Alibaba Cloud MQTT

Function Overview

Issue: 20190914

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. **You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.**
2. **No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.**
3. **The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.**
4. **This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use**

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified , reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates . The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click **OK**.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	It indicates that it is a required value, and only one item can be selected.	`swich {stand | slave}`

Contents

1 Send and receive messages

MQ for MQTT supports message sending and receiving in multiple languages. This topic provides links to sample code of using MQ for MQTT separately and with MQ in combination to send and receive messages in different languages.

Use MQ for MQTT separately to send and receive messages

Language	Sample code
Java	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-java-demo/src/main/java/com/aliyun/openservices/lmq/example/demo/MQ4IoTSendMessageToMQ4IoTUseSignatureMode.ja
C	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-c-demo/src/c/mqttDemo.c
Python	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-python-demo/MQTTSendMessage2MQTT.py
. NET	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-DoNet-demo/MQTTDemo.cs
JavaScript	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-js-demo/lmqdemo.html
iOS	https://github.com/AliwareMQ/lmq-demo/tree/master/lmq-ios-demo/MQTTChatDemo
PHP	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-php-demo/MQTTSendMessageToMQTT.php

Use MQ for MQTT to send messages and MQ to receive messages

Language	Sample code
Java	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-java-demo/src/main/java/com/aliyun/openservices/lmq/example/demo/MQ4IoTSendMessageToRocketMQ.java

Use MQ to send messages and MQ for MQTT to receive messages

Language	Sample code
Java	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-java-demo/src/main/java/com/aliyun/openservices/lmq/example/demo/RocketMQSendMessageToMQ4IoT.java

Use signatures for authentication on MQTT clients

Language	Sample code
Java	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-java-demo/src/main/java/com/aliyun/openservices/lmq/example/demo/MQ4IoTSendMessageToMQ4IoTUseSignatureMode.ja
C	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-c-demo/src/c/mqttDemo.c
Python	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-python-demo/MQTTSendMessage2MQTT.py
. NET	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-DoNet-demo/MQTTDemo.cs
JavaScript	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-js-demo/lmqdemo.html
iOS	https://github.com/AliwareMQ/lmq-demo/tree/master/lmq-ios-demo/MQTTChatDemo

Language	Sample code
PHP	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-php-demo/MQTTConnectUseSignatureMode.php

Use token for authentication on MQTT clients

Language	Sample code
Java	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-java-demo/src/main/java/com/aliyun/openservices/lmq/example/demo/MQ4IoTSendMessageToMQ4IoTUseTokenMode.java
PHP	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-php-demo/MQTTConnectUseTokenMode.php

Use SSL encryption on MQTT clients

Language	Sample code
Java	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-java-demo/src/main/java/com/aliyun/openservices/lmq/example/demo/MQ4IoTSendMessageToMQ4IoTUseSignatureMode.ja
C	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-c-demo/src/c/mqttDemo.c
Python	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-python-demo/MQTTSendMessage2MQTT.py
. NET	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-DoNet-demo/MQTTDemo.cs
JavaScript	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-js-demo/lmqdemo.html
iOS	https://github.com/AliwareMQ/lmq-demo/tree/master/lmq-ios-demo/MQTTChatDemo

Use MQTT clients to send ordered messages to MQ brokers

Language	Sample code
Java	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-java-demo/src/main/java/com/aliyun/openservices/lmq/example/demo/MQ4IoTSendMessageToMQ4IoTUseSignatureMode.ja
PHP	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-php-demo/MQTTSendOrderMessage.php

Query the number of online MQTT clients

Language	Sample code
Java	https://github.com/AliwareMQ/lmq-demo/blob/master/lmq-java-demo/src/main/java/com/aliyun/openservices/lmq/example/demo/QueryOnlineClientNumDemo.java

2 Advanced functions

2.1 Retrieve offline messages

To simplify the offline message retrieving mechanism, AliwareMQ for IoT automatically loads offline messages and delivers them to an MQTT client after the MQTT client successfully establishes a connection and passes permission verification.

You must note the following:

- After the MQTT client establishes a connection, it must pass the permission verification to automatically load offline messages. For example, if you use token verification for an MQTT client, you must upload the token and the MQTT client must pass the verification before it can receive offline messages.
- An offline message takes some time to generate, because the pushed message can be judged as an offline message after the MQTT client acknowledgement times out . Therefore, if the MQTT client experiences transient disconnection and reconnecti on, you may not be able to obtain the offline notification message immediately. The latency is generally 5 to 10 seconds.
- If you have too many offline messages, that is, more than 30 messages, AliwareMQ for IoT sends offline messages in batches (30 messages a batch every 5 seconds).

 Note:

The automatic loading mechanism allows you to replace the original active pulling method for retrieving offline messages, though you can still use the original method.

2.2 Retrieve the online status of the client

You can retrieve the current online status of the Message Queue for MQTT client (referred to here as the client) by using the synchronous query method or asynchronous online/offline notification method.

Message Queue for MQTT must be used with backend MQ services (such as MQ) and work with applications that are deployed on cloud servers (hereinafter referred to as service applications) to complete service processes.

The main scenarios for retrieving the online status of the client are as follows:

- The subsequent operation logic needs to be determined based on whether the client is online or not during the main service process.
- The online status of a specific client needs to be determined during the O&M process.
- Service applications need to trigger some predefined actions when a client goes online or offline.

Basic principle

The Message Queue for MQTT broker supports the following methods for retrieving the online status of the client:

- Call the synchronous query interface

  This method is relatively simple. You can call an HTTP/HTTPS API through an open endpoint to query the online status of a specific client. This method can only query the status of a single client.
- Asynchronous online/offline notification

  This method uses notification messages. When a client goes online or offline, the MQTT broker pushes an online or offline message to backend MQ. Service applications are generally deployed on Alibaba Cloud ECS instances. Service applications can subscribe to this message from backend MQ to retrieve the online /offline events of all clients.

  This method perceives the client status asynchronously and detects the online and offline events rather than the online status of the client. Therefore, cloud-based applications need to analyze the client status based on the timeline of a series of events.

The differences between both methods are as follows:

- The synchronous query method queries the real-time status of a client. Theoretica lly, it is more precise than the asynchronous notification method, but it only supports the status query of a single client at one time.
- Asynchronous online/offline notification messages are based on message decoupling, so status determination is more complex and more likely to be incorrect. However, this method can analyze the running status traces of multiple clients based on events.

Synchronous query method (under public beta)

The synchronous query method is currently under public beta and is only available in the Internet beta test region. It will be available in other regions in the future. The MQTT broker exposes the query method through HTTP/HTTPS.

- Query method

  ```
  https://<mqtt-xxx.mqtt.aliyuncs.com>/route/clientId/get
  ```

  Specifically:

  - *<mqtt-xxx.mqtt.aliyuncs.com>* must be replaced with the endpoint domain of the instance that you purchased.
  - Both HTTP and HTTPS are supported for protocols. Only the GET method is supported for calls.

- Request parameters

  Table 2-1: Request parameters lists the parameters included in a sent request.

  Table 2-1: Request parameters

Name	Type	Description
accessKey	String	The AccessKeyId of the account that is used in the current request.
resource	String	The ID of the client to be queried. Only the client IDs under the current account can be queried.
timestamp	Long	The parameter used to block invalid and expired requests. It must be set to the current actual time, that is, the UNIX timestamp of the current request, in milliseconds. The expiration time of the timestamp is 5 minutes before or after the current actual time.
instanceId	String	The ID of the current instance.

Name	Type	Description
signature	String	The signature of the request parameters, which is calculated based on the AccessKeySecret of the account, with other parameters used as strings to be signed. It is used for permission verification and the prevention of request tampering. For more information about the calculation method, see #unique_7.

- Responses

  After receiving a query request, the MQTT broker verifies the parameters and returns the number of online connections of the current client for a valid query request. Table 2-2: Status codes lists the possible responses.

  Table 2-2: Status codes

HTTP status code	Description
400	This code is returned if the request is invalid. The possible cause is that the URL is incorrect or a parameter is missing.
403	This code is returned if permission verification fails. The possible cause is that the signature calculation is incorrect or that the current account does not have the permission to query the status of the client in the request.

HTTP status code	Description
200	This code is returned if the request is successfully processed.

The response includes the HTTP status code and the HTTP content, which contains the corresponding results in JSON format. Table 2-3: Response parameters lists the response parameters.

Table 2-3: Response parameters

Name	Type	Description
code	Integer	The status code of the request processing result , which indicates whether the current request is successful and the cause in the case of failure.
success	Boolean	Indicates whether the query is successful. Valid values: - true: The query is successful. - false: The query fails.

Name	Type	Description
online	Integer	The online status of the client and the number of maintained TCP connections. Valid values: - 0: The client is offline. - A value greater than 0: The client is online. Note that a value greater than 1 may indicate repeated connections. Due to data synchronization latency on the MQTT broker, there is a small possibility that the value -1 may be returned, indicating that the query result is unknown. In that case, we recommend that you retry the request or make a conservative judgment based on the actual scenario.
message	String	The description of the request processing result, which is used to identify the cause of an exception.

Table 2-4: Error codes describes the error codes.

Table 2-4: Error codes

Code	Description
0	This code is returned if the request is successfully processed.
1	This code is returned if a parameter is missing. In this case, we recommend that you check whether parameter pairs are complete.

Code	Description
2	This code is returned if an error occurs during signature calculation. In this case, we recommend that you check the signature calculation process.
3	This code is returned if the permission verification fails. In this case, we recommend that you check whether the current account has created the group ID of the device.
4	This code is returned if the request timestamp expires. In this case, we recommend that you check whether the value of `timestamp` is within the last 5 minutes.

Asynchronous online/offline notification

As described in Basic principle, if the asynchronous online/offline notification method is used, online and offline events are mapped to backend MQ.

The following uses MQ as backend MQ to demonstrate this.

Procedure

1. Create a topic that corresponds to online and offline events.

   In the Message Queue for MQTT console, create a topic for the device with the target group ID. For information about how to create a topic, see Create a topic and group ID in #unique_8.

   For example, if the target client type corresponds to the group ID `GID_XXX`, then the client ID and topic for this client type are `GID_XXX @@@ YYYYY` and `GID_XXX_MQ TT`, respectively.

   Specifically:

   - `GID_XXX` indicates the group ID created in the Message Queue for MQTT console.
   - `YYYYY` indicates the device ID and is concatenated with the group ID to form the client ID in the format of <*GroupID*>@@@<*DeviceID*>.
   - `_MQTT` indicates the fixed suffix that is required in the name of the topic for this type of event notifications.

   For more information, see #unique_9.
2. Service applications subscribe to this type of notifications.

   Use the topic created in step 1 to receive the online and offline events of the target client. For information about how to receive messages from MQ, see Subscribe to messages.

   The event type is included in an MQ message tag, indicating whether it is an online or offline event. The data format is as follows:

   ```
   RocketMQ Tag : connect / disconnect / tcpclean
   ```

   Specifically:

   - A `connect` event indicates an online event of the client.
   - A `disconnect` event indicates that the client proactively disconnects from the MQTT broker. According to MQTT, the client sends a `disconnect` packet before proactively releasing the TCP connection. The MQTT broker triggers the disconnect message after receiving the `disconnect` packet. If a client SDK

does not send the `disconnect` packet, the MQTT broker cannot receive the disconnect message.

- A `tcpclean` event indicates that the current TCP connection is disconnected. If the current TCP connection is disconnected, a `tcpclean` event is triggered regardless of whether the client has explicitly sent a `disconnect` packet.

 Note:

A `tcpclean` message indicates that the network-layer connection of the client is disconnected. A `disconnect` message only indicates that the client proactively sends a disconnect packet. Some clients may fail to send a `disconnect` message due to unexpected exit. This is subject to the implementation on the clients. Therefore, determine whether a client is offline based on the `tcpclean` event.

Data content is of the JSON type and related keys are as follows:

- `clientId` indicates a specific device.
- `time` indicates the event time.
- `eventType` indicates the event type, which is used by clients to differentiate events.
- `channelId` uniquely identifies each TCP connection.
- `clientIp` indicates the Internet egress IP address used by a client.

Example:

```
clientId :  GID_XXX @@@ YYYYY
time : 1212121212
eventType : connect / disconnect / tcpclean
channelId : 2b9b128104  6046faafe5  e0b458e4XX  XX
clientIp :  192 . 168 . X . X : 133XX
```

To determine whether a client is online, check the last received message while considering the context of online/offline notification messages.

The determination rules are as follows:

- The sequence of online and offline events generated by clients with the same `client ID` is determined by time. That is, a more recent event has a greater timestamp.
- Clients with the same `clientId` may be transiently disconnected multiple times. Therefore, when an offline notification message is received, you need

to check whether the message is related to the current TCP connection based on the `channelId` field. In short, an offline notification message can only overwrite an offline notification message with the same `channelId`. If `channelIds` of offline notification messages are different, the offline notification messages cannot be overwritten even if the value of `time` is more recent.

2.3 P2P messaging model

Message Queue for MQTT supports the point-to-point (P2P) messaging model, in addition to the publish-subscribe messaging model that is supported by the standard MQTT protocol. This topic describes the concepts and principles of the P2P model as well as how to use Message Queue for MQTT to send and receive messages through P2P.

What is the P2P model?

P2P, as its name implies, is a one-to-one messaging model where only one message sender and one message receiver are involved. The publish-subscribe model is usually used in one-to-many or many-to-many message transmission scenarios where one or more message senders and many message receivers are involved.

In the P2P model, the sender knows the information about the expected receiver when sending a message, and expects that the message will be consumed only by a specific client. The sender directly specifies the receiver in a topic when sending a message. The receiver can obtain the message without subscribing to the topic in advance.

The P2P model not only saves the subscription registration cost for the receiver but, due to the independently optimized messaging link, also reduces the push latency.

Differences between the P2P model and the publish-subscribe model

The differences between the P2P model and the publish-subscribe model when used in Message Queue for MQTT are as follows:

- In the publish-subscribe model, the message sender needs to send the messages of the topic to which the receiver has subscribed. In the P2P model, the receiver does not need to subscribe to the topic, and the sender can send messages directly to the target client according to the standards.

- In the publish-subscribe model, the receiver needs to subscribe to the topic in advance to receive messages from the sender. In the P2P model, the receiver does not need to subscribe to the topic in advance. Therefore, the program logic at the receiver is simplified and the subscription cost is saved.

Send P2P messages

When using the MQTT SDK to send P2P messages, you must set the level-2 topic to "p2p" and the level-3 topic to the client ID of the target receiver.

Java example

```
String   p2pTopic  = topic +"/ p2p / GID_xxxx @@@ DEVICEID_0  01 ";
sampleClie  nt . publish ( p2pTopic , message );
```

When using the MQ SDK to send P2P messages, you only need to set the subtopic attribute to the preceding subtopic string because the parent topic and subtopic are set separately.

Java example

```
String   subTopic ="/ p2p / GID_xxxx @@@ DEVICEID_0  01 ";
msg . putUserPro  perties ( PropertyKe  yConst . MqttSecond  Topic ,
subTopic );
```

Table 2-5: Links to sample codes for sending P2P messages provides the links to multi-language sample codes for sending P2P messages.

Table 2-5: Links to sample codes for sending P2P messages

Language	Link
.NET	. NET sample code
C	C sample code
Java	Java sample code
JavaScript	JavaScript sample code
Python	Python sample code
PHP	PHP sample code

Receive P2P messages

The client does not need to subscribe to P2P messages. Instead, it can receive P2P messages after initialization.