目录

[课程内容 2](#_Toc522878564)

[课程目标 2](#_Toc522878565)

[课程准备 2](#_Toc522878566)

[课程知识点 3](#_Toc522878567)

[云端接收设备数据 3](#_Toc522878568)

[云端调用API 4](#_Toc522878569)

[云端接收设备数据Demo 5](#_Toc522878570)

[创建产品，添加设备 6](#_Toc522878571)

[配置服务端订阅 6](#_Toc522878572)

[MNS Demo配置 9](#_Toc522878573)

[发送消息，测试接收 11](#_Toc522878574)

[云端调用API Demo 13](#_Toc522878575)

[引入SDK 13](#_Toc522878576)

[初始化参数 14](#_Toc522878577)

[发起调用 15](#_Toc522878578)

[课程小结 17](#_Toc522878579)

**云端开发指南**

在基于阿里云物联网平台开发物联网应用的过程中，应用*服务*器需要通过平台及时获取*设备*的消息。同时，在业务场景中，用户不可能在每次需要添加*产品*、*设备*或下发消息时都手动在控制台上完成，因此*服务*器需要能够执行创建*产品*和*设备*，向*设备*下发指令等操作。本次课程将向读者介绍应用开发场景下*服务*器接入物联网平台并进行相关操作的方法与流程。

**课程内容**

* 介绍云端接收*设备*数据的方法
* 介绍物联网平台的*服务*器端API
* 结合实例介绍云端开发的操作流程

**课程目标**

* 掌握云端接收*设备*数据的方法
* 了解物联网平台的*服务*器端API
* 掌握云端接入物联网平台，执行相关操作的实现方法

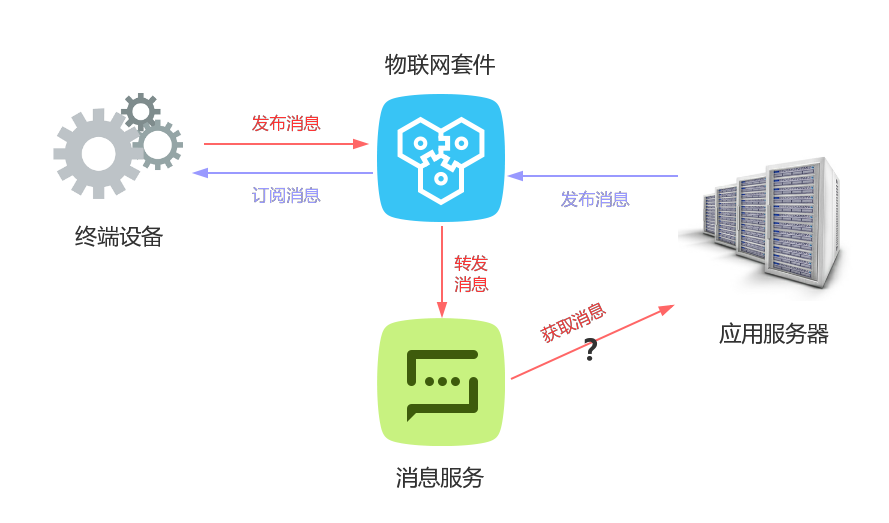
**课程准备**

* 运行Windows系统的 PC机一台
* AIoTKIT开发板开发板一块
* Mk3080 wifi模组
* micro USB连接线
* 安装有AliOS Things Studio插件的VSCode
* AliOS Things 1.3.3版本或者更高版本
* ST-Link驱动程序
* 开通阿里云物联网平台*产品*
* 开通消息*服务产品*
* 配置好Java JDK的IntelliJ IDEA

**课程知识点**

**云端接收设备数据**

*设备*接入到阿里云物联网平台以后，可以基于MQTT协议与平台进行发布/订阅机制的通信。在实际业务场景中，用户的*服务*器端可能需要通过物联网平台快速地接收*设备*发送的业务消息或者*设备*上下线的通知消息，此时就需用户*服务*器接入物联网平台。



物联网平台配合阿里云的消息*服务*（MNS）*产品*可以实现*服务*端订阅*设备*消息的功能。MNS的使用还可以保证消息的可靠性，避免*服务*端不可用时的消息丢失；同时MNS在处理大量消息并发时还有削峰填谷的作用，避免突然的并发压力导致*服务*端不可用。用户*服务*器端通过MNS获取*设备*数据的方式主要有两种：

* 使用MNS队列

用户简单配置后，物联网平台会将*设备*的数据写入MNS队列中，用户*服务*端从队列中即可获取数据。队列模型支持一对一发送和接收消息。

* 使用MNS主题

用户还可以使用物联网平台的*规则引擎*组件对*设备*上报到平台的数据进行解析处理，之后再转发到MNS主题中，用户*服务*端从主题中获取*设备*数据。主题模型支持一对多发布和订阅消息，并且支持多种消息推送方式。

除上述两种使用MNS的方式之外，用户还可以通过*规则引擎*将*设备*数据转发到阿里云的消息队列（MQ）*产品*中，用户*服务*端从消息队列中同样可以获取到数据，MNS与MQ的简要对比如下：

|  |  |
| --- | --- |
| *产品* | 特点 |
| 消息*服务*MNS | 具备更为强大的扩展*服务*功能，如短信、邮件功能 |
| 消息队列MQ | 适合并发高、延时要求低的消息传输场景 |

**云端调用API**

当用户需要将自己的业务系统对接到物联网平台，实现例如通过自己的*服务*器在平台上创建*产品*、*设备*或者向*设备*发送消息等功能时，就需要调用*服务*器端API与物联网平台互通。对API接口的调用是指向API的*服务*端地址发送HTTPS/HTTP GET/POST请求，并且按照API的接口说明在请求中加入相应的参数；根据请求的处理情况，系统会返回相应的处理结果。

GET/POST请求的结构为：http://Endpoint/?Action=xx&Parameters。

* Endpoint：指调用的云*服务*的接入地址。在物联网平台上创建*产品*和*设备*时选择的区域不同，IoT API的*服务*端接入地址也不相同：
  + 华东2（上海）：iot.cn-shanghai.aliyuncs.com，
  + 新加坡：iot.ap-southeast-1.aliyuncs.com，
  + 美国（硅谷）：iot.us-west-1.aliyuncs.com，
  + 日本（东京）：iot.ap-northeast-1.aliyuncs.com，
  + 德国（法兰克福）：iot.eu-central-1.aliyuncs.com。
* Action：指要执行的操作，如使用Pub接口向指定*Topic*发布消息。
* Parameters：指请求参数，由调用每个API时都需要使用的公共请求参数和各个API的自定义参数组成，每个参数之间用&分隔；其中，公共参数中包含API版本号、身份验证等信息。

以调用Pub接口向指定*Topic*发布消息为例，该API的请求参数为：

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 是否必须 | 描述 |
| Action | String | 是 | 要执行的操作，Pub |
| *ProductKey* | String | 是 | 要发送消息的*产品*Key |
| *Topic*FullName | String | 是 | 要接收消息的*Topic*全称，不支持系统*Topic*类 |
| MessageContent | String | 是 | 要发送的消息主体 |
| QoS | Integer | 否 | 指定消息的发送方式。取值为0是至多发送一次，取值为1时至少发送一次。默认值为0 |

返回参数为：

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 描述 |
| RequestId | String | 阿里云为该请求生成的唯一标识符 |
| Success | Boolean | 表示是否调用成功。true表示调用成功，false表示调用失败 |
| ErrorMessage | String | 调用失败时，返回的出错信息 |
| MessageId | String | 成功发送消息后，云端生成的消息ID，用于标识该消息 |

目前物联网平台提供的API完整列表和各API的参数信息可通过链接：https://help.aliyun.com/document\_detail/69893.html进行查阅。

物联网平台还为开发者提供了几种较为普遍的语言对应的SDK，如JAVA、PHP、Python和.NET，用户可以通过链接：https://help.aliyun.com/document\_detail/30579.html查看各个语言Demo的下载地址与使用说明。

**云端接收设备数据Demo**

本实验主要介绍云端如何通过MNS队列来获取*设备*的数据：用户只需要在LP控制台上为*产品*配置“*服务*端订阅”功能，即可将消息自动转发到MNS队列中，云端应用通过监听MNS队列即可简单快速地接收*设备*的消息。

**创建产品，添加设备**

登录物联网平台控制台https://iot.console.aliyun.com，点击“*产品*管理”栏，在“华东2（上海）”区域下新建一个基础版*产品*“MNS\_TEST”（目前*服务*端订阅功能只有基础版*产品*具备），节点类型选择为“*设备*”，并在该*产品*下添加test*设备*。





**配置服务端订阅**

点击创建好的MNS\_TEST*产品*右侧的“查看”按钮，进入所创建的*产品*的详情页。点击 “*服务*端订阅”栏目，可以看到“配置*服务*端订阅”相关信息。



点击“配置”按钮即可进行*服务*端订阅功能的配置。由于该功能会用到消息*服务*MNS*产品*，若以前没有开通MNS*产品*，按照网页提示开通即可。成功开通*服务*后，用户可以根据业务需求选择要向消息*服务*中推送的消息类型，具体包括*设备*上报消息和*设备*状态变化通知消息两类。



* *设备*上报消息

所有具有发布权限的*Topic*中的消息均为“*设备*上报消息”。例如一个*产品*有三个*Topic*类：/pk/${*deviceName*}/get（权限为订阅）、/pk/${*deviceName*}/update（权限为发布）、/pk/${*deviceName*}/update/error（权限为发布）。那么*设备*上报消息指的是pk/${*deviceName*}/update和/pk/${*deviceName*}/update/error对应的该*产品*下所有*设备Topic*中的消息。

* *设备*状态变化通知

当该*产品*下*设备*的状态发生变化，例如上线，下线时，物联网平台会产生*设备*状态变化的通知消息。

消息类型选择完毕后，物联网平台会自动在MNS的“华东2”区域（即创建当前*产品*时所选择的区域）下创建aliyun-iot-${*productKey*}队列，更新配置后大概1分钟后生效，系统会将选中类型的消息推送到该MNS队列里。



物联网平台在将*设备*上报消息和*设备*状态变化通知转发到MNS队列中之前，还会对这些数据进行一层封装。

|  |
| --- |
| {  “messageid”: “12345”,  "messagetype": "status/upload",  "*Topic*": "null/*Topic*",  "payload": {data},  "timestamp": 1469564576  } |

* messageid: IoT平台生成的消息ID，大小为64位;
* messagetype:消息类型，目前包括*设备*状态通知和*设备*上报消息，分别对应status和upload；
* *Topic*: 该消息源自平台中的哪个*Topic*，当messageType=status时，*Topic*为null，当messageType=upload时，*Topic*为具备的*设备Topic*，例如/pk/mydevice/update；
* payload: 用户数据经过Base64编码后的数据，当messageType=status时，数据是平台产生的*设备*上下线通知数据；

|  |
| --- |
| data=  {  "status": "online"(或offline), //*设备*状态  "*productKey*": "xxxxxxxxxxx", //*产品*标识  "*deviceName*": "xxxxxxxxxx", //*设备*标识  "time": "2016-8-11 10:11:12.234", //发送通知时间点  "lastTime": "2016-8-11 10:11:12.123" //状态变更时最后一次通信时间  "clientIp": "xxx.xxx.xxx.xxx" //*设备*端公网出口IP  } |

当messageType=upload时，data即为*设备*发布到*Topic*中的原始数据；

* timestamp:时间戳，以Epoch时间表示。

**MNS Demo配置**

本实验中，我们使用阿里云提供的用户*服务*端接入MNS的Java版本Demo，使用queue模式订阅消息*服务*的队列，以获取*设备*的数据。

* 配置pom依赖与相关参数

从链接：https://help.aliyun.com/document\_detail/27508.html中下载最新版本的MNS Java sample。



解压得到Maven工程，用IntelliJ IDEA软件找到存放该Demo的目录并打开。



然后，在用户目录（“C:\Users\YOURNAME” in Windows）中创建名为“.aliyun-mns.properties”的文件（注意最前面有一个“.”），并填写你的MNS队列对应的*服务*地址、AccessKeyID和AccessKeySecret。

|  |
| --- |
| mns.accountendpoint=http://$accountid.mns. cn-shanghai.aliyuncs.com/  mns.accesskeyid=$your\_accesskeyid  mns.accesskeysecret=$your\_accesskeysecret |

这三个参数可以在消息*服务*控制台（https://mns.console.aliyun.com）上获取：选择正确的队列区域如“华东2”后，点击右上角“获取Endpoint”，其中的公网Endpoint即为MNS队列对应的*服务*地址。注意，将Endpoint复制过去之后，需要将“http(s)”修改为“http”或者“https”。



点击右上角头像栏下的“accesskeys”，即可获取到目前账户所对应的Access Key ID和Access Key Secret。其中，Access Key ID 用于标识访问者身份；Access Key Secret 是用于加密签名字符串和*服务*器端验证签名字符串的密钥，必须严格保密。



文件创建成功后，在IDEA中将src\main\java\com\aliyun\mns\sample\Queue目录下的ComsumerDemo.java文件中的第22行处getQueueRef函数的参数换成刚才IoT自动创建的队列名。

|  |
| --- |
| 21 try{  22 CloudQueue queue = client.getQueueRef("aliyun-iot-\*\*\*");  // replace with your queue name |

**设备端开发**

本实验中，我们首先需要让*设备*向平台上报消息，平台将这些消息转发到MNS队列中后，然后我们才能测试MNS Demo是否能成功地从MNS队列中获取消息。此处只需要*设备*向平台上报消息即可，具体的消息内容不产生影响，因此我们采用与上一次课程中“上报温湿度数据到钉钉群机器人”实验相同的*设备*端代码，让AIoTKIT开发板上报温湿度数据。

更改代码中的三元组信息后，编译工程，之后将代码下载到开发板当中，并复位运行，通过指令连接网络上报数据。

**测试接收**

依照我们对*服务*端订阅功能的配置，此时*设备*上报的消息和*设备*状态变化通知消息都会被转发到MNS队列中。当*设备*发送数十条温湿度消息后刷新页面，我们可以看到队列的活跃消息数增加，运行demo前的活跃消息数为13。



启动ConsumerDemo程序，进行对MNS队列中消息的消费，可见队列活跃消息数减少10条。这是因为ConsumerDemo程序中的for(int i=0;i<10;i++)循环语句，本Demo实现的功能是消费并删除队列中的10条消息。



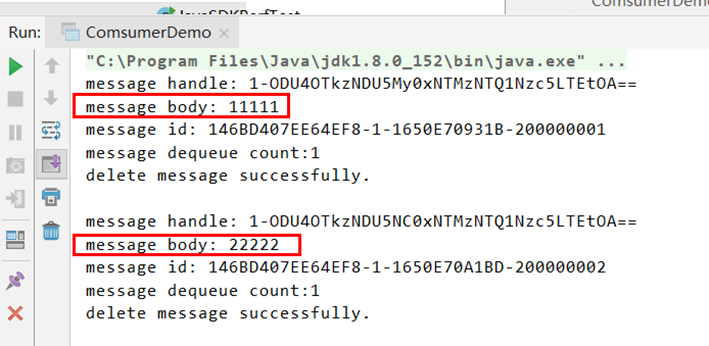
同时，在IDEA控制台上可以看到打印出的所消费消息的信息。



其中，message body即为经过物联网平台封装后发送到MNS队列中的*设备*消息内容，payload为经过Base64编码后的*设备*数据，本实验代码中未进行解码因此看到的不是正常的温湿度数据，用户可自行修改代码解析payload的数据内容。

我们还可以通过在消息队列控制台上找到我们的队列，点击最右侧的“发送消息”，手动模拟快速向该队列中发送消息进行测试，但此时的消息流转链路只包括MNS队列→*服务*端，而不包括*设备*→平台→MNS队列的过程。





其中message body即为我们在消息*服务*控制台上发送的消息内容，可见*服务*器接收正常。

**云端调用API Demo**

物联网平台为我们提供了多种语言版本的*服务*端API使用SDK，如JAVA、PHP、Python和.NET，本实验主要介绍Java版本云端SDK的使用方法。

**引入SDK**

下载Java SDK（链接：https://github.com/aliyun/aliyun-openapi-java-sdk/tree/master/aliyun-java-sdk-iot），用Intelli IDEA导入此maven工程。



在工程的pom.xml文件中添加依赖项：

|  |
| --- |
| <!-- https://mvnrepository.com/artifact/com.aliyun/aliyun-java-sdk-iot -->  <dependency>  <groupId>com.aliyun</groupId>  <artifactId>aliyun-java-sdk-iot</artifactId>  <version>4.0.0</version>  </dependency>  <dependency>  <groupId>com.aliyun</groupId>  <artifactId>aliyun-java-sdk-core</artifactId>  <version>3.2.10</version>  </dependency> |

**初始化参数**

在src\main\java\com\aliyun\iot\client目录下的IotClient.java文件中设置工程运行需要的参数。将文件第38-47行(即try后大括号内的内容)替换为：

|  |
| --- |
| String accessKey = "<your accessKey>";  String accessSecret = "<your accessSecret>";  DefaultProfile.addEndpoint("cn-shanghai", "cn-shanghai", "Iot", "iot.cn-shanghai.aliyuncs.com");  IClientProfile profile = DefaultProfile.getProfile("cn-shanghai", accessKey, accessSecret);  client = new DefaultAcsClient(profile); //初始化SDK客户端 |

其中，accessKeyId和accessKeySecret 可以从阿里云官网控制台上获取 （链接：https://ak-console.aliyun.com/#/accesskey）。

addEndpoint函数的四个参数依次为“endpointName”、“regionId”、 “product”、 “domain”，getProfile函数第一个参数为“regionId”。目前IoT平台可以创建在“华东2（上海）”、“新加坡”、“日本（东京）”、美国（硅谷）”和“德国（法兰克福）”五个区域，我们选择创建在“华东2（上海）”区域,对应的endpointName和regionId即为“cn-shanghai”，*服务*地址domain为iot.cn-shanghai.aliyuncs.com，参数product设置为“Iot”。

**发起调用**

完成SDK的引入和初始化之后，工程便可运行。在Test.java文件中我们可以看到已经封装好的各个接口函数，如负责创建*产品*的函数，注册*设备*的函数等等，通过调用这些接口，开发人员就可以方便地使用 Java 程序来对物联网平台进行操作。

本实验中，我们以调用接口完成*产品*创建、*设备*添加和消息发布这三个基本操作为例，介绍如何调用Demo中封装好的函数完成原先在控制台上完成的操作。我们需要调用的函数有：*产品*创建函数createProductTest，*设备*注册函数registDeviceTest以及消息发布函数PubTest，此时main函数内容如下：

|  |
| --- |
| //创建*产品*  String *productKey* = createProductTest("Test\_sdk", "*服务*端sdk创建");  String *deviceName* = "device";  //注册单个*设备*  registDeviceTest(*productKey*, *deviceName*);  //发送消息  String msg = "hello world";  String *Topic* = "/" + *productKey* + "/" + *deviceName* + "/get";  pubTest(*productKey*, *Topic*, msg); |

运行main函数后，在控制台上可以看到我们新创建的*产品*和新添加的*设备*。





同时由于我们向get这个*Topic*中发布了一条“hello world”消息，*设备Topic*列表中这个*Topic*的发布消息数显示为1。



其它函数的调用方法与此类似，此处不再赘述，读者可以自行尝试，查看结果。

**课程小结**

本次课程学习了云端通过消息*服务*的队列获取*设备*数据的实现方法以及可以代替物联网平台控制台上相关操作的*服务*器端API。基于本次课程的内容，读者便可以基于物联网平台在应用*服务*器上开发符合自身业务需求的应用了。