ALIBABA CLOUD

阿里云

微消息队列MQTT版
功能概述

文档版本：20220308

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 05
2.获取离线MQTT消息 08
3.获取MQTT客户端在线状态 09
4.跨云产品的数据互通 12
4.1. MQTT与RocketMQ的消息结构映射 12
4.2. 跨云产品数据流入 14
4.3. 跨云产品的数据流出 15
4.4. MQTT客户端上下线事件数据流出 16
5.P2P消息收发模式（MQTT） 19

1.消息收发功能

消息收发为微消息队列MQTT版最基本的功能，微消息队列MQTT版支持终端和云端服务接入微消息队列MQTT版服务器，实现终端和终端、终端和云端的消息通信。本文介绍消息收发功能的基本场景和示例代码。

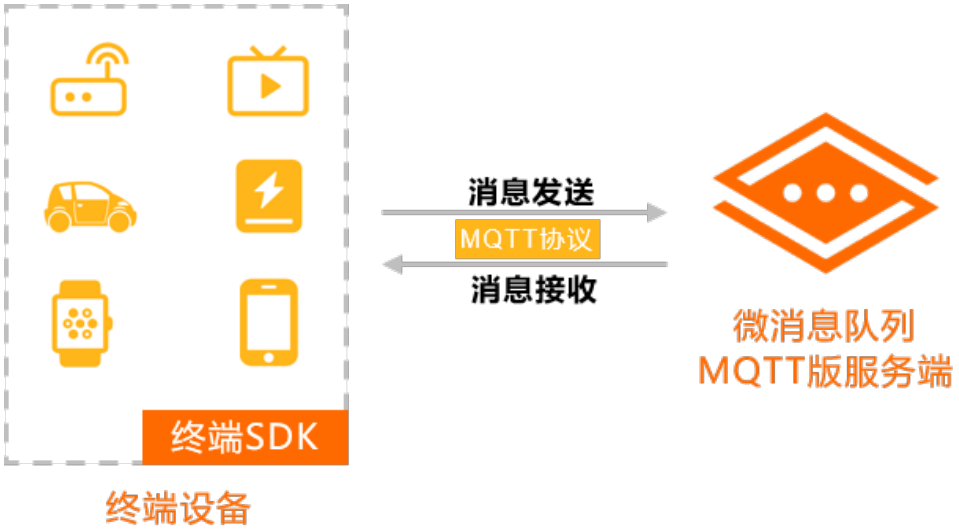
背景信息

微消息队列MQTT版消息收发场景中，消息发送端和消息接收端可分为终端设备和云端服务两种类型。终端设备和云端服务均需要通过微消息队列MQTT版的SDK接入微消息队列MQTT版服务端进行消息通信。

微消息队列MQTT版提供的SDK如下。

- 终端SDK：MQTT的开源SDK，支持MQTT 3.1.1协议，适用于终端设备的开发。
- 云端SDK：阿里云开发的SDK，适用于云上业务应用的开发。支持消息发送、消息订阅以及接收终端SDK上下线事件消息。

终端和终端消息收发

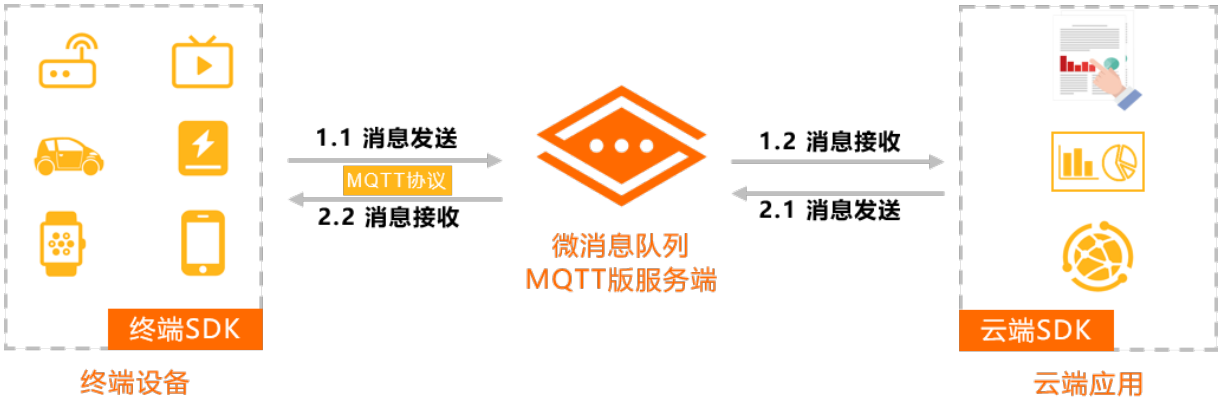


该场景下消息的发送端和消费端都分布在移动终端环境，通过MQTT协议连接到微消息队列MQTT版。发送端和消费端的终端设备均通过开源的终端SDK接入微消息队列MQTT版实现消息收发。

- **即时通信**：例如，两个安装有聊天App的移动手机直接通过微消息队列MQTT版服务端收发聊天信息。
- **智能设备管理**：例如，通过安装在手机上的App向接入到微消息队列MQTT版服务端的共享充电宝下发指令，充电宝收到指令消息后自动弹出。

典型场景示例

终端和云端消息收发



该场景下消息的发送端和消费端分别为移动终端设备和部署在阿里云上的业务应用。通过微消息队列MQTT版实现终端和云端的消息交互。终端设备通过终端SDK接入微消息队列MQTT版服务端，云端应用通过云端SDK接入微消息队列MQTT版服务端。

- **设备状态上报**：消息发送端为终端设备，消费端为云端业务应用。例如，部署在终端环境的海量电子价签定时上报自己的显示状态和节点电量等，部署在云端的管控应用根据上报的数据分析当前在线的电子价签状态，并根据业务需要进行进一步的调整。
- **系统消息推送**：消息发送端为云端业务应用，消费端为终端设备。例如，部署在云端的某游戏应用发送一条停服更新的通告，微消息队列MQTT版服务端将该通告推送至所有下载该游戏的手机终端上，通过手机消息提示给所有游戏用户。
- **消息接收**：云端SDK的订阅模式支持集群消费，即不同的消费端获取不同的消息。

典型场景示例

终端SDK示例代码

> **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，您可以参照Java的示例代码完成其余语言的参数填写。

终端SDK消息收发示例 ❯

WebSocket支持Web应用消息收发示例 ❯

终端设备接入微消息队列MQTT版服务端收发消息时，可以通过以下任意一种鉴权方式对客户端进行鉴权认证。更多鉴权信息，请参见鉴权概述。

> **说明** 如果您使用签名鉴权模式收发消息，User Name和Password的计算方法及验证方法请参见签名鉴权模式。

MQTT签名示例 ❯

MQTT Token示例 ❯

MQTT SSL加密示例 ❯

消息收发

客户端鉴权

云端SDK示例代码

> **说明** 云端SDK暂时只支持Java和Go语言。

云端SDK消息收发示例 ❯

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。

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

您可以在消费端通过设置QoS和cleanSession两个参数来决定客户端是否需要获取离线消息。

QoS和cleanSession的组合关系

QoS级别	cleanSession=true	cleanSession=false
QoS0	无离线消息，在线消息只尝试推一次。	无离线消息，在线消息只尝试推一次。
QoS1	无离线消息，在线消息保证可达。	有离线消息，所有消息保证可达。
QoS2	无离线消息，在线消息保证只推一次。	暂不支持。

QoS和cleanSession参数的详细说明，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服务端支持将上下线消息直接推送到部署在阿里云服务器上的云端应用。
  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。
- **异步上下线事件通知**
  - 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。
  - 服务端需要对客户端的上下线数据进行统计分析，并根据客户端的在线状态推送消息。

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**
  同步查询方式可以通过调用以下API接口获取客户端状态：
  - QuerySessionByClientId
  - BatchQuerySessionByClientIds
- **异步上下线事件通知**
  异步上下线事件通知方式下，云端服务可通过调用微消息队列MQTT版提供的云端SDK获取客户端上下线消息。SDK下载，请参见版本说明。
  获取客户端上下线事件的示例代码如下：

```
package com.aliyun.openservices.lmq.example;
import com.alibaba.fastjson.JSONObject;
import com.alibaba.mqtt.server.ServerConsumer;
import com.alibaba.mqtt.server.callback.StatusListener;
import com.alibaba.mqtt.server.config.ChannelConfig;
import com.alibaba.mqtt.server.config.ConsumerConfig;
import com.alibaba.mqtt.server.model.StatusNotice;
public class MQTTClientStatusNoticeProcessDemo {
    public static void main(String[] args) throws Exception {
         /**
         * 您创建的微消息队列MQTT版的实例接入点。
         * 云端SDK使用的接入点需要提交工单提交工单获取。
         * 接入点地址必须填写分配的域名，不得使用IP地址直接连接，否则可能会导致服务端异常。
         */
        String domain = "domain";
         /**
         * 使用的协议和端口必须匹配，该参数值固定为5672。
         */
        int port = 5672;
         /**
         * 您创建的微消息队列MQTT版的实例ID。
         */
        String instanceId = "instanceId";
         /**
         * AccessKey ID，阿里云身份验证，在阿里云RAM控制台创建。获取方式，请参见获取AccessKey。
         */
        String accessKey = "accessKey";
         /**
         * AccessKey Secret，阿里云身份验证，在阿里云RAM控制台创建。仅在签名鉴权模式下需要设置。获取
方式，请参见获取AccessKey。
         */
        String secretKey = "secretKey";
         /**
         * 您在微消息队列MQTT版控制台创建的Group的ID。
         * */
        String mqttGroupId = "mqttGroupId";
        ChannelConfig channelConfig = new ChannelConfig();
        channelConfig.setDomain(domain);
        channelConfig.setPort(port);
        channelConfig.setInstanceId(instanceId);
        channelConfig.setAccessKey(accessKey);
        channelConfig.setSecretKey(secretKey);
        ServerConsumer serverConsumer = new ServerConsumer(channelConfig, new ConsumerConfi
g());
        serverConsumer.start();
        serverConsumer.subscribeStatus(mqttGroupId, new StatusListener() {
            @Override
            public void process(StatusNotice statusNotice) {
                System.out.println(JSONObject.toJSONString(statusNotice));
            }
        });
    }
}
```

更多信息

MQTT客户端上下线事件数据流出

4.跨云产品的数据互通

4.1. MQTT与RocketMQ的消息结构映射

本文针对使用
消息队列RocketMQ版
SDK与微消息队列MQTT版交互的场景，提供交互中所涉及的消息结构和属性字段的映射关系，方便您更好的理解和组合使用这两个产品。
微消息队列MQTT版是一款面向移动端的网关产品，可通过数据互通规则实现与其他阿里云产品，例如

消息队列RocketMQ版

的交互。规则的详细信息，请参见规则管理。

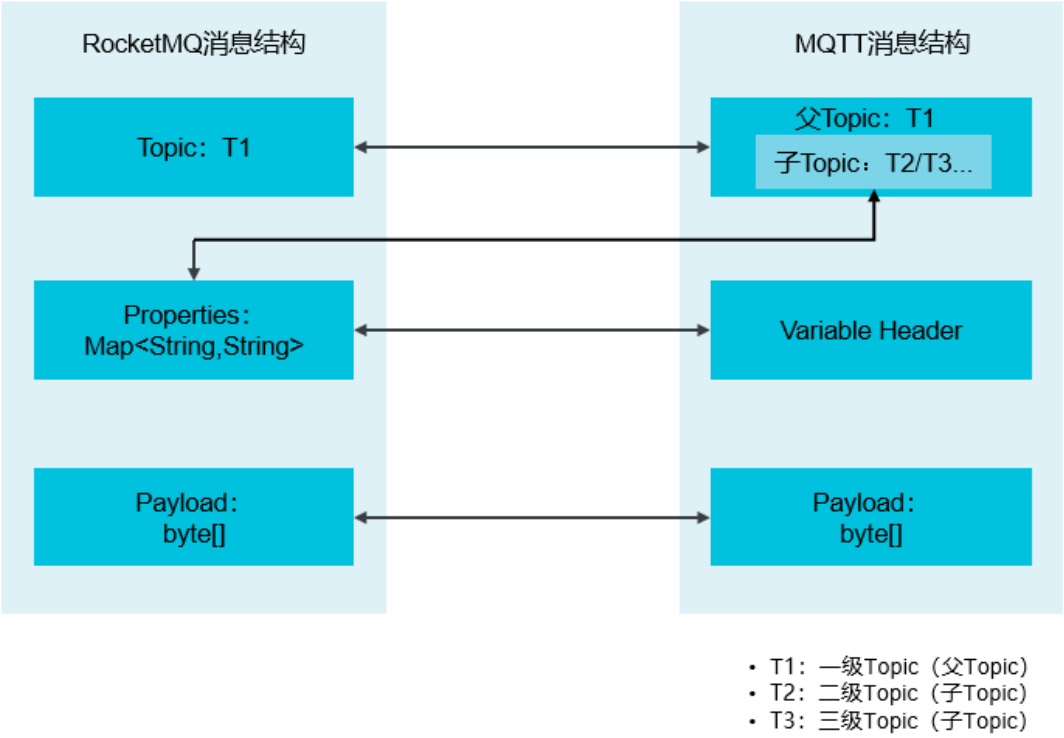
如果单独使用微消息队列MQTT版，则无需关注本文提供的映射关系，一切遵循标准MQTT协议规范即可。

微消息队列MQTT版的详细介绍，请参见什么是微消息队列MQTT版和名词解释。

消息结构映射

微消息队列MQTT版和

消息队列RocketMQ版
都是基于发布/订阅（Pub/Sub）模型的消息系统，两者概念上存在很多相似之处，下图列举了关键概念的区别和映射关系。



如上图所示，在微消息队列MQTT版中Topic是多级结构，而

消息队列RocketMQ版
的Topic仅有一级，因此，微消息队列MQTT版中的一级Topic映射到
消息队列RocketMQ版
的Topic，而子级Topic，包括二级、三级、n级Topic则映射到
消息队列RocketMQ版

的消息属性（Properties）中。

消息队列RocketMQ版
协议中的消息（Message）可以拥有自定义属性（Properties），而MQTT协议目前的版本不支持属性，但为了方便溯源MQTT协议中的Header信息和设备信息，微消息队列MQTT版的部分信息将被映射到
消息队列RocketMQ版
的消息属性中，方便使用
消息队列RocketMQ版
的SDK接入的用户获取。

> **说明** 在
> 消息队列RocketMQ版
> 中设置属性字段来映射微消息队列MQTT版的具体步骤，请参见下文属性字段映射的表格。

消息队列RocketMQ版
和微消息队列MQTT版的消息负载（Payload）均是您的业务消息的数据序列化结果，
消息队列RocketMQ版
和微消息队列MQTT版不会对业务消息再做进一步的编解码处理。

属性字段映射

目前，微消息队列MQTT版和

消息队列RocketMQ版
支持的属性字段映射关系如下表所示。使用
消息队列RocketMQ版
和微消息队列MQTT版的SDK的应用交互时，可以通过读写这些属性字段来设置或获取信息。
QoS、cleanSession、Topic以及Client ID的详细解释，请参见名词解释。

MQTT参数	RocketMQ属性Key	属性可选值	说明
QoS	qoslevel	0、1、2	消息队列RocketMQ版发给微消息队列MQTT版消息时可以设置，如果不设置，默认为“1”；微消息队列MQTT版发给消息队列RocketMQ版的消息可以直接读取。
cleanSession	cleansessionflag	true、false	消息队列RocketMQ版发给微消息队列MQTT版客户端P2P消息时设置，如不设置，默认为“true”；其他消息不可以设置，微消息队列MQTT版发给消息队列RocketMQ版的消息可以直接读取。
MQTT子级Topic	mqttSecondTopic	具体的子级Topic字符串	消息队列RocketMQ版发给微消息队列MQTT版客户端消息时如果需要子级Topic来做过滤，则设置，如不设置，默认为空；微消息队列MQTT版发给消息队列RocketMQ版的消息可以直接读取。

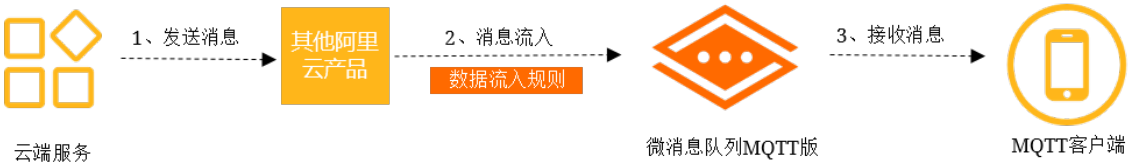
MQTT参数	RocketMQ属性Key	属性可选值	说明
客户端接收消息显示Topic	mqttRealTopic	业务上希望客户端收到消息时显示的子级字符串	消息队列RocketMQ版发给微消息队列MQTT版客户端消息时如果希望客户端收到消息后显示成指定的子级Topic名称，则可以设置；一般用于P2P消息，若不设置，P2P消息默认使用自己固定的Topic；微消息队列MQTT版发给消息队列RocketMQ版的消息时无该属性。
clientId	clientId	具体的Client ID字符串	不可设置，微消息队列MQTT版发给消息队列RocketMQ版消息时，用于追踪该消息的发送源的Client ID。

4.2. 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、使用限制以及微消息队列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。



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。

注意事项

微消息队列MQTT版通过配置的流入规则和

消息队列RocketMQ版
进行跨云产品的消息交互时，若
消息队列RocketMQ版
的服务端进行升级，则在升级期间，可能会造成微消息队列MQTT版客户端消费消息延迟。产品升级动态，请

参见公告列表。

使用限制

- 当前仅支持将
  消息队列RocketMQ版
  的数据导入微消息队列MQTT版。
- 轻量版实例不支持数据流入规则。

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

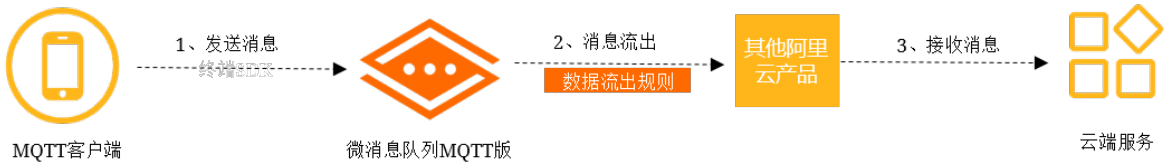
如需了解控制台上的操作，请参见数据流入规则管理。

4.3. 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、使用限制以及微消息队列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。



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。

使用限制

- 当前仅支持将微消息队列MQTT版的数据流出至
  消息队列RocketMQ版

。

- 轻量版实例不支持数据流出规则。

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

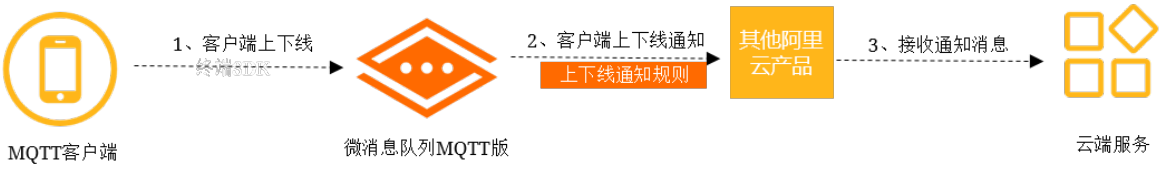
如需了解控制台上的操作，请参见数据流出规则管理。

4.4. 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、使用限制以及微消息队列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。

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。详细信息，请参见获取MQTT客户端在线状态。

使用限制

- 当前仅支持将

消息队列RocketMQ版
的数据导入微消息队列MQTT版。

- 轻量版实例不支持客户端上下线通知规则。

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和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。
   创建规则的详细步骤，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。示例代码详细信息，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.XX.XX: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。

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
JavaScript	JavaScript示例代码

语言	链接
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。