阿里云
微消息队列MQTT

功能概述

文档版本：20190914

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 消息收发功能

微消息队列 MQTT 支持多语言的消息收发。本文提供单独使用微消息队列 MQTT 收发消息以及与消息队列 MQ 结合使用时需使用的代码示例的链接。

单独使用微消息队列 MQTT 收发消息

语言	链接
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

微消息队列 MQTT 发送 MQ 接收

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

MQ 发送微消息队列 MQTT 接收

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

微消息队列 MQTT 客户端使用签名模式

语言	链接
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/MQTTConnectUseSignatureMode.php

微消息队列 MQTT 客户端使用 Token 模式

语言	链接
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

微消息队列 MQTT 客户端使用 SSL 加密

语言	链接
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

微消息队列 MQTT 客户端发送顺序消息到 MQ

语言	链接
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

查询微消息队列 MQTT 客户端在线数量

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

2 高级功能

2.1 获取离线 MQTT 消息

为了简化离线消息获取机制，微消息队列 MQTT 系统在客户端成功建立连接并通过权限校验后，会自动加载离线消息并下发到客户端。

其中需要注意的是：

- 客户端建立连接后，需要通过权限校验才能自动加载离线消息。例如，若您使用的是 Token 验证的方式，则需要完成 Token 上传并通过校验后才会收到离线消息。
- 离线消息生成需要一定的时间，因为推送的消息需要等待客户端的 ack 超时才会被判成离线消息。所以，如果客户端闪断重连，不一定马上可以获取到刚刚的离线消息。延迟时间一般在 5 秒 ~ 10 秒左右。
- 如果您的离线消息过多，即大于 30 条，微消息队列 MQTT 系统会分批（5 秒一次，每次 30 条）下发离线消息。

说明：

对于部分老用户来说，有了自动加载机制，可不再使用原来的主动拉取的方式获取离线消息，但继续保留也无影响。

2.2 获取 MQTT 客户端在线状态

您可通过调用同步查询接口或使用异步上下线通知的方式获取微消息队列 MQTT 客户端（下文简称为客户端）当前在线情况。

微消息队列 MQTT 需结合后端存储消息队列产品，如消息队列 MQ 等使用，来和部署在云端服务器上的应用（下文简称为业务应用）完成业务流程。

获取客户端在线状态这个功能主要的应用场景如下：

- 主业务流程中需要根据客户端是否在线来决定后续运行逻辑；
- 运维过程需要判断特定客户端当前是否在线；
- 业务应用需要在客户端上线或者下线时触发一些预定义的动作。

基本原理

微消息队列 MQTT 服务器（下文简称为 MQTT 服务器）提供以下方式获取客户端在线状态：

- 调用同步查询接口

  该方式相对简单，即通过开放的接入点地址调用 HTTP/HTTPS 方式的 API 查询某个特定客户端的当前实时状态，适用于对单个客户端的状态判断。

- 异步上下线通知

  该方式使用消息通知，在客户端上线和下线事件触发时，MQTT 服务器会向后端存储消息队列推送一条上下线消息。业务应用一般部署在阿里云的服务器上，业务应用通过向后端存储消息队列订阅这条消息来获取所有客户端的上下线动作。

  该方式属于异步感知客户端的状态，且感知到的是上下线事件，而非在线状态，云端应用需要根据事件发生的时间序列分析出客户端的状态。

两种查询方式的区别如下：

- 同步查询接口是查询当前客户端的实时状态，理论上比异步通知的方式更精确，但只能查询单个客户端状态。
- 异步上下线通知因为采用消息解耦，状态判断更加复杂，且误判可能性更大，但该方法可以基于事件分析多个客户端的运行状态轨迹。

同步查询接口（公测期间）

同步查询接口功能目前处于公测期间，仅限公网测试地域（Region）使用，后续会在其他地域开放服务。MQTT 服务器对外通过 HTTP/HTTPS 协议方式暴露查询接口。

- 查询接口

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

  其中：

  - *<mqtt-xxx.mqtt.aliyuncs.com>* 需替换为您所购买的实例的接入点域名。
  - 协议可选 HTTP 或者 HTTPS，调用方式仅限 GET 方法。

- 请求参数

  发送请求的时候所需带上的参数如表 2-1: 请求参数列表 所述。

  表 2-1: 请求参数列表

参数名称	参数类型	说明
accessKey	String	当前请求使用的账号的 AccessKeyId。

参数名称	参数类型	说明
resource	String	需要查询的客户端 Client ID，只能查询当前账号拥有的 Client ID。
timestamp	Long	用于阻止非法过期请求，需要设置为当前真实时间，即该参数即构建当前请求的 UNIX 毫秒时间戳。时间戳过期时间为当前真实时间前后 5 分钟。
instanceId	String	当前使用的实例 ID。
signature	String	请求参数的签名，将其他参数作为待签名字符串，使用账户 AccessKeySecret 计算得到，用于权限验证和防止请求篡改。具体计算方式请参见#unique_7。

- 返回结果

  MQTT 服务器接收到查询请求后，会验证接口参数，对于合法的查询请求返回当前客户端的在线连接数。具体返回情况如表 2-2: 状态码列表 所述。

  表 2-2: 状态码列表

HTTP 状态码	说明
400	非法请求。原因可能是接口 URL 不正确或参数缺失。
403	权限验证失败。原因可能是签名计算错误，或者没有权限查询对应的 Client ID 状态。
200	请求处理成功。

  返回结果中，除 HTTP 状态码，HTTP Content 中也会包含对应的返回结果，返回结果为 JSON 格式，具体的字段映射如表 2-3: 返回参数列表 所述。

  表 2-3: 返回参数列表

字段名称	类型	说明
code	Integer	请求处理结果状态码，用于判断当前请求是否成功和失败原因。

字段名称	类型	说明
success	Boolean	查询是否成功，取值说明如下： - “true” 代表查询成功。 - “false” 代表查询失败。
online	Integer	当前客户端的在线状态以及维持的 TCP 连接数。取值说明如下： - “0” 代表客户端不在线。 - 大于 “0” 代表客户端在线。应用方应注意该字段大于 “1” 可能存在重复连接情况。因服务端数据同步延迟，会有很小概率出现返回 “-1” 的情况，代表查询结果未知，建议应用重试，或者基于实际场景做保守判断。
message	String	请求处理结果描述，用于判断异常原因。

其中错误码（Error Code）说明如表 2-4: 错误码列表 所述。

表 2-4: 错误码列表

code	说明
0	代表请求处理成功。
1	参数缺失。建议检查参数对是否完整。
2	签名计算错误。建议检查签名计算过程。
3	权限验证错误。建议检查当前账号是否创建过该设备所属的 Group ID。
4	请求时间戳过期。建议检查 `timestamp` 字段是否为最近 5 分钟内。

异步上下线通知

如上文基本原理所述，如果使用异步上下线通知的方式，上下线事件会映射到后端存储消息队列中。

下文以消息队列 MQ 作为后端存储消息队列的情况为例说明。

操作步骤

1. 创建上下线事件对应的 Topic。

   您需关注哪些 Group ID 分组的设备，就在微消息队列 MQTT 控制台创建对应的 Topic。创建 Topic 的步骤请参见#unique_8中的创建 Topic 和 Group ID 部分。

   例如您需要关注 Group ID 为 GID_XXX 类型的所有客户端，那么这类客户端对应的 Client ID 和 Topic 分别是 GID_XXX@@@YYYYY 和 GID_XXX_MQTT。

   其中：

   - GID_XXX 是在微消息队列 MQTT 控制台上创建的 Group ID。
   - YYYYY 是设备 ID，与 Group ID 以“<*GroupID*>@@@<*DeviceID*>” 模式构成 Client ID。
   - _MQTT 是该类事件通知消息的 Topic 命名所必需的固定后缀。

   更多信息请参见#unique_9。
2. 业务应用订阅该类通知消息。

   使用步骤 1 中创建的 Topic，即可收到关注的客户端的上下线事件。消息队列 MQ 的接收程序请参见订阅消息。

   事件类型放在消息队列 MQ 的 Tag 中，代表上线或下线。数据格式如下：

   ```
   MQ Tag: connect/disconnect/tcpclean
   ```

   其中：

   - connect 事件代表客户端上线动作。
   - disconnect 事件代表客户端主动断开连接。按照 MQTT 协议，客户端主动断开 TCP 连接之前应该发送 disconnect 报文，MQTT 服务器在收到 disconnect 报文后触发该类型消息。如果某些客户端 SDK 没有按照协议发送 disconnect 报文，MQTT 服务器相应无法收到该消息。
   - tcpclean 事件代表实际的 TCP 连接断开。无论客户端是否显示发送过 disconnect 报文，只要当前 TCP 连接断开就会触发 tcpclean 事件。

    说明：

tcpclean 消息代表客户端网络层连接的真实断开。对应的，disconnect 消息仅仅代表客户端是主动发送了下线报文。受限于客户端的实现，有时候客户端异常退出会导致 disconnect 消息并没有正常发送。因此判断客户端下线请使用 tcpclean 事件。

数据内容为 JSON 类型，相关的 Key 说明如下：

- clientId 代表具体设备；
- time 代表本次事件的时间；
- eventType 代表事件类型，供客户端区分事件类型；
- channelId 代表每个 TCP 连接的唯一标识；
- clientIp 代表客户端使用的公网出口 IP 地址。

示例：

```
clientId: GID_XXX@@@YYYYY
time:1212121212
eventType:connect/disconnect/tcpclean
channelId:2b9b1281046046faafe5e0b458e4XXXX
clientIp: 192.168.X.X:133XX
```

判断客户端当前是否在线不能仅仅根据收到的最后一条消息的状态，而需要结合上下线消息的前后关联来判断。

具体判断规则如下：

- 同一个 clientId 的客户端，产生上下线事件的先后顺序以时间为准，基本原则为时间戳越大则越新。
- 同一个 clientId 的客户端，可能存在多次闪断，因此，当收到下线消息时，一定要根据 channelId 字段判断是否是当前的 TCP 连接。简而言之，下线消息只能覆盖 channelId 相同的下线消息，如果下线消息的 channelId 不一样，尽管 time 较新，也不能覆盖。

2.3 P2P 消息收发模式（MQTT）

除了标准 MQTT 协议所支持的发布/订阅（Pub/Sub）消息收发模式外，微消息队列 MQTT 还支持点对点（Point to Point，简称 P2P）模式。本文介绍 P2P 模式的概念、原理以及如何使用微消息队列 MQTT 点对点收发消息。

什么是 P2P 模式

P2P，顾名思义，是一对一的消息收发模式，即只有一个消息发送者和一个消息接收者。而 Pub/Sub 模式通常用于一对多或多对多的消息群发场景，即拥有一个或多个消息发送者和多个消息接收者的场景。

在 P2P 模式中，发送者发送消息时已经明确该消息预期的接收者信息，并明确该消息只需要被特定的单个客户端消费。发送者发送消息时通过 Topic 信息直接指定接收者，接收者无需提前订阅即可获取该消息。

P2P 模式不仅可以为接收者节省注册订阅关系的成本，此外，由于收发消息的链路有单独的优化，还可以降低推送延迟。

P2P 模式和 Pub/Sub 模式的区别

在微消息队列 MQTT 中使用 P2P 模式收发消息与使用 Pub/Sub 的普通模式收发消息的区别如下所述：

- 发送消息时，Pub/Sub 模式下，发送者需要按照和接收者约定好的 Topic 发送消息；而 P2P 模式下，发送者无需事先约定传输消息的 Topic，发送者可以直接按照规范发送消息到目标的接收者。
- 接收消息时，Pub/Sub 模式下，接收者需要按照和发送者约定好的 Topic 提前订阅才能收到消息；而 P2P 模式下接收者无需事先订阅即可接收消息，从而简化接收者的程序逻辑，节省订阅成本。

发送 P2P 消息

使用 MQTT SDK 发送 P2P 消息时，需将二级 Topic 设为 “p2p”，将三级 Topic 设为目标接收者的 Client ID。

Java 示例

```
String p2pTopic =topic+"/p2p/GID_xxxx@@@DEVICEID_001";
sampleClient.publish(p2pTopic,message);
```

使用消息队列 MQ 的 SDK 发送 P2P 消息时，由于一级 Topic 和子级 Topic 是分开设置的，因此只需要将子级 Topic 属性设置成上述的子级 Topic 字符串。

Java 示例

```
String subTopic="/p2p/GID_xxxx@@@DEVICEID_001";
msg.putUserProperties(PropertyKeyConst.MqttSecondTopic, subTopic);
```

表 2-5: 发送 P2P 消息的示例代码链接 提供了发送 P2P 消息的多语言代码示例的链接。

表 2-5: 发送 P2P 消息的示例代码链接

语言	链接
.NET	.NET 示例代码
C	C 示例代码

语言	链接
Java	Java 示例代码
JavaScript	JavaScript 示例代码
Python	Python 示例代码
PHP	PHP 示例代码

接收 P2P 消息

接收消息的客户端无需任何订阅处理，只需要完成客户端的初始化即可收到 P2P 消息。