ALIBABA CLOUD

阿里云

微消息队列MQTT版

微消息队列MQTT公共云合集

文档版本：20220422

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.最佳实践

1.1. 快速使用MQTT的Java SDK收发消息（跨产品数据流入）

本文介绍如何将其他阿里云产品的数据导入微消息队列MQTT版。本文以当前仅支持的

消息队列RocketMQ版

数据互通为例进行说明。

前提条件

- 安装IDE。您可以使用IntellIJ IDEA或者Eclipse，本文以IntellIJ IDEA为例。
- 下载安装JDK。

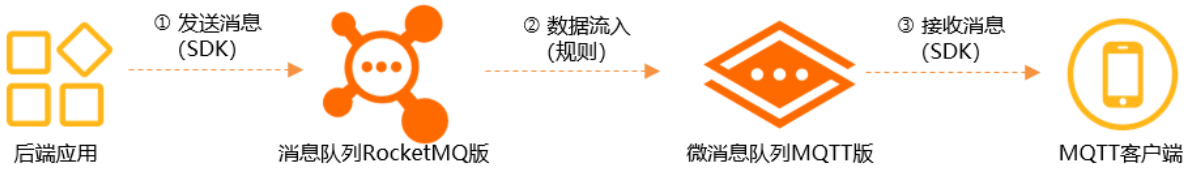
背景信息

本文以公网环境中的Java SDK为例说明如何将

消息队列RocketMQ版

数据流入至微消息队列MQTT版。

此场景下可使用多语言的第三方开源SDK来实现消息收发。更多信息，请参见SDK下载。



如上图所示，您部署在公网地域的后端应用数据需要传送至公网环境中的MQTT客户端，后端应用和MQTT客户端均通过Java语言开发。数据从

消息队列RocketMQ版

流入至微消息队列MQTT版是通过创建相应的数据流入规则实现。两个产品的服务端通过各自产品提供的Java SDK分别与各自的客户端实现消息收发。

> **注意**
>
> 消息队列RocketMQ版
>
> 和微消息队列MQTT版的Topic不能跨地域使用，因此，本文中所有资源都应在公网地域创建。详细信息，请参见Topic地域化。

网络访问

微消息队列MQTT版同时提供了**公网接入点**和**VPC 接入点**。接入点说明如下：

- 在物联网和移动互联网的场景中，客户端推荐使用**公网接入点**接入。
- **VPC 接入点**仅供一些特殊场景使用。因为一般而言，涉及部署在云端服务器上的应用的场景，建议使用服务端消息产品例如

消息队列RocketMQ版

实现。

> **注意** 客户端使用接入点连接服务时务必使用域名接入，不得直接使用域名背后的IP地址直接连接，因为IP地址随时会变化。在以下使用情况中出现的问题微消息队列MQTT版产品方概不负责：
>
> - 客户端不使用域名接入而是使用IP地址接入，产品方更新了域名解析导致原有IP地址失效。
> - 客户端网络对IP地址设置网络防火墙策略，产品方更新了域名解析后新IP地址被您的防火墙策略拦截。

本文以公网接入点为例。微消息队列MQTT版与

消息队列RocketMQ版

的应用场景对比和消息属性映射关系请参见以下文档：

- MQTT与Rocket的应用场景对比
- MQTT与Rocket的消息结构映射

使用流程

后端应用消息发送至MQTT客户端的使用流程如下所示。

步骤一：创建MQTT实例并获取接入点

> **注意** 轻量版实例不支持数据流入规则、数据流出规则及上下线通知规则，如需使用规则功能，请您在创建实例时选择其他类型的实例。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在**实例列表**页面左上角单击**创建实例**。
5. 在弹出的付费方式面板中，**付费方式**固定为**包年包月**，您无需设置，直接在面板左下角单击**确定**。
6. 在弹出的实例规格面板中，按需选择您需要购买的实例规格，选中**微消息队列 MQTT 版（包年包月）服务协议**，然后单击**立即购买**。

7. 在订单支付面板，根据提示完成支付。
8. 在支付成功页面单击**返回控制台**。
9. 回到微消息队列MQTT版控制台，在左侧导航栏单击**实例列表**，并将地域切换为您所购买的实例所对应的地域。
10. 在**实例列表**页面中，单击您所购买实例的名称或在其**操作**列单击**详情**，进入**实例详情**页面。
11. 在**实例详情**页面单击**接入点**页签，即可看到实例的接入点信息，本示例以**公网接入点**为例。

步骤二：创建父级Topic

MQTT协议支持多级Topic，父级Topic需在控制台创建，子级Topic无需创建，使用时直接在代码中设置即可。命名格式为：父级Topic和各子级Topic间均使用正斜线（/）隔开，*<父级Topic名称>/<二级Topic名称>/<三级Topic名称>*，例如，SendMessage/demo/producer。父级Topic和子级Topic总长度不能超过64个字符。Topic详细信息，请参见名词解释。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。

4. 在实例列表中找到目标实例，在其**操作**列中，选择**更多 > Topic 管理**。
5. 在**Topic 管理**页面左上角，单击**创建 Topic**。
6. 在创建Topic面板中，输入要创建的Topic**名称**和**描述**，然后在左下角单击**确定**。
   您可以在**Topic 管理**页面查看刚创建的Topic。

步骤三：创建Group ID

Group ID详细信息，请参见名词解释。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在实例列表中找到目标实例，在其**操作**列中，选择**更多 > Group 管理**。
5. 在**Group 管理**页面的左上角，单击**创建 Group**。
6. 在创建Group面板中，输入**Group ID**，然后在左下角单击**确定**。
   您可以在**Group 管理**页面查看刚创建的Group。

步骤四：创建数据流入规则

规则中填写的参数需与您创建的资源保持一致。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在实例列表中找到目标实例，在其**操作**列，选择**更多 > 规则管理**。
5. 在**规则管理**页面左上角，单击**创建规则**。
6. 在**创建规则**页面完成以下操作。

i. 在**配置基本信息**配置向导页面，填写规则的基本信息，然后单击**下一步**。

参数	取值示例	说明
规则ID	111111	规则的全局唯一标识，说明如下： ■ 只能包含字母、数字、短划线（-）和下划线（_），至少包含一个字母或数字。 ■ 名称长度限制在3~64字符之间，长于64字符将被自动截取。 ■ 创建后无法更新。
描述	migrate from rocketmq	对规则的描述。
状态	启用	是否启用当前规则，取值说明如下： ■ 启用 ■ 停用
规则类型	数据流入	创建的规则类型，取值说明如下： ■ 数据流出：用于将微消息队列MQTT版的数据导出至其他阿里云产品。详细信息，请参见跨云产品的数据流出。 ■ 数据流入：用于将其他阿里云产品的数据导入至微消息队列MQTT版。详细信息，请参见跨云产品数据流入。 ■ 上下线通知：用于将获取的微消息队列MQTT版客户端上下线事件数据导出至其他阿里云产品。详细信息，请参见MQTT客户端上下线事件数据流出。

ii. 在**配置规则源**配置向导页面，配置数据源，然后单击**下一步**。

参数	取值示例	说明
源服务类型	消息队列 RocketMQ 版	指定您需将哪个源云产品的数据流入至微消息队列MQTT版。 **说明** 当前仅支持消息队列RocketMQ版。
RocketMQ 实例	MQ_INST_13801563067*****_ByOD2jQ	指定源云产品的实例ID，即消息队列RocketMQ版的实例ID。 **说明** 仅支持选择和微消息队列MQTT版实例为同一地域的云产品实例。
Topic	TopicA	指定源云产品的资源键值，即消息队列RocketMQ版的Topic。TopicA的消息将流转至目标微消息队列MQTT版的Topic。

iii. 在**配置规则目标**配置向导页面，配置数据的流转目标，然后单击**创建**。

参数	取值示例	说明
Topic	TopicB	指定您需要将其他源云产品的数据导入至微消息队列MQTT版的哪个目标Topic。

您可以在**规则管理**的规则列表查看到刚创建的数据流入规则。

步骤五：调用Java SDK收发消息

1. 下载第三方的开源Java SDK。下载地址为Eclipse Paho Java Client。
2. 下载阿里云微消息队列MQTT版的Java SDK的Demo示例作为您代码开发的参考。下载地址为mqtt-java-demo。
3. 解压该Demo工程包至您指定的文件夹。
4. 在IntellIj IDEA中，导入解压后的文件以创建相应的工程，并确认*pom.xml*中已包含以下依赖。

```
<dependencies>
        <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
            <version>1.10</version>
        </dependency>
        <dependency>
            <groupId>org.eclipse.paho</groupId>
            <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
            <version>1.2.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>4.5.2</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.48</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun.openservices</groupId>
            <artifactId>ons-client</artifactId>
            <version>1.8.5.Final</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-onsmqtt</artifactId>
            <version>1.0.3</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-core</artifactId>
            <version>4.5.0</version>
        </dependency>
</dependencies>
```

> **说明** ons-client的版本信息，请参见版本说明。

5. 在RocketMQSendMessageToMQ4IoT.java类中，按代码注释说明填写相应参数，主要涉及步骤一至步骤三所创建MQTT资源以及您在

   消息队列RocketMQ版

   创建的相应资源，然后执行Main函数运行代码实现消息收发。

   示例代码如下。

```
package com.aliyun.openservices.lmq.example.demo;
import com.aliyun.openservices.lmq.example.util.ConnectionOptionWrapper;
import com.aliyun.openservices.ons.api.Message;
import com.aliyun.openservices.ons.api.ONSFactory;
import com.aliyun.openservices.ons.api.Producer;
import com.aliyun.openservices.ons.api.PropertyKeyConst;
```

```
import com.aliyun.openservices.ons.api.PropertyKeyConst;
import com.aliyun.openservices.ons.api.SendResult;
import java.util.Properties;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallbackExtended;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;
/**
 * 阿里云生产环境中，
消息队列RocketMQ版除了公网地域，其他地域的实例不允许在本地使用，必须在对应地域的ECS机器上部署使用
。
 */
public class RocketMQSendMessageToMQ4IoT {
    public static void main(String[] args) throws Exception {
        /**
         * 初始化
消息队列RocketMQ版发送客户端，实际业务中一般部署在服务端应用中。
         */
        Properties properties = new Properties();
        /**
         * 设置
消息队列RocketMQ版Group ID，在
消息队列RocketMQ版控制台创建。
         */
        properties.setProperty(PropertyKeyConst.GROUP_ID, "GID-XXXXX");
        /**
         * AccessKey ID，阿里云身份验证，在阿里云RAM控制台创建。
         */
        properties.put(PropertyKeyConst.AccessKey, "XXXX");
        /**
         * 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
。
         */
        properties.put(PropertyKeyConst.SecretKey, "XXXX");
        /**
         * 设置TCP接入点，该接入点为
消息队列RocketMQ版实例的接入点。进入
消息队列RocketMQ版控制台实例详情页面获取。
         */
        properties.put(PropertyKeyConst.NAMESRV_ADDR, "XXXX");
        /**
         * 设置
消息队列RocketMQ版的Topic，在
消息队列RocketMQ版控制台创建。
         *
消息队列RocketMQ版和微消息队列MQTT版配合使用时，RocketMQ客户端仅操作一级Topic。
         */
        final String parentTopic = "XXXXX";
        Producer producer = ONSFactory.createProducer(properties);
```

```
        producer.start();
        ////////////////////////////////////////////////////////////////////////
////////////////////////////////
        /**
         * 初始化微消息队列MQTT版接收客户端，实际业务中微消息队列MQTT版一般部署在移动终端环境。
         */
        /**
         * 您在控制台创建的微消息队列MQTT版的实例ID。
         */
        String instanceId = "XXXXX";
        /**
         * 设置接入点，进入微消息队列MQTT版控制台实例详情页面获取。
         */
        String endPoint = "XXXXXX.mqtt.aliyuncs.com";
        /**
         * AccessKey ID，阿里云身份验证，在阿里云RAM控制台创建。
         */
        String accessKey = "XXXX";
        /**
         * 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
。
         */
        String secretKey = "XXXX";
        /**
         * MQTT客户端ID，由业务系统分配，需要保证每个TCP连接都不一样，保证全局唯一，如果不同的客
户端对象（TCP连接）使用了相同的clientId会导致连接异常断开。
         * clientId由两部分组成，格式为GroupID@@@DeviceID，其中GroupID在微消息队列MQTT版控制
台创建，DeviceID由业务方自己设置，clientId总长度不得超过64个字符。
         */
        String clientId = "GID_XXXX@@@XXXXX";
        /**
         * 微消息队列MQTT版支持子级Topic，用来做自定义的过滤，此处为示例，可以填写任何字符串。
         * 需要注意的是，完整的Topic长度不得超过128个字符。
         */
        final String subTopic = "/testMq4Iot";
        final String mq4IotTopic = parentTopic + subTopic;
        /**
         * QoS参数代表传输质量，可选0，1，2。详细信息，请参见名词解释。
         */
        final int qosLevel = 0;
        ConnectionOptionWrapper connectionOptionWrapper = new ConnectionOptionWrapper(i
nstanceId, accessKey, secretKey, clientId);
        final MemoryPersistence memoryPersistence = new MemoryPersistence();
        /**
         * 客户端协议和端口。客户端使用的协议和端口必须匹配，如果是SSL加密则设置ssl://endpoint:
8883。
         */
        final MqttClient mqttClient = new MqttClient("tcp://" + endPoint + ":1883", cli
entId, memoryPersistence);
        /**
         * 设置客户端发送超时时间，防止无限阻塞。
         */
        mqttClient.setTimeToWait(5000);
        final ExecutorService executorService = new ThreadPoolExecutor(1, 1, 0, TimeUni
```

```
t.MILLISECONDS,
            new LinkedBlockingQueue<Runnable>());
        mqttClient.setCallback(new MqttCallbackExtended() {
            @Override
            public void connectComplete(boolean reconnect, String serverURI) {
                /**
                 * 客户端连接成功后就需要尽快订阅需要的Topic。
                 */
                System.out.println("connect success");
                executorService.submit(new Runnable() {
                    @Override
                    public void run() {
                        try {
                            final String topicFilter[] = {mq4IotTopic};
                            final int[] qos = {qosLevel};
                            mqttClient.subscribe(topicFilter, qos);
                        } catch (MqttException e) {
                            e.printStackTrace();
                        }
                    }
                });
            }
            @Override
            public void connectionLost(Throwable throwable) {
                throwable.printStackTrace();
            }
            @Override
            public void messageArrived(String s, MqttMessage mqttMessage) throws Except
ion {
                /**
                 * 消费消息的回调接口，需要确保该接口不抛异常，该接口运行返回即代表消息消费成功。
                 * 消费消息需要保证在规定时间内完成，如果消费耗时超过服务端约定的超时时间，对于可
靠传输的模式，服务端可能会重试推送，业务需要做好幂等去重处理。
                 */
                System.out.println(
                    "receive msg from topic " + s + " , body is " + new String(mqttMess
age.getPayload()));
            }
            @Override
            public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {
                System.out.println("send msg succeed topic is : " + iMqttDeliveryToken.
getTopics()[0]);
            }
        });
        mqttClient.connect(connectionOptionWrapper.getMqttConnectOptions());
        for (int i = 0; i < 10; i++) {
            /**
             * 使用RocketMQ客户端发消息给MQTT客户端时，Topic指定为一级父Topic，Tag指定为MQ2MQ
TT。
             */
            Message msg = new Message(parentTopic, "MQ2MQTT", "hello mq send mqtt msg".
getBytes());
            /**
             * 使用RocketMQ客户端发消息给MQTT客户端时，可以通过MqttSecondTopic属性设置MQTT的
```

```
子级Topic属性。
             */
            msg.putUserProperties(PropertyKeyConst.MqttSecondTopic, subTopic);
            SendResult result = producer.send(msg);
            System.out.println(result);
            /**
             * 发送P2P消息，设置子级Topic。
             */
            msg.putUserProperties(PropertyKeyConst.MqttSecondTopic, "/p2p/" + clientId)
;
            result = producer.send(msg);
            System.out.println(result);
        }
        Thread.sleep(Long.MAX_VALUE);
    }
}
```

结果验证

完成消息收发后，您可在微消息队列MQTT版控制台查询轨迹以验证消息是否发送并接收成功。详细信息，请参见消息轨迹查询。

更多信息

- 快速使用MQTT的Java SDK收发消息（终端和终端消息收发）
- 快速使用MQTT的Java SDK收发消息（跨产品数据流出）
- 快速使用MQTT的Java SDK收发消息（客户端上下线通知）

1.2. 快速使用MQTT的Java SDK收发消息（跨产品数据流出）

本文介绍如何将微消息队列MQTT版的数据导出至其他阿里云产品。本文以当前仅支持的

消息队列RocketMQ版

数据互通为例进行说明。

前提条件

- 安装IDE。您可以使用IntellIJ IDEA或者Eclipse，本文以IntellIJ IDEA为例。
- 下载安装JDK。

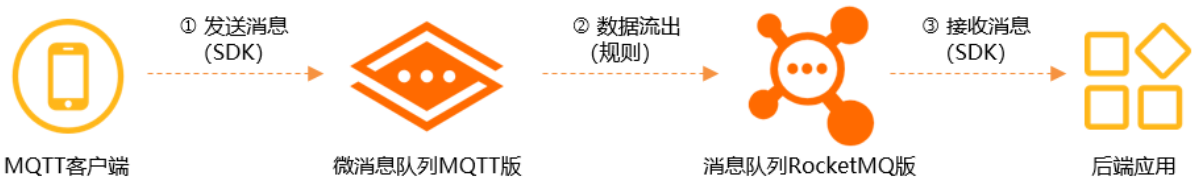
背景信息

本文以公网环境中的Java SDK为例说明如何将微消息队列MQTT版的数据导出至

消息队列RocketMQ版

。

此场景下可使用多语言的第三方开源SDK来实现消息收发。更多信息，请参见SDK下载。



如上图所示，您部署在公网环境的MQTT客户端数据需要传送至公网地域的后端应用，后端应用和MQTT客户端均通过Java语言开发。数据从微消息队列MQTT版导出至

消息队列RocketMQ版

是通过配置数据流出规则实现。两个产品的服务端通过各自产品提供的Java SDK分别与各自的客户端实现消息收发。

> **注意**
>
> 消息队列RocketMQ版
>
> 和微消息队列MQTT版的Topic不能跨地域使用，因此，本文中所有资源都应在公网地域创建。详细信息，请参见Topic地域化。

网络访问

微消息队列MQTT版同时提供了**公网接入点**和**VPC 接入点**。接入点说明如下：

- 在物联网和移动互联网的场景中，客户端推荐使用**公网接入点**接入。
- **VPC 接入点**仅供一些特殊场景使用。因为一般而言，涉及部署在云端服务器上的应用的场景，建议使用服务端消息产品例如

  消息队列RocketMQ版

  实现。

> **注意** 客户端使用接入点连接服务时务必使用域名接入，不得直接使用域名背后的IP地址直接连接，因为IP地址随时会变化。在以下使用情况中出现的问题微消息队列MQTT版产品方概不负责：
>
> - 客户端不使用域名接入而是使用IP地址接入，产品方更新了域名解析导致原有IP地址失效。
> - 客户端网络对IP地址设置网络防火墙策略，产品方更新了域名解析后新IP地址被您的防火墙策略拦截。

本文以公网接入点为例。微消息队列MQTT版与

消息队列RocketMQ版

的应用场景对比和消息属性映射关系请参见以下文档：

- MQTT与Rocket的应用场景对比
- MQTT与Rocket的消息结构映射

使用流程

MQTT客户端消息发送至后端应用的流程如下图所示。

步骤一：创建MQTT实例并获取接入点

> **注意** 轻量版实例不支持数据流入规则、数据流出规则及上下线通知规则，如需使用规则功能，请您在创建实例时选择其他类型的实例。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在弹出的付费方式面板中，**付费方式**固定为**包年包月**，您无需设置，直接在面板左下角单击**确定**。
5. 在**实例列表**页面左上角单击**创建实例**。
6. 在弹出的实例规格面板中，按需选择您需要购买的实例规格，选中**微消息队列 MQTT 版（包年包月）服务协议**，然后单击**立即购买**。

7. 在订单支付面板，根据提示完成支付。
8. 在支付成功页面单击**返回控制台**。
9. 回到微消息队列MQTT版控制台，在左侧导航栏单击**实例列表**，并将地域切换为您所购买的实例所对应的地域。
10. 在**实例列表**页面中，单击您所购买实例的名称或在其**操作**列单击**详情**，进入**实例详情**页面。
11. 在**实例详情**页面单击**接入点**页签，即可看到实例的接入点信息，本示例以**公网接入点**为例。

步骤二：创建父级Topic

MQTT协议支持多级Topic，父级Topic需在控制台创建，子级Topic无需创建，使用时直接在代码中设置即可。命名格式为：父级Topic和各子级Topic间均使用正斜线（/）隔开，*<父级Topic名称>/<二级Topic名称>/<三级Topic名称>*，例如，SendMessage/demo/producer。父级Topic和子级Topic总长度不能超过64个字符。Topic详细信息，请参见名词解释。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在实例列表中找到目标实例，在其**操作**列中，选择**更多 > Topic 管理**。
5. 在**Topic 管理**页面左上角，单击**创建 Topic**。
6. 在创建Topic面板中，输入要创建的Topic**名称**和**描述**，然后在左下角单击**确定**。
   您可以在**Topic 管理**页面查看刚创建的Topic。

步骤三：创建Group ID

Group ID详细信息，请参见名词解释。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在实例列表中找到目标实例，在其**操作**列中，选择**更多 > Group 管理**。
5. 在**Group 管理**页面的左上角，单击**创建 Group**。
6. 在创建Group面板中，输入**Group ID**，然后在左下角单击**确定**。
   您可以在**Group 管理**页面查看刚创建的Group。

步骤四：创建数据流出规则

规则中填写的参数需与您创建的资源保持一致。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在实例列表中找到目标实例，在其**操作**列，选择**更多 > 规则管理**。
5. 在**规则管理**页面左上角，单击**创建规则**。
6. 在**创建规则**页面完成以下操作。

i. 在**配置基本信息**配置向导页面，填写规则的基本信息，然后单击**下一步**。

参数	取值示例	说明
规则ID	111111	规则的全局唯一标识，说明如下： ■ 只能包含字母、数字、短划线（-）和下划线（_），至少包含一个字母或数字。 ■ 名称长度限制在3~64字符之间，长于64字符将被自动截取。 ■ 创建后无法更新。
描述	migrate from rocketmq	对规则的描述。
状态	启用	是否启用当前规则，取值说明如下： ■ 启用 ■ 停用
规则类型	数据流出	创建的规则类型，取值说明如下： ■ 数据流出：用于将微消息队列MQTT版的数据导出至其他阿里云产品。详细信息，请参见跨云产品的数据流出。 ■ 数据流入：用于将其他阿里云产品的数据导入至微消息队列MQTT版。详细信息，请参见跨云产品数据流入。 ■ 上下线通知：用于将获取的微消息队列MQTT版客户端上下线事件数据导出至其他阿里云产品。详细信息，请参见MQTT客户端上下线事件数据流出。

ii. 在**配置规则源**配置向导页面，配置数据源，然后单击**下一步**。

参数	取值示例	说明
Topic	TopicA	指定您需导出数据的源Topic，即微消息队列MQTT版的Topic。

iii. 在**配置规则目标**配置向导页面，配置数据的流转目标，然后单击**创建**。

参数	取值示例	说明
目标服务类型	消息队列 RocketMQ 版	指定您需将源Topic的数据转发至的目标云产品。 **说明** 当前仅支持消息队列RocketMQ版。
RocketMQ 实例	MQ_INST_13801563067*****_BbyOD2jQ	指定目标云产品的实例ID，即消息队列RocketMQ版的实例ID。 **说明** 仅支持选择和微消息队列MQTT版实例为同一地域的云产品实例。
Topic	TopicB	指定目标云产品的资源键值，即消息队列RocketMQ版的Topic。源Topic的数据将流转至TopicB。

您可以在**规则管理**的规则列表查看到刚创建的数据流出规则。

步骤五：调用Java SDK收发消息

1. 下载第三方的开源Java SDK。下载地址为Eclipse Paho Java Client。
2. 下载阿里云微消息队列MQTT版的Java SDK的Demo示例作为您代码开发的参考。下载地址为mqtt-java-demo。
3. 解压该Demo工程包至您指定的文件夹。
4. 在IntelliJ IDEA中，导入解压后的文件以创建相应的工程，并确认*pom.xml*中已包含以下依赖。

```
<dependencies>
        <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
            <version>1.10</version>
        </dependency>
        <dependency>
            <groupId>org.eclipse.paho</groupId>
            <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
            <version>1.2.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>4.5.2</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.48</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun.openservices</groupId>
            <artifactId>ons-client</artifactId>
            <version>1.8.5.Final</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-onsmqtt</artifactId>
            <version>1.0.3</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-core</artifactId>
            <version>4.5.0</version>
        </dependency>
</dependencies>
```

> **说明** ons-client的版本信息，请参见版本说明。

5. 在MQ4IoTSendMessageToRocketMQ.java类中，按代码注释说明填写相应参数，主要涉及步骤一至步骤三所创建的MQTT资源以及您在

   消息队列RocketMQ版

   创建的相应资源，然后执行Main函数运行代码实现消息收发。

   示例代码如下。

```
package com.aliyun.openservices.lmq.example.demo;
import com.aliyun.openservices.lmq.example.util.ConnectionOptionWrapper;
import com.aliyun.openservices.ons.api.Action;
import com.aliyun.openservices.ons.api.ConsumeContext;
import com.aliyun.openservices.ons.api.Consumer;
import com.aliyun.openservices.ons.api.Message;
```

```
import com.aliyun.openservices.ons.api.Message;
import com.aliyun.openservices.ons.api.MessageListener;
import com.aliyun.openservices.ons.api.ONSFactory;
import com.aliyun.openservices.ons.api.PropertyKeyConst;
import java.util.Properties;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallbackExtended;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;
/**
 * 阿里云生产环境中，
消息队列RocketMQ版除了公网地域，其他地域的实例不允许在本地使用，必须在对应地域的ECS机器上部署使用
。
 */
public class MQ4IoTSendMessageToRocketMQ {
    public static void main(String[] args) throws Exception {
        /**
         * 初始化
消息队列RocketMQ版接收客户端，实际业务中一般部署在服务端应用中。
         */
        Properties properties = new Properties();
        /**
         * 设置
消息队列RocketMQ版Group ID，在
消息队列RocketMQ版控制台创建。
         */
        properties.setProperty(PropertyKeyConst.GROUP_ID, "GID-XXXXX");
        /**
         * AccessKey ID，阿里云身份验证，在阿里云RAM控制台创建。
         */
        properties.put(PropertyKeyConst.AccessKey,"XXXX");
         /**
         * 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
。
         */
        properties.put(PropertyKeyConst.SecretKey, "XXXX");
         /**
         * 设置TCP接入点，该接入点为
消息队列RocketMQ版实例的接入点。
消息队列RocketMQ版控制台实例详情页面获取。
         */
        properties.put(PropertyKeyConst.NAMESRV_ADDR, "http://xxxxx.XXXXX.mq-internet.a
liyuncs.com");
        /**
         * 设置
消息队列RocketMQ版的Topic，在
消息队列RocketMQ版控制台创建。
         *
消息队列RocketMQ版和微消息队列MQTT配合使用时，RocketMQ客户端仅操作一级Topic。
         */
        final String parentTopic = "XXXXX";
        Consumer consumer = ONSFactory.createConsumer(properties);
        consumer.subscribe(parentTopic, "*", new MessageListener() {
            public Action consume(Message message, ConsumeContext consumeContext) {
```

```
                System.out.println("recv msg:" + message);
                return Action.CommitMessage;
            }
        });
        consumer.start();
        ////////////////////////////////////////////////////////////////////////////
////////////////////////////////
        /**
         * 初始化微消息队列MQTT版发送客户端，实际业务中微消息队列MQTT版一般部署在移动终端环境。
         */
        /**
         * 您在控制台创建的微消息队列MQTT的实例ID。
         */
        String instanceId = "XXXXX";
         /**
         * 设置接入点，进入微消息队列MQTT版控制台实例详情页面获取。
         */
        String endPoint = "XXXXXX.mqtt.aliyuncs.com";
         /**
         * AccessKey ID，阿里云身份验证，在阿里云RAM控制台创建。
         */
        String accessKey = "XXXX";
        /**
         * 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
。
         */
        String secretKey = "XXXX";
        /**
         * MQTT客户端ID，由业务系统分配，需要保证每个TCP连接都不一样，保证全局唯一，如果不同的客
户端对象（TCP连接）使用了相同的clientId会导致连接异常断开。
         * clientId由两部分组成，格式为GroupID@@@DeviceID，其中GroupID在微消息队列MQTT版控制
台创建，DeviceID由业务方自己设置，clientId总长度不得超过64个字符。
         */
        String clientId = "GID_XXXX@@@XXXXX";
       /**
         * 微消息队列MQTT版支持子级Topic，用来做自定义的过滤，此处为示例，可以填写任何字符串。
         * 需要注意的是，完整的Topic长度不得超过128个字符。
         */
        final String mq4IotTopic = parentTopic + "/" + "testMq4Iot";
        /**
         * QoS参数代表传输质量，可选0，1，2。详细信息，请参见名词解释。
         */
        final int qosLevel = 0;
        ConnectionOptionWrapper connectionOptionWrapper = new ConnectionOptionWrapper(i
nstanceId, accessKey, secretKey, clientId);
        final MemoryPersistence memoryPersistence = new MemoryPersistence();
         /**
         * 客户端协议和端口。客户端使用的协议和端口必须匹配，如果是SSL加密则设置ssl://endpoint:
8883。
         */
        final MqttClient mqttClient = new MqttClient("tcp://" + endPoint + ":1883", cli
entId, memoryPersistence);
        /**
         * 设置客户端发送超时时间，防止无限阻塞。
```

```
         */
        mqttClient.setTimeToWait(5000);
        mqttClient.setCallback(new MqttCallbackExtended() {
            @Override
            public void connectComplete(boolean reconnect, String serverURI) {
                /**
                 * 客户端连接成功后就需要尽快订阅需要的Topic。
                 */
                System.out.println("connect success");
            }
            @Override
            public void connectionLost(Throwable throwable) {
                throwable.printStackTrace();
            }
            @Override
            public void messageArrived(String s, MqttMessage mqttMessage) throws Except
ion {
            }
            @Override
            public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {
                System.out.println("send msg succeed topic is : " + iMqttDeliveryToken.
getTopics()[0]);
            }
        });
        mqttClient.connect(connectionOptionWrapper.getMqttConnectOptions());
        for (int i = 0; i < 10; i++) {
            MqttMessage message = new MqttMessage("hello mq4Iot pub sub msg".getBytes()
);
            message.setQos(qosLevel);
            /**
             *  发送普通消息时，Topic必须和接收方订阅的Topic一致，或者符合通配符匹配规则。
             */
            mqttClient.publish(mq4IotTopic, message);
        }
        Thread.sleep(Long.MAX_VALUE);
    }
}
```

结果验证

完成消息收发后，您可在微消息队列MQTT版控制台查询轨迹以验证消息是否发送并接收成功。详细信息，请参见消息轨迹查询。

更多信息

- 快速使用MQTT的Java SDK收发消息（终端和终端消息收发）
- 快速使用MQTT的Java SDK收发消息（跨产品数据流入）
- 快速使用MQTT的Java SDK收发消息（客户端上下线通知）

1.3. 快速使用MQTT的Java SDK收发消息（客户端上下线通知）

MQTT客户端的上下线事件将会触发MQTT服务端生成一条通知消息，微消息队列MQTT版支持将该条消息数据导出至其他阿里云产品，并使用MQTT的Java SDK实现MQTT客户端与后端应用收发消息。本文以当前仅支持的

消息队列RocketMQ版

数据互通为例进行说明。

前提条件

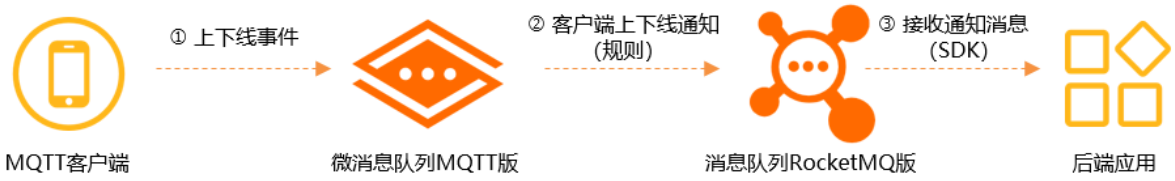
- 安装IDE。您可以使用IntellIJ IDEA或者Eclipse，本文以IntellIJ IDEA为例。
- 下载安装JDK。

背景信息

您可以通过配置客户端上下线通知规则来将MQTT客户端的上下线事件通知的数据导出至其他阿里云产品。

本文以公网环境中的Java SDK为例说明微消息队列MQTT版如何将MQTT客户端上下线事件通知的消息发送至后端应用。

此场景下可使用多语言的第三方开源SDK来实现消息收发。更多信息，请参见SDK下载。



如上图所示，您的后端应用和MQTT客户端均通过Java语言开发。您部署在公网环境中的MQTT客户端上线或下线时，会触发MQTT服务生成一条事件消息，您可以通过配置客户端上下线通知规则将该通知消息经过

消息队列RocketMQ版

发送至后端应用。两个产品的服务端通过各自产品提供的Java SDK分别与各自的客户端实现消息收发。

客户端上下线通知功能的更多信息，请参见MQTT客户端上下线事件数据流出。

> **注意**
>
> 消息队列RocketMQ版
>
> 和微消息队列MQTT版的Topic不能跨地域使用，因此，本文中所有资源都应在公网地域创建。详细信息，请参见Topic地域化。

网络访问

微消息队列MQTT版同时提供了**公网接入点**和**VPC 接入点**。接入点说明如下：

- 在物联网和移动互联网的场景中，客户端推荐使用**公网接入点**接入。
- **VPC 接入点**仅供一些特殊场景使用。因为一般而言，涉及部署在云端服务器上的应用的场景，建议使用服务端消息产品例如

  消息队列RocketMQ版

  实现。

> **注意** 客户端使用接入点连接服务时务必使用域名接入，不得直接使用域名背后的IP地址直接连接，因为IP地址随时会变化。在以下使用情况中出现的问题微消息队列MQTT版产品方概不负责：
>
> - 客户端不使用域名接入而是使用IP地址接入，产品方更新了域名解析导致原有IP地址失效。
> - 客户端网络对IP地址设置网络防火墙策略，产品方更新了域名解析后新IP地址被您的防火墙策略拦截。

本文以公网接入点为例。微消息队列MQTT版与

消息队列RocketMQ版

的应用场景对比和消息属性映射关系请参见以下文档：

- MQTT与Rocket的应用场景对比
- MQTT与Rocket的消息结构映射

使用流程

MQTT客户端上下线通知的消息收发流程如下图所示。

步骤一：创建MQTT实例并获取接入点

> **注意** 轻量版实例不支持数据流入规则、数据流出规则及上下线通知规则，如需使用规则功能，请您在创建实例时选择其他类型的实例。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在**实例列表**页面左上角单击**创建实例**。
5. 在弹出的付费方式面板中，**付费方式**固定为**包年包月**，您无需设置，直接在面板左下角单击**确定**。
6. 在弹出的实例规格面板中，按需选择您需要购买的实例规格，选中**微消息队列 MQTT 版（包年包月）服务协议**，然后单击**立即购买**。

7. 在订单支付面板，根据提示完成支付。
8. 在支付成功页面单击**返回控制台**。
9. 回到微消息队列MQTT版控制台，在左侧导航栏单击**实例列表**，并将地域切换为您所购买的实例所对应的地域。
10. 在**实例列表**页面中，单击您所购买实例的名称或在其**操作**列单击**详情**，进入**实例详情**页面。
11. 在**实例详情**页面单击**接入点**页签，即可看到实例的接入点信息，本示例以**公网接入点**为例。

步骤二：创建父级Topic

MQTT协议支持多级Topic，父级Topic需在控制台创建，子级Topic无需创建，使用时直接在代码中设置即可。命名格式为：父级Topic和各子级Topic间均使用正斜线（/）隔开，*<父级Topic名称>/<二级Topic名称>/<三级Topic名称>*，例如，SendMessage/demo/producer。父级Topic和子级Topic总长度不能超过64个字符。Topic详细信息，请参见名词解释。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。

4. 在实例列表中找到目标实例，在其**操作**列中，选择**更多 > Topic 管理**。
5. 在**Topic 管理**页面左上角，单击**创建 Topic**。
6. 在创建Topic面板中，输入要创建的Topic**名称**和**描述**，然后在左下角单击**确定**。
   您可以在**Topic 管理**页面查看刚创建的Topic。

步骤三：创建Group ID

Group ID详细信息，请参见名词解释。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在实例列表中找到目标实例，在其**操作**列中，选择**更多 > Group 管理**。
5. 在**Group 管理**页面的左上角，单击**创建 Group**。
6. 在创建Group面板中，输入**Group ID**，然后在左下角单击**确定**。
   您可以在**Group 管理**页面查看刚创建的Group。

步骤四：创建客户端上下线通知规则

规则中填写的参数需与您创建的资源保持一致。

1. 登录微消息队列MQTT版控制台。
2. 在左侧导航栏单击**实例列表**。
3. 在顶部菜单栏选择地域。
4. 在实例列表中找到目标实例，在其**操作**列，选择**更多 > 规则管理**。
5. 在**规则管理**页面左上角，单击**创建规则**。
6. 在**创建规则**页面完成以下操作。

i. 在**配置基本信息**配置向导页面，填写规则的基本信息，然后单击**下一步**。

参数	取值示例	说明
规则ID	111111	规则的全局唯一标识，说明如下： ■ 只能包含字母、数字、短划线（-）和下划线（_），至少包含一个字母或数字。 ■ 名称长度限制在3~64字符之间，长于64字符将被自动截取。 ■ 创建后无法更新。
描述	migrate from rocketmq	对规则的描述。
状态	启用	是否启用当前规则，取值说明如下： ■ 启用 ■ 停用
规则类型	上下线通知	创建的规则类型，取值说明如下： ■ 数据流出：用于将微消息队列MQTT版的数据导出至其他阿里云产品。详细信息，请参见跨云产品的数据流出。 ■ 数据流入：用于将其他阿里云产品的数据导入至微消息队列MQTT版。详细信息，请参见跨云产品数据流入。 ■ 上下线通知：用于将获取的微消息队列MQTT版客户端上下线事件数据导出至其他阿里云产品。详细信息，请参见MQTT客户端上下线事件数据流出。

ii. 在**配置规则源**配置向导页面，配置数据源，然后单击**下一步**。

参数	取值示例	说明
Group ID	GID_Client_Status	指定需导出数据的设备组。Group ID的详细信息，请参见名词解释。

iii. 在**配置规则目标**配置向导页面，配置数据的流转目标，然后单击**创建**。

参数	取值示例	说明
目标服务类型	消息队列 RocketMQ 版	指定您需将微消息队列MQTT版客户端上下线通知流转至哪个目标云产品。 **说明** 当前仅支持消息队列RocketMQ版。
RocketMQ 实例	MQ_INST_13801563067*****_BbyOD2jQ	指定目标云产品的实例ID，即消息队列RocketMQ版的实例ID。 **说明** 仅支持选择和微消息队列MQTT版实例为同一地域的云产品实例。
Topic	TopicB	指定目标云产品的资源键值，即消息队列RocketMQ版的Topic。微消息队列MQTT版客户端上下线通知信息将流转至TopicB。

您可以在**规则管理**的规则列表查看到刚创建的上下线通知规则。

步骤五：调用Java SDK收发消息

1. 下载第三方的开源Java SDK。下载地址为Eclipse Paho Java Client。
2. 下载阿里云微消息队列MQTT版的Java SDK的Demo示例作为您代码开发的参考。下载地址为mqtt-java-demo。
3. 解压该Demo工程包至您指定的文件夹。
4. 在IntelliJ IDEA中，导入解压后的文件以创建相应的工程，并确认*pom.xml*中已包含以下依赖。

```
<dependencies>
        <dependency>
            <groupId>commons-codec</groupId>
            <artifactId>commons-codec</artifactId>
            <version>1.10</version>
        </dependency>
        <dependency>
            <groupId>org.eclipse.paho</groupId>
            <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
            <version>1.2.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>4.5.2</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.48</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun.openservices</groupId>
            <artifactId>ons-client</artifactId>
            <version>1.8.5.Final</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-onsmqtt</artifactId>
            <version>1.0.3</version>
        </dependency>
        <dependency>
            <groupId>com.aliyun</groupId>
            <artifactId>aliyun-java-sdk-core</artifactId>
            <version>4.5.0</version>
        </dependency>
</dependencies>
```

> **说明** ons-client的版本信息，请参见版本说明。

5. 在MQTTClientStatusNoticeProcessDemo.java类中，按代码注释说明填写相应参数，主要涉及步骤一至步骤三所创建MQTT资源以及您在

   消息队列RocketMQ版

   创建的相应资源，然后执行Main函数运行代码实现消息收发。

   示例代码如下。

```
package com.aliyun.openservices.lmq.example.demo;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import com.aliyun.openservices.ons.api.Action;
import com.aliyun.openservices.ons.api.ConsumeContext;
import com.aliyun.openservices.ons.api.Consumer;
```

```
import com.aliyun.openservices.ons.api.Message;
import com.aliyun.openservices.ons.api.MessageListener;
import com.aliyun.openservices.ons.api.ONSFactory;
import com.aliyun.openservices.ons.api.PropertyKeyConst;
import java.util.Map;
import java.util.Properties;
import java.util.Set;
public class MQTTClientStatusNoticeProcessDemo {
    public static void main(String[] args) {
        /**
         * 初始化消息队列RocketMQ版接收客户端，实际业务中一般部署在服务端应用中。
         */
        Properties properties = new Properties();
        /**
         * 设置消息队列RocketMQ版Group ID，在消息队列RocketMQ版控制台创建。
         */
        properties.setProperty(PropertyKeyConst.GROUP_ID, "GID_XXXX");
        /**
         * AccessKey ID，阿里云身份验证，在阿里云RAM控制台创建。
         */
        properties.put(PropertyKeyConst.AccessKey, "XXXX");
         /**
         * 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
。
         */
        properties.put(PropertyKeyConst.SecretKey, "XXXX");
        /**
         * 设置TCP接入点，该接入点为消息队列RocketMQ版实例的接入点。进入消息队列RocketMQ版控制
台实例详情页面获取。
         */
        properties.put(PropertyKeyConst.NAMESRV_ADDR, "XXXX");
        /**
         * 使用消息队列RocketMQ版消费端来处理MQTT客户端的上下线通知时，订阅的Topic为上下线通知T
opic。
         */
        final String parentTopic = "GID_XXXX_MQTT";
        /**
         * 客户端状态数据，实际生产环境中建议使用数据库或者Redis等外部持久化存储来保存该信息，避
免应用重启丢失状态，本示例以单机内存版实现为例。
         */
        MqttClientStatusStore mqttClientStatusStore = new MemoryHashMapStoreImpl();
        Consumer consumer = ONSFactory.createConsumer(properties);
        /**
         *  此处仅处理客户端是否在线，因此只需要关注connect事件和tcpclean事件即可。
         */
        consumer.subscribe(parentTopic, "connect||tcpclean", new MqttClientStatusNotice
Listener(mqttClientStatusStore));
        consumer.start();
        String clientId = "GID_XXXXX@@@XXXXX";
        while (true) {
            System.out.println("ClientStatus :" + checkClientOnline(clientId, mqttClien
tStatusStore));
            try {
                Thread.sleep(1000);
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
    /**
     * 处理上下线通知的逻辑。
     * 实际部署过程中，消费上下线通知的应用可能部署多台机器，因此客户端在线状态的数据可以使用数据库或者Redis等外部共享存储来维护。
     * 其次需要单独做消息幂等处理，以免重复接收消息导致状态机判断错误。
     */
    static class MqttClientStatusNoticeListener implements MessageListener {
        private MqttClientStatusStore mqttClientStatusStore;
        public MqttClientStatusNoticeListener(
            MqttClientStatusStore mqttClientStatusStore) {
            this.mqttClientStatusStore = mqttClientStatusStore;
        }
        @Override
        public Action consume(Message message, ConsumeContext context) {
            try {
                JSONObject msgBody = JSON.parseObject(new String(message.getBody()));
                System.out.println(msgBody);
                String eventType = msgBody.getString("eventType");
                String clientId = msgBody.getString("clientId");
                String channelId = msgBody.getString("channelId");
                ClientStatusEvent event = new ClientStatusEvent();
                event.setChannelId(channelId);
                event.setClientIp(msgBody.getString("clientIp"));
                event.setEventType(eventType);
                event.setTime(msgBody.getLong("time"));
                /**
                 * 首先存储新的事件。
                 */
                mqttClientStatusStore.addEvent(clientId, channelId, eventType, event);
                /**
                 * 读取当前channel的事件列表。
                 */
                Set<ClientStatusEvent> events = mqttClientStatusStore.getEvent(clientId
, channelId);
                if (events == null || events.isEmpty()) {
                    return Action.CommitMessage;
                }
                /**
                 * 如果事件列表里上线和下线事件都已经收到，则当前channel已经掉线，可以清理掉这个channel的数据。
                 */
                boolean findOnlineEvent = false;
                boolean findOfflineEvent = false;
                for (ClientStatusEvent clientStatusEvent : events) {
                    if (clientStatusEvent.isOnlineEvent()) {
                        findOnlineEvent = true;
                    } else {
                        findOfflineEvent = true;
                    }
```

```
                }
                if (findOnlineEvent && findOfflineEvent) {
                    mqttClientStatusStore.deleteEvent(clientId, channelId);
                }
                return Action.CommitMessage;
            } catch (Throwable e) {
                e.printStackTrace();
            }
            return Action.ReconsumeLater;
        }
    }
    /**
     * 根据状态表判断一个clientId是否有活跃的TCP连接。
     * 1.如果没有channel表，则客户端一定不在线。
     * 2.如果channel表非空，检查一下channel数据中是否仅包含上线事件，如果有则代表有活跃连接，客户端在线。
     * 如果全部的channel都有掉线断开事件则客户端一定不在线。
     *
     * @param clientId
     * @param mqttClientStatusStore
     * @return
     */
    public static boolean checkClientOnline(String clientId,
        MqttClientStatusStore mqttClientStatusStore) {
        Map<String, Set<ClientStatusEvent>> channelMap = mqttClientStatusStore.getEvent
sByClientId(clientId);
        if (channelMap == null) {
            return false;
        }
        for (Set<ClientStatusEvent> events : channelMap.values()) {
            boolean findOnlineEvent = false;
            boolean findOfflineEvent = false;
            for (ClientStatusEvent event : events) {
                if (event.isOnlineEvent()) {
                    findOnlineEvent = true;
                } else {
                    findOfflineEvent = true;
                }
            }
            if (findOnlineEvent & !findOfflineEvent) {
                return true;
            }
        }
        return false;
    }
}
```

结果验证

完成消息收发后，您可在微消息队列MQTT版控制台查询轨迹以验证消息是否发送并接收成功。详细信息，请参见消息轨迹查询。

更多信息

- 快速使用MQTT的Java SDK收发消息（终端和终端消息收发）
- 快速使用MQTT的Java SDK收发消息（跨产品数据流入）
- 快速使用MQTT的Java SDK收发消息（跨产品数据流出）

2.常见问题

本文主要列举使用微消息队列MQTT版时遇到的常见问题。

详细信息

- 购买预付费（包年包月）实例后产生额外费用的原因
- 收到短信告警提示“使用非法的Group ID”
- 短信告警提示离线消息存储数量超过系统限制
- 连接微消息队列MQTT版的服务端时异常断开
- 之前订阅过的Topic仍然在推送消息
- 消息队列MQTT版的客户端存在流量限制

3.视频专区

3.1. 如何快速使用微消息队列MQTT版的Java SDK收发消息？

本文提供使用微消息队列MQTT版的Java SDK收发消息的快速入门视频。

4.联系我们

如果您准备使用或正在使用微消息队列MQTT版，有任何疑问和建议，欢迎您搜索钉钉群号35228338加入钉钉群与我们交流。

5.相关协议

5.1. 阿里云产品服务协议（通用）

通用服务条款与专用服务条款详情，请参见阿里云产品服务协议（通用）。

5.2. 微消息队列MQTT版服务等级协议

微消息队列MQTT版提供的服务可用性等级指标及赔偿方案，请查看微消息队列MQTT版服务等级协议。