



微消息队列MQTT版 快速入门

文档版本: 20220524



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。

- 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。
- 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。
- 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 <i>,</i> 是用户必须 了解的内容。	大意 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。	⑦ 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 listinstanceid
[] 或者 [alb]	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	05
2.开通服务并授权	06
3.创建资源	08
4.快速使用MQTT的Java SDK收发消息(终端和云端消息收发)	10
5.快速使用MQTT的Java SDK收发消息(终端和终端消息收发)	16
6.快速使用MQTT.fx模拟SDK收发消息	21

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
 - o 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。
- 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。

三()阿里云)	☆ 工作	台 📑 华东2(上海) 🗸		Q 搜索	费用	工单	ICP 备案	企业	支持	▶_	Ō	?	简体	0
微消息队列 MQTT 版 概览		 ●●●● 您正在使用阿里云微消息 微消息队列 MQTT 版 / 概览 	息队列 MQTT 版新版 UI,您可以点起	击右侧的"返回旧版"按钮,返回之前的 U	JI.								×	
实例列表 文档 SDK 文档 ^[2]	^		您尚未开通微消息队列 微消息队列 MQTT 版是专为移动!	J MQTT 版 ^{互联网(MI)、物联网(IoT)领域设计}	快捷	入口 ocketMQ	控制台							
帮助文档 🖸	<		的渦息产品,覆盖互动直播、金器 Apps、智能设备、车联网等多种胚 WebSocket 等协议的全面支持, 注 C2C、C2B、B2C 等业务场景之间 消息并发。	愈好,智能餐饮、即时聊天、移动 並用场景;通过对 MQTT、 生核端和云之间的双向通信,实现 的消息通信,可支撑干万级设备与	常用 产品:	链接 介绍	帮助文档	á	快速入I	כי	SDI	ĸ		
		[免费开通 视频介绍]	产品: API]	定价 文档	场景方象	名 列群	Demo _	工程	常贝]问题		

6. 在服务开通面板,选中消息队列Rocket MQ版服务协议并单击立即开通。

⑦说明 微消息队列MQTT版是
消息队列Rocket MQ版
系列产品之一,开通
消息队列Rocket MQ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。

⑦ 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。
- 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。

							前往购买页 🖸	\times
连接数上限	100个	17	Ŷ	5 1 -	\uparrow	1万个		
	2万个	5万~	\uparrow	10万	` 个			
	同时在线连接数:任一时刻在 计算规则参考链接 >>	单个实例上存在的	的客户端 TCP 追	主接的数量 。				
消息TPS上限	100条/秒	500条	/秒	1千条	/秒	3千条/秒		
	5千条/秒	1万条	/秒	2万条	/秒			
	消息收发TPS:指每秒钟使用很 计算规则参考链接 >>	說消息队列支持的	1协议,进行上	行数据发送和下	行数据接收的》	肖息总条数。		
订阅关系数上限	1千个	1万~	\uparrow	5万~	个	10万个		
	20万个	50万	\uparrow	100万	亦个			
	订阅关系数:用户在微消息队 计算规则参考链接 >>	列服务端注册保留	留的订阅规则的	数量。				
购买数量	- <u>1</u> +							
购买时长	1个月 2个月	3个月	4个月	5个月	6个月	更多时长 ▼		
	到期自动续费							
服务协议	✓ 微消息队列 MQTT 版 (包	年包月) 服务协议	×					
					总配置费用		立即始至	
							立即购关	

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。

数据统计 接入点 SDK 中新配置接入点地址。点	油設置了解異体例用力式。					
网络	Endpoint	标准协议跳口	SSL \$#□	WebSocket 第日	WebSocket SSL/TLS 第日	Flash 第日
公网接入点 🚱	post-onmqtt.allyuncs.com	1883	8883	80	443	843
VPC 接入点 😡	post-cn	1883	8883	80	443	843

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<mark>JDK</mark>。

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

⑦ 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。

<depende:< th=""><th>ncles></th></depende:<>	ncles>
	<dependency></dependency>
	<groupid>commons-codec</groupid>
	<artifactid>commons-codec</artifactid>
	<version>1.10</version>
	<dependency></dependency>
	<proupid>org.eclipse.paho</proupid>
	<pre><artifactid>org.eclipse.paho.client.mqttv3</artifactid></pre>
	<version>1.2.2</version>
	<dependency></dependency>
	<proupid>org.apache.httpcomponents</proupid>
	<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>
<td>encies></td>	encies>

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.mgttv3.MgttCallbackExtended;
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MOTT版的实例ID。
       */
      String instanceId = "XXXXX";
       /**
       * 设置终端SDK的接入点,进入微消息队列MOTT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MOTT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 gosLevel = 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() { QOverride 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(); } QOverride 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
*/
         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>
</dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependency></dependen
```

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

* 设直云端SDK的接入点,请参见接入点况明甲的云端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 ConsumerC onfig()); serverConsumer.start(); serverConsumer.subscribeTopic(firstTopic, new MessageListener() { QOverride public void process (String msgId, MessageProperties messageProperties, byte [] payload) { System.out.println("Receive:" + msgId + "," + JSONObject.toJSONString(m essageProperties) + "," + 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<mark>JDK</mark>。

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D*-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

⑦ 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-javademo。
- 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.mgttv3</artifactId>
            <version>1.2.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>4.5.2</version>
        </dependency>
        <dependency>
            <proupId>com.alibaba</proupId>
            <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.lmg.example.demo;
import com.aliyun.openservices.lmg.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.mgttv3.IMgttDeliveryToken;
import org.eclipse.paho.client.mgttv3.MgttCallbackExtended;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mgttv3.MgttException;
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
0
       */
       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() { QOverride 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(); } }); } QOverride 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(mgttMess 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

    {{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<mark>消息轨迹查询</mark>。

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。

⑦ 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。

⑦ 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。

							前往购买页 🖸 💙
连接数上限	100个	17	\uparrow	5千/	\uparrow	1万个	
	2万个	5万·	\uparrow	10万	` 个		
	同时在线连接数:任一时刻在! 计算规则参考链接 >>	单个实例上存在的	的客户端 TCP 追	生接的数量。			
消息TPS上限	100条/秒	500条	/秒	1千条	/秒	3千条/秒	
	5千条/秒	1万条	/秒	2万条	/秒		
	消息收发TPS:指每秒钟使用微 计算规则参考链接 >>	m消息队列支持的	动协议,进行上	行数据发送和下行	行数据接收的演	肖息总条数。	
订阅关系数上限	1千个	1万·	\uparrow	5万~	\uparrow	10万个	
	20万个	50万	个	100万	亦个		
	订阅关系数:用户在微消息队3 计算规则参考链接 >>	列服务端注册保留	留的订阅规则的)数量。			
购买数量	- <u>1</u> +						
购买时长	1个月 2个月	3个月	4个月	5个月	6个月	更多时长 🗸 🔻	
	到期自动续费						
服务协议	✓ 微消息队列 MQTT 版 (包)	年包月)服务协	īχ.				
	L						
					白配響進田		-)
					404 HUEL92/T		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。

数据统计 接入点						
SDK 中所配置接入点地址。点	a击这里了解具体使用方式。					
网络	Endpoint	标准协议跳口	SSL \$R□	WebSocket \$#	WebSocket SSL/TLS 第□	Flash 第日
公网接入点 🔘	post-on	1883	8883	80	443	843
VPC 接入点 😡	post-cn	1883	8883	80	443	843

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。

Edit Connection Profiles						\times
M2M Eclipse MQTT Test	Profile Name Profile Type		s M	<u>) </u>		
	MQTT Broker Profile Settings Broker Address	post-cn .	nqtt.aliyuncs.com			
	Broker Port	1883				
	Client ID	GID_MQTT_test@@@	111	Generat	e	
	General User Credentials	SSL/TLS Proxy	LWT			
	Connection Timeout	30				
	Keep Alive Interval	60				
	Clean Session Auto Reconnect	~				
	Max Inflight	10				
	MQTT Version	✓ Use Default				
		Clear Publish History				
		Clear Subscription Hi	story			
+ -	Revert			Cancel	ОК	Apply
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 <i>MQT</i> MQTT服务器。	T Broker	, 表示:	连接

微消息队列MQTT版

快速入门·快速使用MQTT.fx模拟SDK 收发消息

参数	说明	取值示例				
MQTT Broker Profile Settings						
		本示例以公网接入点为例。 <i>post-c</i> <i>n-st21xgs****.mqtt.aliyuncs.co</i> <i>m</i>				
Broker Address	步骤二中获取的微消息队列MQTT 版实例的接入点。	⑦ 说明 post-cn- st21xgs****为您购买的微消 息队列MQTT版实例ID。				
	连接微消息队列MQTT版的协议端					
Broker Port	□。 • 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。</deviceid></groupid></deviceid </groupid>	GID_MQTT_test@@@111 本示例中 <i><deviceid< i="">>自定义 为<i>111, <groupid< i="">>为GID_MQTT_ test</groupid<></i></deviceid<></i>				
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	Signature LTAI4GBY9J8e7YukuUi e**** post-cn-st21xgs****				
Password	在通过后才能进行,有急收及。用广 名和密码计算方式请参见 <mark>签名鉴权</mark> 模式。	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				

⑦ 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。

🌚 MQTT.fx - 1.7.1		- 🗆 X
File Extras Help		
MQTT Test	Connect Disconnect	
Publish Subscribe Scripts	Broker Status Log	
» home/garden/fountain	▼ Publish	QoSO QoS1 QoS2 Retained OFV

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。

← post-cn-	1×p#1904						
实例详情	* Group ID		* Device ID				
Topic 管理	GID_MQTT_test	~	111	۵	查询		
Group 管理			Client ID 中 @@@ 后面的部分	÷			
规则管理	设备状态						
设备状态查询	Client ID						
设备轨迹查询	Client ID	GID_MQ11_test@@@1				在現状态	 任 後
消息轨迹查询	客户端地址	/113.140.				cleanSession 🕜	見
资源报表	最后更新时间	2021年7月15日21:59:20	6				

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。

⑦ 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。

WQTT.fx - 1.7.1	-	_		\times
File Extras Help				
MQTT Test V 😵 Connect Disconnect				• •
Publish Subscribe Scripts Broker Status Log				
MQTT_test_topic Qos0	QoS1 0	QoS 2	Autoscroll	0;*
MQTT_test, topic				

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。

← post-cn-	2494-904			实例 ID ; 状态● 服务中 缆格关型 基础板 地
实例详情	创建 Topic 您可以按照 Topic 名称进行精确匹配搜索	Q		
Topic 管理	名称 辛	订阅关系清理周期 😡 🕴	创建时间,小	損作
Group 管理	MQTT_test_topic 123	n/a 😡	今天 22:11:42	洋橋 在线设备数 快速体验 权限策略示例 更多▼
規則管理		n/a 😧	2021年5月20日	洋橋 在线设备数 快速体验 权限策略示例 更多▼

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。

快速体验消息	奴发 ×
发送方式	控制台 Docker SDK
* 消息内容	hello mqtt
子级 Topic 👔	请输入子级 Topic(非必填)
	举个例子,如果子级 Topic 为 "classroomA" 那么订阅了 MQTT_test_topic/classroomA 的客户端将会收到这条消息。
() 运行 Docke 您可以通过使用 验消费消息。	r 容器模拟消息接收客户端 用我们的 SDK,在您的应用中消费 Topic 中的消息。您也可以拷贝,替换,并执行下面的 Docker 命令行,来快速体
docker run -e TC -e INSTANC -e ENDPOIM -e DEVICE_ -e GROUP_I -e AK="您说 -e SK="您说 registry.c	PIC="MQTT_test_topic" \
确定 取消	

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。

🌚 MQTT.fx - 1.7.1		_	\Box \times
File Extras Help			
MQTT Test	Connect		-
Publish Subscribe Scripts Broker Status Lo	g		
MQTT_test_topic	Subscribe	QoS1 QoS1 QoS2	Autoscroll 08-
MQTT_test_topic Dump Messages Mute Unsu	MQTT_test_topic		1 QoS 0
Topics Collector (0) Scan Stop	MQTT_test_topic		1
	hello mqtt		2000

使用MQTT.fx发送消息

- 1. 在MQTT.fx客户端上方单击Publish页签。
- 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。

1.7.1 WQTT.fx - 1.7.1	_	
File Extras Help		
MQTT Test Connect Disconnect		-
Publish Subscribe Scripts Broker Status Log MQTT_test_topic Publish	Qo50 Qo51 Qo52	Retained 0°
mqtt send message		

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。

🜚 MQTT.fx - 1.7.1	1						\times
File Extras Help							
MQTT Test		•	Disconnect				•
Publish Subscribe	Scripts Broker Statu	s Log					
MQTT_test_topic		Sub:	cribe	Qo50 Qo51	QoS 2	Autoscroll	0 °▼
MQTT_test_topic	Dumo Messages Mute	2 Unsubscribe	MQTT_test_topic				QoS C
	Dump Messages Mate	Unsubscribe	MQTT_test_topic				QoS C
Tania Callestar (0)	5	Chan all	MQTT_test_topic				2
Topics Collector (0)	Stall	Stop	16-07-2021 10:34:14.38054616				QoS C
			mqtt send message				

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。

← post-cn	1001004				案例 ID post-cn-	状态 ● 服务中	规格类型 基础版
实例详情	Message ID	Group ID	Device ID		时间范围		
Topic 管理	请输入 Message ID	GID_MQTT_test	× 111	٥	最近1小时	m	查询
Group 管理			Client ID 中 @@@ 后面的部分				
规则管理							
设备状态查询	Message ID	c	lient ID	说明	时间		操作
没备外达赛次	0BC1F142C7211D44BCFA	c	SID_MQTT_test@@@111	推送至MQTT客户端	2021-07-16 10:34:15.710		浦思轨迹
(2回4)(2000)	0BC1F142C7211D44BCFA	0	iD_MQTT_test@@@111	MQTT客户颁发送	2021-07-16 10:34:15.693		消息轨迹
满意轨迹查询	0BC1F07511531D44BCF/	0	BID_MQTT_test@@@111	推送至MQTT春户端	2021-07-16 11:00:14.390		满思轨迹
资源报表							
等实检验							

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。

MQTT Test Connect Disconnect	•
Publish Subscribe Scripts Broker Status Log	
2021-07-1610:01:23,207INF0 MqttFX ClientModel: adtRecentSubscriptionTopic : de jensd.mqtfx.entities.Topic@Sd6aafa02021-07-1610:01:23,207INF0 MqttFX ClientModel: adtRecentSubscriptionTopic : de jensd.mqtfx.entities.Topic@Sd6aafa02021-07-1610:01:23,211INF0 MqttFX ClientModel: sucessfully subscribed to topic (MQTT_test_topic (QOS 0)2021-07-1610:07:51,654INF0 MqttFX ClientModel: sucessfully subscribed to topic (MQTT_test_topic (2010)2021-07-1610:20:43,155INF0 MqttFX ClientModel: messageArrived() with topic: MQTT_test_topic (MQTT_test_topic (2020)2021-07-1610:20:43,155INF0 MqttFX ClientModel: rebuildMessagesList()2021-07-1610:20:43,155INF0 MqttFX ClientModel: sucessfully unsubscribed from topic: MQTT_test_topic2021-07-1610:20:56,157INF0 MqttFX ClientModel: sucessfully unsubscribed2021-07-1610:21:64,262INF0 MqttFX ClientModel: sucessfully unsubscribed from topic: MQTT_test_topic (QOS 0, Retai2021-07-1610:21:64,362INF0 MqttFX ClientModel: attempt to addPublishTopic2021-07-1610:21:64,362INF0 MqttFX ClientModel: attempt to addPublishTopic2021-07-1610:21:64,365INF0 MqttFX ClientModel: attempt to addPublishTopic2021-07-1610:21:64,365INF0 MqttFX ClientModel: attempt to addPublishTopic2021-07-1610:21:64,365INF0 MqttFX ClientModel: attempt to	ned: fa' , Retain (QoS 0