ALIBABA CLOUD

阿里云

微消息队列MQTT版

功能概述

文档版本：20210115

阿里云

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
危险	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。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会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，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	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。	单击**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。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	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。	`switch {active|stand}`

目录

1.消息收发功能

微消息队列MQTT版支持多语言的消息收发。本文提供单独使用微消息队列MQTT版收发消息以及与消息队列RocketMQ版结合使用的示例代码的链接。

示例代码使用说明

针对不同的使用场景，本文提供的示例代码所覆盖的语言并不代表该场景下仅支持该语言。示例代码可为您实际的消息收发提供指引与参考。例如，本文列举的第七种场景 RocketMQ 发送消息 MQTT 订阅消息示例，虽然仅有 Java 的示例代码，但实际支持的语言不仅只有 Java。您可以参照 Java 的示例代码完成其余语言的参数填写。各语言的 Demo 仍在不断补充，敬请期待。

微消息队列MQTT版所支持的客户端 SDK 请参见 SDK 下载。

示例代码列表

1. MQTT 消息收发示例 ❯
2. MQTT 签名示例 ❯
3. MQTT Token 示例 ❯
4. MQTT SSL 加密示例 ❯
5. WebSocket 支持 Web 应用消息收发示例 ❯
6. MQTT 发送消息 RocketMQ 订阅消息示例 ❯
7. RocketMQ 发送消息 MQTT 订阅消息示例 ❯
8. MQTT 发送顺序消息 RocketMQ 订阅顺序消息示例 ❯

> 说明 如果您使用签名鉴权模式收发消息，且需对签名进行验证，请参见使用控制台验证签名。

2.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版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。
MQTT 微消息队列MQTT

注意事项

- 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版系统会分批（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。

设置方法

您需要通过设置 QoS 和 cleanSession 两个参数来决定是否需要获取离线消息。详情请参见名词解释。

3.获取 MQTT 客户端在线状态

微消息队列MQTT版提供同步查询和异步上下线事件通知两种方式，来获取 MQTT 客户端在线状态。本文介绍这两种方式的基本原理、应用场景、具体差异以及实现方式。

基本原理

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

- 同步查询
  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 OpenAPI 查询某个特定客户端的当前实时状态，适用于对单个或多个客户端的状态判断。
- 异步上下线事件通知
  该方式使用消息通知，在客户端上线和下线事件触发时，MQTT 服务端会通过上下线通知规则向其他阿里云产品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，业务应用通过向其他阿里云产品订阅这条消息来获取所有客户端的上下线动作。

  > **说明** 当前支持的其他阿里云产品仅包含消息队列RocketMQ版。

  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。

应用场景

两种获取 MQTT 客户端在线状态的方式分别应用于以下场景：

- 同步查询
  - 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。
- 异步事件通知
  - 服务端需要在客户端上线或者下线时触发一些预定义的动作。

同步查询与异步事件通知的差异

两种查询方式的区别如下：

- 同步查询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。
- 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。异步通知虽然存在一定复杂度和误判，但更加适合大规模的客户端的状态统计。

实现方式

- 同步查询接口
  - QuerySessionByClientId
  - BatchQuerySessionByClientIds
- 异步事件通知规则
  - MQTT 客户端上下线事件数据流出
  - 上下线通知规则管理

4.跨云产品的数据互通

4.1. 跨云产品数据流入

您可通过配置微消息队列MQTT版的数据流入规则来自定义数据从其他阿里云产品流入到微消息队列MQTT版。本文介绍跨产品数据流入的原理、应用场景以及微消息队列MQTT版与其他阿里云产品的资源映射关系。

基本原理

如需将其他阿里云产品的数据导入微消息队列MQTT版，您需创建数据流入规则。该规则用于从您配置的阿里云产品中读取数据并将数据通过 MQTT 协议推送到 MQTT 客户端，从而实现直接调用阿里云产品的 API 发送数据到 MQTT 客户端。

> **注意** 当前，仅支持将消息队列RocketMQ版的数据导入微消息队列MQTT版。

应用场景

指令下发场景

部署在云端的后台管控服务发送指令到消息队列RocketMQ版，微消息队列MQTT版产品根据配置的数据流入规则，将消息队列RocketMQ版的 Topic 映射到 MQTT 的 Topic，然后将对应的数据推送到目标的 MQTT 设备端。

资源映射方式

数据流入规则的映射粒度为微消息队列MQTT版的父级 Topic，即支持将其他阿里云产品的某个资源的数据导入到微消息队列MQTT版的父级 Topic 中。针对 MQTT 协议推送的子级 Topic，通过其他拓展属性的方式配置。

映射关系

MQTT 资源	其他阿里云产品	其他阿里云产品资源	数据包定义
MQTT Topic	消息队列RocketMQ版	消息队列RocketMQ版的 Topic	MQTT 与 RocketMQ 的消息结构映射

更多信息

如需了解控制台上的操作，请参见数据流入规则管理。

4.2. 跨云产品的数据流出

您可通过配置微消息队列MQTT版的数据流出规则来自定义数据从微消息队列MQTT版流出至其他阿里云产品。本文介绍数据流出的原理、应用场景以及微消息队列MQTT版与其他阿里云产品的资源映射关系。

基本原理

如需将微消息队列MQTT版的数据导出至其他阿里云产品，您需创建数据流出规则。该规则用于将 MQTT 客户端发送的消息导出到您配置的其他阿里云产品中，从而实现直接调用云产品的 API 读取 MQTT 客户端发送的消息。

> 注意 当前，仅支持将微消息队列MQTT版的数据导出至消息队列RocketMQ版。

应用场景

设备数据上报

海量设备通过 MQTT 协议上报状态数据到微消息队列MQTT版，配置数据流出规则将微消息队列MQTT版的 Topic 映射到消息队列RocketMQ版的 Topic 后，可以直接在后台服务启动消息队列RocketMQ版的消费者消费处理上报数据。

规则映射方式

数据流出规则的映射粒度为微消息队列MQTT版父级 Topic，即支持将微消息队列MQTT版的父级 Topic 下所有的消息导出到其他阿里云产品的某个资源中。针对 MQTT 协议的子级 Topic，通过其他拓展属性的方式配置。

映射关系

MQTT 资源	其他阿里云产品	其他阿里云产品资源	数据包定义
MQTT Topic	消息队列RocketMQ版	消息队列RocketMQ版的 Topic	MQTT 与 RocketMQ 的消息结构映射

更多信息

如需了解控制台上的操作，请参见数据流出规则管理。

4.3. MQTT 客户端上下线事件数据流出

您可通过配置微消息队列MQTT版的客户端上下线通知规则，将获取的 MQTT 客户端上下线事件数据导出至其他阿里云产品。该方法为异步上下线通知。本文介绍客户端上下线通知的原理、应用场景以及微消息队列MQTT版与其他阿里云产品的资源映射关系。

基本原理

在客户端上线和下线事件触发时，MQTT 服务器会根据您配置的客户端上下线通知规则，向后端其他云产品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云产品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。

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。

> 注意 当前支持的其他后端云产品仅有消息队列RocketMQ版。

应用场景

客户端上下线通知主要的应用场景为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。

例如，客户端在线状态聚合。此场景中，MQTT 客户端产生上下线等状态变更，微消息队列MQTT版会根据您配置的客户端状态通知规则，将状态变更封装后转发到

消息队列RocketMQ版
消息的方式来实现客户端状态数据的聚合和统计。

> **说明** 针对其他场景，推荐您使用同步查询接口来获取客户端在线状态。详情请参见获取 MQTT 客户端在线状态。

资源映射方式

同一个微消息队列MQTT版 Group ID 下的所有客户端的状态变更通知都会转发到您配置的同一个其他阿里云产品的资源里。

映射关系

MQTT 资源	其他阿里云产品	其他阿里云产品资源	数据包定义
MQTT Group ID	消息队列RocketMQ版	消息队列RocketMQ版的 Topic	MQTT 与 RocketMQ 的消息结构映射

操作流程

如上文所述，如果使用异步上下线通知的方式，您需创建客户端上下线事件通知规则，将上下线通知的消息导出至后端云产品中。下文以使用

消息队列RocketMQ版
后端云产品为例进行说明。

1. 创建上下线通知规则。
   您需关注哪些 Group ID 分组的设备，就在微消息队列MQTT版控制台创建规则时，选定相应的 Group ID。
   创建规则的步骤请参见创建上下线通知规则。
2. 业务应用订阅该类通知消息。
   通过步骤 1 中创建的规则，即可收到关注的客户端的上下线事件。
   消息队列RocketMQ版
   的接收程序请参见订阅消息。示例代码详情请参见 MQTTClientStatusNoticeProcessDemo.java。
   事件类型放在
   消息队列RocketMQ版
   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。

示例如下：

```
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。一个 channelId 代表一个 TCP 连接，只会存在一个 connect 事件和一个 close 事件。

异步上下线通知 Java 示例代码 ▼

更多信息

如需了解控制台上的操作，请参见上下线通知规则管理。

5.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版还支持点对点（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版点对点收发消息。
MQTT 微消息队列MQTT

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版中使用 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RocketMQ版
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);
```

发送 P2P 消息的多语言代码示例的链接如下表所示。

语言	链接
.NET	.NET 示例代码
C	C 示例代码
Java	Java 示例代码

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

> **说明** 不支持使用 Go SDK 收发 P2P 消息。

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。