data Processing

Last updated：2021-09-26 17:42:37

Once a rule is created, you can write SQL statements to process the data in a certain type of topics. The IoT Hub console provides a simplified way to enter SQL statements.

For example, if you want to extract three fields `action`, `targetDevice`, and `count` from the JSON message in the `E23VBC3GE8/device_02/event` topic and then filter them by **count <= 3** to get the final processed data for further forwarding, use the following rule:

Action

When you extract the desired fields from the topic, you should consider performing some operations on them, such as forwarding or storage. Currently supported operations include:

- Data forwarding to another topic
- Data forwarding to a third-party service
- Data forwarding to CKafka
- Data forwarding to a CMQ topic
- Data forwarding to a CMQ queue
- Data forwarding to CTSDB
- Data forwarding to TencentDB for MySQL
- Data forwarding to TencentDB for MongoDB

When triggering a forwarding action, the rule engine will encapsulate the payload reported by the device in JSON format as shown below:

1. JSON sample for forwarding to CMQ/CKafka:

```
{
"MsgType": "Publish",
"Topic": "AD4GVS5549/device/data",
"Seq": 13107192,
"PayloadLen": 17,
"Payload": "dGhpcyBpcyBhIGV4YW1wbGU=",
"ProductId": "AD4GVS5549",
"DeviceName": "device",
"Time": "2018-08-14 15:12:05",
}
```

The descriptions of each field are as follows:

- MsgType: valid values: Publish (forwarding to configured message queue), Forward (forwarding to the hit rule engine), StatusChange (status change).
- PayloadLen: length of the payload of the message reported by the device in bytes.
- Payload: payload of the original message, which will be Base64-encoded by default.
- Event: it is present only for messages of the `StatusChange` type. Valid values: Online (connection), Offline (disconnection).
- Time: the timestamp when the forwarding action is triggered.

2. JSON sample for forwarding to a third-party service (http forward):

```
{
"devicename":"device",
"payload":{
"params":{
"power_switch":1,
"color":1,
"brightness":32
}
```

```
    },
    "productid":"AD4GVS5549",
    "seq":2,
    "timestamp":1587109346,
    "topic":"AD4GVS5549/device/data"
    }
```

The descriptions of each field are as follows:

- devicename: device name defined on the IoT Hub platform.
- payload: payload of the original message. If it is in JSON format, it will be passed through for forwarding; if it is in binary format, it will be Base64-encoded.
- seq: internal auto-incrementing unique message identifier in `int` type.
- timestamp: the Unix timestamp when the forwarding action is triggered.

Notes

Field definition

- A field can contain up to 300 asterisks, commas, dots, spaces, letters, and digits and cannot be empty.
- A field indicates the key in JSON. If the data is in binary format, the field cannot be used for filtering, and `*` can be used to forward all binary data.
- The reported JSON data can be in nested JSON format, such as {"device_status":{"switch":"on"}}, where `device_status.switch` can be used to get the value of `switch`.
- SQL and JSON subarrays are not supported.

Topic wildcard definition

If you want to listen on multiple topics, you can use `#` and `+` wildcards to define them.

- `#` represents 0 or more arbitrary topic segments, which can only be placed at the end of the topic.
- `+` represents 1 arbitrary topic segment, which can be placed in the middle of the topic.

For example, `house_monitor/+/get`

- Can listen on topics such as `house_monitor/thermometer/get` and `house_monitor/door/get`.
- Cannot listen on the topic `house_monitor/door/switch/get`, because `+` can represent only 1 topic segment.

For example, `house_monitor/#`

- Can listen on topics such as `house_monitor/thermometer` and `house_monitor/door/switch/get`.
- Cannot listen on `house/#/get`, because `#` can only be placed at the end of the topic.

Condition definition

The [condition] expression is used to filter messages in the topic. Only when a message satisfies the [condition] expression will it be extracted for further processing. Supported expressions are listed in the table below:

Operator	Description	Example
=	Equal to	color = 'red'
<>	Not equal to	color <> 'red'
AND	Logic AND	color = 'red' AND siren = 'on'
OR	Logic OR	color = 'red' OR siren = 'on'
( )	The content in the parentheses is a complete entity	color = 'red' AND (siren = 'on' OR siren ='isTest')
+	Arithmetic addition	age = 4 + 5
-	Arithmetic subtraction	age = 5 - 4
/	Division	age = 20 / 4
*	Multiplication	age = 5 * 4
%	Remainder	age = 0 % 6
<	Less than	5 < 6
<=	Less than or equal to	5 <= 6
>	Greater than	6 > 5
>=	Greater than or equal to	6 >= 5

Rule Function

Last updated：2021-08-20 16:11:59

The rule engine provides a wide variety of functions, which you can use in the corresponding values of the rule engine fields, conditions, and database fields to process data in diverse ways.

Supported Functions

Function	Description
productId()	Returns the ID of the product from which the message comes
deviceName()	Returns the name of the device from which the message comes
timestamp()	Returns the current Unix system timestamp in seconds
topic()	Returns the original topic from which the message comes
topic(n)	Returns the segment n separated by `/` in the original topic from which the message comes
payloadLen()	Returns the length of the payload in bytes
bin_to_dec()	Converts the binary data to a decimal integer
to_hex ()	Converts the entered original message to a hexadecimal string
randint(min,max)	Returns a random integer between `min` and `max`
upper(string)	Returns an uppercase string (the entered message should be in JSON format, and the function object is the corresponding `key` value. For example, if the entered message is `"tencent":"iot"`, then `upper(tencent)=IOT`)
lower(string)	Returns a lowercase string (the entered message should be in JSON format, and the function object is the corresponding `key` value)
crypto(field,String)	Encrypts the value of `field`, where the second parameter `String` is an algorithm string. Valid values: MD5, SHA1, SHA256, SHA384, SHA512. (The entered message should be in JSON format, and the function object is the corresponding `key` value)
concat(string1, string2)	Concatenates strings, such as `concat(deviceid,'a')` or `concat(field1,field2)`
requestId()	Returns the ID of the message generated by IoT Hub
newuuid()	Returns a random UUID string

Function	Description
replace(source, substring, replacement)	Replaces `substring` in `source` with `replacement`
substring(source, start, end)	Returns the string between `start` (inclusive) and `end` (not inclusive)

Samples

A home temperature and humidity monitor device `dev00` sends the following messages to the cloud:

```
{"room1":{"temperature":31,"humidity":"63%"},
"room2":{"temperature":26,"humidity":"63%"}}
```

Three temperature and humidity monitor devices `dev00` , `dev01` , and `dev02` under the temperature and humidity monitor product monitor the temperature and humidity in six rooms `room1` , `room2` , `room3` ... `room6` respectively, and only when the temperature in room `room1` is above 30°C, the data needs to be transferred to the MySQL database for processing. In this example, the settings of the rule engine are as follows:

Filter Data ⑦

Edit SQL Debugging

Field	action,targetDevice,count
Topic	$ota/report/░░░░░░░░░░/door1
Condition	cout<=3
Current SQL	SELECT action,targetDevice,count FROM '$ota/report/░░░░░░░░░░/door1' WHERE cout<=3

Add Action ✕

ⓘ This action will insert data into TencentDB for MySQL. View Document

Action

Forward data to TencentDB for MySQL ▾

Region *

[_____] ▾

Instance *

[_____] ▾

MySQL Database *

[_____] ▾

Data Table *

[_____] ▾

Instance Account ⓘ *

[_____]

Password ⓘ *

[_____]

Data Fields

Field Name ⓘ	Value ⓘ		
table_temperature	${temp}	+	−
table_humidity	${hum}	+	−
prodductId	${productId()}	+	−
deviceName	${deviceName()}	+	−

☐ Use batch configuration ⓘ

[Save] Cancel

Data Forwarding to Another Topic

Last updated：2021-08-23 11:33:40

Overview

Machine-to-Machine (M2M) communication among different devices can be achieved by forwarding the desired message fields to another topic. You can enter the following topic information:

- **Enter a topic name**

  For example, `${productId}/house_monitor/thermometer` can forward messages satisfying the rule to this topic.

- **Enter a topic name with variables**

  For example, `${productId}/${house}/device`, where `house` in `${}` represents a variable name, which is the content of the field selected in the SELECT statement.

Samples

The following sample shows how a forwarding topic with variables works. Assume that the following rule is defined:

```
SELECT temperature as t, house
FROM house_monitor/thermometer/get
WHERE house="tencent" AND temperature > 40
```

This rule extracts the values of the `t` and `house` fields from the message. Assume that the content of the `house` field is `tencent`.

If you define forwarding to the `house_monitor/${house}/app` topic, the rule engine will replace the `${house}` variable in this topic with `tencent` and send the values of the `t` and `house` fields to the `house_monitor/tencent/app` topic.

The entire forwarding process is as shown below:
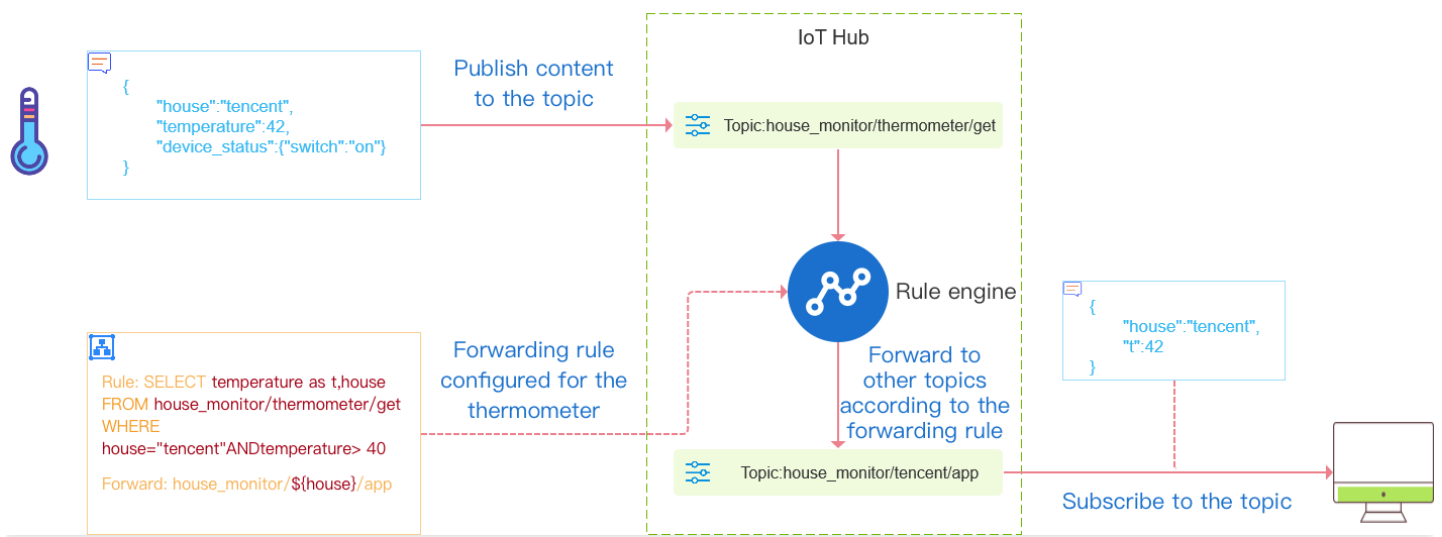
Configuration

1. Log in to the IoT Hub console, select **Rule Engine** on the left sidebar, and click the rule to be configured.

2. On the rule details page, click **Add Action**.

3. In the pop-up **Add Action** window, enter relevant information and click **Save**.
   - Select "republish" as the action type.

○ Enter the name of the topic to forward to.

IoT Hub will forward the reported data to this topic.

QoS Levels of Forwarded Messages

The quality of service (QoS) level of a message will not change when it is forwarded from the source topic to another topic.

- If a message published by a device is at QoS 0, the rule engine will forward it according to QoS 0; if it is at QoS 1, the rule engine will forward it according to QoS 1.
- At QoS 0, if message forwarding fails, the system will discard the message. At QoS 1, if message forwarding fails, the system will retry forwarding the message at intervals of 3s, 6s, and 9s in sequence, and if all three retries fail, the system will store the message in the offline message queue.

Data Forwarding to Third-Party Service

Last updated：2021-08-31 12:03:50

Overview

When forwarding a message field extracted through a rule to a third-party service, you can customize how to process the data. This is the most flexible way for you to process the message.

> Note：
>
> The third-party service must be HTTP- or HTTPS-based. To configure forwarding to a third-party service, you need to provide a URL and port number supporting HTTP or HTTPS. After successful forwarding by the rule engine, the third-party service will receive packets from `42.193.134.62`.

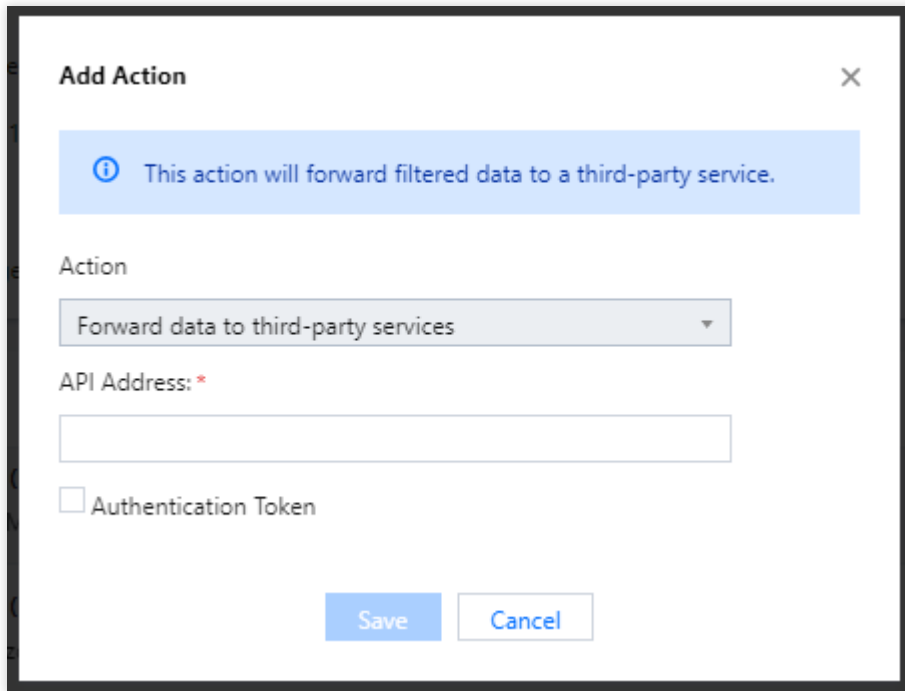The figure below shows the entire process of forwarding data to a third-party service:
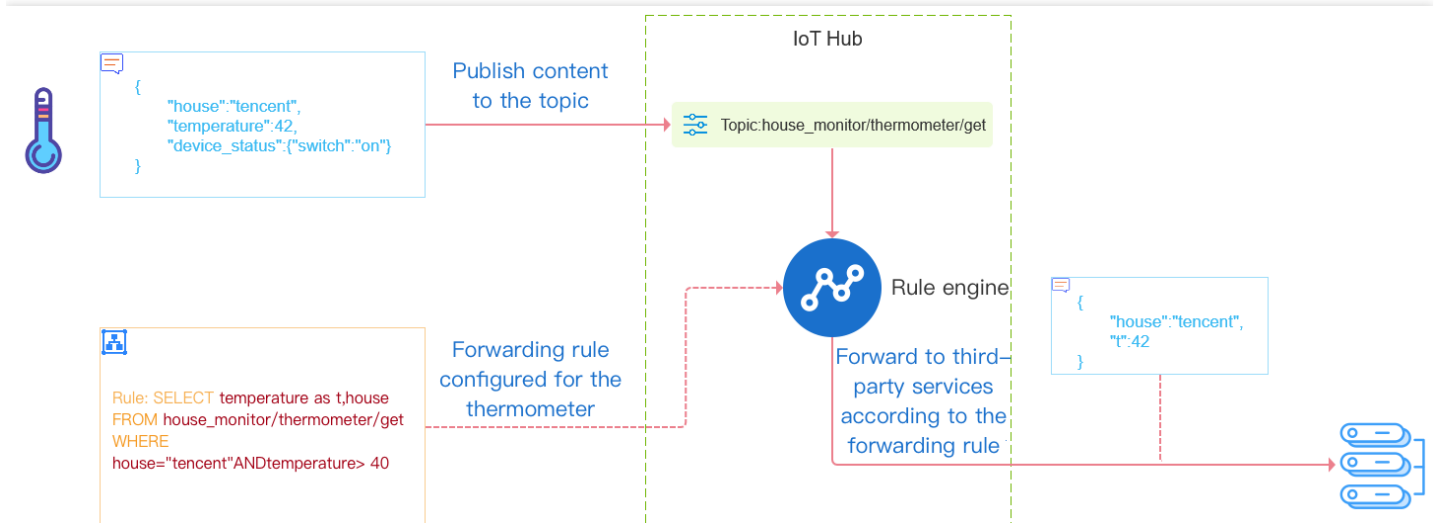
For more information on the content and format of the data to be forwarded, please see Data Processing.

Entering Server Configuration

1. Log in to the IoT Hub console and select **Rule Engine** on the left sidebar.
2. Click the rule to be configured to enter the rule details page and click **Add Action**.
3. In the pop-up **Add Action** window, enter relevant information and click **Save**.

- Select "forward" as the action type.
- Enter your HTTP or HTTPS service address. IoT Hub will forward data reported by devices to this address.
- Please select **Authentication Token** and enter the corresponding token of your service. You can enter any token for signature generation (this token will be compared with the token contained in the URL of the API for authentication).

Verifying Message Source as IoT Hub

> Note：
>
> To ensure the stable use of your backend, please select **Authentication Token**.

Request ID

If **Authentication Token** is selected for forwarding to the third-party service (i.e., HTTP forwarding), IoT Hub will add the following fields to the header of the HTTP or HTTPS request:

Parameter	Description
Signature	`Signature` combines the `Token` parameter entered in **Add Action** and the `Timestamp` and `Nonce` parameters in the request
Timestamp	Timestamp

Parameter	Description
Nonce	Random number

1. The `Token` , `Timestamp` , and `Nonce` parameters are sorted in lexicographical order.

2. The three parameter strings are concatenated into one string for SHA1 encryption.

3. After getting the encrypted string, you can compare it with `Signature` and verify whether the request comes from IoT Hub.

The sample code used to verify `Signature` in PHP is as follows:

```php
private function checkSignature()
{
$signature = $_GET["signature"];
$timestamp = $_GET["timestamp"];
$nonce = $_GET["nonce"];

$token = TOKEN;
$tmpArr = array($token, $timestamp, $nonce);
sort($tmpArr, SORT_STRING);
$tmpStr = implode( $tmpArr );
$tmpStr = sha1( $tmpStr );

if( $tmpStr == $signature ){
return true;
}else{
return false;
}
}
```

For example, the relevant parameters of a request are as follows, and `Token` is set to `aaa` .

```
Nonce: IkOaKMDalrAzUTxC
Signature: c259ed29ec13ba7c649fe0893007401a36e70453
Timestamp: 1604458421
```

The string after sorting is `1604458421IkOaKMDalrAzUTxCaaa` , and the final SHA1 result is calculated as `c259ed29ec13ba7c649fe0893007401a36e70453` .

Service address verification

1. When the rule engine is enabled, IoT Hub will send a GET request to the entered server URL, and the following fields will be added to the header of the GET request:

Parameter	Description
Signature	`Signature` combines the `Token` parameter entered in **Add Action** and the `Timestamp` and `Nonce` parameters in the request
Timestamp	Timestamp
Nonce	Random number
Echostr	Random string

Sample message sent by IoT Hub to the third-party service:

```
GET / HTTP/1.1
Host: **.**.**.**:4443
User-Agent: Go-http-client/1.1
Content-Type: application/json
Echostr: UPWIAFASvDUFcTEE
Nonce: testrance
Signature: abb6c316a8134596d825c5a1295bfa6f7657664d
Timestamp: 1623149590
Accept-Encoding: gzip
```

2. If the third-party service confirms that the GET request comes from IoT Hub, it needs to return the content of the `Echostr` parameter as-is in the `body` .

Sample message replied by the third-party service to IoT Hub:

```
HTTP/1.1 200 OK
Date: Tue, 08 Jun 2021 10:53:10 GMT
Content-Length: 16
Content-Type: text/plain; charset=utf-8

UPWIAFASvDUFcTEE
```

3. IoT Hub verifies the content of the returned `Echostr` parameter to check whether the server URL is valid.

Resending Mechanism

The resending mechanism is used to send the message again in case of a failure in the message forwarding process, which makes sure that the message is received. The details are as follows:

- If message forwarding fails, the system will retry forwarding at intervals of 1s, 3s, and 10s in sequence. If all three retries fail, the message will be discarded.

- If you have configured the forward error action, then after three unsuccessful retries, the message will be forwarded again according to the configured action. If forwarding still fails, the message will be discarded.

Data Forwarding to CKafka

Last updated：2021-09-08 12:45:50

Overview

The rule engine allows you to configure rules to forward eligible data reported by devices to CKafka, and then your application server can read the data from CKafka for processing. This takes advantage of CKafka's high throughput to create a highly available message linkage.

The figure below shows the entire process of forwarding data to CKafka by the rule engine:
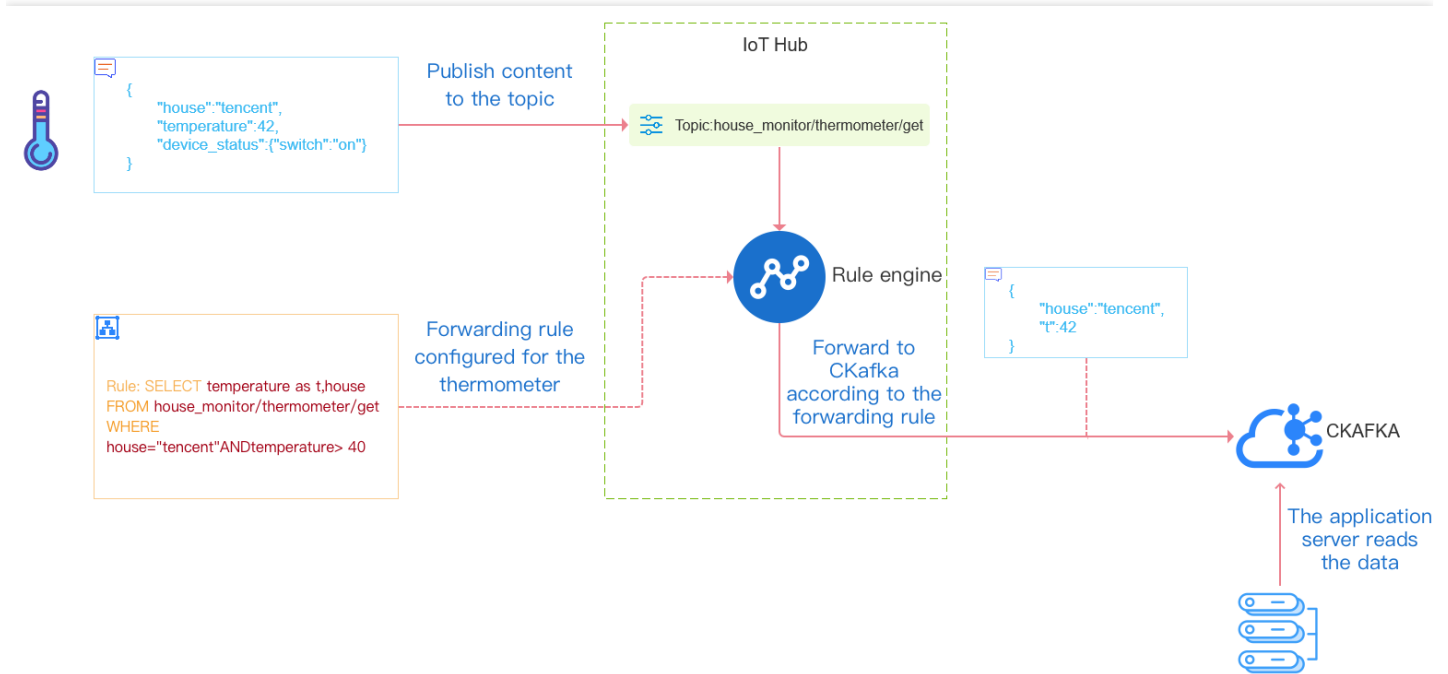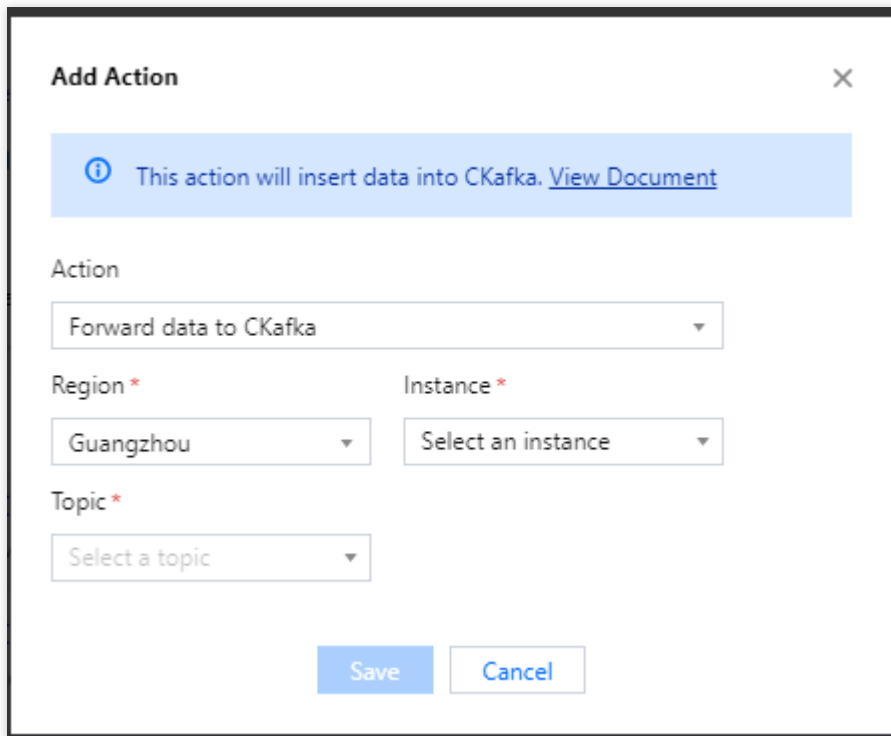
Configuration

1. Log in to the IoT Hub console and click **Rule Engine** on the left sidebar.

2. Go to the rule engine page and click the rule to be configured.

3. On the rule details page, click **Add Action**.

> Note：

You will be prompted to authorize access to CKafka upon the first use. You need to click **Authorize Now** before you can proceed.

4. In the **Add Action** pop-up window, select **Forward data to CKafka**, CKafka instance, and topic and click **Save**.

5. After the above configuration is completed, IoT Hub will forward eligible data reported by devices to the configured CKafka instance. You can refer to Process Overview to read and process the data on your own application server.

Resending Mechanism

The resending mechanism is used to send the message again in case of a failure in the message forwarding process, which makes sure that the message is received. The details are as follows:

- If message forwarding fails, the system will retry forwarding at intervals of 1s, 3s, and 10s in sequence. If all three retries fail, the message will be discarded.
- If you have configured the forward error action, then after three unsuccessful retries, the message will be forwarded again according to the configured action. If forwarding still fails, the message will be discarded.

Data Forwarding to TDMQ

Last updated：2022-03-11 18:33:02

Overview

The rule engine allows you to configure rules to forward eligible data reported by devices to the message queue TDMQ topic. After subscribing to a TDMQ topic through TencentCloud API, you can receive messages pushed from the topic. The message push mechanism of the TDMQ topic provides the capability to receive messages asynchronously with high reliability.

The following graph shows the entire process of forwarding data to a TDMQ topic by the rule engine:

Configuration

1. Log in to the IoT Hub console and click **Rule Engine** on the left sidebar.
2. Click the rule to be configured.
3. On the rule details page, click **Add Action**.

> Note：
> You will be prompted to authorize access to the TDMQ topic if this is your first time using the rule engine.
> Click **Authorize Now** and continue to create.

4. In the **Add Action** pop-up window, select **Forward data to TDMQ**, region, and topic and click **Save**.

Resending Mechanism

The resending mechanism is used to send the message again in case of a failure in the message forwarding process, which makes sure that the message is received. The details are as follows:

- If message forwarding fails, the system will retry forwarding at intervals of 1s, 3s, and 10s in sequence. If all three retries fail, the message will be discarded.
- If you have configured the "action for forwarding failure", then after three unsuccessful retries, the message will be forwarded again according to the configured action. If forwarding still fails, the message will be discarded.

Data Forwarding to CTSDB

Last updated：2021-09-08 14:25:52

Overview

The rule engine allows you to configure rules to forward eligible data reported by devices to CTSDB, and then your application server can read the data from CTSDB for processing. This takes advantage of CTSDB's high storage compression rate and aggregate display for massive amounts of data, effectively meeting the daily needs of devices for data storage, analysis, and visual display.

The figure below shows the entire process of forwarding data to CTSDB by the rule engine:
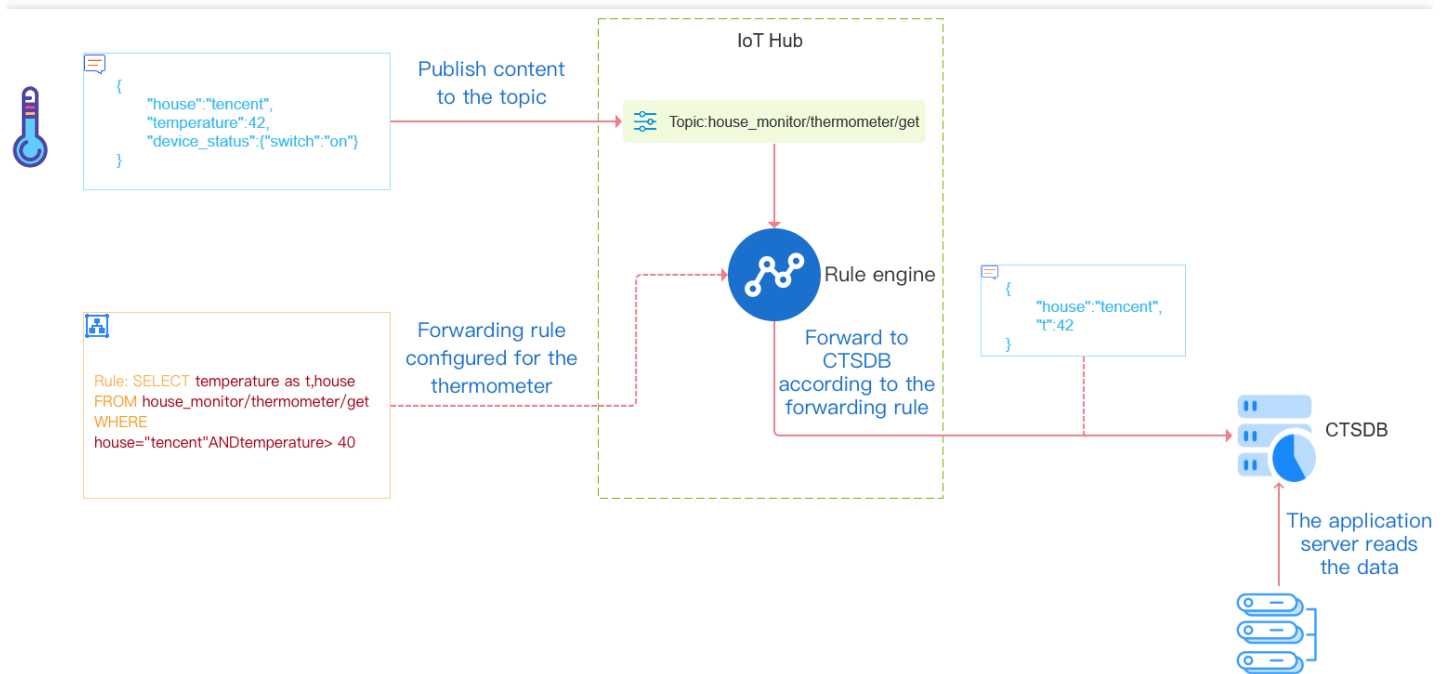
Configuration

1. Log in to the IoT Hub console and select **Rule Engine** on the left sidebar.
2. On the rule details page, click **Add Action**.

> Note：
>
> You will be prompted to authorize access to CTSDB upon the first use. You need to click **Authorize Now** before you can proceed.
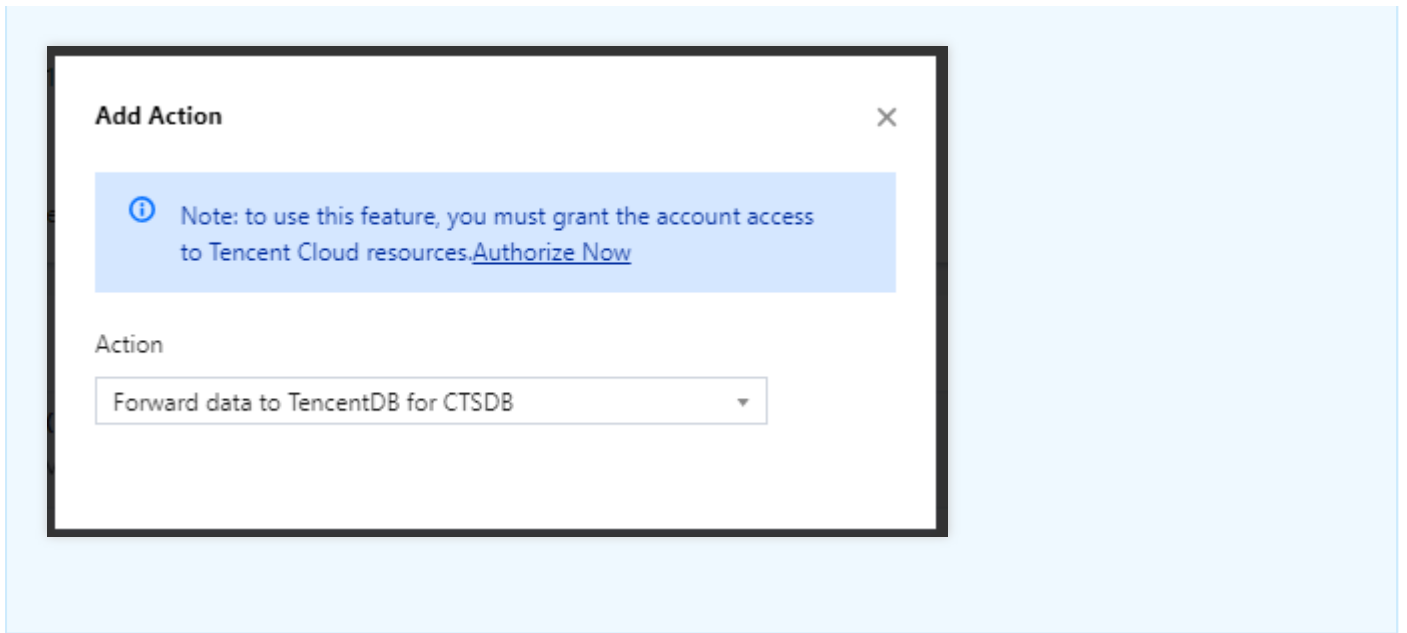
3. In the pop-up **Add Action** window, select **Forward data to TencentDB for CTSDB**, select the CTSDB region and instance, enter the basic information and the fields to be forwarded, and click **Save**.

After the above configuration is completed, IoT Hub will forward eligible data reported by devices to the configured CTSDB instance. You can refer to Overview to read the data on your own application server for processing or aggregate, search for, and query the data in the CTSDB console.

Configuration Parameter Description

- **Instance Account**: this is the account name entered when you create the CTSDB instance before configuring the rule engine.
- **Password**: this is the account password entered when you create the CTSDB instance before configuring the rule engine.
- **Metric**: this configures to which CTSDB metric to forward the data. If the metric is not present when the rule engine is configured, IoT Hub will create it automatically.
- **Timestamp**: this is the timestamp when the data is written to CTSDB. Currently, 4 types of configurations are supported:
  - The field value of the original message imported through `${}` .
  - System function.
  - timestamp(): the current system time of the message hitting the rule engine.
  - Constant: it needs to be a Unix timestamp in seconds; if it is left empty, the current time of the message hitting the rule engine will be used by default.

> Note：
>
> If you change the unit of the timestamp of the CTSDB metric to a unit other than second (such as millisecond) after the rule is created, subsequent data writes may fail.

- **Data Fields**: for the type, you can select tag or field type in CTSDB; for the restriction on field name, please see Overview; the value has 3 configuration methods: field value of the original message imported through `${}` , constant, or fixed value.

Advanced Configuration Description

The advanced configuration items are suitable for scenarios where the device-reported data fields are dynamically expanding and cannot be preconfigured. For example, if there are many sensors on the devices that need to transfer data, but different devices have different specifications and configurations, and the number of sensors is variable, then you can use the following advanced configuration to store the data from all device sensors to CTSDB through the rule

engine:

> Note：
>
> - Default Storage Type: the default storage type of dynamically expanding storage fields in CTSDB is tag.
> - Key: this is the JSON key that needs to traverse the expanding storage. IoT Hub will traverse the JSON key value nesting under this key with _ as the connector and finally store the data in CTSDB. The following example shows the JSON results and configuration retrieved by SQL SELECT in the rule engine (multiple subkeys can be configured) as well as the data actually stored in CTSDB:

Resending Mechanism

The resending mechanism is used to send the message again in case of a failure in the message forwarding process, which makes sure that the message is received. The details are as follows:

- If message forwarding fails, the system will retry forwarding at intervals of 1s, 3s, and 10s in sequence. If all three retries fail, the message will be discarded.
- If you have configured the forward error action, then after three unsuccessful retries, the message will be forwarded again according to the configured action. If forwarding still fails, the message will be discarded.

Data Forwarding to TencentDB for MySQL

Last updated：2021-08-23 11:33:40

Overview

The rule engine allows you to configure forwarding rules to forward eligible data reported by devices to TencentDB for MySQL. After you create an instance and table in the TencentDB for MySQL console or through TencentCloud API, specified fields in device messages can be written to the corresponding TencentDB for MySQL table.

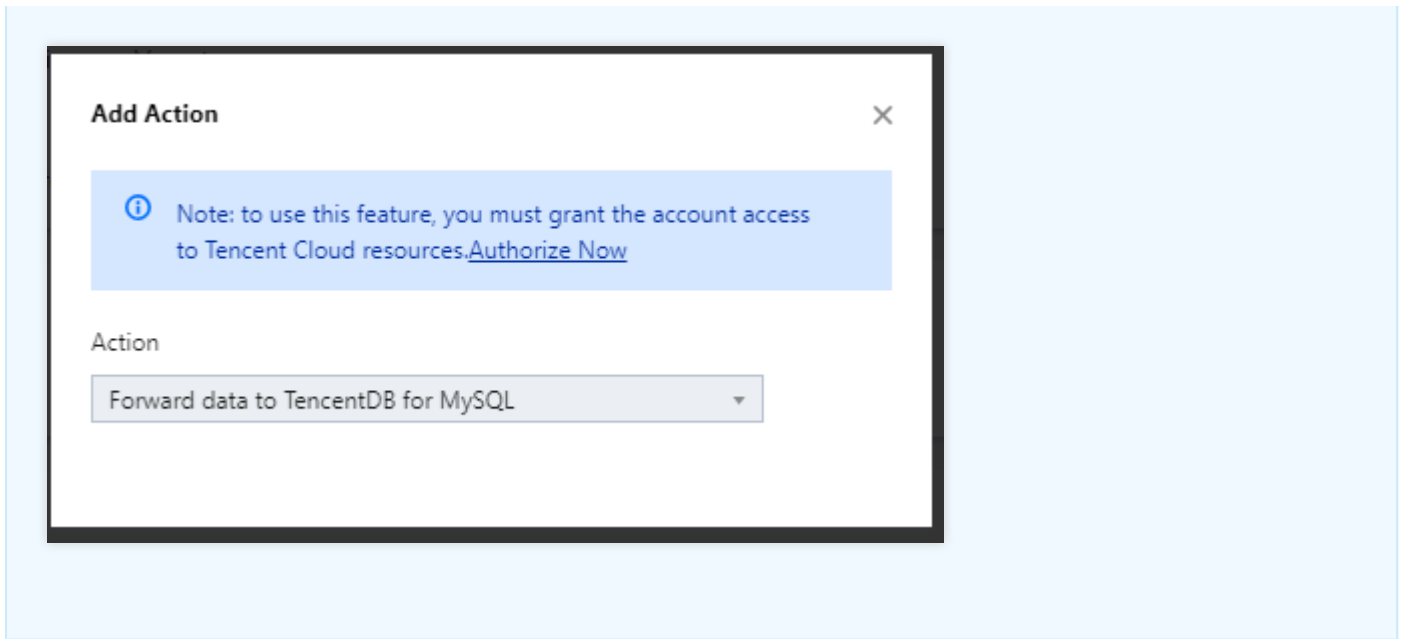The figure below shows the entire process of forwarding data to TencentDB for MySQL by the rule engine:
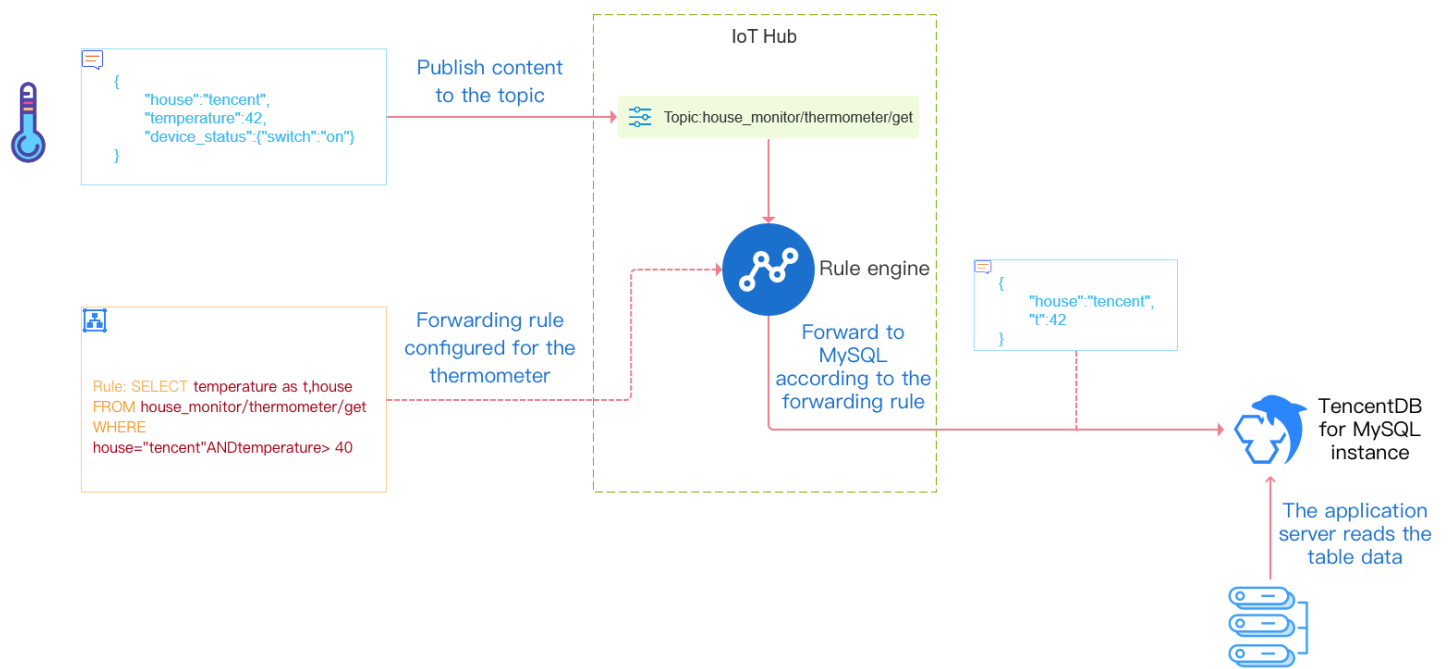
Configuration

1. Log in to the IoT Hub console and click **Rule Engine** on the left sidebar.

2. Go to the rule engine page and click the rule to be configured.

3. On the rule details page, click **Add Action**.

> Note：
>
> You will be prompted to authorize access to TencentDB for MySQL upon the first use. You need to click **Authorize Now** before you can proceed.

4. In the **Add Action** pop-up window, select **Forward data to TencentDB for MySQL**. After successful authorization, you need to configure the information of the TencentDB for MySQL instance and written fields as shown below. The configuration is divided into the following steps:

After successful forwarding, the information displayed in TencentDB for MySQL is as shown below:

Configuration Description

The configuration is divided into the following steps:

1. Select the region and TencentDB for MySQL instance.
2. Enter the username of the TencentDB for MySQL instance you just created.
3. Enter the login password of the instance.
4. Select the name of the database to be written to. If no database has been created under the created TencentDB for MySQL instance, go to the TencentDB for MySQL console to create one. For detailed directions, please see DMC Overview.
5. Select the table to be written to. If no table has been created under the created database, go to the TencentDB for MySQL console to create one.
6. Configure the fields to be written. There are two columns here: "field name" and "value". "Field name" corresponds to the field in the database table, indicating the field to be written. "Value" indicates the value of the field to be written. The value source can be a message body (which must be in JSON format to support value extraction) or a constant entered here.

> Note：
>
> - If the source is a message body, use "${}" to import the fields in the message body. If you want to specify a constant, just enter the corresponding value, such as 5 (number) or hello (string).
> - You must create the database, table, and fields in TencentDB for MySQL first before you can write data to the database.

For more information, please see DMC Overview.

Resending Mechanism

The resending mechanism is used to send the message again in case of a failure in the message forwarding process, which makes sure that the message is received. The details are as follows:

- If message forwarding fails, the system will retry forwarding at intervals of 1s, 3s, and 10s in sequence. If all three retries fail, the message will be discarded.
- If you have configured the "action for forwarding failure", then after three unsuccessful retries, the message will be forwarded again according to the configured action. If forwarding still fails, the message will be discarded.

Data Forwarding to TencentDB for MongoDB

Last updated：2021-08-23 11:33:41

Overview

The rule engine allows you to configure forwarding rules to forward eligible data reported by devices to TencentDB for MongoDB. After you create an instance in the TencentDB for MongoDB console or through TencentCloud API, device messages can be written to the corresponding TencentDB for MongoDB set.

The figure below shows the entire process of forwarding data to TencentDB for MongoDB by the rule engine:
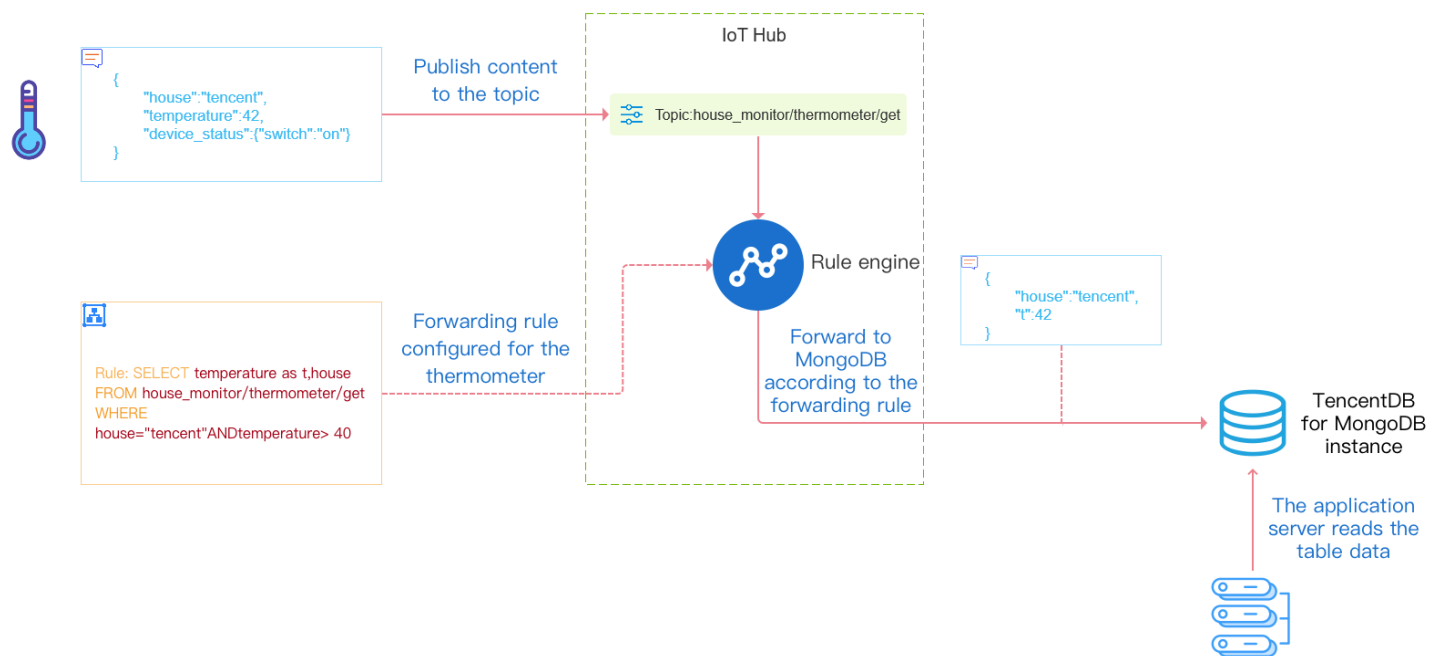
Configuration

1. Log in to the IoT Hub console and click **Rule Engine** on the left sidebar.
2. Go to the rule engine page and click the name of the rule to be configured.
3. On the rule details page, click **Add Action**.

> Note：
>
> You will be prompted to authorize access to TencentDB for MongoDB upon the first use. You need to click
>
> **Authorize Now** before you can proceed.
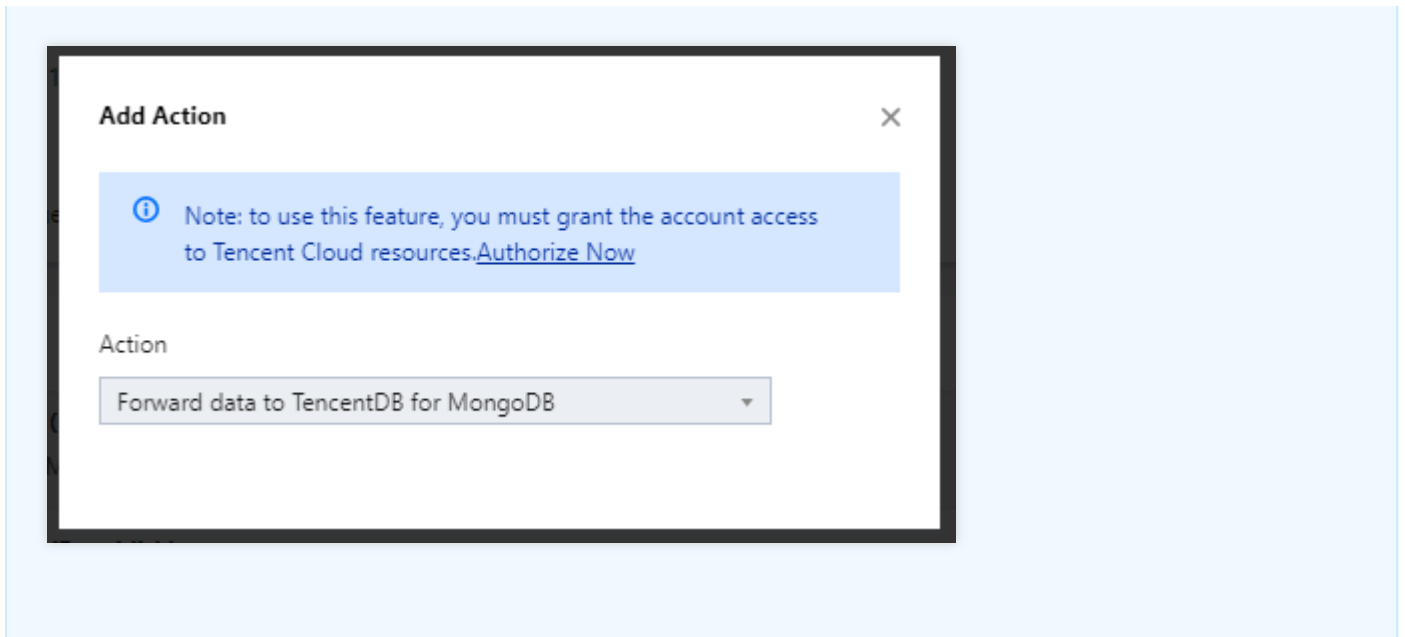
4. On the action adding page, select **Forward data to TencentDB for MongoDB**.
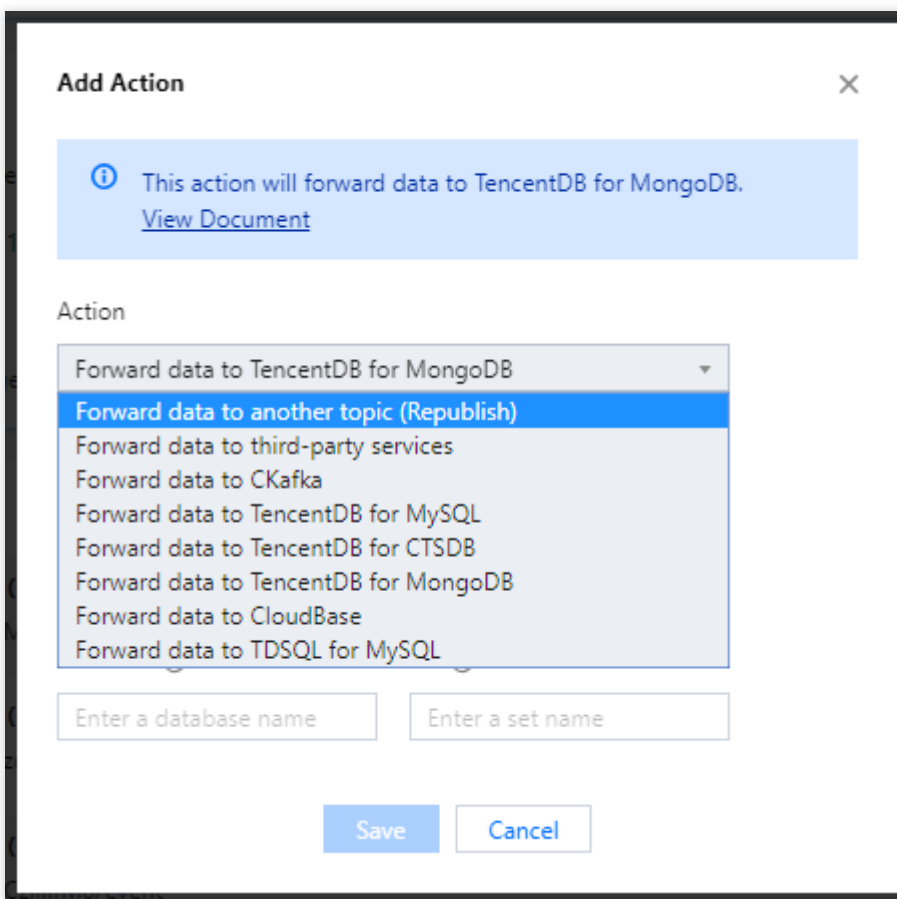
5. After successful authorization, you need to configure the TencentDB for MongoDB instance information as shown
   below. The configuration is divided into the following steps:

     1. Select the region and TencentDB for MongoDB instance. If there is no instance under the account, click
   Create Instance to redirect to the TencentDB for MongoDB console and create one.

ii. Enter the username of the TencentDB for MongoDB instance (`mongouser` by default).

iii. Enter the login password of the TencentDB for MongoDB instance.

iv. Enter the name of the database to be written to.

v. Enter the name of the set to be written to.

Resending Mechanism

The resending mechanism is used to send the message again in case of a failure in the message forwarding process, which makes sure that the message is received. The details are as follows:

- If message forwarding fails, the system will retry forwarding at intervals of 1s, 3s, and 10s in sequence. If all three retries fail, the message will be discarded.
- If you have configured the "action for forwarding failure", then after three unsuccessful retries, the message will be forwarded again according to the configured action. If forwarding still fails, the message will be discarded.

Data Forwarding to Tencent CloudBase

Last updated：2022-09-26 11:57:33

Overview

The rule engine allows you to configure forwarding rules to forward eligible data reported by devices to Tencent CloudBase (TCB). You can create an environment in the TCB console.

The figure shows the entire process of forwarding data to TCB by the rule engine:
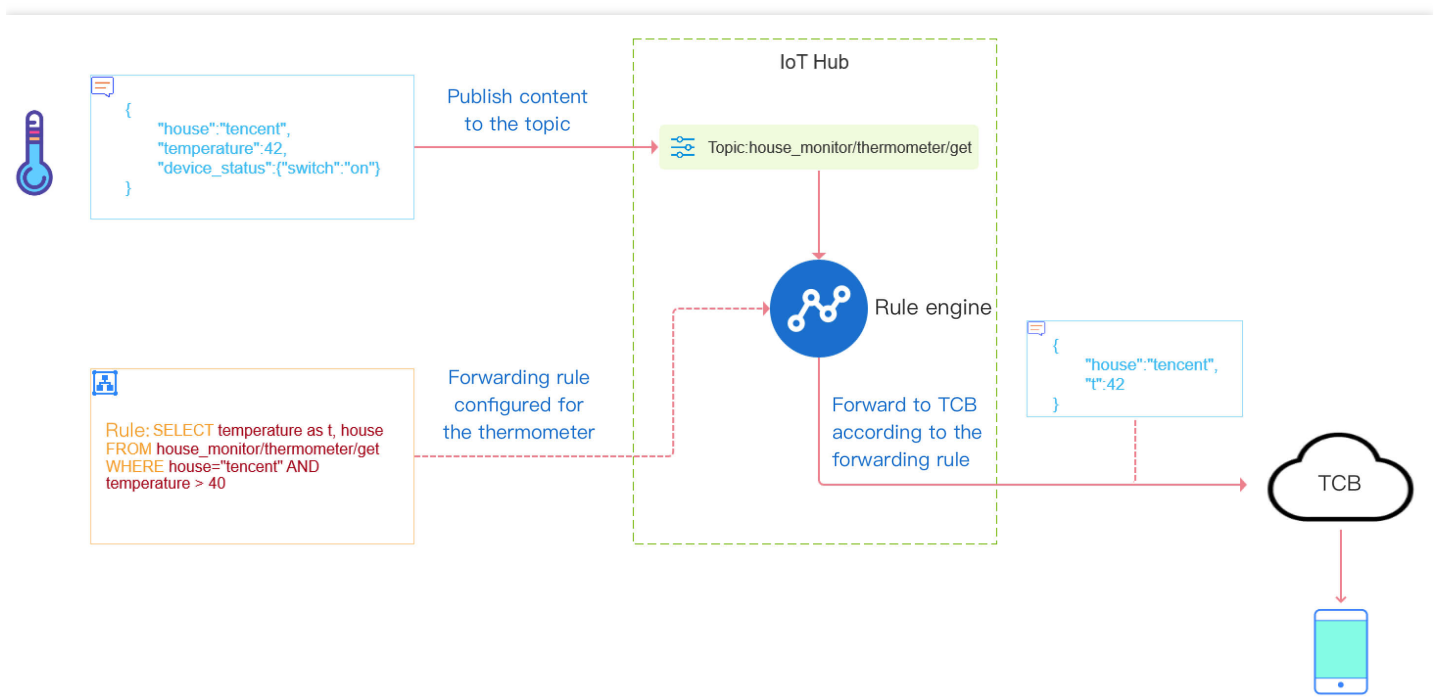
Configuration

1. Log in to the IoT Hub console and click **Rule Engine** on the left sidebar.
2. Click the rule to be configured.
3. On the rule details page, click **Add Action**.

> Note：
> You will be prompted to authorize access to TCB upon the first use. You need to click **Authorize Now** before you can proceed.

4. In the **Add Action** pop-up window, select **Forward data to Tencent CloudBase**, environment, and function and click **Save**.

> Note：
>
> Currently, data can be forwarded only to functions in TCB. You need to create a development environment and function in TCB before you can select them here.

Resending Mechanism

The resending mechanism is used to send the message again in case of a failure in the message forwarding process, which makes sure that the message is received. The details are as follows:

- If message forwarding fails, the system will retry forwarding at intervals of 1s, 3s, and 10s in sequence. If all three retries fail, the message will be discarded.
- If you have configured the **Forward Error Action** item, then after three unsuccessful retries, the message will be forwarded again according to the configured action. If forwarding still fails, the message will be discarded.

Data Forwarding to TDSQL for MySQL

Last updated：2022-09-26 11:47:25

Overview

The rule engine allows you to configure forwarding rules to forward eligible data reported by devices to TDSQL for MySQL. After you create an instance and table in the TDSQL for MySQL console or through TencentCloud API, specified fields in device messages can be written to the corresponding TDSQL for MySQL table.

The figure shows the entire process of forwarding data to TDSQL for MySQL by the rule engine:
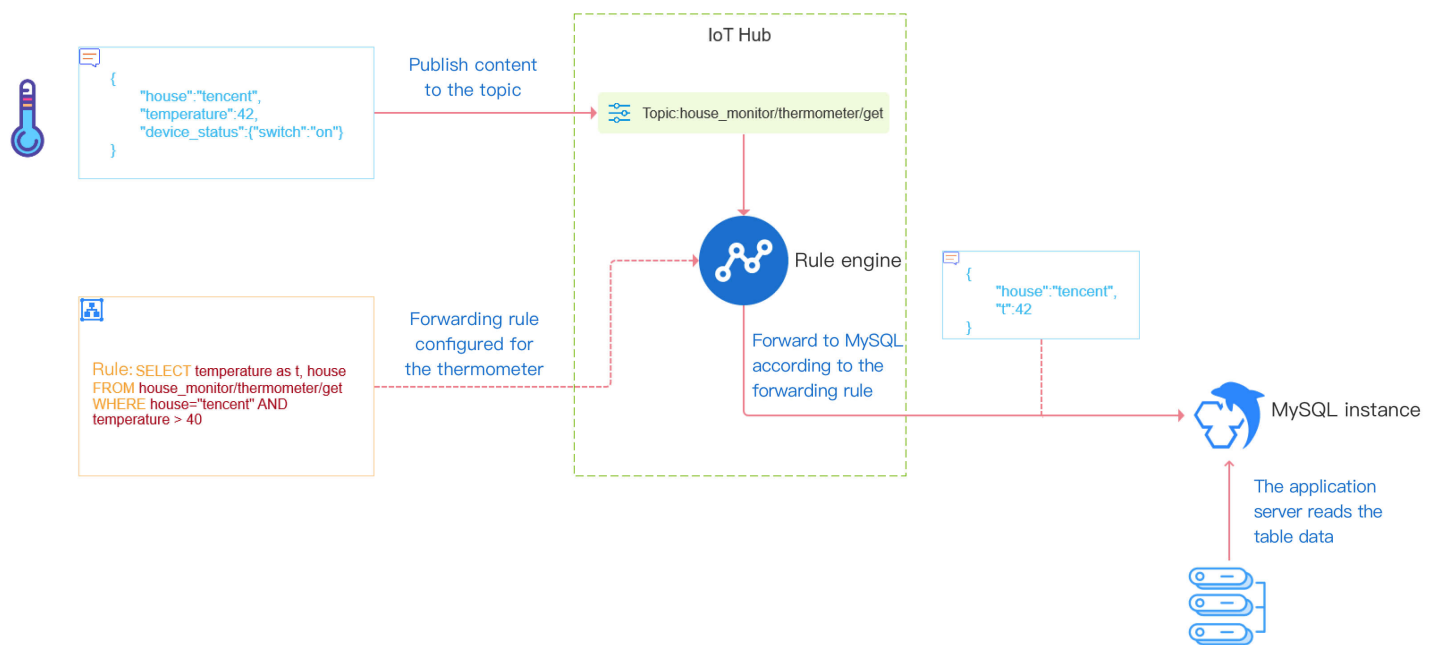
Configuration

1. Log in to the IoT Hub console and click **Rule Engine** on the left sidebar.
2. Click the **Rule Name** of the rule to be configured.
3. Click **Add Action**. You will be prompted to authorize access to the TDSQL instance if this is your first time using

   the rule engine. Click **Authorize Now** and continue to create.
4. After successful authorization, in the **Add Action** pop-up window, select **Forward data to TDSQL for MySQL**.
   You need to configure the information of the TDSQL for MySQL instance and written fields as shown below. The

configuration is divided into the following steps:

5. After completing the configuration, click **Save**.

Configuration Description

The configuration is divided into the following steps:

1. Select a region and a TDSQL for MySQL instance.
2. Enter the username of the TDSQL for MySQL instance you just created.
3. Enter the login password of the instance.
4. Select the name of the database to be written to. If no database has been created under the created TDSQL for MySQL instance, go to the TDSQL for MySQL console to create one.
5. Select the table to be written to. If no table has been created under the created database, go to the TDSQL for MySQL console to create one.
6. Configure the fields to be written. There are two columns here: "field name" and "value". "Field name" corresponds to the field in the database table, indicating the field to be written. "Value" indicates the value of the field to be written. The value source can be a message body (which must be in JSON format to support value extraction) or a constant entered here.

> Note
> - If the source is a message body, use `"${}"` to import the fields in the message body. If you want to specify a constant, just enter the corresponding value, such as 5 (number) or hello (string).
> - You must create the database, table, and fields in TDSQL for MySQL first before you can write data to the database.

Resending Mechanism

The resending mechanism is used to send the message again in case of a failure in the message forwarding process, which makes sure that the message is received. The details are as follows:

- If message forwarding fails, the system will retry forwarding at intervals of 1s, 3s, and 10s in sequence. If all three retries fail, the message will be discarded.
- If you have configured the **Forward Error Action** item, then after three unsuccessful retries, the message will be forwarded again according to the configured action. If forwarding still fails, the message will be discarded.

Sub-account Access to IoT Hub
Creating Sub-account

Last updated：2021-09-08 14:28:20

Overview

This document describes how to add a sub-account (with a **collaborator** as an example) under the root account.

Directions

1. Log in to the Tencent Cloud console as the **root account** and select **Cloud Products** > **Cloud Access Management** to enter the CAM console.
2. Select **Users** > **User List** on the left sidebar and click **Create User**.
3. On the user type selection page that pops up, select **Collaborator** or **Sub-user**.
4. Enter relevant information such as **username**, **login account**, **mobile** and **email** as required on the collaborator creation page and click **Next**.
5. Select a user group for the new collaborator. If there is no existing user group, click **Create User Group**.
6. Select **Select policies from the policy list**, search for "IoT" in the list, select the IoT-related policies as shown below, and click **Done**.

After the collaborator is created, you can view the collaborator information in the user list.

Subsequent Steps

Click the **collaborator username** to enter the user information management page and create an API key for the collaborator. This key is used by the collaborator to access the root account's IoT resources through RESTful APIs.

Sub-account Permission Control

Last updated：2021-08-30 17:58:56

Overview

This document describes how to grant sub-accounts the product-/device-level access control permissions.

- The product-level access control permissions allow sub-accounts to access and control the products created by themselves or created for them by the root account.
- The device-level access control permissions allow the sub-accounts to access and control only the devices created for them by the root account.

Authorization by Creating Policy by Policy Syntax

Creating policy

1. Log in to the CAM console and click **Policies** on the left sidebar.
2. Go to the policy management page and click **Create Custom Policy**.
3. On the **Select Policy Creation Method** page that pops up, select **Create by Policy Syntax**.
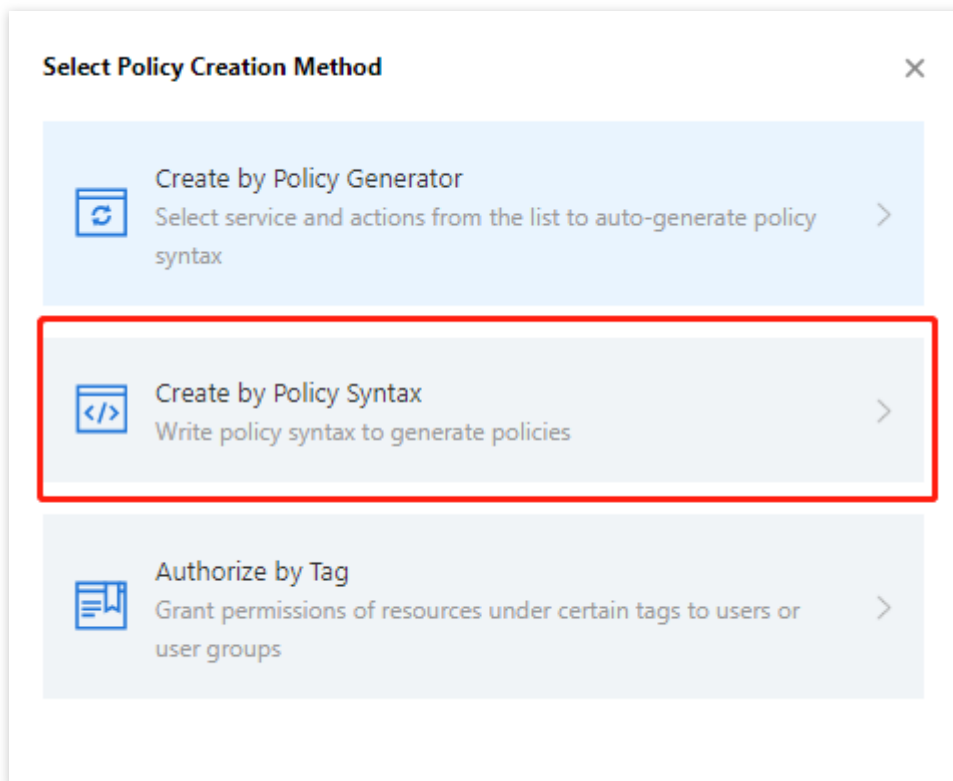
4. Select **Blank Template** and click **Next**.

5. Enter the custom policy name and edit the policy content based on the policy template. Below is the sample code:


```
{
"version": "2.0",
"statement": [
{
"action": [
"iotcloud:CreateProduct"
],
"resource": "*",
"effect": "deny"
},
{
"action": [
"iotcloud:*"
],
"resource": "*",
"effect": "allow",
```

```
"condition": {
"string_equal_if_exist": {
"product": [
"${productID1}",
"${productID2}",
"${productID3}"
]
}
}
}
]
}
```

Associating policy

1. After the custom policy is created, go to the User List page.
2. Select the sub-account to which to grant the permissions and click **Associate Policy** in the **Permissions** column.
3. Search for the name of the policy just created, select it, and click **OK** to grant the permissions defined in it.

Policy description

- The policy template below indicates that the sub-account is not allowed to create products. To disable other permissions for the sub-account, you can write the permission API names in `action`. For example, writing `iotcloud::DeleteDevice` there prohibits the deletion of devices by the sub-account.

```
{
"action": [
"iotcloud:CreateProduct"
],
"resource": "*",
"effect": "deny"
}
```

- The policy template below indicates that other permissions (such as device creation and deletion) are granted. However, these operations can only be performed under the specified product, subject to the PID entered in the product list (you can replace `${productID\*}` with the `productID` of the product in IoT Hub for authorization).

```
{
"action": [
"iotcloud:*"
],
"resource": "*",
"effect": "allow",
"condition": {
```

```
"string_equal_if_exist": {
"product": [
"${productID1}",
"${productID2}",
"${productID3}"
]
}
}
}
```

At this point, you can get the basic product information in the IoT Hub console.

Authorization by Tag

Creating device tag

1. Log in to the IoT Hub console and go to the product information page. If no products and devices are added, you need to add a product and device first. For detailed directions, please see Device Connection Preparations.

2. Click **Tag** on the **Device Information** page, click **Add**, and enter information such as key and value to add a device tag.

- Tag key: it can contain up to 16 letters, digits, and underscores.
- Tag value: it can contain up to 16 letters, digits, and underscores.

3. After editing, click **Confirm** to add the tag information, and the corresponding tag content will be displayed in the device information.

Creating and associating policy

1. Log in to the CAM console and click **Policy** on the left sidebar.
2. Go to the policy management page and click **Create Custom Policy**.

3. On the **Select Policy Creation Method** page that pops up, select **Authorize by Tag**.
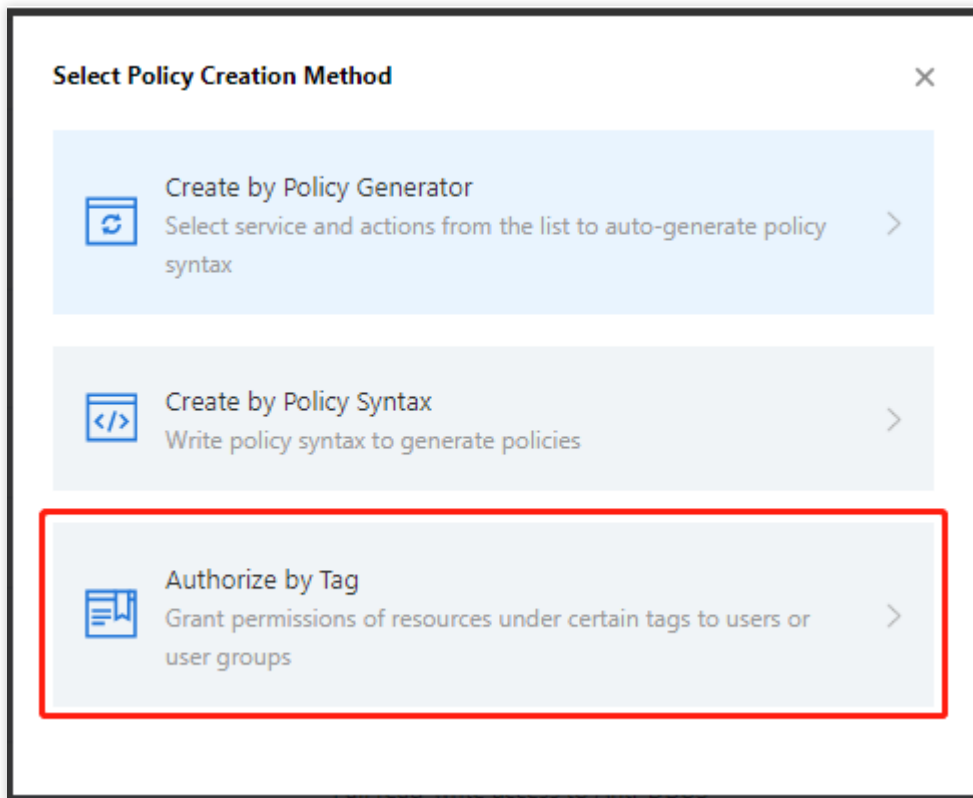
4. Select a user or user group and tag key and value.

Note: you can enter multiple tags for one single device, and the tag keys and values can be duplicate if they are on different devices. You can select multiple tag keys and values when selecting resources. You can also select a group of tag keys and values to assign resources. Such a group can assign one or multiple device resources to a sub-account.

← Authorize by Tag

1 **Tag Policy Generator** > ② Check and Finish

ⓘ Authorize users and user groups to perform operations of the services associated with specified tags

Users   subaccount ⊗   ▼

User Groups   Please select   ▼

Tags ⓘ   label01   ▼   01   ▼   ✕

     + Add

If existing tags do not meet your requirements, please create one ↗ in the console.

The selected users and user groups will be authorized to perform corresponding operations of the services below if such services are associated with all selected tags.

Cloud Service	Action Name	Description	Operation

Add Services and Operations

Next

5. After selection, click **Next** to enter the check page.

Here, the policy name and policy information can be modified. After confirming that everything is correct, click **Done** to create and associate the policy.

6. Due to the limit in the IoT Hub console, after device resources are assigned to a sub-user, the sub-user can enter the device information page and view authorized device resources only after getting the product and device list information. Therefore, you also need to authorize the product and device lists by creating a policy by policy syntax. The authorization code is as follows:

```
{
"version": "2.0",
"statement": [
{
"action": [
"iotcloud:DescribeProducts",
"iotcloud:DescribeDevices"
],
"resource": "qcs::iotcloud:::ProductId/*",
```

```
"effect": "allow"
    }
  ]
}
```

7. After completing the operation, the authorized sub-user can manage the corresponding device resources in the console.

Unauthorized device resources cannot be viewed.

Firmware Upgrade

Last updated：2021-11-26 11:07:16

Overview

This document describes how to quickly use the firmware update service in IoT Hub.

Directions

Adding firmware

1. Log in to the IoT Hub console, click **Firmware Update** on the left sidebar to enter the firmware list page, and you can view all firmware files in the project.

2. Click **Add Firmware** to add a firmware file.

- Firmware Name: it can contain up to 32 letters, digits, underscores, minus signs, and parentheses and must begin with a letter or digit.
- Product: select the product of the firmware to be uploaded.
- Version: it can contain 1–32 letters, digits, dots, hyphens, and underscores.
- Firmware File: the firmware file to be uploaded must be a .bin file or a .tar, .gz, or .zip package and cannot exceed 1,024 MB in size.
- Firmware Description: it is the description and record of the firmware to be uploaded and can contain 0–100 characters.
- Up to 100 firmware files can be uploaded under each account. If you want to upload more, you need to delete some old firmware versions first.

3. After successful upload, the firmware will be displayed in the list. You can perform operations such as updating, adding, deleting, querying, and modifying the firmware file and viewing its details.

Updating firmware

Select the target firmware version for update and click **Firmware Update** on the right of the firmware list to initiate an update task. You can batch update devices by firmware version number or device name.

Updating by firmware version number

1. Go to the firmware update page, which displays the information of the target firmware, such as firmware name, product, and version number.

2. Select **Firmware version** for **Batch Update By**.

- Source Versions: select one or multiple firmware version numbers in the drop-down list for update.
- Devices: you can select all or specified devices on the firmware version numbers for update as the target devices. The specified device update feature is usually used for beta test verification of firmware content. If you select **Custom** for **Devices**, click **Select Device** next to the drop-down list, and then you can batch select

target devices for update among all devices of the product.

Firmware Update   ✕

Firmware Name	firmware
Product	prod1
Version	1.1.1

Batch Update By ⓘ [Firmware version] [Device name]

Source Versions [0.1 ▾]

Devices [Custom ▾] [Select Device]

Timeout Period ⓘ [− | 15 | +] min

[Save] [Cancel]

3. Click **Save**, and the system will perform the update task by delivering the selected target firmware to the target devices.

> Note：
>
> Update by firmware version requires the target devices to report their currently running firmware versions. If the versions are not reported, you can select updating by device name as described below.

Updating by device name

1. Go to the firmware update page, which displays the information of the target firmware, such as firmware name, product, and version number.

2. Select **Device name** for **Batch Update By**.

3. Upload the list of devices whose firmware needs to be updated. Click **Download Template** to get the template file, enter the correct `DeviceName` values in the template, and click **Upload File** to upload it. Up to 10,000 devices can be updated at a time, and only CSV files are supported.

4. Click **Save**, and the system will perform the update task by delivering the firmware to the target devices.

Viewing firmware details

1. Click **View Details** on the right of the firmware list to view firmware details.

2. On the firmware details page, you can view the firmware details, statistics of devices updated to the firmware version, and update task management list.

- Firmware Information: it includes the firmware name, product, version number, signature, signature algorithm, adding time, and description. You can click **Edit** in the top-right corner to modify the firmware name and description.
- Firmware Update Statistics: it includes the total number of devices in all batch update tasks of the firmware and the numbers of corresponding devices in firmware update tasks in different update status.

○ Task management list:

■ Click **Tasks** to view all update tasks of this firmware. There are four update task status: not started, creating, created, and creation failed.

Task Management

Tasks	Devices				Enter a task ID 🔍
Tasks					
Task ID	Task Type	Task Status	Added		Operation
	Batch update	Created successfully	2021-08-09 15:49:26		View Details
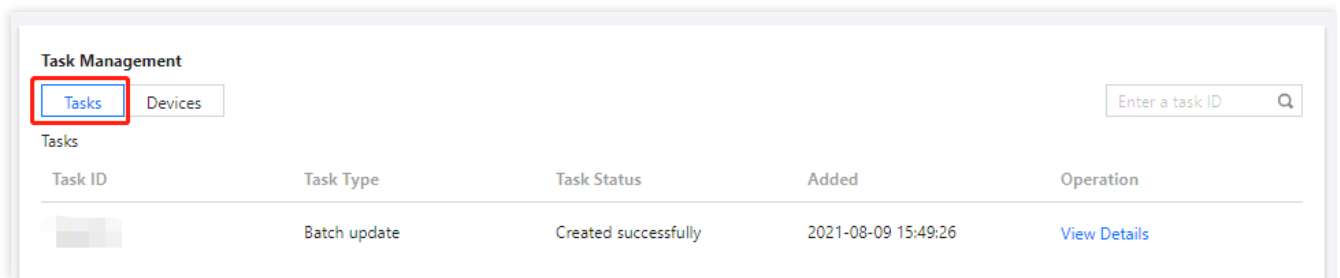
■ Click **Devices** to view the detailed records of devices in all update tasks of this firmware. There are five device update status: to be pushed, pushed, updating, updated, and update failed.
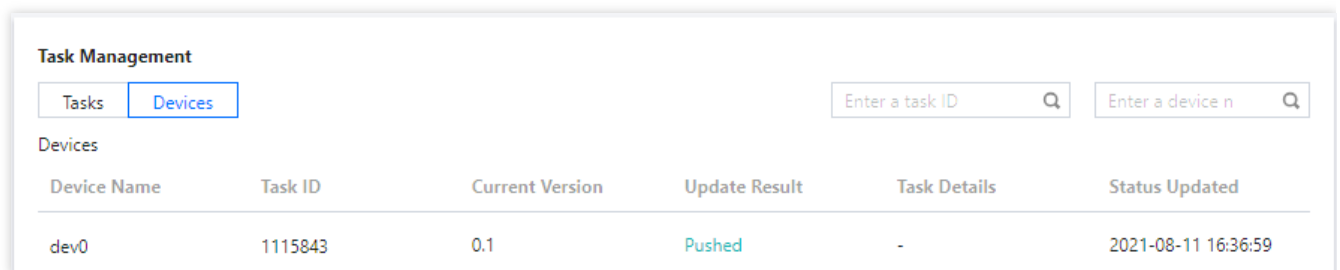
Task Management

Tasks	Devices			Enter a task ID 🔍	Enter a device n 🔍
Devices					
Device Name	Task ID	Current Version	Update Result	Task Details	Status Updated
dev0	1115843	0.1	Pushed	-	2021-08-11 16:36:59

3. In **Tasks** or **Devices** in task management, click **View Details** on the right of a task to enter the task details page, where you can view the list of target devices, update status, and numbers of devices in different update status in

this task.
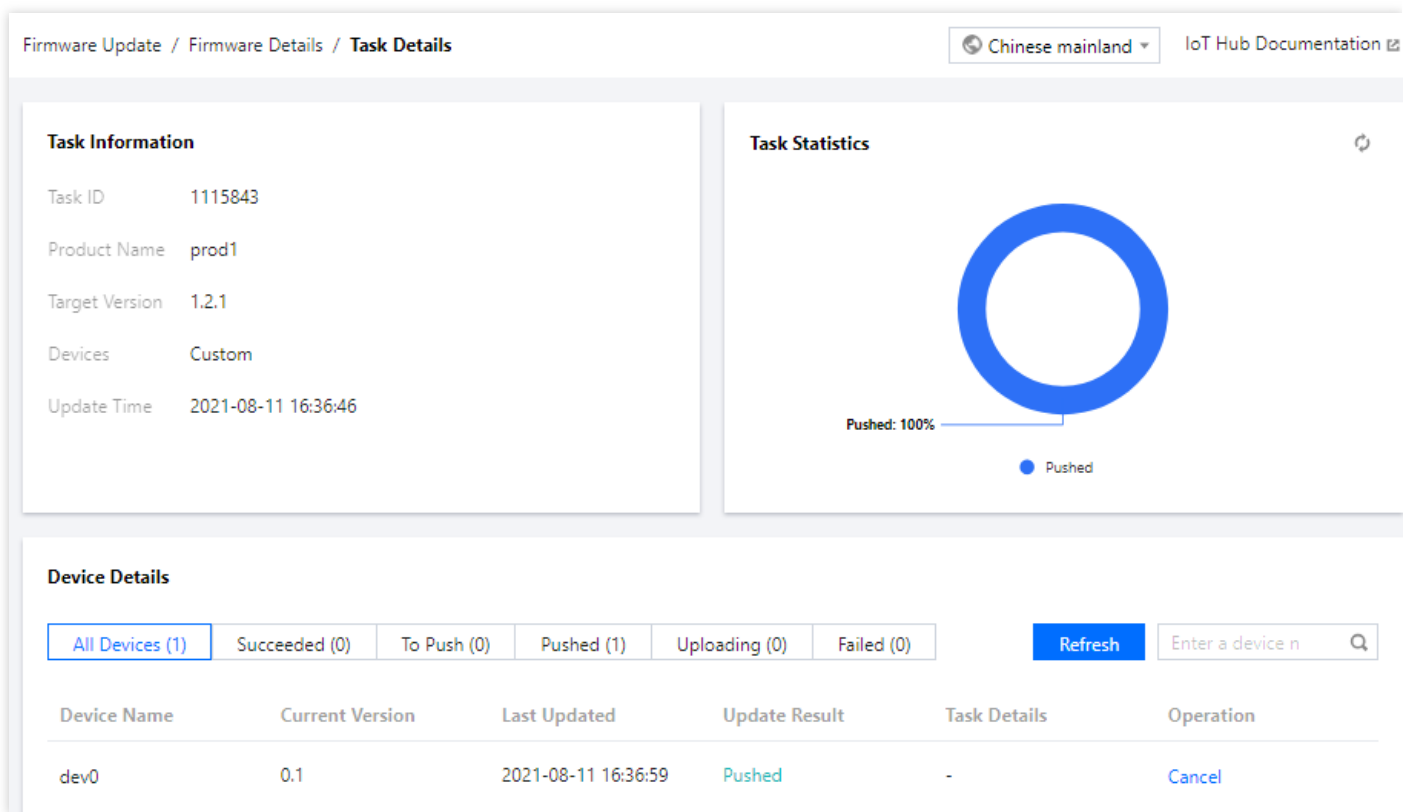
In the device details list, you can view the current update status and status details of all target devices in the batch update task. If the update status is **To push** or **Pushed**, no status details will be displayed. If the update status is **Updating**, the status details will display the combination of **Downloading** and **Burning** and the update progress values in percentage of each device. If the update status is **Failed**, the status details will display error messages.

In addition, you can cancel or retry device update on the right of the device details list based on the update progress. Devices whose update is canceled will be marked as **Failed**. For such devices, you can click **Retry** to update them again.

Resource Management

Last updated：2021-08-31 12:07:28

Overview

This document describes how to use the resource management feature to quickly deliver resources to devices or upload device resources to the IoT Hub platform.

Platform Resources

The platform resource feature allows you to deliver resources uploaded to the IoT Hub platform to devices.

Adding resource

1. Log in to the IoT Hub console, click **Resources** on the left sidebar, and select **Platform Resources** to enter the resource list page and view all uploaded resources currently on the platform.

2. Click **Add Resource** to add a resource by entering the resource information.
   - Resource Name: it can contain up to 32 letters, digits, underscores, minus signs, and parentheses and must begin with a letter or digit.
   - Product: select the product of the resource to be uploaded.
   - Resource: the resource file to be uploaded cannot exceed 1,024 MB in size.
   - Resource Description: it is the description and record of the resource to be uploaded and can contain 0–100 characters.

3. After successful upload, click **Save**, and the added resource will be displayed in the list. You can perform operations such as delivering, adding, deleting, querying, and modifying the resource and viewing its details.

> Note：
> You can upload up to 1,000 pieces or 1 GB of resource files under each account. If the limit is exceeded, please delete some old resources first.

Delivering resource

Select the resource to be delivered and click **Deliver Resource** on the right of the resource list to initiate a delivery task.

Resources can be delivered to a single device or batch delivered to multiple devices as detailed below:

- Delivery to One Device
- Batch Delivery to Multiple Devices

1. Go to the resource delivery page, select **Single Device**, and select the target device for delivery in the drop-down list.

2. Click **Save**, and the system will perform the delivery task by delivering the selected resource to the selected target device.

Viewing resource details

1. In the resource list, click **View Details** on the right to view the resource details.

2. On the resource details page, you can view the resource details, statistics of devices to which the resource has been delivered, and delivery task management list.

- Resource Information: it includes the resource name, resource signature, signature algorithm, adding time, and resource description.
- Delivery Statistics: it includes the total number of devices in the resource delivery task and the numbers of devices in different delivery status.
- Task management list:

- Click **Tasks** to view all delivery tasks of this resource. There are four delivery task status: not started, creating, created, and creation failed.

- Click **Devices** to view the detailed records of devices in all delivery tasks of this resource. There are five device delivery status: to push, pushed, delivering, delivered, and delivery failed.

3. In **Tasks** or **Devices** in task management, click **View Details** on the right of a task to enter the task details page, where you can view the list of target devices, delivery status, and numbers of devices in different delivery status in

this task.

In the device details list, you can view the current delivery status and status details of all target devices in the delivery task.

- If the delivery status is **To push** or **Pushed**, no status details will be displayed.
- If the delivery status is **Delivering**, the status details will display **Delivering** and the delivery progress values in percentage of each device.
- If the delivery status is **Delivery failed**, the status details will display the error messages.

In addition, you can cancel or retry resource delivery on the right of the device details list based on the delivery progress. Devices to which delivery is canceled will be marked as **Delivery failed**. For such devices, you can click **Retry** to deliver the resource again.

Device Resources

The device resource feature allows you to directly download and view resources uploaded by devices to the platform in the console.

Directions

1. A device uploads its resource to the platform over the resource upload communication protocol. After successful upload, the resource will be displayed in the resource list in the IoT Hub console.

2. Log in to the IoT Hub console, click **Resources** on the left sidebar, select **Device Resources** to enter the resource list page, where you can view all resources uploaded by devices and perform operations such as downloading and deleting resources.

Certificate Management

Last updated：2022-09-26 12:02:27

Overview

This document describes how to use the certificate management feature to quickly authenticate devices with private CA certificates.

Directions

To use a private certificate, you must first apply for a CA certificate from a certificate authority (CA) and then upload it to the IoT Hub platform.

Uploading certificate

1. Log in to the IoT Hub console, click **Certificate Management** on the left sidebar, and you can view all CA certificates uploaded to the platform.
2. Click **Add Certificate** to add a new CA certificate. Then, enter and upload the relevant information of the certificate.

   - CA Certificate Name: It can contain up to 32 characters including letters, numbers, underscores, hyphens, and @.
   - Upload CA Certificate: Upload a CA certificate issued by a certificate authority. Only CER, CRT, and PEM files are allowed.
   - Authentication Code: Used to generate an authentication certificate.
   - Upload Certificate: Use the CA certificate's private key and authentication code to generate a certificate to verify the correctness of the uploaded CA certificate. Only CER, CRT, and PEM files are allowed.
3. After successful upload, click **Save**, and the added CA certificate will be displayed in the list.

> Note：
> You can upload up to ten CA certificate files under one account.

Authenticating with custom CA certificate

1. Log in to the IoT Hub console, click **Products** > **Create Product** on the left sidebar, and enter the following information:

   - Region: It is **Guangzhou** by default.
   - Product Type: Select **General**.
   - Product Name: Enter a custom name, which can contain up to 40 characters including letters, numbers, underscores, hyphens, and @ symbols.
   - Authentication Method: Select **Certificate**.
   - CA Certificate: Select the name of the certificate you created.
   - **Data Format**:
     - JSON: You can match the rules and extract the content based on the data.
     - Custom: No data parsing is performed.

2. Click **Save**. The successfully created product will be displayed on the **Products** page.
3. On the **Products** page, click the **Product Name** to enter the product details page.
4. Click **Devices** > **Add Device** to upload the device certificate.

5. Click **Save**.
6. Click **Back to Devices** > **Product Settings** to view the basic product information, download the device CA certificate, and use the device certificate and private key for device link authentication.

Generating testing CA certificate

> Note：
>
> The CA certificate generated by this method is only used for testing, and you should apply for an official CA certificate from a certificate authority.

The following uses OpenSSL as an example to describe how to generate a testing CA certificate:

1. Prepare the CA certificate configuration file to get `ca.conf` with the following content:

```
[ req ]
default_bits = 4096
distinguished_name = req_distinguished_name

[ req_distinguished_name ]
```

```
countryName = Country Name (2 letter code)
countryName_default = CN
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Tencent
localityName = Locality Name (eg, city)
localityName_default = Shenzhen
organizationName = Organization Name (eg, company)
organizationName_default = Tencent IoT
commonName = Common Name (e.g. server FQDN or YOUR name)
commonName_max = 64
commonName_default = Tencent CA Test
```

2. Generate the CA certificate key to get `ca.key` with the following command:

```
openssl genrsa -out ca.key 4096
```

3. Generate a CA certificate issuance request to get `ca.csr` with the following command:

```
openssl req -new -sha256 -out ca.csr -key ca.key -config ca.conf
```

4. Generate a CA root certificate to get `ca.crt` with the following command:

```
openssl x509 -req -days 3650 -sha256 -extfile openssl.cnf -extensions v3_ca -i
n ca.csr -signkey ca.key -out ca.crt
```

Generating authentication certificate

The following uses OpenSSL as an example to describes how to generate an authentication certificate:

1. Generate the key pair of an authentication certificate with the following command:

```
openssl genrsa -out verificationCert.key 2048
```

2. Use the authentication code in the **Add Certificate** dialog box to create a CSR file with the following command:

```
openssl req -new -key verificationCert.key -out verificationCert.csr
```

Copy the **Authentication Code** from the **Add Certificate** dialog box and paste it as the value of the `Common Name` field.

```
Common Name (e.g. server FQDN or YOUR name) []: 9f5cfb6ec0fcbdffd94473491bbb05
2e339e5b7beff4d7ed46420b697****
```

3. Use the CA certificate, private key, and the CSR file generated in step 2 to create an authentication certificate with the following command:

```
openssl x509 -req -in verificationCert.csr -CA ca.crt -CAkey ca.key -CAcreates
erial -out verificationCert.crt -days 300 -sha512
```

Here, `ca.crt` and `ca.key` are the CA certificate and its private key file you obtained from your CA.

Issuing device certificate and private key

The following uses OpenSSL as an example to describe how to use a CA certificate to issue a device certificate and private key.

1. Generate a device private key with the following command:

```
openssl genrsa -out dev_01.key 2048
```

2. Create a CSR file with the following command:

```
openssl req -new -key dev_01.key -out dev_01.csr
```

The value of the `Common Name` field is the product ID + device name as follows:

```
Common Name (e.g. server FQDN or YOUR name) []: U58***2YLJdev_01
```

3. Use the CA certificate, private key, and the CSR file generated in step 2 to create a device certificate with the following command:

```
openssl x509 -req -in dev_01.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
dev_01.crt -days 3650 -sha512 -extfile openssl.cnf -extensions v3_req
```

Here, `ca.crt` and `ca.key` are the CA certificate and its private key file you obtained from your certificate authority.