ALIBABA CLOUD

阿里云

微消息队列MQTT版
快速入门

文档版本：20220524

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}`

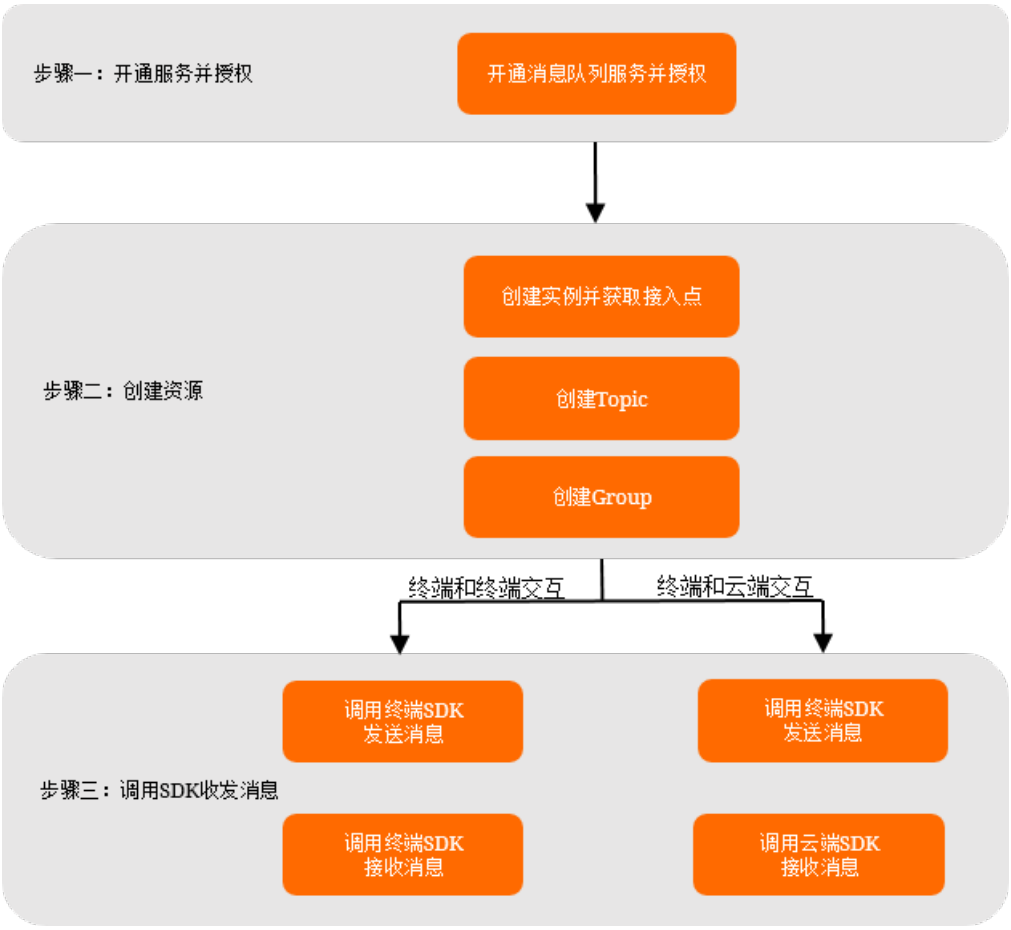
目录

1.快速入门概述

微消息队列MQTT版支持使用多种语言的SDK接入服务端实现消息收发，本文以Java为例介绍不同场景下的消息收发流程。

操作流程

使用SDK收发消息流程如下图所示。



具体操作步骤如下：

1. 开通服务并授权

   开通消息队列服务，并授予RAM用户操作微消息队列MQTT版资源的权限。

2. 创建资源

   在微消息队列MQTT版控制台创建实例、Topic和Group资源。

3. 调用SDK收发消息
   - 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（终端和云端消息收发）

操作视频

该视频以终端和终端交互流程为例演示如何使用Java SDK收发消息。

2.开通服务并授权

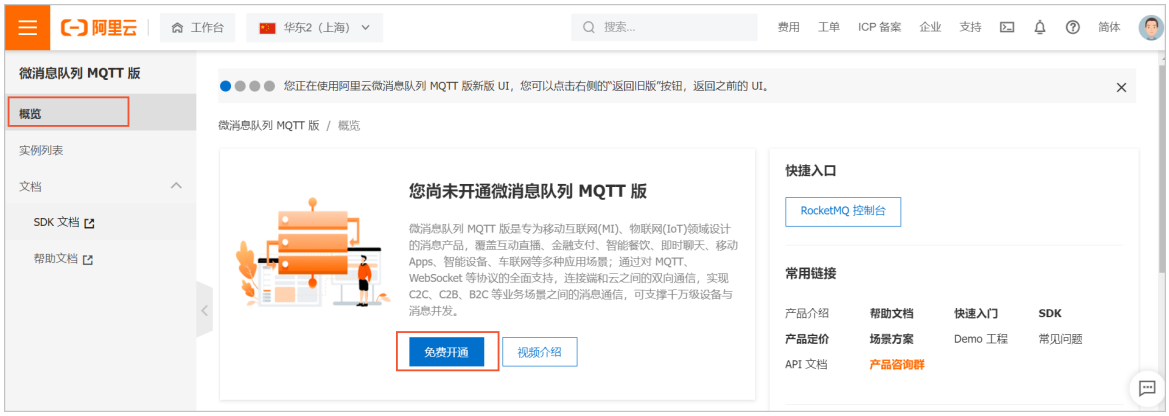
在阿里云官方网站开通消息队列服务后方可开始使用微消息队列MQTT版。如果您的账号为RAM用户，必须先为RAM用户进行授权，才能通过控制台或API访问相应的微消息队列MQTT版资源，并使用SDK收发消息。

前提条件

您已注册阿里云账号并完成实名认证。更多信息，请参见注册阿里云账号。

步骤一：开通消息队列服务

1. 打开微消息队列MQTT版产品页。
2. 在页面右上角单击**登录**。
3. 在**登录**页面输入您的阿里云账号和密码，并单击**登录**。
4. 在微消息队列MQTT版的产品页，单击**管理控制台**。
   页面跳转至微消息队列MQTT版控制台。
5. 在**概览**页面，单击**免费开通**。



6. 在服务开通面板，选中**消息队列RocketMQ版服务协议**并单击**立即开通**。

> **说明** 微消息队列MQTT版是消息队列RocketMQ版系列产品之一，开通消息队列RocketMQ版服务即开通微消息队列MQTT版服务。开通服务不收取费用。

7. 单击**关闭**回到**概览**页面。
   刷新页面，可看到**概览**页面出现资源分布等信息，说明服务开通成功。

（RAM用户必选）步骤一：为RAM用户授权

若您开通消息队服务使用的是账号是RAM用户，必须先为RAM用户进行授权，才能访问微消息队列MQTT版的资源。若您的账号是阿里云账号，默认拥有使用微消息队列MQTT版资源的权限，您可以跳过该步骤。

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，选择**身份管理 > 用户**。
3. 在**用户**页面，单击目标RAM用户**操作**列的**添加权限**。

4. 在**添加权限**面板，为RAM用户添加权限。
   i. 选择授权应用范围。
      - **整个云账号**：权限在当前阿里云账号内生效。
      - **指定资源组**：权限在指定的资源组内生效。

        > **说明** 指定资源组授权生效的前提是该云服务已支持资源组。更多信息，请参见支持资源组的云服务。

   ii. 输入授权主体。

      授权主体即需要授权的RAM用户，系统会自动填入当前的RAM用户，您也可以添加其他RAM用户。

   iii. 选择权限策略。

      > **说明** 每次最多绑定5条策略，如需绑定更多策略，请分次操作。

5. 单击**确定**。
6. 单击**完成**。

微消息队列MQTT版提供以下系统策略，您可以根据权限范围为RAM用户授予相关权限。

权限策略名称	说明
AliyunMQFullAccess	管理微消息队列MQTT版的权限，等同于阿里云账号的权限，被授予该权限的RAM用户具有所有消息收发权限且有控制台所有功能操作权限。
AliyunMQPubOnlyAccess	微消息队列MQTT版的发布权限，被授予该权限的RAM用户具有使用阿里云账号所有资源通过SDK发送消息的权限。
AliyunMQSubOnlyAccess	微消息队列MQTT版的订阅权限，被授予该权限的RAM用户具有使用阿里云账号所有资源通过SDK订阅消息的权限。
AliyunMQReadOnlyAccess	微消息队列MQTT版的只读权限，被授予该权限的RAM用户仅有通过访问控制台或调用管控API读取资源信息的权限。

后续步骤

创建资源

3.创建资源

在使用SDK收发消息前，您需要在微消息队列MQTT版控制台创建相关资源，包括实例、Topic和Group。您在调用SDK时需要填写这些创建的资源信息。

前提条件

开通服务并授权

创建实例并获取接入点

实例是用于微消息队列MQTT版服务的虚拟机资源，会存储消息主题（Topic）和客户端相关（Group ID）信息。

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。

创建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

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。

后续步骤

快速使用MQTT的Java SDK收发消息（终端和终端消息收发）

4.快速使用MQTT的Java SDK收发消息（终端和云端消息收发）

本文介绍如何快速使用微消息队列MQTT版的Java SDK实现MQTT终端和云端服务的消息收发。

前提条件

- 创建资源
- 获取AccessKey
- 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

MQTT终端和云端服务交互流程如下图所示。终端设备和云端服务可分别通过对应的SDK接入微消息队列MQTT版，实现终端和云端服务的双向通信。

本文以公网环境为例，介绍使用Java SDK实现消息收发。更多消息收发示例代码，请参见终端Demo工程或云端Demo工程。

接入点说明

终端和云端服务与微消息队列MQTT版通信时，需要在各自的SDK代码中设置微消息队列MQTT版实例的接入点信息，通过接入点和微消息队列MQTT版服务端连接。

- **终端SDK接入点格式**

  使用终端SDK接入微消息队列MQTT版时，需要填写的接入点格式如下：

  - **公网接入点：** *MQTT实例ID*.mqtt.aliyuncs.com
  - **VPC 接入点：** *MQTT实例ID*-internal-vpc.mqtt.aliyuncs.com

  终端SDK接入点也可以直接在微消息队列MQTT版控制台**实例详情**页面的**接入点**页签中查看。

- **云端SDK接入点格式**

  使用云端SDK接入微消息队列MQTT版时，需要填写的接入点格式如下：

  > 注意　仅实例内核版本为V3.3.0且实例地域属于中国内地的实例支持云端SDK接入。

  - **公网接入点：** *MQTT实例ID*-server-internet.mqtt.aliyuncs.com
  - **VPC 接入点：** *MQTT实例ID*-server-internal.mqtt.aliyuncs.com

> 说明　*MQTT实例ID*可在微消息队列MQTT版控制台**实例详情**页面的**基础信息**区域查看。

终端SDK接入点和云端SDK接入点同时支持**公网接入点**和**VPC 接入点**。**公网接入点**为本地公网环境访问的IP地址，一般用于物联网和移动互联网场景中；**VPC 接入点**为云上私网访问的IP地址，一般用于云端应用接入微消息队列MQTT版。

> **注意** SDK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：
>
> - 终端或云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。
> - 终端或云端网络侧对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。

调用终端SDK发送消息

1. 下载第三方的开源Java SDK。下载地址为Eclipse Paho Java Client。
2. 下载终端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</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>
```

5. 在MQ4IoTProducerDemo.java类中，按代码注释说明填写相应参数，主要涉及您已在创建资源中所创建

的MQTT资源信息。然后执行Main函数运行代码完成消息发送。

示例代码如下。

```
package com.aliyun.openservices.lmq.example.demo;
import com.aliyun.openservices.lmq.example.util.ConnectionOptionWrapper;
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;
public class MQ4IoTProducerDemo {
    public static void main(String[] args) throws Exception {
        /**
         * 您创建的微消息队列MQTT版的实例ID。
         */
        String instanceId = "XXXXX";
        /**
         * 设置终端SDK的接入点，进入微消息队列MQTT版控制台实例详情页面的接入点页签查看。
         * 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，否则可能会导致客户端异常。
         */
        String endPoint = "XXXXX.mqtt.aliyuncs.com";
        /**
         * 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 = "XXXXX";
        /**
         * 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 = "XXXXX";
        /**
         * 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_XXXXX@@@XXXXX";
        /**
         * 微消息队列MQTT版消息的一级Topic，需要在控制台创建才能使用。
         * 如果使用了没有创建或者没有被授权的Topic会导致鉴权失败，服务端会断开客户端连接。
         */
        final String parentTopic = "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，可以填写任意字符串。
         * 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);
        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");
            }
            @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++) {
            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);
            /**
             * 微消息队列MQTT版支持点对点消息，即如果发送方明确知道该消息只需要给特定的一个设备接收，且知道对端的clientId，则可以直接发送点对点消息。
             * 点对点消息不需要经过订阅关系匹配，可以简化订阅方的逻辑。点对点消息的Topic格式规范是 {{parentTopic}}/p2p/{{targetClientId}}。
             */
            String receiverId = "xxx";
            final String p2pSendTopic = parentTopic + "/p2p/" + receiverId;
            message = new MqttMessage("hello mq4Iot p2p msg".getBytes());
            message.setQos(qosLevel);
            mqttClient.publish(p2pSendTopic, message);
        }
        Thread.sleep(Long.MAX_VALUE);
    }
}
```

调用云端SDK接收消息

1. 下载微消息队列MQTT版提供的云端SDK。下载地址为云端SDK版本说明。
2. 下载云端SDK的Demo示例做为您代码开发的参考。下载地址为mqtt-server-sdk-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>com.alibaba.mqtt</groupId>
            <artifactId>server-sdk</artifactId>
            <version>1.0.0.Final</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.48</version>
        </dependency>
</dependencies>
```

5. 在MQTTConsumerDemo.java类中，按代码注释说明填写相应参数，主要涉及您已在创建资源中所创建好的MQTT资源信息。然后执行Main函数运行代码完成消息接收。

   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.MessageListener;
import com.alibaba.mqtt.server.config.ChannelConfig;
import com.alibaba.mqtt.server.config.ConsumerConfig;
import com.alibaba.mqtt.server.model.MessageProperties;
public class MQTTConsumerDemo {
    public static void main(String[] args) throws Exception {
        /**
```

```
         * 设置云端SDK的接入点，请参见接入点说明中的云端SDK接入点格式。
         * 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 = "port";
        /**
         * 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";
        /**
         * 微消息队列MQTT版消息的一级Topic，需要在控制台创建才能使用。
         * 如果使用了没有创建或者没有被授权的Topic会导致鉴权失败，服务端会断开客户端连接。
         */
        String firstTopic = "firstTopic";
        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 ConsumerConfig());
        serverConsumer.start();
        serverConsumer.subscribeTopic(firstTopic, new MessageListener() {
            @Override
            public void process(String msgId, MessageProperties messageProperties, byte[] payload) {
                System.out.println("Receive:" + msgId + "," + JSONObject.toJSONString(messageProperties) + "," + new String(payload));
            }
        });
    }
}
```

> **说明** 云端SDK消息发送的示例代码，请参见MQTTProducerDemo.java。

5.快速使用MQTT的Java SDK收发消息（终端和终端消息收发）

本文介绍如何快速使用微消息队列MQTT版的Java SDK实现MQTT终端与终端消息的收发。

前提条件

- 创建资源
- 获取AccessKey
- 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

微消息队列MQTT版最简单的使用场景即MQTT终端和终端交互，消息生产者和消费者均分布在终端设备。各终端设备均通过终端SDK微消息队列MQTT版与微消息队列MQTT版服务端连接实现消息收发。

本文以公网环境为例，说明如何使用Java SDK完成消息收发。

接入点说明

终端和云端服务与微消息队列MQTT版通信时，需要在各自的SDK代码中设置微消息队列MQTT版实例的接入点信息，通过接入点和微消息队列MQTT版服务端连接。

- **终端SDK接入点格式**

  使用终端SDK接入微消息队列MQTT版时，需要填写的接入点格式如下：

  - **公网接入点：** *MQTT实例ID*.mqtt.aliyuncs.com
  - **VPC 接入点：** *MQTT实例ID*-internal-vpc.mqtt.aliyuncs.com

  终端SDK接入点也可以直接在微消息队列MQTT版控制台**实例详情**页面的**接入点**页签中查看。

- **云端SDK接入点格式**

  使用云端SDK接入微消息队列MQTT版时，需要填写的接入点格式如下：

  > 注意　仅实例内核版本为V3.3.0且实例地域属于中国内地的实例支持云端SDK接入。

  - **公网接入点：** *MQTT实例ID*-server-internet.mqtt.aliyuncs.com
  - **VPC 接入点：** *MQTT实例ID*-server-internal.mqtt.aliyuncs.com

> 说明　*MQTT实例ID*可在微消息队列MQTT版控制台**实例详情**页面的**基础信息**区域查看。

终端SDK接入点和云端SDK接入点同时支持**公网接入点**和**VPC 接入点**。**公网接入点**为本地公网环境访问的IP地址，一般用于物联网和移动互联网场景中；**VPC 接入点**为云上私网访问的IP地址，一般用于云端应用接入微消息队列MQTT版。

> **注意** SDK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：
>
> - 终端或云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。
> - 终端或云端网络侧对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。

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</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>
```

5. 在MQ4IoTSendMessageToMQ4IoTUseSignatureMode.java类中，按代码注释说明填写相应参数，主要涉及您已在创建资源中所创建的MQTT资源信息。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 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;
public class MQ4IoTSendMessageToMQ4IoTUseSignatureMode {
    public static void main(String[] args) throws Exception {
        /**
         * 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 = "XXXXX.mqtt.aliyuncs.com";
        /**
         * AccessKey ID，阿里云身份验证，在阿里云RAM控制台创建。
         */
        String accessKey = "XXXXX";
        /**
         * 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 = "XXXXX";
        /**
         * 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_XXXXX@@@XXXXX";
        /**
         * 微消息队列MQTT版消息的一级Topic，需要在控制台创建才能使用。
         * 如果使用了没有创建或者没有被授权的Topic会导致鉴权失败，服务端会断开客户端连接。
         */
        final String parentTopic = "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，可以填写任意字符串。
         * 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);
        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++) {
            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);
            /**
             * 微消息队列MQTT版支持点对点消息，即如果发送方明确知道该消息只需要给特定的一个设备接
收，且知道对端的clientId，则可以直接发送点对点消息。
             * 点对点消息不需要经过订阅关系匹配，可以简化订阅方的逻辑。点对点消息的Topic格式规范
是 {{parentTopic}}/p2p/{{targetClientId}}。
             */
            final String p2pSendTopic = parentTopic + "/p2p/" + clientId;
            message = new MqttMessage("hello mq4Iot p2p msg".getBytes());
            message.setQos(qosLevel);
            mqttClient.publish(p2pSendTopic, 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（终端和云端消息收发）

6.快速使用MQTT.fx模拟SDK收发消息

MQTT.fx是一款基于Eclipse Paho、使用Java语言编写的MQTT客户端。支持Windows、Mac和Linux操作系统，可用于验证设备是否可以连接微消息队列MQTT版，并通过Topic发布和订阅消息。本文以Windows系统为例，介绍如何使用MQTT.fx模拟SDK接入微消息队列MQTT版并进行消息收发。

前提条件

下载并安装MQTT.fx v1.7.1。

背景信息

微消息队列MQTT版最简单的使用场景即MQTT客户端消息的自发自收。如下图所示，您可以使用MQTT.fx作为MQTT客户端，在MQTT.fx客户端配置相关参数后接入微消息队列MQTT版实现消息的发送和接收。

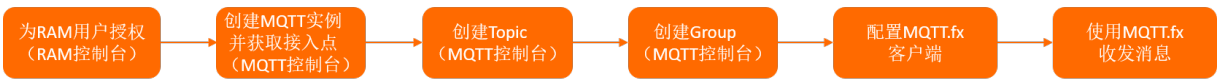
微消息队列MQTT版同时提供了**公网接入点**和**VPC 接入点**。

- 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。例如涉及部署在云端服务器上的应用场景。

本示例以**公网接入点**为例接入微消息队列MQTT版。

使用流程

使用MQTT.fx收发消息的流程如下图所示。



（RAM用户必选）步骤一：为RAM用户授权

若您开通消息队服务使用的是账号是RAM用户，必须先为RAM用户进行授权，才能访问微消息队列MQTT版的资源。若您的账号是阿里云账号，默认拥有使用微消息队列MQTT版资源的权限，您可以跳过该步骤。

1. 使用阿里云账号登录RAM控制台。
2. 在左侧导航栏，选择**身份管理 > 用户**。
3. 在**用户**页面，单击目标RAM用户**操作**列的**添加权限**。
4. 在**添加权限**面板，为RAM用户添加权限。

i. 选择授权应用范围。
   - **整个云账号**：权限在当前阿里云账号内生效。
   - **指定资源组**：权限在指定的资源组内生效。

   > **说明** 指定资源组授权生效的前提是该云服务已支持资源组。更多信息，请参见支持资源组的云服务。

ii. 输入授权主体。

   授权主体即需要授权的RAM用户，系统会自动填入当前的RAM用户，您也可以添加其他RAM用户。

iii. 选择权限策略。

   > **说明** 每次最多绑定5条策略，如需绑定更多策略，请分次操作。

5. 单击**确定**。
6. 单击**完成**。

微消息队列MQTT版提供以下系统策略，您可以根据权限范围为RAM用户授予相关权限。

权限策略名称	说明
AliyunMQFullAccess	管理微消息队列MQTT版的权限，等同于阿里云账号的权限，被授予该权限的RAM用户具有所有消息收发权限且有控制台所有功能操作权限。
AliyunMQPubOnlyAccess	微消息队列MQTT版的发布权限，被授予该权限的RAM用户具有使用阿里云账号所有资源通过SDK发送消息的权限。
AliyunMQSubOnlyAccess	微消息队列MQTT版的订阅权限，被授予该权限的RAM用户具有使用阿里云账号所有资源通过SDK订阅消息的权限。
AliyunMQReadOnlyAccess	微消息队列MQTT版的只读权限，被授予该权限的RAM用户仅有通过访问控制台或调用管控API读取资源信息的权限。

步骤二：创建MQTT实例并获取接入点

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。

步骤三：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。

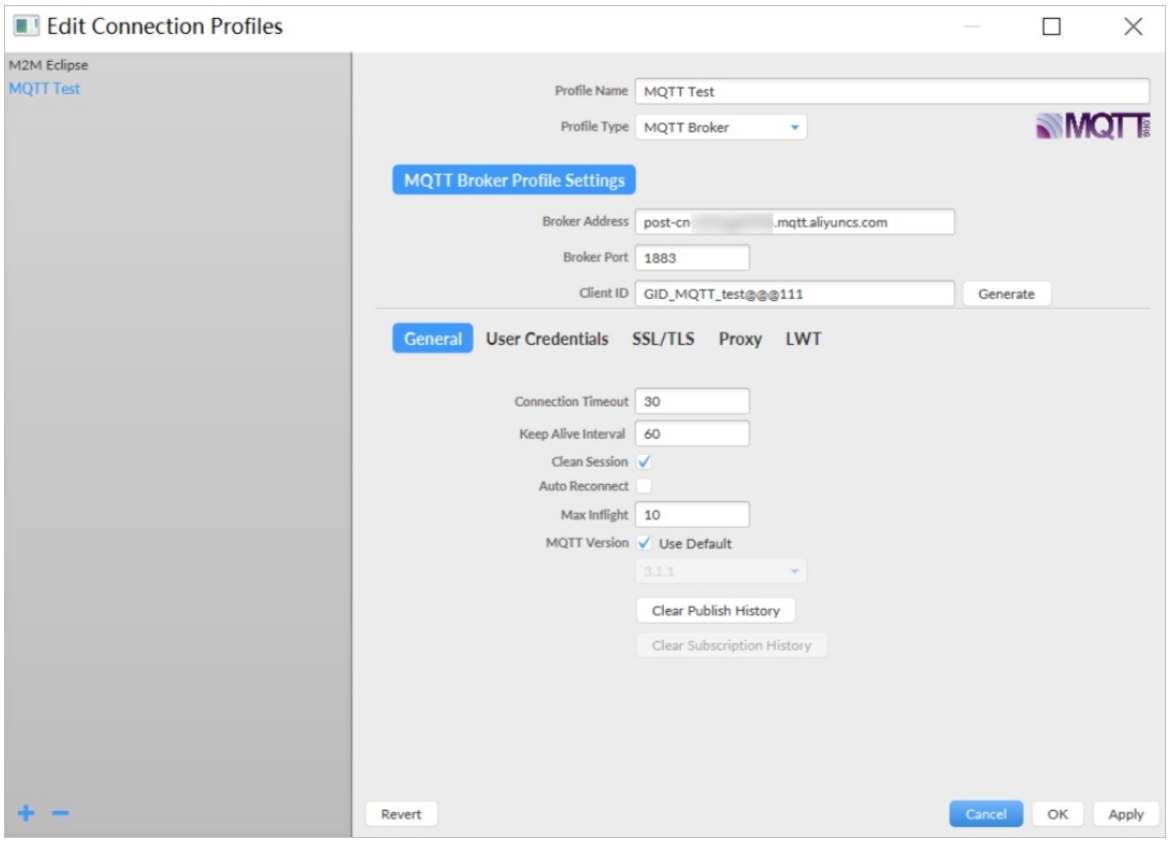
步骤四：创建Group

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。

步骤五：配置MQTT.fx接入微消息队列MQTT版

1. 打开MQTT.fx客户端，在其顶部菜单栏中选择**Extras > Edit Connection profiles**。
2. 在**Edit Connection Profiles**页面中配置相关参数，然后单击**OK**。

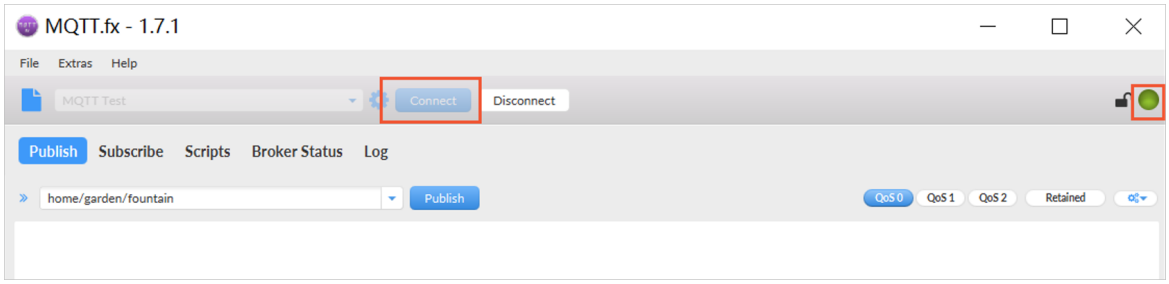


参数	说明	取值示例
Profile Name	配置文件名称。	*MQTT Test*
Profile Type	配置的连接类型。	固定选择*MQTT Broker*，表示连接MQTT服务器。

参数	说明	取值示例
MQTT Broker Profile Settings		
Broker Address	步骤二中获取的微消息队列MQTT版实例的接入点。	本示例以公网接入点为例。*post-cn-st21xgs****.mqtt.aliyuncs.com* **说明** post-cn-st21xgs****为您购买的微消息队列MQTT版实例ID。
Broker Port	连接微消息队列MQTT版的协议端口。 ○ MQTT协议端口：1883 ○ SSL端口：8883	*1883*
Client ID	客户端的唯一标识，要求全局唯一。Client ID由两部分组成，组成形式为*<GroupID>*@@@*<DeviceID>*。*<GroupID>*为步骤四中创建的Group ID，*<DeviceID>*为您自定义的设备ID。Client ID的长度限制为64个字符，不允许使用不可见字符，具体限制，请参见使用限制。	*GID_MQTT_test@@@111* 本示例中*<DeviceID>*自定义为*111*，*<GroupID>*为*GID_MQTT_test* 。
User Credentials		
User Name	客户端接入微消息队列MQTT版需要输入用户名和密码完成认证，认证通过后才能进行消息收发。用户名和密码计算方式请参见签名鉴权模式。	*Signature|LTAI4GBY9J8e7YukuUie****|post-cn-st21xgs****
Password		*p3Mxc2PDZet09sHhurTJAg3J****
SSL/TLS		
Enable SSL/TLS	是否使用SSL或TLS加密协议。	取消选中
Protocol	协议版本。	*TLSv1.2*

说明 其他参数均使用系统默认值，无需设置。您也可以根据实际场景自定义参数值。

3. 参数配置完成后，单击**Connect**进行连接。
   - 右侧绿灯亮起，表示MQTT.fx和微消息队列MQTT版已成功连接。
   - 右侧红灯亮起，表示连接失败，你可以单击**Log**查看日志，根据日志信息修改配置并重新尝试连接。

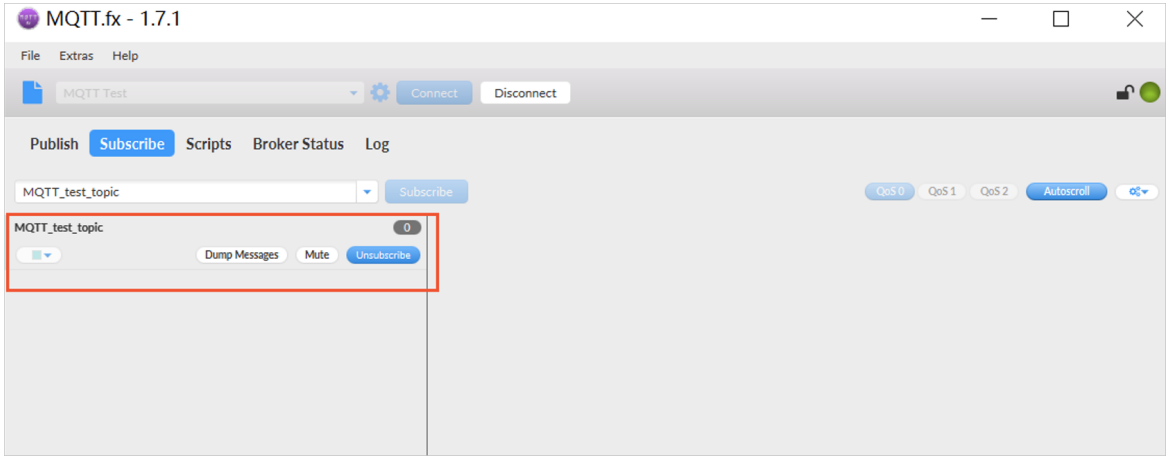


您可以在微消息队列MQTT版控制台对应实例下，查看设备状态，预期设备状态为**在线**。

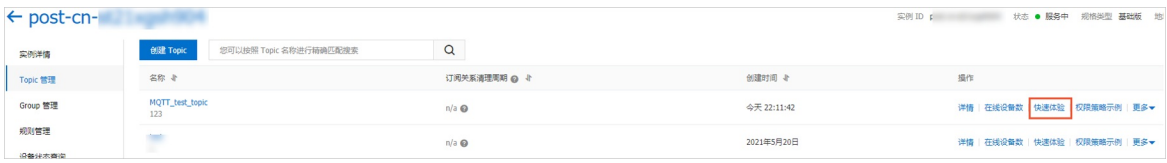
使用MQTT.fx订阅消息

1. 在MQTT.fx客户端上方单击**Subscribe**页签。
2. 在**Subscribe**页签中，在左侧Topic文本框输入步骤三中创建的Topic的名称，然后再单击文本框右侧的**Subscribe**。本示例以MQTT_test_topic为例。订阅成功后，该Topic会显示在订阅列表中。

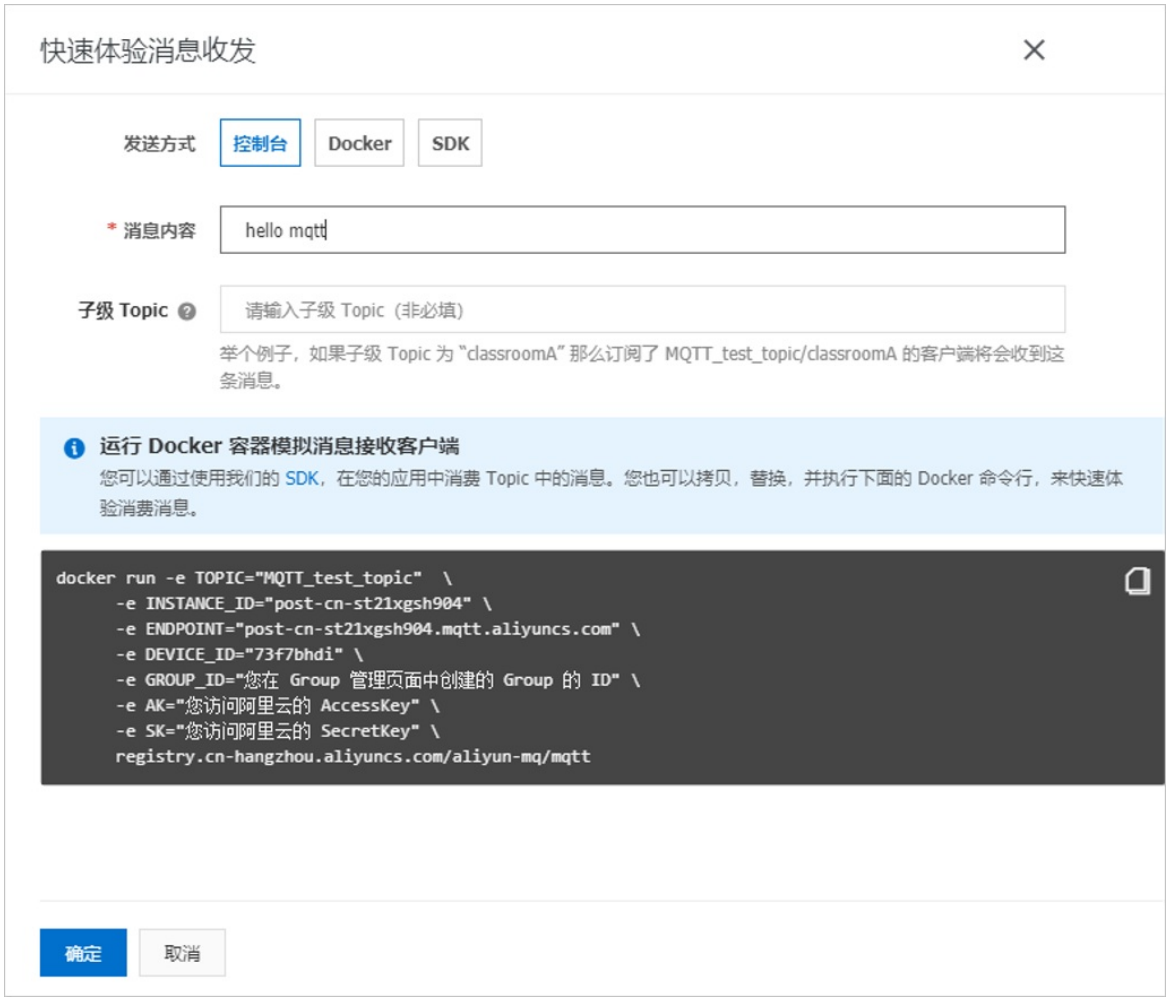
> **说明** 本文以父级Topic为例，若需要订阅子级Topic，直接在Topic文本框中父级Topic名称后加上子级Topic名称即可，父级Topic和各子级Topic间均使用正斜线（/）隔开，例如MQTT_test_topic/subscribe1。Topic详细信息，请参见名词解释。



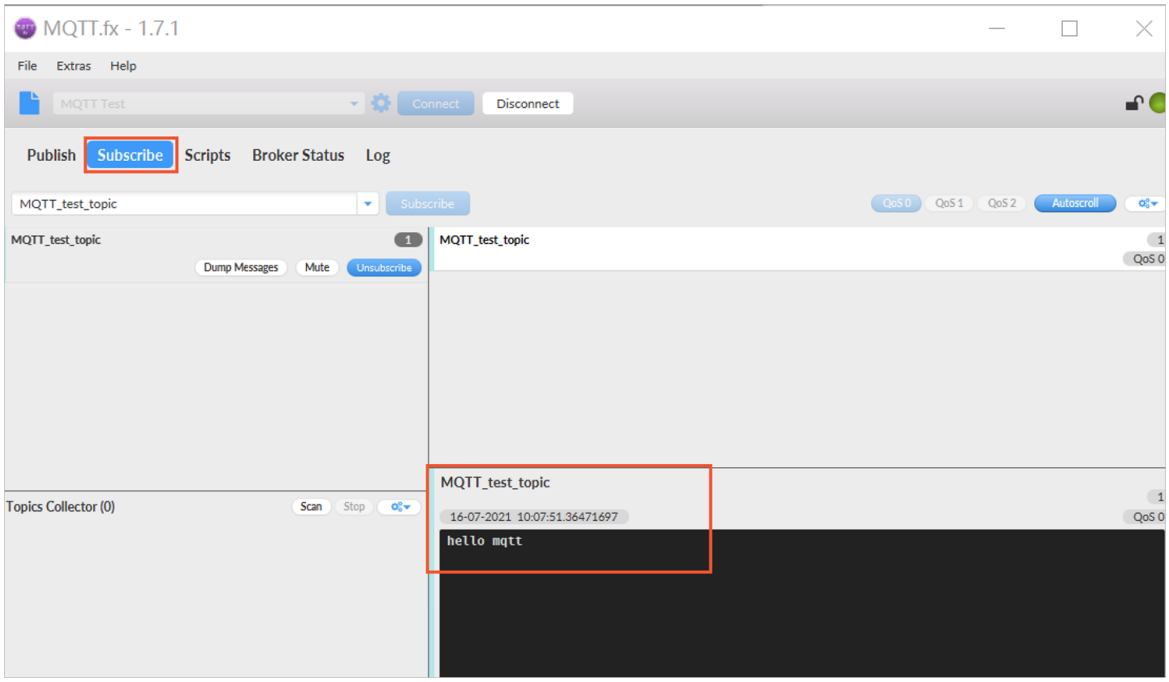
3. 在微消息队列MQTT版控制台对应实例下，单击**Topic 管理**，找到订阅的Topic，在其**操作**列单击**快速体验**。



4. 在**快速体验消息收发**面板中，选择**发送方式**为**控制台**并在**消息内容**文本框中输入要发送的消息内容，例如：hello mqtt，然后单击**确定**。

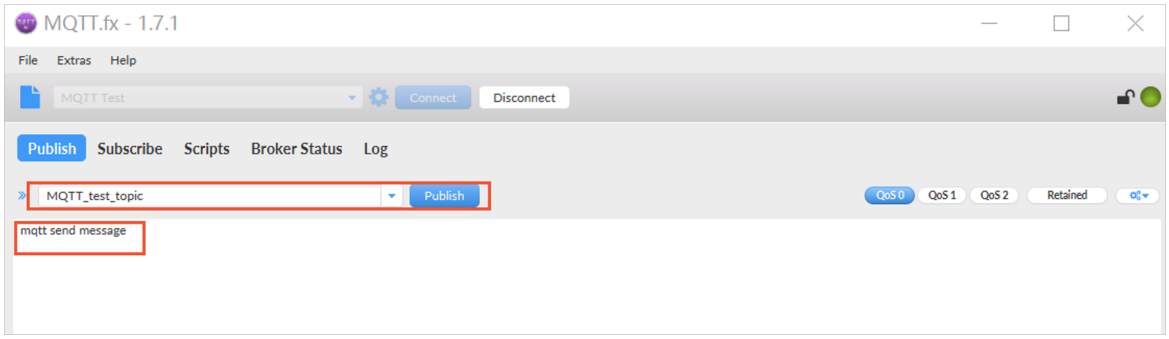


回到MQTT.fx客户端，在**Subscribe**页签中可以查看订阅的Topic接收到的消息。

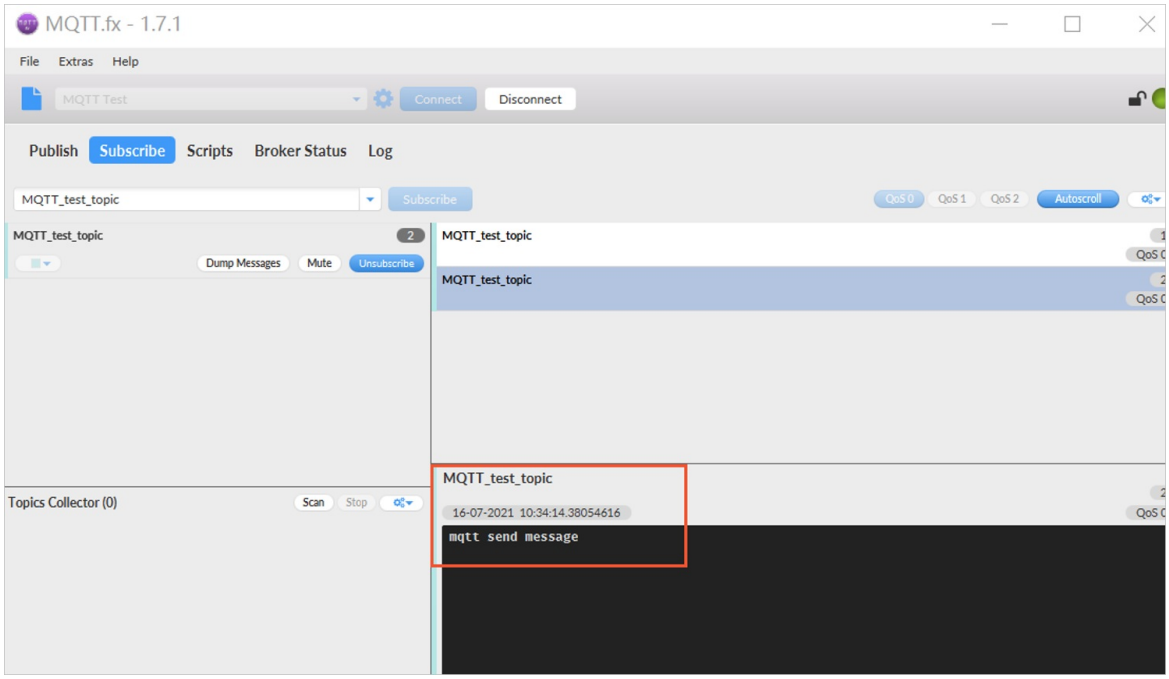


使用MQTT.fx发送消息

1. 在MQTT.fx客户端上方单击**Publish**页签。
2. 在**Publish**页签中，在左侧Topic文本框输入步骤三中创建的Topic的名称，本示例以MQTT_test_topic为例，然后在下面的消息文本框中输入要发送的消息内容，例如：mqtt send message，然后单击Topic文本框右侧的**Publish**发送消息。

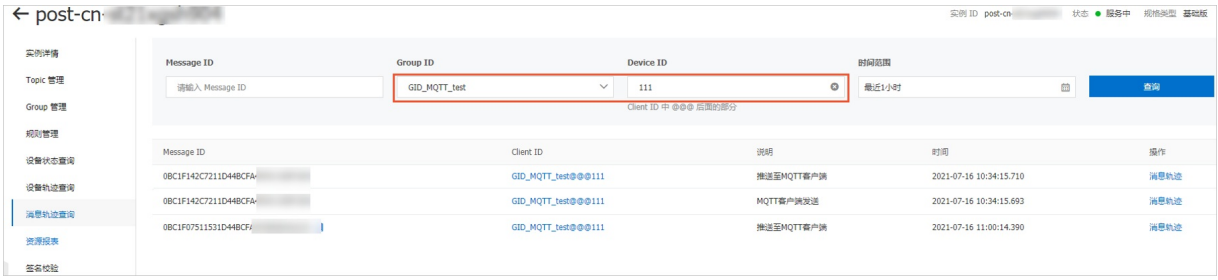


单击**Subscribe**页签，您可以查看到对应订阅的Topic已接收到刚才发送的消息。



查看消息轨迹

进入微消息队列MQTT版控制台，在**消息轨迹查询**页面输入**Group ID**和**Device ID**然后单击**查询**。



返回3条消息轨迹。从下至上依次为：

- 第1条：MQTT微消息队列MQTT版控制台发送到MQTT.fx的消息。

- 第2条：MQTT.fx客户端发送的消息。
- 第3条：推送至MQTT.fx的消息。第2条和第3条轨迹的Message ID相同，即通过MQTT.fx完成了同一条消息的自收发。

查看日志

在MQTT.fx中，单击Log页签可查看操作日志和错误日志。