ALIBABA CLOUD

阿里云

物联网平台

设备接入

文档版本：20201023

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.创建产品

使用物联网平台的第一步：在控制台创建产品。产品是设备的集合，通常是一组具有相同功能定义的设备集合。例如：产品指同一个型号的产品，设备就是该型号下的某个设备。

操作步骤

1. 登录物联网平台控制台。
2. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。

   
3. 在左侧导航栏，选择**设备管理 > 产品**，单击**创建产品**。
4. 按照页面提示填写信息，然后单击**确认**。

   创建产品

参数	描述
产品名称	为产品命名。产品名称在账号内具有唯一性。例如，可以填写为产品型号。支持中文、英文字母、日文、数字、下划线（_）、短划线（-）、@符号和英文圆括号，长度限制4~30个字符，一个中文或日文占2个字符。
所属品类	相当于产品模板。 ○ **标准品类**：物联网平台已为标准品类预定义了功能模板。 例如，**能源管理 > 电表**品类已预定义用电量、电压、电流、总累积量等电表标准功能。选择该品类，创建的产品具有预定义的功能。您可以在该产品的**产品详情**页**功能定义**页签下，编辑、修改、新增功能。 ○ **自定义品类**：产品创建成功后，需根据实际需要，自定义物模型。
节点类型	产品下设备的类型。 ○ **直连设备**：直连物联网平台，且不能挂载子设备，也不能作为子设备挂载到网关下的设备。 ○ **网关子设备**：不直接连接物联网平台，而是通过网关设备接入物联网平台的设备。网关与子设备说明，请参见网关与子设备。 ○ **网关设备**：可以挂载子设备的直连设备。网关具有子设备管理模块，可以维持子设备的拓扑关系，将与子设备的拓扑关系同步到云端。
接入网关协议	节点类型选择为**网关子设备**的参数。表示该产品下的设备作为子设备与网关的通讯协议类型。 ○ 自定义：表示子设备和网关之间是其它标准或私有协议。 ○ Modbus：表示子设备和网关之间的通讯协议是Modbus。 ○ OPC UA：表示子设备和网关之间的通讯协议是OPC UA。 ○ ZigBee：表示子设备和网关之间的通讯协议是ZigBee。 ○ BLE：表示子设备和网关之间的通讯协议是BLE。

参数	描述
连网方式	直连设备和网关设备的连网方式。 ○ Wi-Fi ○ 蜂窝（2G/3G/4G） ○ 以太网 ○ LoRaWAN 说明 首次选择LoRaWAN时，需要单击下方提示中的立即授权，前往RAM控制台授权IoT使用*AliyunIOTAccessingLinkWANRole*角色访问LinkWAN服务。 ○ 其他
入网凭证	当连网方式选择为LoRaWAN时，需提供入网凭证名称。 若无凭证，请单击创建凭证，进入物联网络管理平台，添加专用凭证，并为凭证授权用户，详细操作请参见入网开通。 使用凭证创建的产品，将作为一个节点分组，自动同步到物联网络管理平台的节点分组列表中。
数据格式	设备上下行的数据格式。 ○ ICA标准数据格式（Alink JSON）：是物联网平台为开发者提供的设备与云端的数据交换协议，采用JSON格式。 ○ 透传/自定义：如果您希望使用自定义的串口数据格式，可以选择为透传/自定义。 您需在控制台提交数据解析脚本，将上行的自定义格式的数据转换为Alink JSON格式；将下行的Alink JSON格式数据解析为设备自定义格式，设备才能与云端进行通信。 说明 使用LoRaWAN接入网关的产品仅支持透传/自定义。

参数	描述
认证方式	设备接入物联网平台的安全认证方式。产品创建成功后，认证方式不可变更。可选： ○ 设备密钥：使用物联网平台为设备生成的DeviceSecret进行设备认证签名计算。 使用DeviceSecret签名计算，可参见MQTT-TCP连接通信。 ○ ID²：ID²认证提供设备与物联网平台的双向身份认证能力，通过建立轻量化的安全链路（iTLS）来保障数据的安全性。 说明 ■ 仅华东2（上海）地域支持ID²认证方式。 ■ 连网方式选择为*LoRaWAN*的产品不支持ID²认证方式。 ■ 选择使用ID²认证，需购买ID²服务。请参见IoT设备身份认证（ID²）用户手册。 ○ X.509证书：使用X.509数字证书进行设备身份认证。 说明 ■ 仅华东2（上海）地域支持X.509证书。 ■ 连网方式选择为*LoRaWAN*的产品不支持X.509证书。 在产品下创建设备后，物联网平台为设备生成唯一的X.509证书。您可以在设备的设备详情页，查看和下载该设备的X.509证书。 使用X.509证书进行设备身份认证的设备端配置说明，请参见使用X.509证书认证。
产品描述	可输入文字，用来描述产品信息。字数限制为100。
资源组	将该产品划归为某个资源组。 默认选择*账号全部资源*，可以选择已创建的资源组。通过资源组管理，可以授予指定子账号查看和操作该产品，而未授权的子账号则不可以查看和操作该产品。关于资源组，请参见什么是资源组。 说明 该功能目前仅适用于白名单用户。如有需要，请提交工单，提供您的阿里云账号UID，申请成为白名单用户。 产品创建成功后，可以在资源管理控制台变更产品所属的资源组。

后续步骤

1. 进行产品开发。

   在左侧导航栏，选择**设备管理 > 产品**，在产品列表中，单击产品对应的**查看**，进入产品详情页。单击相应页签，查看产品信息、Topic类列表，设置自定义Topic、功能定义（物模型）、数据解析脚本服务端订阅等。

2. 进行设备开发。

单击产品详情页的**设备开发**页签，跟随界面提示进行设备创建、功能开发、证书烧录、连接验证，完成设备开发，接入物联网平台。详细信息，请参见设备接入指南。

3. 在该产品的详情页中，单击**发布**，发布产品。

发布前需确认：产品各项信息已设置完成、设备开发调试工作已完成、产品已具备上线发布条件。

产品发布后，产品状态变为**已发布**，此时产品信息仅支持查看，不支持修改和删除操作。

已发布的产品支持**撤销发布**。

2.创建设备

2.1. 单个创建设备

产品指某一类设备，创建完产品后，需要为设备创建身份。您可以创建单个设备，也可以批量创建设备。本文介绍单个设备的创建。

操作步骤

1. 登录物联网平台控制台。
2. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。

3. 在左侧导航栏，选择**设备管理 > 设备**。
4. 在**设备**页，单击**添加设备**。
5. 在**添加设备**对话框中，输入设备信息，单击**确认**。

参数	描述
产品	选择产品。新创建的设备将继承该产品定义好的功能和特性。 说明 若该产品关联了其他平台，请确保您的账户下有足够的激活码用于创建设备。
DeviceName	设置设备名称。 设备名称在产品内具有唯一性。支持英文字母、数字、短划线（-）、下划线（_）、at符号（@）、点号（.）和英文冒号（:），长度限制为4~32个字符。 说明 DeviceName可以为空。为空时，由物联网平台生成一个全局唯一标识符作为设备的DeviceName。
备注名称	设置备注名称。支持中文、英文字母、日文、数字和下划线（_），备注名称长度为4~64个字符，一个中文或日文占2个字符。

执行结果

设备创建成功后，将自动弹出**查看设备证书**对话框。您可以查看、复制设备证书信息。设备证书由设备的ProductKey、DeviceName和DeviceSecret组成，是设备与物联网平台进行通信的重要身份认证，建议您妥善保管。

参数	说明
ProductKey	设备所隶属产品的Key，即物联网平台为产品颁发的全局唯一标识符。
DeviceName	设备在产品内的唯一标识符。DeviceName与设备所属产品的ProductKey组合，作为设备标识，用来与物联网平台进行连接认证和通信。

参数	说明
DeviceSecret	物联网平台为设备颁发的设备密钥，用于认证加密。需与DeviceName成对使用。

后续步骤

查看设备信息，请参见管理设备。

设备创建完成后，设备状态显示未激活。请参见Link SDK文档开发设备端SDK，然后接入物联网平台，激活设备。

设备接入物联网平台实践案例，可参见：

- 使用MQTT.fx接入物联网平台
- Android Things接入物联网平台
- Ruff开发板接入物联网平台

2.2. 批量创建设备

产品指某一类设备，创建完产品后，需要为具体设备创建身份。您可以创建单个设备，也可以批量创建设备。本文介绍如何批量创建设备。

操作步骤

1. 登录物联网平台控制台。
2. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。

   
3. 在左侧导航栏，选择**设备管理 > 设备**。
4. 在**设备**页，单击**批量添加**。
5. 选择产品。新创建的设备将继承该产品的功能和特性。

   > **说明** 若该产品关联了其他平台，请确保您的账户下有足够的激活码用于创建设备。

   

6. 选择设备的添加方式。
   - **自动生成**：无需为设备指定DeviceName。填写设备数量后，系统将为每个设备自动生成由字母、数字随机组合成的DeviceName。
   - **批量上传**：需要为所有设备指定名称。单击**下载.csv模板**下载表格模板，在模板中填写DeviceName、Nickname，然后将填好的表格上传至控制台。

> **说明** 填写模板时，请注意：
>
> - DeviceName支持英文字母、数字、短划线（-）、下划线（_）、at符号（@）、点号（.）和英文冒号（:），长度限制为4~32个字符。DeviceName在产品维度具有唯一性，不可重复。
> - Nickname为备注名称，是选填项，支持中文、英文字母、日文、数字和下划线（_），备注名称长度为4~64个字符，一个中文或日文占2个字符。
> - 一个文件中最多可包含1,000条记录。
> - 文件大小不超过2 MB。

7. 单击**确认**，完成创建批量设备。若批量上传的设备名称列表中有不合法的名称，将出现错误提示。请单击**下载不合法列表**，查看不合法的设备名称。根据设备名称规范，修改设备名称，再重新上传文件。
8. 设备创建成功后，单击**下载设备证书**，下载本批次设备的设备证书，用于在产线上统一烧录。

> **说明** 如果创建产品时，选择认证方式为**X.509证书**，批量创建设备成功后，返回的X.509证书下载URL（CertUrl）有效期为30日，请及时下载。
>
> 下载包中，包含成功生成的X.509证书和密钥，以及证书生成失败的设备列表txt文件。如果没有失败的设备，则没有txt文件。

执行结果

批量创建设备成功。可在**设备**页的**批次管理**页签下，查看该批次设备信息和下载设备证书。

- 单击**查看详情**，查看对应批次设备的详细信息。
- 单击**下载CSV**，下载该批次设备的证书。

后续步骤

查看设备信息，请参见管理设备。

设备创建完成后，设备状态显示**未激活**。请参见Link SDK文档开发设备端SDK，然后接入物联网平台，激活设备。

设备接入物联网平台实践案例，可参见：

- 使用MQTT.fx接入物联网平台
- Android Things接入物联网平台
- Ruff开发板接入物联网平台

2.3. 创建LoRa设备

物联网平台支持创建LoRa产品和设备。创建LoRa产品后，可以根据本文操作，创建LoRa设备。您可以单个创建LoRa设备，也可以批量操作。

前提条件

- 已创建连网方式为LoRaWAN的产品，参见创建产品。
- 已确定设备的证书信息。证书信息有两种组合：
  - DevEUI、PIN Code：阿里云颁发的证书信息，一般印刷于设备外显标签上。

- DevEUI、JoinEui、AppKey：对于购买时开启了Link WAN的实例，您可使用该组证书信息单个创建设备，不支持批量创建。当设备无阿里云颁发的证书信息时，用户自定义该组证书信息：DevEUI、JoinEui定义为16位HEX，且JoinEui不能以d896e0开头，AppKey定义为32位HEX。

以上两种证书信息中，DevEUI都是LoRa设备的唯一标识符，采用LoRaWAN协议标准规范。请确保DevEUI在产品下唯一，且已烧录到设备中。

单个创建设备

1. 登录物联网平台控制台。
2. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。

3. 左侧导航栏选择**设备管理 > 设备**，单击**添加设备**。
4. 在对话框中选择已创建的连网方式为LoRaWAN的产品。新创建的设备将继承该产品定义好的功能和特性。
5. 填写设备的证书信息。
   - 对于公共实例和购买时未开启Link WAN的实例，填写DevEUI和PIN Code：
   - 对于购买时开启了Link WAN的实例，可以选择：
     - **阿里云颁发**：填写DevEUI和PIN Code。
     - **用户自定义**：填写DevEUI、JoinEui和AppKey。
6. 单击**确认**，完成设备创建。
   设备创建完成后，将自动弹出**查看设备证书**弹框。您可以查看、复制LoRa设备的证书信息。

批量创建设备

1. 登录物联网平台控制台。
2. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。

3. 左侧导航栏选择**设备管理 > 设备**，单击**批量添加**。
4. 选择已创建的连网方式为LoRaWAN的产品。新创建的设备将继承该产品定义好的功能和特性。
5. 单击**下载.csv模板**下载表格模板，在模板中填写DevEUI和PIN Code，然后将填好的表格上传至控制台。
6. 单击**确认**，完成设备创建。

后续步骤

您可以参见物联网络管理平台文档搭建物联网所需的网络服务和开发设备端（即网关开发和节点开发）。

> **说明** 在物联网平台创建LoRa设备后，您无需再在物联网络管理平台录入设备信息和配置数据流转。

2.4. 管理设备

在物联网平台成功创建设备后，您可以在控制台管理、查看具体设备信息。

管理账号下的设备

1. 登录物联网平台控制台。
2. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。

3. 在左侧导航栏，选择**设备管理 > 设备**，进入**设备**页。

任务	操作步骤
查看具体产品下设备信息	在页面左上方选择某个产品。
搜索设备	输入设备名称、设备备注名称或设备标签搜索具体设备，支持模糊搜索。
查看具体设备信息	单击设备对应的**查看**。
管理AI-BOX设备	单击AI-BOX设备对应的**边缘管理**，跳转到视频服务控制台中该设备的详情页。AI-BOX设备使用说明，请参见视频边缘智能服务文档。 **说明** ○ 仅针对启用了视频服务的实例。您购买的其他类型实例不提供AI-BOX设备管理。 ○ AI-BOX设备的产品所属品类为**标准品类 > 摄像头边缘节点**。
删除某个设备	单击设备对应的**删除**。 **说明** 设备删除后，该设备证书信息将失效，该设备在物联网平台上的数据记录随之删除。

查看具体设备信息

在设备列表中，单击设备对应的**查看**按钮，进入**设备详情**页。

任务	操作步骤
激活设备	未激活状态表明设备未接入物联网平台。您可以参见下载设备端SDK进行设备开发，然后接入物联网平台，将设备激活。
查看设备信息	查看设备基本信息，包括设备证书信息、固件信息、扩展信息、标签信息等内容。
查看设备数据	在**物模型数据**页签下： • 选择**运行状态**页签，查看设备上报的当前属性值，属性记录数据和期望属性值。 • 选择**事件管理**页签，查看设备上报的事件记录。 • 选择**服务调用**页签，查看设备服务调用记录。

任务	操作步骤
查看设备日志	在**日志服务**页签下，可以查看云端运行日志。开启**设备本地日志上报**开关后，还可以查看设备本地日志。日志具体说明，请参见云端运行日志、设备本地日志。
查看设备分组	在**分组**页签下，查看设备的分组信息。可以单击**添加此设备到分组**将设备添加到已有的分组。设备分组的具体说明，请参见设备分组。

3.下载设备端SDK

物联网平台提供各类设备端SDK，简化开发过程，使设备快速上云。

物联网 设备端 SDK 开发

前提条件

在设备端开发之前，您需要首先完成控制台所需操作，获取设备开发阶段的必要信息，包括设备信息、Topic信息等。具体操作指导，请参见创建产品、创建设备、消息通信Topic、新增物模型等章节。

基于设备端SDK开发

您可以在设备中集成物联网平台提供的SDK，实现物联网平台接入。设备开发完成，接入物联网平台，设备激活，在物联网平台显示在线。

设备接入流程请参见设备接入引导。

实际开发中，请根据开发时使用的语言、平台，选用合适的设备端SDK，包含：

- C SDK
- Android SDK
- NodeJS SDK
- Java SDK
- Python SDK
- iOS SDK

> **说明** 不同语言、平台的SDK功能可能有所不同，具体请查询各SDK支持的功能。

如果以上SDK不满足您的需求，可以按照以下模板发送邮件至 `linkkitSDK-query@list.alibaba-inc.com` 联系我们。

- 邮件主题：*SDK开发语言或平台咨询*
- 邮件内容：

```
公司名称：
联系人：
联系电话：
设备开发语言或平台：
需求描述：
贵司的产品规模及开发计划：
```

泛化协议SDK

阿里云物联网平台支持基于MQTT、CoAP和HTTP协议的通信，其他类型协议，如消防协议GB/T 26875.3-2011等暂不支持。如果您的设备使用物联网平台不支持的协议，可能无法直接接入物联网平台。您可以使用泛化协议SDK，快速构建桥接服务，搭建设备或平台与阿里云物联网平台的双向数据通道。

基于Alink协议开发

如果提供的设备端SDK无法满足您的需求，可以参见Alink协议文档自行开发。

使用开源MQTT客户端接入物联网平台的示例，请参见：

- Paho-MQTT Go接入示例
- Paho-MQTT C#接入示例
- Paho-MQTT C接入示例
- Paho-MQTT Java接入示例
- Paho-MQTT Android接入示例

4.设备安全认证

为保障设备安全，物联网平台为设备颁发证书，包括产品证书（ProductKey和ProductSecret）与设备证书（DeviceName和DeviceSecret）。其中，设备证书与设备一一对应，以确保设备的唯一合法性。设备通过协议接入IoT Hub之前，需依据不同的认证方案，上报产品证书和设备证书，云端通过认证后，方可接入物联网平台。针对不同的使用环境，物联网平台提供了多种认证方案。

物联网平台目前提供三种认证方案，分别是：

- 一机一密：每台设备烧录自己的设备证书。
- 一型一密：同一产品下设备烧录相同产品证书。
- 子设备认证：网关连接上云后，子设备的认证方案。

三种方案在易用性和安全性上各有优势，您可以根据设备所需的安全等级和实际的产线条件灵活选择，方案对比如下图所示。

认证方案对比

对比项	一机一密	一型一密	子设备注册
设备端烧录信息	ProductKey、DeviceName、DeviceSecret	ProductKey、ProductSecret	ProductKey
云端是否需要开启	无需开启，默认支持。	需打开动态注册开关	需打开动态注册开关
是否需要预注册DeviceName	需要，确保产品下DeviceName唯一	需要，确保产品下DeviceName唯一	需要预注册
产线烧录要求	逐一烧录设备证书，需确保设备证书的安全性	批量烧录相同的产品证书，需确保产品证书的安全存储	子设备批量烧录相同的产品证书，但需要确保网关的安全性
安全性	较高	一般	一般
是否有配额限制	有，单个产品50万上限	有，单个产品50万上限	有，单个网关最多可注册200个子设备
其他外部依赖	无	无	依赖网关的安全性保障

4.1. 概述

设备接入物联网平台之前，需通过身份认证。目前，物联网平台支持使用设备密钥、ID²和X.509证书进行设备身份认证。

设备密钥认证

在创建产品时，**认证方式**选择为**设备密钥**，物联网平台会为设备颁发ProductSecret、DeviceSecret等密钥。设备接入物联网平台时，需使用密钥进行身份认证。

针对不同的使用环境，物联网平台提供了使用密钥认证的四种认证方案。

- 一机一密：每台设备烧录自己的设备证书（ProductKey、DeviceName和DeviceSecret）。
- 一型一密预注册：同一产品下设备烧录相同产品证书（ProductKey和ProductSecret）。开通产品的动

态注册功能，设备通过动态注册获取DeviceSecret。

- 一型一密免预注册：同一产品下设备烧录相同产品证书（ProductKey和ProductSecret）。开通产品的动态注册功能，通过动态注册，设备不获取DeviceSecret，而是获取ClientID与DeviceToken的组合。
- 子设备动态注册：网关连接上云后，子设备通过动态注册获取DeviceSecret。

四种方案在易用性和安全性上各有优势，您可以根据设备所需的安全等级和实际的产线条件灵活选择。方案对比，如下表所示。

认证方案对比

对比项	一机一密	一型一密预注册	一型一密免预注册	子设备动态注册
设备端烧录信息	ProductKey、DeviceName、DeviceSecret	ProductKey、ProductSecret	ProductKey、ProductSecret	ProductKey
云端是否需要开启动态注册	无需开启，默认支持。	需打开动态注册开关。	需打开动态注册开关。	需打开动态注册开关。
是否需要提前在物联网平台创建设备，注册DeviceName	需要，产品下DeviceName唯一。	需要，产品下DeviceName唯一。	不需要。	需要，确保产品下DeviceName唯一。
产线烧录要求	逐一烧录设备证书，需确保设备证书的安全性。	批量烧录相同的产品证书，需确保产品证书的安全存储。	批量烧录相同的产品证书，需确保产品证书的安全存储。	• 网关可以本地获取子设备ProductKey。 • 将子设备ProductKey烧录在网关上。
安全性	较高	一般	一般	一般
是否有配额限制	有，单个产品50万上限。	有，单个产品50万上限。	有，单个产品50万上限。	有，单个网关最多可注册1500个子设备。
其他外部依赖	无	无	无	依赖网关的安全性保障。

ID²认证

阿里云提供IoT设备身份认证ID²（Internet Device ID）。ID²是一种物联网设备的可信身份标识，具备不可篡改、不可伪造、全球唯一等安全属性。

在创建产品时，认证方式选择为ID²，设备接入物联网平台时，使用ID²身份认证。

使用ID²认证，需购买ID²服务。ID²服务购买方式和使用指南，请参见IoT设备身份认证（ID²）用户手册

> 说明
> - 目前仅华东2（上海）地域支持ID²认证。
> - 连网方式选择为LoRaWAN的产品不支持ID²认证。

X.509证书认证

X.509是由国际电信联盟（ITU-T）制定的数字证书标准，具有通信实体鉴别机制。目前物联网平台华东2（上海）地域支持使用X.509证书进行设备身份认证。

使用X.509证书的操作流程：

1. 在创建产品时，**认证方式**选择为**X.509证书**。
2. 在该产品下创建设备，物联网平台会为设备颁发X.509证书和密钥。
3. 开发设备端，将X.509数字证书和密钥烧录到设备上。

   设备端上身份认证配置，请参见使用X.509证书认证。

> 说明
> - 仅MQTT协议直连的设备可使用X.509证书认证。
> - 目前仅华东2（上海）地域支持X.509证书认证。
> - 连网方式为LoRaWAN的产品不支持X.509证书认证。
> - 设备身份认证方式设置后，不可更改。

4.2. 一机一密

一机一密认证方法，即预先为每个设备烧录其唯一的设备证书（ProductKey、DeviceName和DeviceSecret）。当设备与物联网平台建立连接时，物联网平台对其携带的设备证书信息进行认证。认证通过，物联网平台激活设备，设备与物联网平台间才可传输数据。

背景信息

一机一密认证方式的安全性较高，推荐使用。

使用流程示意图：

操作步骤

1. 创建产品。在物联网平台控制台创建产品，具体步骤，请参见创建产品。
2. 添加设备。

   在已创建产品下添加设备，并获取设备证书信息，具体步骤，请参见单个创建设备、批量创建设备。
3. 产线烧录。
   i. 下载设备端SDK。
   ii. 初始化设备端SDK。在设备端SDK中，填入设备证书。初始化一机一密认证方式的设备端SDK，请参见Link SDK文档中，各语言SDK《设备认证》、《认证与连接》文档。
   iii. 根据实际需求，完成设备端SDK开发，如OTA开发、子设备接入、设备物模型开发、设备影子开发等。
   iv. 在产线上，将已开发完成的设备SDK烧录至设备中。
4. 设备联网。设备上电联网后，携带设备证书发起认证请求。请参见MQTT-TCP连接通信、CoAP连接通信、HTTP连接通信。
5. 云端激活。物联网平台对设备证书进行校验。认证通过后，与设备建立连接，设备便可通过发布消息至Topic和订阅Topic消息，与物联网平台进行数据通信。

4.3. 一型一密

一型一密认证方式下，同一产品下所有设备可以烧录相同固件，包含相同的产品证书（ProductKey和ProductSecret）。设备发送激活请求时，物联网平台进行身份确认，认证通过，下发设备接入所需信息。

背景信息

> 说明
> - 采用一型一密认证方式，设备烧录相同固件，存在产品证书泄露风险。您可以在**产品详情**页面，手动关闭动态注册开关，拒绝新设备的认证请求。
> - 一型一密动态注册时必须使用TLS加密，如果您的设备端SDK无法运行TLS加密，则无法使用一型一密认证方式，请采用**一机一密**认证方式。

一型一密认证使用流程示意图：

图中有两种使用方式：

- 一型一密预注册：

  设备联网前，需要在物联网平台预注册设备DeviceName，建议采用设备的MAC地址、IMEI、SN码等作为DeviceName。物联网平台为设备颁发DeviceSecret。

  云端鉴权成功后，设备采用设备证书（ProductKey、DeviceName和DeviceSecret）与云端建立通信连接。

  支持通过MQTT通道、HTTP通道进行一型一密预注册认证。

- 一型一密免预注册：

  不需要在物联网平台预注册设备DeviceName，便于使用物联网卡卡号等作为DeviceName。

  云端鉴权成功后，设备采用ProductKey、ProductSecret、ClientID和DeviceToken与云端建立通信连接。

  支持通过MQTT通道进行一型一密预注册认证。

操作步骤

1. 创建产品。在**物联网平台控制台**创建产品，具体步骤，请参见**创建产品**。
2. 开启动态注册。在已创建产品的**产品详情**页面，开启动态注册开关。系统将进行短信验证，以确认是您本人操作。

   > 说明 若设备发出激活请求时，系统校验发现该开关未开启，将拒绝新设备的动态激活请求。已激活设备不受影响。

   

3. 添加设备。
   - 一型一密预注册：

     在已创建产品下添加设备，具体步骤，请参见**批量创建设备**或**单个创建设备**。

     因设备激活时会校验DeviceName，建议您采用可以直接从设备中读取到的ID，如设备的MAC地址、IMEI或SN码等，作为DeviceName使用。

物联网平台为设备颁发DeviceSecret。设备初始状态为未激活。

- 一型一密免预注册，请跳过本步骤。

4. 产线烧录。
   i. 下载设备端SDK。

   > 注意 对于一型一密免预注册方式，请在设备端使用4.X版C SDK。该SDK包含DAS（设备取证服务），可以对设备可能产生的安全事件进行风控。物联网平台不承担因设备端没有使用该SDK而导致的安全风险。

   ii. 初始化设备端SDK，开通设备端SDK动态注册。在设备端SDK中，填入产品证书（ProductKey和ProductSecret）。开通设备端SDK动态注册，请参见C SDK的MQTT动态注册、C SDK的HTTPS动态注册，以及Link SDK文档中，其他语言SDK的《认证与连接》文档。
   iii. 根据实际需求，完成设备端SDK开发，如OTA开发、子设备接入、设备物模型开发、设备影子开发等。
   iv. 在产线上，将已开发完成的设备SDK烧录至设备中。
5. 设备联网。设备上电联网后，携带ProductKey、ProductSecret、DeviceName发起认证请求。请参见基于MQTT通道的设备动态注册、基于HTTP通道的设备动态注册。
6. 云端激活。
   - 一型一密预注册：

     物联网平台校验通过后，下发已在第3步中为该设备颁发的DeviceSecret。至此，设备获得连接云端所需的设备证书（ProductKey、DeviceName和DeviceSecret），可以与云端建立连接，进行数据通信。

     > 说明
     > - 同一组设备证书只能用于激活一个物理设备。
     > - 若DeviceName名下已激活物理设备A，但物理设备B需要使用这个DeviceName，则您可以在物联网平台上删除设备A，作废设备A的DeviceSecret，再使用原DeviceName重新添加设备，激活物理设备B。
     > - 若设备因丢失DeviceSecret等原因需要重新激活，可以通过ResetThing接口重置设备，然后将设备重新联网激活，云端下发的DeviceSecret不变。

   - 一型一密免预注册：

     物联网平台校验通过后，下发ClientID、DeviceToken。设备后续通过ProductKey、ProductSecret和下发的ClientID、DeviceToken与云端建立连接，进行数据通信。

> 说明 一型一密免预注册情况下，物联网平台允许最多5个物理设备使用同一组ProductKey、ProductSecret、DeviceName进行激活，为这些物理设备下发不同的ClientID、DeviceToken。
>
> 当一个DeviceName名下有多个不同ClientID的物理设备时，物联网平台控制台**产品详情**页将提示“当前产品下有设备同时有两个ClientID”。您可以指定保留唯一物理设备，或清除所有物理设备：
>
> a. 在**产品详情**页，单击该提示后的**查看**，跳转到产品下的风险设备列表。
>
> b. 在**设备管理 > 设备**，单击列表中设备对应的**查看**，进入**设备详情**页，页面显示当前连接的ClientID，单击ClientID右侧的**切换**或**清除**。
>
> - **切换**：从下拉列表选择ClientID，通过该ClientID对应设备的首次连接时间，或者单击**日志服务**，通过该ClientID对应设备的**云端运行日志**判断其是否为需要保留的物理设备。选择要保留的物理设备的ClientID，单击**确认**，使用其他ClientID的物理设备将被禁止连接。
> - **清除**：所有物理设备都将被禁止连接。

4.4. 使用X.509证书认证

X.509证书是一种用于通信实体鉴别的数字证书。物联网平台支持基于MQTT协议直连的设备使用X.509证书进行认证。

限制说明

- 仅MQTT协议直连的设备可使用X.509证书认证。
- 目前仅华东2（上海）地域支持X.509证书认证。
- 连网方式为LoRaWAN的产品不支持X.509证书认证。
- 设备身份认证方式设置后，不可更改。

生成X.509证书

设备的X.509证书由物联网平台颁发。

1. 登录物联网平台控制台。
2. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。

   
3. 在**设备管理 > 产品**，创建认证方式为**X.509证书**的产品，详细操作请参见创建产品。

   

4. 在**设备管理 > 设备**，在新建的产品下创建设备，详细操作请参见单个创建设备、批量创建设备。物联网平台创建设备时，会为设备颁发X.509证书和密钥。

   > 说明 对于使用X.509证书进行安全认证的设备，您需要在设备端烧录X.509证书。如果同时设备使用一型一密认证或子设备通过网关接入物联网平台功能，则您还需分别烧录ProductSecret或DeviceSecret。
5. 下载设备的X.509证书和密钥。
   - 下载批量创建设备的X.509证书和密钥。

a. 在设备管理 > 设备 > 批次管理，单击设备批次对应的下载CSV按钮，下载设备证书信息表格。

b. 根据下载表格中的CertUrl地址，下载设备X.509证书和密钥。

> 说明 CertUrl地址有效期为30天。请在批量创建设备后的30天内下载X.509证书信息。

- 下载单个设备的X.509证书和密钥。两种方式：
  - 在设备管理 > 设备，单击设备对应的查看，进入设备详情页。单击X.509证书对应的下载按钮，下载证书信息。
  - 调用云端API QueryDeviceCert获取证书信息。

设备端认证配置

目前仅C语言版的设备端Link SDK支持X.509认证方式。请访问C SDK获取，下载开发代码Demo。

使用X.509证书的设备连接物联网平台的域名和端口如下：

- 连接域名：*x509.itls.cn-shanghai.aliyuncs.com*
- 端口：*1883*

以下以C语言版的设备端Link SDK为例。

1. 下载根证书，用于双向认证时，校验服务端证书。
2. 配置设备认证的MQTT连接。

   如果您使用C语言Link SDK Demo，请在 *\src\mqtt\examples*目录下*mqtt_example.c*文件中进行配置。

   - 将设备证书信息（ProductKey、DeviceName和DeviceSecret）设置为空字符串。

     使用X.509证书认证，建立认证MQTT连接时，不传入设备证书（ProductKey、DeviceName和DeviceSecret）。设备连接物联网平台成功后，由云端下发设备证书信息（ProductKey和DeviceName）给设备。

     示例：

     ```
     char g_product_key[IOTX_PRODUCT_KEY_LEN + 1]     = "";
     char g_device_name[IOTX_DEVICE_NAME_LEN + 1]     = "";
     char g_device_secret[IOTX_DEVICE_SECRET_LEN + 1]   = "";
     ```

   - 注册ITE_IDENTITY_RESPONSE事件，用于MQTT建连成功后，接收云端下发的ProductKey和DeviceName。

     示例：

     ```
     int identity_response_handle(const char *payload)
     {
        EXAMPLE_TRACE("identify: %s", payload);

        return 0;
     }

     IOT_RegisterCallback(ITE_IDENTITY_RESPONSE, identity_response_handle);
     ```

> **说明** 您还可以配置设备端将接收到的ProductKey和DeviceName在进行固化。如果不做固化，设备每次连接物联网平台都会重新获取ProductKey和DeviceName。

3. 配置设备的X.509证书和其对应的密钥（PrivateKey），用于连接认证。

   如果您使用C语言Link SDK Demo，请在 *\wrappers\tls*目录下的*HAL_TLS_mbedtls.c*文件中替换以下信息。

   - 将g_cli_crt取值替换为您的设备X.509证书。
   - 将g_cli_key取值替换为X.509证书的密钥。

   示例：

```
const char *g_cli_crt = \
{
  \
  "-----BEGIN CERTIFICATE-----\r\n"
  "Your X.509 certificate"
  "-----END CERTIFICATE-----\r\n"
};

const char *g_cli_key = \
{
  \
  "-----BEGIN RSA PRIVATE KEY-----\r\n"
  "Your X.509 PrivateKey"
  "-----END RSA PRIVATE KEY-----\r\n"
};
```

> **说明** 如果您不使用阿里云提供的设备端Link SDK，而是自己开发设备端，需：
>
> - 将设备X.509证书和密钥配置到安全库中。
> - 将设备证书信息（ProductKey、DeviceName和DeviceSecret）设置为空字符串，认证通过后由云端下发ProductKey和DeviceName。
>
>   设备端从Topic `/ext/auth/identity/response` 中获取云端下发的ProductKey和DeviceName。该Topic无需订阅。
>
>   消息payload格式：
>
> ```
> {
>   "productKey":"***",
>   "deviceName":"***"
> }
> ```

设备再次连接

设备认证通过，获得ProductKey和DeviceName，并将ProductKey和DeviceName固化后，如果设备下线，再重新建立与物联网平台的MQTT连接时，传入的CONNECT报文参数：

连接域名	x509.itls.cn-shanghai.aliyuncs.com:1883
可变报头（variable header）：Keep Alive	CONNECT指令中需包含Keep Alive（保活时间）。保活心跳时间取值范围为30秒至1,200秒。如果心跳时间不在此区间内，物联网平台会拒绝连接。建议取值300秒以上。如果网络不稳定，将心跳时间设置高一些。
MQTT的CONNECT报文参数	mqttClientId: clientId+"|securemode=2,signmethod=hmacsha1,timestamp=132323232|" mqttUsername: deviceName+"&"+productKey mqttPassword: "" mqttClientId包含的参数说明如下。其中，|| 内为扩展参数。 • clientId：表示客户端ID，建议使用设备的MAC地址或SN码，64字符内。 • timestamp：表示当前时间毫秒值，可以不传递。 • signmethod：表示签名算法类型。支持hmacmd5，hmacsha1和hmacsha256。 • securemode：取值为*2*，表示TLS直连模式。 说明 因为使用X.509数字证书认证，物联网平台不再校验签名值，mqttPassword设置为空。

4.5. 使用ID²认证

阿里云提供IoT设备身份认证ID²（Internet Device ID）。ID²是一种物联网设备的可信身份标识，具备不可篡改、不可伪造、全球唯一的安全属性。

背景信息

ID²设备身份认证服务提供设备与云端的双向身份认证和链路加密功能。

使用ID²认证的设备，有两种注册方式：

- 只需在控制台创建产品，无需创建设备。设备使用ID²认证通过后，物联网平台会根据设备端上报的设备名称，自动注册设备。
- 在控制台创建产品后，开启ID²白名单校验，然后注册设备。未注册的设备接入物联网平台时不能通过认证。

> 说明
> - 目前仅华东2（上海）地域支持ID²证书认证。
> - 对于您购买的实例，请确保已启用ID²设备身份认证。若创建实例时未启用，可通过为实例升配启用该功能。
> - 设备身份认证方式设置后，不可更改。

操作步骤

1. 登录物联网平台控制台。
2. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。

   
3. 在**设备管理 > 产品**，创建认证方式为ID²的产品，详细操作请参见创建产品。

   创建产品

   在物联网平台控制台创建ID²产品后，产品会自动列入IoT设备身份认证控制台的产品列表。
4. 登录IoT设备身份认证控制台。
   - 对于您购买的实例，登录物联网平台控制台，找到对应的实例，单击实例进入**实例详情**页，在**公网终端节点区**域选择**管理ID²**，复制ID²控制台地址，粘贴到浏览器地址栏并按回车键，进入实例对应的ID²设备身份认证控制台。
   - 对于公共实例，单击进入IoT设备身份认证控制台。
5. 在左侧导航栏，选择**使用管理 > 产品管理**，查看创建的ID²产品，如下图所示。

   
6. 为产品分配ID²授权。

   > 说明
   > - 购买的ID²授权仅能用于指定产品下的设备。
   > - 每一个设备至少使用一个ID²授权。购买之后，您还可以根据业务需要，随时为该产品购买更多ID²授权。

   - 对于您购买的实例，单击上图中的**分配ID²**，将ID²授权，即启用ID²设备身份认证时购买的ID² License，分配给产品。
   - 对于公共实例，单击上图中的**购买授权**，购买ID²授权并分配给产品。

     购买成功后，可在IoT设备身份认证控制台的产品列表中，单击产品对应的**查看**，进入**产品详情**页查看ID²授权信息。
7. （可选）开启ID²白名单校验，注册设备。开启后，未注册的设备接入物联网平台时不能通过认证。
   i. 登录物联网平台控制台。
   ii. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。

   
   iii. 选择**设备管理 > 产品**，单击ID²产品对应的**查看**。
   iv. 在**产品详情**页，开启**ID²白名单校验**开关。系统将进行短信验证，以确认是您本人操作。

v. 在**设备管理 > 设备**，在ID²产品下创建设备，详细操作请参见单个创建设备、批量创建设备。

8. 烧录ID²到设备端。
   - 向设备端SDK集成ID² Client SDK。

     阿里云提供ID² Client SDK，用于设备端身份认证开发。请访问ID² Client SDK，获取SDK和查看编译说明。
   - 配置设备端调用ID²认证接口。Client SDK中已封装了对ID²载体和接口的操作。设备上线时，需调用ID² Client SDK的认证接口激活ID²。具体接口说明，请参见认证接口。

   阿里云IoT设备身份认证服务提供芯片、模组和设备级别的合作方案，详情请参见IoT设备身份认证文档。
9. 设备接入物联网平台。
   - 向设备端SDK传入设备信息。

     

     - 将PRODUCT_KEY和PRODUCT_SECRET替换为您的产品对应的信息。
     - 设备名称DEVICE_NAME：
       - 未开启ID²白名单校验时，传入自定义设备名称，需保证设备名称在产品下唯一。
       - 开启ID²白名单校验后，传入的设备名称需已在物联网平台注册。
     - 使用ID²认证的设备不再使用DeviceSecret进行认证，所以DEVICE_SECRET可传入任意值，系统不会做校验。
   - MQTT连接参数：
     - 接入域名：
       - 对于您购买的实例，请登录物联网平台控制台，找到对应的实例，单击实例进入**实例详情**页，在**公网终端节点区域**选择**管理ID²**，查看ID²认证地址，即为接入域名。
       - 对于公共实例，接入域名为 `${YourProductKey}.itls.cn-shanghai.aliyuncs.com` 。请将*${YourProductKey}*替换为您的产品对应的信息。
     - 端口：1883。
     - mqttClientId： `clientId+"|securemode=8,signmethod=hmacsha1,timestamp=2524608000000,authtype=id2,instanceId=xxxxx|"` 。

       *instanceId*为实例ID，仅在使用您购买的实例时传入，请登录物联网平台控制台，在**实例概览**页面查看。

执行结果

设备通过ID²认证通过后，成功接入物联网平台。物联网平台自动注册该设备，您可以在**设备管理 > 设备**查看该设备。

5.设备获取设备证书

5.1. 概述

物理设备可通过两种方式获取物联网平台颁发的设备证书（ProductKey、DeviceName和DeviceSecret）：设备厂商将证书烧录到设备上和设备上电联网后从云端获取证书。

- 烧录设备证书

  该方案是设备厂商获取到物联网平台颁发的设备证书后，在产线上将证书烧录到设备。设备上电联网之后，使用该证书连接到阿里云物联网平台。本方案需要设备厂商对自己的产线进行改造，使产线具有烧录证书的能力。

  具体方案，请参见将证书烧录至设备。

- 设备从云端获取证书

  该方案是设备上电联网后，自动获取IP地址，并连接设备厂商的云端服务器获取证书。在生产时，设备厂商无需为此类设备烧录设备证书。设备上电联网后，再从厂商的云端服务器获取物联网平台颁发的设备证书，继而连接阿里云物联网平台。采用这种方案可以不用在产线上设计证书烧录过程，可加快设备的量产速度。

  具体方案，请参见设备从云端获取证书。

5.2. 将证书烧录至设备

本文介绍在产线上将证书（ProductKey、DeviceName和DeviceSecret）烧录至设备的方案。

本方案需要您对设备产线做相应的改造，具体改造方案需您自行设计。下面仅介绍可用的烧录方案。

获取设备证书

创建设备时，系统会自动生成设备证书。您可以按以下方式获取设备证书，将获得的证书写入您自己的数据库或者文件中。

- 在物联网平台控制台上单个创建设备后，获取设备证书。
  - 创建成功后，将自动弹出**添加完成**对话框，单击**前往查看**或**一键复制设备证书**，获取设备证书。
  - 在**设备列表**页签下，单击设备对应的**查看**按钮，进入**设备详情**页**设备信息**页签下，查看设备证书。
- 在物联网平台控制台上批量创建设备后，下载该批次设备的证书文件。
  - 创建成功后，在**添加完成**提示框中，单击**下载设备证书**，即可下载该批次设备证书。
  - 在**设备**页的**批次管理**页签下，单击产品对应的**下载CSV**，下载产品下所有设备的证书。
- 调用云端API创建设备后，物联网平台将生成的设备证书返回给您的应用。

> **说明** 创建设备操作可以参考以下文档：
>
> 1. 创建产品操作，请参见创建产品。
> 2. 注册设备。
>   - 在物联网平台控制台单个创建设备，请参见单个创建设备。
>   - 在物联网平台控制台批量创建设备，请参见批量创建设备。
>   - 调用云端API创建设备。物联网平台提供单个注册设备的API RegisterDevice和批量注册设备的API BatchRegisterDevice、BatchRegisterDeviceWithApplyId。获取云端SDK和调用API的方法，请参见下载云端SDK。

证书烧录方式

获取到设备证书之后，可以在产线上启动一个服务器，用于分发设备证书。编程器、烧录器或设备可向该证书分发服务器申请证书，并将获得的证书烧录到设备的NVRAM或Flash。

支持两种烧录方式，您可以根据实际情况，选择使用相应的证书烧录方案。流程如下图所示。

烧录证书流程

两种烧录方式的说明如下。

- 使用编程器或烧录器烧录设备证书。

  需要您对现有的编程器或烧录器程序进行改造，让PC可以向证书分发服务器申请设备证书，然后通过编程器或烧录器将设备证书烧入到芯片或设备上。

  此方案需要在产线上部署多台烧录器或编程器，进行证书烧录。您可以根据设备产量的大小，增加或减少烧录器或编程器的数量。

- 设备主动获取证书。

  需要开发设备固件，使设备上电后，自动检测是否有有效的证书。当发现无有效证书时，主动向证书分发服务器申请设备证书，然后将获得的证书写入NVRAM或Flash。

  此方案无需在产线上部署烧录器或编程器，并且多个设备可以同时向证书分发服务器申请设备证书。

5.3. 设备从云端获取证书

本方案不在设备上烧录设备证书，而是设备上电联网后，向您的服务器发起请求，获取设备证书（ProductKey、DeviceName和DeviceSecret）。

原理

本方案中，您需要部署自己的设备证书分发服务器，开发相应的服务器API和设备信息数据表。

证书分发服务器收到来自设备的获取证书请求时，调用上述API。该API的业务逻辑为：根据请求中的设备标识查询设备信息数据表，根据查询结果，进行以下后续操作。

- 没有查到传入的设备标识，则返回设备非法错误。
- 有对应的设备标识，且已有设备证书，则返回设备证书。
- 有对应的设备标识，但没有设备证书，则调用物联网平台API RegisterDevice注册设备身份，获取证书后，再发送给设备。

设备获得证书之后，再使用该证书连接阿里云物联网平台。

参与该过程的各个角色的时序图如下所示。

> 说明
> - 设备需要能自动获取IP地址，并连接您的证书分发服务器。
> - 证书分发服务器由您自行设计实现。
> - 您需要确保设备连接到证书分发服务器的安全性和可靠性。

服务器API设计建议

建议您按以下方法设计该API。

- 请求参数：

参数	说明
deviceId	设备在您的系统中的唯一标识。自定义，可以是设备MAC或者SN等。

- 返回参数：

参数	说明
productKey	对应物联网平台颁发的设备证书的ProductKey。
deviceName	对应物联网平台颁发的设备证书的DeviceName。
deviceSecret	对应物联网平台颁发的设备证书的DeviceSecret。

设备信息数据表设计建议

建议您按以下方法设计设备信息数据表。

表格属性：

表格属性	建议取值
数据表名称	device_table
数据生命周期	-1
最大数据版本	1
数据有效版本偏差	86400
主键	deviceId，字符串（String），分区键。

数据字段：

字段	说明
deviceId	设备唯一标识。自定义，可以是设备MAC或者SN等。
registerTime	设备注册时间。

字段	说明
activateTime	设备激活时间。
productKey	对应物联网平台颁发的设备证书的ProductKey。
deviceName	对应物联网平台颁发的设备证书的DeviceName。
deviceSecret	对应物联网平台颁发的设备证书的DeviceSecret。
IotId	物联网平台为该设备颁发的设备ID，该设备在物联网平台上的唯一标识。

6.消息通信Topic

物联网平台中，云端和设备端通过 Topic 来实现消息通信。设备上报消息至指定的Topic中，并从Topic中订阅消息。云端将指令下发到Topic中，并订阅具体Topic来获取设备信息。

6.1. 什么是Topic

物联网平台中，服务端和设备端通过Topic来实现消息通信。Topic是针对设备的概念，Topic类是针对产品的概念。

产品Topic类

为了方便海量设备基于Topic进行通信，简化授权操作，物联网平台增加了产品Topic类的概念。Topic类是一类Topic的集合，例如，产品的自定义Topic类 `/${YourProductKey}/${YourDeviceName}/user/update` 是具体设备Topic `/${YourProductKey}/device1/user/update` 、 `/${YourProductKey}/device2/user/update` 等的集合。

登录物联网平台控制台，在对应实例下，选择**设备管理 > 产品**，单击产品对应的**查看**，进入**产品详情**页，单击**Topic类列表**页签，可以在**基础通信Topic**、**物模型通信Topic**、**自定义Topic**页签查看相应的Topic类。关于这三类Topic的说明，请参见Topic分类。

Topic类列表中：

- Topic类以正斜线（/）进行分层，区分每个类目。其中，有两个类目为既定类目：*${YourProductKey}*表示产品的标识符ProductKey，*${YourDeviceName}*表示设备名称。
- 操作权限：
  - **发布**表示设备可以往该Topic发布消息。
  - **订阅**表示设备可以订阅该Topic，从而获取消息。

设备Topic

产品的Topic类不用于通信，只是定义Topic，用于消息通信的是具体的设备Topic。

您创建设备后，产品的所有Topic类会自动映射到设备上，生成具体设备Topic，您无需单独为每个设备创建Topic。设备Topic格式和产品Topic类格式一致，区别在于Topic类中的变量*${YourDeviceName}*，在Topic中是具体的设备名称（DeviceName）。

设备Topic自动生成示意图

设备Topic只能被该设备用于消息通信，例如，Topic： `/${YourProductKey}/device1/user/update` 归属于设备device1，所以只能被设备device1用于发布或订阅消息，而不能被设备device2用于发布或订阅消息。

设备发送SUB指令订阅某个Topic后，登录物联网平台控制台，在对应实例下，选择**设备管理 > 设备**，单击设备对应的**查看**，进入**设备详情**页，单击**Topic列表**页签，**已订阅Topic列表**展示了设备订阅成功的所有Topic。云端系统可以通过其中的Topic发送下行消息。

单击列表中已订阅的自定义Topic对应的**发布消息**，可通过该Topic从云端发布一条消息到设备端。使用通配符的自定义Topic除外，详情请参见带通配符的自定义Topic。

设备可以通过UNSUB指令取消与指定Topic的订阅关系，取消成功后，该Topic从**已订阅Topic列表**中删除。

如果您需要管控单个设备的消息收发，请在物联网平台控制台对应实例下的**设备列表**页或在服务端调用DisableThing接口，禁用该设备；或在业务上管控发送给设备的消息。

Topic分类

物联网平台将Topic分为三类。

类别	说明
基础通信Topic	物联网平台预定义的基础功能通信Topic，包含： • OTA升级相关Topic。各Topic的用途和消息数据格式，请参见OTA升级。 • 设备标签相关Topic。各Topic的用途和消息数据格式，请参见设备标签。 • 时钟同步相关Topic。时钟同步功即NTP服务，请参见NTP服务。 • 设备影子相关Topic。各Topic的用途和消息数据格式，请参见设备影子数据流。 • 配置更新相关Topic。各Topic的用途和消息数据格式，请参见远程配置。 • 广播Topic。调用云端API PubBroadcast向订阅了该Topic的所有设备发送广播消息，实现批量控制设备。
物模型通信Topic	物联网平台预定义的物模型通信Topic。各物模型功能Topic消息的数据格式，请参见设备属性、事件、服务。 说明 在云端，不可以调用Pub接口向物模型通信Topic发送消息。 在云端通过物模型功能远程控制设备，请调用SetDeviceProperty或SetDevicesProperty设置设备属性值；调用InvokeThingService或InvokeThingsService调用设备服务。
自定义Topic	您可以根据业务需求，在产品的**Topic类列表**页自定义Topic类，具体操作请参见自定义Topic类。 Topic类是一个Topic模版配置，编辑更新某个Topic类后，可能对产品下所有设备使用该类Topic通信产生影响。建议在设备研发阶段设计好，设备上线后不再变更Topic类。

6.2. 自定义Topic

本文介绍如何为产品自定义Topic类。自定义Topic类将自动映射到该产品下的所有设备中。

操作步骤

1. 登录物联网平台控制台。
2. 在**实例概览**页，找到对应的实例，单击实例进入**实例详情**页。
3. 在左侧导航栏，选择**设备管理 > 产品**。
4. 在**产品**页面，找到需要自定义Topic类的产品，并单击对应操作栏中的**查看**按钮。
5. 在**产品详情**页面，单击**Topic类列表 > 自定义Topic > 定义Topic类**。
6. 配置参数，单击**确认**。

自定义Topic

参数	描述
设备操作权限	设备对该Topic的操作权限，可设置为**发布**、**订阅**、**发布和订阅**。
Topic类	将Topic类填充完整。类目命名只能包含字母、数字和下划线（_），每级类目不能为空。 说明 只有设备操作权限为**订阅**时，才可以使用通配符+和#自定义Topic类，以便设备实现批量订阅Topic。通配符使用方法请参见带通配符的自定义Topic 。
描述	可输入文字，描述该Topic类。

带通配符的自定义Topic

物联网平台支持在设备操作权限为**订阅**的自定义Topic中使用两种通配符，以便设备实现批量订阅Topic。

通配符	描述
#	#只能出现在Topic的最后一个类目，代表本级及下级所有类目。 例如：自定义Topic `/a1aycMA****/${deviceName}/user/#` 。设备device1订阅 `/a1aycMA****/device1/user/#` ，表示订阅以 `/a1aycMA****/device1/user/` 为开头的全部Topic，包含 `/a1aycMA****/device1/user/update` 、 `/a1aycMA****/device1/user/update/error` 等Topic。
+	代表本级所有类目。 例如：自定义Topic `/a1aycMA****/${deviceName}/user/+/error` 。设备device1订阅 `/a1aycMA****/device1/user/+/error` ，表示订阅 `/a1aycMA****/device1/user/get/error` 、 `/a1aycMA****/device1/user/update/error` 等Topic。

由于带通配符的Topic实质为一组Topic的集合，因此带通配符的Topic不支持在设备的**Topic列表**页面执行**发布消息**操作，将消息发布到已订阅该Topic的设备。

自定义Topic通信

服务端调用Pub，可向指定的自定义Topic发布消息。设备通过订阅该Topic，接收来自服务端的消息。

使用自定义Topic通信的示例，请参见使用自定义Topic进行通信。

7.使用开放协议自主接入

7.1. MQTT协议接入

7.1.1. MQTT协议规范

MQTT是基于TCP/IP协议栈构建的异步通信消息协议，是一种轻量级的发布、订阅信息传输协议。MQTT在时间和空间上，将消息发送者与接受者分离，可以在不可靠的网络环境中进行扩展，适用于设备硬件存储空间有限或网络带宽有限的场景。物联网平台支持设备使用MQTT协议接入。

支持版本

目前物联网平台支持MQTT标准协议接入，兼容3.1.1和3.1版本协议，具体的协议请参见 MQTT 3.1.1和MQTT 3.1协议文档。

与标准MQTT的区别

- 支持MQTT的PUB、SUB、PING、PONG、CONNECT、DISCONNECT和UNSUB等报文。
- 支持clean session。
- 不支持will、retain msg。
- 支持QoS 0、QoS 1，不支持QoS 2。
- 不支持SUB QoS，消息QoS以发送方（PUB）指定为准。
- 基于原生的MQTT Topic上支持RRPC同步模式，服务器可以同步调用设备并获取设备回执结果。

安全等级

- TCP通道TLS直连模式：安全级别高。

  > 说明
  > - 支持TLS协议1.0、1.1和1.2版本，强烈建议您的设备使用TLS 1.2加密。因TLS 1.0、1.1版本较老，可能有安全风险。
  > - 设备端Link SDK已配置V1.2版本的TLS协议，您无需自行配置。

- TCP直连模式（数据不加密）：安全级别低。
- TCP直连模式，并使用芯片级加密（ID²硬件集成）：安全级别高。

Topic规范

Topic定义及分类，请查看什么是Topic。

系统默认通信类Topic可前往控制台设备详情页查看，功能类Topic可前往具体功能文档页查看。

7.1.2. MQTT-TCP连接通信

本文档主要介绍基于TCP的MQTT连接，连接方式为MQTT客户端直连。

背景信息

在进行MQTT CONNECT协议设置时，注意：

- 如果同一个设备证书（ProductKey、DeviceName和DeviceSecret）或同一组ProductKey、DeviceName、ClientID、DeviceToken同时用于多个物理设备连接，可能会导致客户端频繁上下线。因为新设备连接认证时，原设备会被迫下线，而设备被下线后，又会自动尝试重新连接。
- MQTT连接模式中，设备端Link SDK断开后会自动重连。您可以通过日志服务查看设备行为。

MQTT客户端直连

1. （可选）推荐使用TLS加密。
   - 设备端Link SDK已配置TLS加密，您无需自行配置。
   - 若您自行开发设备端SDK，需要下载根证书。根证书使用方法，请参见mbed TLS。
2. 使用MQTT客户端连接服务器。连接方法，请参见开源MQTT客户端。 如果需了解MQTT协议，请参见MQTT官方文档 。

   > **说明** 若使用第三方代码，阿里云不提供技术支持。

3. MQTT连接。

   建议您使用设备端SDK接入物联网平台。如果您自行开发接入，连接参数如下。

接入域名	○ 对于您购买的实例，接入域名请在物联网平台控制台，找到对应的实例，单击实例进入实例详情查看。 ○ 公共实例的接入域名：`${YourProductKey}.iot-as-mqtt.${YourRegionId}.aliyuncs.com:1883` 。其中： ■ *${YourProductKey}*：请替换为设备所属产品的ProductKey。可登录物联网平台控制台，在对应实例的设备详情页获取。 ■ *${YourRegionId}*：请参见地域和可用区
可变报头（variable header）：Keep Alive	CONNECT指令中需包含Keep Alive（保活时间）。保活心跳时间取值范围为30秒~1200秒。如果心跳时间不在此区间内，物联网平台会拒绝连接。建议取值300秒以上。如果网络不稳定，将心跳时间设置高一些。

	○ 一机一密、一型一密预注册认证方式：使用设备证书（ProductKey、DeviceName和DeviceSecret）连接。
	`mqttClientId: clientId+"|securemode=3,signmethod=hmacsha1,timestamp=132323232|"`
	`mqttUsername: deviceName+"&"+productKey`
	`mqttPassword: sign_hmac(deviceSecret,content)`
	■ mqttClientId：格式中 `||` 内为扩展参数。
	■ clientId：表示客户端ID，建议使用设备的MAC地址或SN码，64个字符内。
	■ securemode：表示目前安全模式，可选值有2（TLS直连模式）和3（TCP直连模式）。
	■ signmethod：表示签名算法类型。支持hmacmd5，hmacsha1和hmacsha256，默认为hmacmd5。
	■ timestamp：表示当前时间毫秒值，可以不传递。

MQTT的CONNECT报文参数	■ mqttPassword：sign签名需把提交给服务器的参数按字典排序后，根据signmethod加签。签名计算示例，请参见MQTT连接签名示例。 ■ content的值为提交给服务器的参数（ProductKey、DeviceName、timestamp和clientId），按照字母顺序排序， 然后将参数值依次拼接。 示例： 假设 clientId = 12345，deviceName = device， productKey = pk， timestamp = 789，signmethod=hmacsha1，deviceSecret=secret ，那么使用TCP方式提交给MQTT的参数如下： mqttclientId=12345|securemode=3,signmethod=hmacsha1,timestamp=789| mqttUsername=device&pk mqttPassword=hmacsha1("secret","clientId12345deviceNamedeviceproductKeypktimestamp789").toHexString(); 加密后的Password为二进制转16制字符串，示例结果为： FAFD82A3D602B37FB0FA8B7892F24A477F85**** ○ 一型一密免预注册认证方式：使用ProductKey、DeviceName、ClientID、DeviceToken连接。 mqttClientId: clientId+"|securemode=-2,authType=connwl|" mqttUsername: deviceName+"&"+productKey mqttPassword: deviceToken ■ mqttClientId：格式中 || 内为扩展参数。 ■ clientId、deviceToken：设备动态注册时获得的ClientID、DeviceToken，请参见基于MQTT通道的设备动态注册。 ■ securemode：表示目前安全模式，采用一型一密免预注册时，固定取值为-2。 ■ authType：表示认证方式，采用一型一密免预注册时，固定取值为regnwl。

示例

使用开源MQTT客户端接入物联网平台的示例，请参见：

- Paho-MQTT Go接入示例
- Paho-MQTT C#接入示例
- Paho-MQTT C接入示例
- Paho-MQTT Java接入示例
- Paho-MQTT Android接入示例

MQTT保活

设备端在保活时间间隔内，至少需要发送一次报文，包括ping请求。

如果物联网平台在保活时间内无法收到任何报文，物联网平台会断开连接，设备端需要进行重连。

连接保活时间的取值范围为30秒~1200秒。建议取值300秒以上。

7.1.3. 基于MQTT通道的设备动态注册

直连设备可通过MQTT通道进行动态注册，即使用一型一密连接认证方式连接物联网平台。设备先基于TLS建立与物联网平台的连接，获取TCP连接所需信息，再断开连接，并重新建立TCP连接进行通信。下面介绍动态注册流程。

前提条件

已完成一型一密文档中的以下步骤：

1. 创建产品。
2. 开启动态注册。
3. 添加设备。
4. 产线烧录。

动态注册流程

流程

1. 设备发送CONNECT报文，报文中包含动态注册参数，请求建立连接。

   > 说明 目前，动态注册只支持使用TLS建立连接，不支持TCP直连；动态注册时，云端不会校验MQTT连接的Keep Alive（保活时间），因此可以不用设置Keep Alive时间。

   - MQTT连接域名：
     - 对于您购买的实例，接入域名请在物联网平台控制台，找到对应的实例，单击实例进入**实例详情**查看。
     - 公共实例的连接域名为 `${YourProductKey}.iot-as-mqtt.${YourRegionId}.aliyuncs.com:1883` 。其中：
       - *${YourProductKey}*：请替换为设备所属产品的ProductKey。可登录物联网平台控制台，在对应实例的**设备详情**页获取。
       - *${YourRegionId}*：请参见地域和可用区替换为您的Region ID。
   - CONNECT报文的动态注册参数：
     - 当设备属于您购买的实例，且使用一型一密免预注册认证方式时，动态注册参数如下：

       ```
       mqttClientId: clientId+"|securemode=-2,authType=xxxx,random=xxxx,signmethod=xxxx,instanceId=xxxx|"
       mqttUserName: deviceName+"&"+productKey
       mqttPassword: sign_hmac(productSecret,content)
       ```

     - 当设备属于公共实例，或使用一型一密预注册认证方式时，动态注册参数如下：

       ```
       mqttClientId: clientId+"|securemode=2,authType=xxxx,random=xxxx,signmethod=xxxx|"
       mqttUserName: deviceName+"&"+productKey
       mqttPassword: sign_hmac(productSecret,content)
       ```

参数说明：

- mqttClientId

  参数取值中包含的详细参数如下表所示。

参数	说明
clientId	客户端ID。建议使用设备的MAC地址或SN码，长度在64个字符内。
securemode	安全模式。 ■ 一型一密预注册认证方式：固定取值为2。 ■ 一型一密免预注册认证方式：固定取值为-2。
authType	一型一密认证方式，不同类型将返回不同的认证参数： ■ register：一型一密预注册认证方式，返回DeviceSecret。 ■ regnwl：一型一密免预注册认证方式，返回DeviceToken、ClientID。
random	随机数。您自定义随机数。
signMethod	签名算法。目前支持hmacmd5、hmacsha1、hmacsha256。
instanceId	实例ID。请登录物联网平台控制台，在实例概览页面查看。

- mqttUserName

  组成结构：`deviceName+"&"+productKey`

  示例：`device1&al123456789`

- mqttPassword

  计算方法：`sign_hmac(productSecret,content)`

  其中，content的值是提交给服务器的必需参数和值（deviceName、productKey、random）按照字母顺序排序、拼接（无拼接符号）的字符串。然后，将content的值通过mqttClientId中的signMethod指定的算法，使用产品的ProductSecret进行签名计算。

  示例：`hmac_sha1(h1nQFYPZS0mW****, deviceNamedevice1productKeyal123456789random123)`

2. 物联网平台返回CONNECT ACK。

   返回0，则表示建连成功，即动态注册成功。

   建连失败，则需根据返回的错误码，确定错误原因。

   设备发送连接请求后，物联网平台返回的结果状态码和说明如下表。

结果码	消息	说明
0	CONNECTION_ACCEPTED	动态注册成功。

结果码	消息	说明
2	IDENTIFIER_REJECTED	参数错误。原因可能是： ○ 必填参数缺失或格式错误。 ○ 您使用了TCP直连注册。动态注册只能使用TLS通道。
3	SERVER_UNAVAILABLE	云端错误。请稍后再试。
4	BAD_USERNAME_OR_PASSWORD	动态注册失败，鉴权未通过。 请检查传入的mqttUserName和mqttPassword取值是否正确。

3. 建立连接后，物联网平台使用Topic：`/ext/register`，根据CONNECT报文中的authType，返回不同的认证参数：

   > 说明 设备无需订阅推送证书的Topic。

   - authType取值为register：一型一密预注册认证方式，返回DeviceSecret。

     物联网平台推送的消息Payload格式如下：

     ```
     {
       "productKey" : "xxx",
       "deviceName" : "xxx",
       "deviceSecret" : "xxx"
     }
     ```

   - authType取值为regnwl：一型一密免预注册认证方式，返回ClientID、DeviceToken。

     物联网平台推送的消息Payload格式如下：

     ```
     {
       "productKey" : "xxx",
       "deviceName" : "xxx",
       "clientId" : "xxx",
       "deviceToken" : "xxx"
     }
     ```

4. 设备收到并保存DeviceSecret，或ClientID和DeviceToken的组合，断开当前MQTT连接。

   设备可以通过发送DISCONNECT报文或直接断开TCP连接，断开当前连接。

   如果设备未断开此连接，15秒之后，物联网平台会主动断开连接。

   如果您使用Eclipse Paho MQTT客户端，设置`MqttConnectOptions.setAutomaticReconnect(false)`关闭自动重连。否则，注册成功并TCP断连后，重连逻辑会发起新的动态注册请求。

5. 设备使用DeviceSecret，或使用ClientID和DeviceToken的组合，再次发起MQTT连接请求，建立设备

与物联网平台的连接，进行消息通信。详情请参见MQTT-TCP连接通信。

7.1.4. MQTT-WebSocket连接通信

物联网平台支持基于WebSocket的MQTT协议。您可以首先使用WebSocket建立连接，然后在WebSocket通道上，使用MQTT协议进行通信，即MQTT over WebSocket。

背景信息

使用WebSocket方式主要有以下优势：

- 使基于浏览器的应用程序可以像普通设备一样，具备与服务端建立MQTT长连接的能力。
- WebSocket方式使用443端口，消息可以顺利穿过大多数防火墙。

操作步骤

1. 证书准备。

   WebSocket可以使用ws和wss两种方式，ws就是普通的WebSocket连接，wss就是增加了TLS加密。如果使用wss方式进行安全连接，需要使用和TLS直连一样的根证书。

2. 客户端选择。

   直接使用官方客户端，只需要替换连接URL即可。其他语言版本客户端或者是自主接入，请参考开源MQTT客户端参考，使用前请阅读相关客户端的说明，是否支持WebSocket方式。

3. 连接说明。

   使用WebSocket方式进行连接，区别主要在MQTT连接URL的协议和端口号，MQTT连接参数和TCP直接连接方式完全相同，其中要注意securemode参数，使用wss方式连接时securemode=2，使用ws方式连接时securemode=3。

   - 接入域名：
     - 对于您购买的实例，接入域名请在物联网平台控制台，找到对应的实例，单击实例进入**实例详情**查看。
     - 公共实例的接入域名：`${YourProductKey}.iot-as-mqtt.${YourRegionId}.aliyuncs.com` 。其中：
       - *${YourProductKey}*：请替换为设备所属产品的ProductKey。可登录物联网平台控制台，在对应实例的**设备详情**页获取。
       - *${YourRegionId}*：请参见地域和可用区替换为您的Region ID。
   - 端口：443。
   - 可变报头（variable header）：Keep Alive。

     Connect指令中需包含Keep Alive（保活时间）。保活心跳时间取值范围为30至1200秒。如果心跳时间不在此区间内，物联网平台会拒绝连接。建议取值300秒以上。如果网络不稳定，将心跳时间设置高一些。

     设备端在保活时间间隔内，至少需要发送一次报文，包括PING请求。

     如果物联网平台在保活时间内无法收到任何报文，物联网平台会断开连接，设备端需要进行重连。
   - MQTT的Connect报文参数如下：

```
mqttClientId: clientId+"|securemode=3,signmethod=hmacsha1,timestamp=132323232|"
mqttUsername: deviceName+"&"+productKey
mqttPassword: sign_hmac(deviceSecret,content)sign签名需要把以下参数按字典序排序后，再根据signmethod加签。
content=提交给服务器的参数（productKey,deviceName,timestamp,clientId），按照字母顺序排序，然后将参数值依次拼接
```

其中，

- clientId：表示客户端ID，建议mac或sn，64字符内。
- timestamp：表示当前时间毫秒值，可选。
- mqttClientId：格式中 || 内为扩展参数。
- signmethod：表示签名算法类型。
- securemode：表示目前安全模式，可选值有2（wss协议）和3（ws协议）。

参考示例，如果预置前提如下：

```
clientId = 12345，deviceName = device， productKey = pk， timestamp = 789，signmethod=hmacsha1，deviceSecret=secret
```

- 使用ws方式
  - 连接域名

    ```
    ws://pk.iot-as-mqtt.cn-shanghai.aliyuncs.com:443
    ```

  - 连接参数

    ```
    mqttclientId=12345|securemode=3,signmethod=hmacsha1,timestamp=789|
    mqttUsername=device&pk
    mqttPasswrod=hmacsha1("secret","clientId12345deviceNamedeviceproductKeypktimestamp789").toHexString();
    ```

- 使用wss方式
  - 连接域名

    ```
    wss://pk.iot-as-mqtt.cn-shanghai.aliyuncs.com:443
    ```

  - 连接参数

    ```
    mqttclientId=12345|securemode=2,signmethod=hmacsha1,timestamp=789|
    mqttUsername=device&pk
    mqttPasswrod=hmacsha1("secret","clientId12345deviceNamedeviceproductKeypktimestamp789").toHexString();
    ```

建议您使用设备端SDK接入物联网平台。如果您自行开发接入，可参考MQTT连接签名示例。

7.1.5. MQTT连接签名示例

若您不使用阿里云提供的设备端SDK，而是使用其他方式，自己进行开发使您的设备使用MQTT协议与物联网平台连接，您可以参见本文提供的签名代码示例进行MQTT连接签名。

MQTT IoT 物联网平台 设备 签名 物联网 签名函数

说明

推荐您使用阿里云提供的设备端SDK。使用阿里云提供的任何一种语言的设备端SDK，则不用您自己配置签名机制。请访问下载设备端SDK查看阿里云提供的SDK下载路径。

如果您不使用阿里云提供的设备端SDK，而是使用其他方式将您的设备接入物联网平台，需了解：

- 需要您自己保证连接的稳定性、MQTT连接保活和MQTT连接断开重连。
- 不使用设备端SDK连接阿里云物联网平台导致的连接问题，阿里云不负责相关的技术支持。
- 如果您要使用物联网平台提供的OTA、物模型、一型一密等多种功能，需您自己去编写这些功能的实现。这将会耗费较多的开发时间、以及bug修复时间。

签名计算代码示例

若您不使用阿里云物联网平台设备端SDK，可单击以下链接，访问相关代码示例页面。

- sign_mqtt.c：实现签名函数的代码示例。
- sign_api.h：定义签名函数用到的数据结构的代码示例。
- sign_sha256.c：签名函数可能使用的算法实现代码示例。如果您自己的平台上有HMACSHA256的实现，可以不编译本文件，但是需要提供函数 `utils_hmac_sha256()` 给*sign_mqtt.c*中的API调用。
- 用于测试签名函数的代码示例。

签名函数API接口说明

函数原型	`int32_t IOT_Sign_MQTT(iotx_mqtt_region_types_t region,`
	`iotx_dev_meta_info_t *meta,`
	`iotx_sign_mqtt_t *signout);`

函数功能	根据输入的IoT设备身份认证信息，输出连接到阿里云物联网平台时所需要的域名、MQTT ClientID、MQTT Username、MQTT Password。之后，您可以将这些信息提供给MQTT Client用于连接阿里云物联网平台。

输入参数	输入参数内容包括：
	• region：指定设备需要连接的阿里云站点。
	代码示例：
	typedef enum {
	IOTX_CLOUD_REGION_SHANGHAI, /* Shanghai */
	IOTX_CLOUD_REGION_SINGAPORE, /* Singapore */
	IOTX_CLOUD_REGION_JAPAN, /* Japan */
	IOTX_CLOUD_REGION_USA_WEST, /* America */
	IOTX_CLOUD_REGION_GERMANY, /* Germany */
	IOTX_CLOUD_REGION_CUSTOM, /* Custom setting */
	IOTX_CLOUD_DOMAIN_MAX /* Maximum number of domain */
	} iotx_mqtt_region_types_t;
	• meta：指定设备的身份认证信息。
	说明 API调用者需为meta分配内存。
	代码示例：
	typedef struct _iotx_dev_meta_info {
	char product_key[IOTX_PRODUCT_KEY_LEN + 1];
	char product_secret[IOTX_PRODUCT_SECRET_LEN + 1];
	char device_name[IOTX_DEVICE_NAME_LEN + 1];
	char device_secret[IOTX_DEVICE_SECRET_LEN + 1];
	} iotx_dev_meta_info_t;
	其中包含的参数：
	○ product_key：设备所属产品的ProductKey。
	○ product_secret：设备所属产品的ProductSecret。
	○ device_name：设备名称DeviceName。
	○ device_secret：设备的DeviceSecret。

输出参数	signout：输出的数据，该数据将用于MQTT连接。
	代码示例：
	`typedef struct {`
	`  char hostname[DEV_SIGN_HOSTNAME_MAXLEN];`
	`  uint16_t port;`
	`  char clientid[DEV_SIGN_CLIENT_ID_MAXLEN];`
	`  char username[DEV_SIGN_USERNAME_MAXLEN];`
	`  char password[DEV_SIGN_PASSWORD_MAXLEN];`
	`} iotx_sign_mqtt_t;`
	其中包含参数：
	• hostname：完整的阿里云物联网站点域名。
	• port：阿里云站点的端口号。
	• clientid：MQTT建立连接时需要指定的ClientID。建议使用设备的MAC地址或SN码，64字符内。
	• username：MQTT建立连接时需要指定的Username。由设备名DeviceName、符号（&）和产品ProductKey组成，格式：`deviceName+"&"+productKey`。示例：`Device1&alSseIs****`。
	• password：MQTT建立连接时需要指定的Password。把提交给服务器的参数按字典排序并拼接后，使用hmacsha256方法和设备的DeviceSecret，加签生成Password。
	具体参数说明，请参见MQTT-TCP连接通信。

返回值	• 0：表示成功。 • -1：表示输入参数非法而失败。

签名API使用示例

以下以sign_test.c中的测试代码为例。

```
#include <stdio.h>
#include <string.h>
#include "sign_api.h"  //包含签名所需的各种数据结构定义

//下面的几个宏用于定义设备的阿里云身份认证信息：ProductKey、ProductSecret、DeviceName、DeviceSecr
et
//在实际产品开发中，设备的身份认证信息应该是设备厂商将其加密后存放于设备Flash中或者某个文件中，
//设备上电时将其读出后使用
#define EXAMPLE_PRODUCT_KEY    "a1X2bEn****"
#define EXAMPLE_PRODUCT_SECRET  "7jluWm1zql7b****"
#define EXAMPLE_DEVICE_NAME    "example1"
#define EXAMPLE_DEVICE_SECRET   "ga7XA6KdlEeiPXQPpRbAjOZXwG8y****"

int main(int argc, char *argv[])
{
    iotx_dev_meta_info_t meta_info;
    iotx_sign_mqtt_t sign_mqtt;

    memset(&meta_info, 0, sizeof(iotx_dev_meta_info_t));
    //下面的代码是将上面静态定义的设备身份信息赋值给meta_info
    memcpy(meta_info.product_key, EXAMPLE_PRODUCT_KEY, strlen(EXAMPLE_PRODUCT_KEY));
    memcpy(meta_info.product_secret, EXAMPLE_PRODUCT_SECRET, strlen(EXAMPLE_PRODUCT_SECRET))
;
    memcpy(meta_info.device_name, EXAMPLE_DEVICE_NAME, strlen(EXAMPLE_DEVICE_NAME));
    memcpy(meta_info.device_secret, EXAMPLE_DEVICE_SECRET, strlen(EXAMPLE_DEVICE_SECRET));

    //调用签名函数，生成MQTT连接时需要的各种数据
    IOT_Sign_MQTT(IOTX_CLOUD_REGION_SHANGHAI, &meta_info, &sign_mqtt);

    ...

}
```

7.1.6. 基于IPv6的MQTT连接通信

物联网平台支持设备端使用基于IPv6协议的MQTT通道接入物联网平台。

背景信息

- 目前，仅华东2（上海）地域支持基于IPv6协议的MQTT通道。

- 环境验证测试时，请使用以下域名和端口，通过MQTT直连方式接入物联网平台。

  测试环境使用域名：`ipv6.itls.cn-shanghai.aliyuncs.com`

  端口：1883

  传输加密：TLSv1.2

  > **说明** 请勿将测试域名用于正式使用场景。

设备端接入物联网平台

正式环境中，设备端必须通过产品对应的正式MQTT连接域名接入物联网平台。

1. 登录工单系统，提交工单，申请开通产品对应的正式MQTT连接域名的AAAA记录。

   产品对应的正式MQTT连接域名：`${YourProductKey}.iot-as-mqtt.cn-shanghai.aliyuncs.com` 。请将*${YourProductKey}*替换为您的产品的PorductKey。

2. 下载用于TLS加密的根证书。

3. 开发设备端，配置MQTT连接。

   建议您使用阿里云提供的设备端SDK接入物联网平台。如果您自行开发设备端，签名时可参见MQTT连接签名示例。

   需配置的信息如下表。

字段	具体信息
连接域名和端口	`${YourProductKey}.iot-as-mqtt.cn-shanghai.aliyuncs.com:1883` 请将*${YourProductKey}*替换为您的产品的PorductKey。
可变报头（variable header）：Keep Alive	CONNECT指令中需包含Keep Alive（保活时间）。保活心跳时间取值范围为30至1200秒。如果心跳时间不在此区间内，物联网平台会拒绝连接。建议取值300秒以上。如果网络不稳定，将心跳时间设置高一些。

字段	具体信息
MQTT的CONNECT报文参数	mqttClientId: clientId+"|securemode=3,signmethod=hmacsha1,timestamp=132323232|" mqttUsername: deviceName+"&"+productKey mqttPassword: sign_hmac(deviceSecret,content) mqttPassword：sign签名需把提交给服务器的参数按字典排序后，根据signmethod加签。 content的值为提交给服务器的参数（ProductKey、DeviceName、timestamp和clientId），按照字母顺序排序，然后将参数值依次拼接。 ○ clientId：表示客户端ID，建议使用设备的MAC地址或SN码，64字符内。 ○ timestamp：表示当前时间毫秒值，可以不传递。 ○ mqttClientId：格式中 || 内为扩展参数。 ○ signmethod：表示签名算法类型。支持hmacmd5，hmacsha1和hmacsha256，默认为hmacmd5。 ○ securemode：表示目前安全模式，可选值有2（TLS直连模式）和3（TCP直连模式）。 示例： 假设 clientId = 12345，deviceName = device， productKey = pk， timestamp = 789，signmethod=hmacsha1，deviceSecret=secret ，那么使用TCP方式提交给MQTT的参数如下： mqttclientId=12345|securemode=3,signmethod=hmacsha1,timestamp=789| mqttUsername=device&pk mqttPassword=hmacsha1("secret","clientId12345deviceNamedeviceproductKeypktimestamp789").toHexString(); 加密后的Password为二进制转16制字符串，示例结果为： FAFD82A3D602B37FB0FA8B7892F24A477F85****

更多关于MQTT-TCP连接信息，请参见MQTT-TCP连接通信。

7.2. CoAP协议接入

7.2.1. CoAP协议规范

本文介绍物联网平台支持的CoAP协议规范。

协议版本

支持 RFC 7252 Constrained Application Protocol协议，具体请参见：RFC 7252。

通道安全

使用 DTLS v1.2保证通道安全，具体请参见：DTLS v1.2。

开源客户端

http://coap.technology/impls.html。

> 说明 若使用第三方代码，阿里云不提供技术支持。

限制

- 仅华东2（上海）、华北2（北京）、华南1（深圳）地域支持CoAP通信。
- 暂时不支持资源发现。
- 仅支持UDP协议，目前支持DTLS和对称加密两种安全模式。

说明

- URI规范，CoAP的URI资源和MQTT Topic保持一致，参见MQTT协议规范。
- Topic规范和MQTT Topic一致，CoAP协议内 `coap://host:port/topic/${topic}` 接口对于所有 `${topic}` 和MQTT Topic可以复用。
- 客户端缓存认证返回的token是请求的令牌。
- 传输的数据大小依赖于MTU的大小，建议在1 KB以内。
- 如设备在10分钟内使用CoAP协议上报过数据，则设备在物联网平台控制台显示为在线状态。

7.2.2. CoAP连接通信

物联网平台支持CoAP协议连接通信。CoAP协议适用在资源受限的低功耗设备上，尤其是NB-IoT的设备使用。本文介绍基于CoAP协议进行设备接入的流程，及使用DTLS和对称加密两种认证方式下的自主接入流程。

基础流程

基于CoAP协议将NB-IoT设备接入物联网平台的流程如下图所示。

基础流程说明如下：

1. 在设备端NB-IoT模块中，集成阿里云物联网平台SDK。厂商在物联网平台控制台申请设备证书（ProductKey、DeviceName和DeviceSecret）并烧录到设备中。
2. NB-IoT设备通过运营商的蜂窝网络进行入网。需要联系当地运营商，确保设备所属地区已经覆盖NB网络，并已具备NB-IoT入网能力。
3. 设备入网成功后，NB设备产生的流量数据及产生的费用数据，将由运营商的M2M平台管理。此部分平台能力由运营商提供。
4. 设备开发者可通过CoAP/UDP协议，将设备采集的实时数据上报到阿里云物联网平台，借助物联网平台，实现海量亿级设备的安全连接和数据管理能力。并且，可通过规则引擎，将数据转发至阿里云的大数据产品、云数据库、表格存储等服务中进行处理。
5. 物联网平台提供相关的数据开放接口和消息推送服务，可将数据转发到业务服务器中，实现设备资产与

实际应用的快速集成。

使用对称加密自主接入

1. 连接CoAP服务器。Endpoint地址：
   - 对于您购买的实例，接入域名请在物联网平台控制台，找到对应的实例，单击实例进入**实例详情**查看。
   - 华东2（上海）地域，公共实例的Endpoint地址为 `${YourProductKey}.coap.cn-shanghai.link.aliyuncs.com:${port}` 。
     - *${YourProductKey}*：请替换为设备所属产品的ProductKey。可登录物联网平台控制台，在对应实例的**设备详情**页获取。
     - `${port}` ：端口。使用对称加密时端口为5682。
2. 设备认证。

   设备认证请求：

```
POST /auth
Host: ${YourEndpoint}
Port: 5682
Accept: application/json or application/cbor
Content-Format: application/json or application/cbor
payload: {"productKey":"a1NUjcV****","deviceName":"ff1a11e7c08d4b3db2b1500d8e0e55","clientId"
:"a1NUjcV****&ff1a11e7c08d4b3db2b1500d8e0e55","sign":"F9FD53EE0CD010FCA40D14A9FE******", "s
eq":"10"}
```

设备认证参数说明

参数	说明
Method	请求方法，只支持POST方法。
URL	URL地址，取值：/auth。
Host	Endpoint地址。
Port	端口，取值：5682。
Accept	设备接收的数据编码方式。目前，支持两种方式：*application/json*和*application/cbor*。
Content-Format	设备发送给物联网平台的上行数据的编码格式，目前，支持两种方式： *application/json*和*application/cbor*。
payload	设备认证信息内容，JSON数据格式。具体参数，请参见下表Payload说明。

Payload 说明

字段名称	是否必需	说明
productKey	是	设备证书信息中ProductKey的值，是物联网平台为产品颁发的全局唯一标识。可从物联网平台控制台对应实例下的**设备详情**页获取。
deviceName	是	设备证书信息中DeviceName的值，在注册设备时自定义的或自动生成的设备名称。可从物联网平台控制台对应实例下的**设备详情**页获取。
ackMode	否	通信模式。取值： ○ *0*：request/response 是携带模式，即客户端发送请求到服务端后，服务端处理完业务，回复业务数据和ACK。 ○ *1*：request/response 是分离模式，即客户端发送请求到服务端后，服务端先回复一个确认ACK，然后再处理业务后，回复业务数据。 若不传入此参数，则默认为携带模式。
sign	是	签名。 您需根据签名计算方法：`signmethod(DeviceSecret,content)`，计算出的值作为sign的值。支持hmacmd5和hmacsha1方法。 签名计算所需参数： ○ signmethod：签名方法，需与您的传入signmethod取值一致。 ○ DeviceSecret：设备的DeviceSecret。可在物联网平台控制台对应实例下的**设备详情**页查看。 ○ content：是将所有提交给服务器的参数（除version、sign、resources和signmethod外），按照英文字母升序，依次拼接排序（无拼接符号）。 说明 用于签名计算的参数值需与设备认证请求中提交的参数值一致。 签名计算示例： hmac_md5(mRPVdzSMu2nVBxzK77ERPIMxSYIv****, clientIda1NUjcV****&ff1a11e7c08d4b3db2b1500d8e0e55deviceNameff1a11e7c08d4b3db2b1500d8e0e55productKeya1NUjcV****seq10timestamp1524448722000)
signmethod	否	算法类型，支持hmacmd5和hmacsha1。默认是hmacmd5。
clientId	是	客户端ID，长度需在64字符内。建议使用设备的的MAC地址或SN码作为clientId的值。
timestamp	否	时间戳。目前，时间戳不做时间窗口校验。

返回结果示例：

```
{"random":"ad2b3a5eb51d6****","seqOffset":1,"token":"MZ8m37hp01w1SSqoDFzo001050****.ad2b"}
```

返回参数说明

字段名称	说明
random	用于后续上、下行加密，组成加密Key。
seqOffset	认证seq偏移初始值。
token	设备认证成功后，返回的Token值。

3. 上报数据。

上报数据请求：

```
POST /topic/${topic}
Host: ${YourEndpoint}
Port: 5682
Accept: application/json or application/cbor
Content-Format: application/json or application/cbor
payload: ${your_data}
CustomOptions: number:2088(标识token), 2089(seq)
```

上报数据参数说明

字段名称	是否必需	说明
Method	是	请求方法，只支持POST方法。
URL	是	传入格式：`/topic/${topic}`。其中，变量*${topic}*需替换为设备数据上行Topic。
Host	是	Endpoint地址。
Port	是	端口，取值：5682。
Accept	是	设备接收的数据编码方式。目前，支持两种方式：*application/json*和*application/cbor*。
Content-Format	是	上行数据的编码格式，服务端对此不做校验。目前，支持两种方式：*application/json*和*application/cbor*。

字段名称	是否必需	说明
payload	是	待上传的数据经高级加密标准（AES）加密后的数据。 说明 AES加密时，Transform为 AES/CBC/PKCS5Padding ，初始向量IV为 543yhjy97ae7fyfg ，Key由sha256算法生成。 Key生成示例： 假设 deviceSecret=zPwChiLh0EaifR809D5Rc6LDIC6A**** ，设备认证返回 random=8fe3c8d50e10**** 。 i. 将deviceSecret和random按照 ${deviceSecret},${random} 格式组成以下字符串。 zPwChiLh0EaifR809D5Rc6LDIC6A****,8fe3c8d50e10**** ii. 使用sha256对以上字符串的UTF-8编码结果进行加密，并转换为16进制字符串。 59ea5ac1cb092e5910c405821119959e5297516d185b71e344735cf3f268**** iii. 从上一步得到的字符串的第17位开始，截取长度为32位的字符串（subString(16,48)），得到密钥。 10c405821119959e5297516d185b71e3

字段名称	是否必需	说明
CustomOptions	是	option值有2088和2089两种类型，说明如下： ○ 2088：表示token，取值为设备认证后返回的token值。 说明 每次上报数据都需要携带token信息。如果token失效，需要重新进行设备认证，获取token。 ○ 2089：表示seq，取值需比设备认证后返回的seqOffset值更大，且在认证生效周期内不重复的随机值。建议设置为根据每次请求数据包中的seq递增值，使用上一栏所介绍的方法进行AES加密填充。 option返回示例： number:2090(云端消息ID) token和seq除了写在options中，还可以写在URI中，例如 /topic/${topic}?token=xxxx&seq=xxxx 。如果同时存在于options和URI中，以options为准。

消息上行成功后，返回成功状态码，同时返回物联网平台生成的消息ID。

使用DTLS自主接入

1. 连接CoAP服务器。 Endpoint地址：
   - 对于您购买的实例，接入域名请在物联网平台控制台，找到对应的实例，单击实例进入**实例详情**查看。
   - 华东2（上海）地域，公共实例的Endpoint地址为 ${YourProductKey}.coap.cn-shanghai.link.aliyuncs.com:${port} 。
     - *${YourProductKey}*：请替换为设备所属产品的ProductKey。可登录物联网平台控制台，在对应实例的**设备详情**页获取。
     - *${port}*：端口。使用DTLS时，端口为5684。
2. 如果您使用我们提供的设备端SDK，则DTLS安全通道默认使用PSK加密算法。如果您未使用我们提供的设备端SDK，则需要下载DTLS安全通道根证书，自行使用DTLS库连接物联网平台，PSK对应的加密方式如下：

```
psk_id: "${authType}" + "|" + "${signMethod}" + "|" + "${productKey}" + "&" + "${deviceName}" + "timestamp"
psk: signMethod(DeviceSecret, "${productKey}" + "&" + "${deviceName}" + "${timestamp}")
```

字段说明

字段	是否必需	说明
authType	是	认证类型，这里设为固定值：devicename。
signMethod	是	算法类型，支持hmacmd5、hmacsha1、hmacsha256。
productKey	是	设备所属产品的ProductKey。
deviceName	是	设备名称，DeviceName。
DeviceSecret	是	设备的DeviceSecret
timestamp	是	时间戳。

3. 设备认证。使用auth接口认证设备，获取Token。上报数据时，需携带Token信息。

   设备认证请求：

```
POST /auth
Host: ${YourEndpoint}
Port: 5684
Accept: application/json or application/cbor
Content-Format: application/json or application/cbor
payload: {"productKey":"ZG1EvTE****","deviceName":"NlwaSPXsCpTQuh8FxBGH","clientId":"mylight
1000002","sign":"bccb3d2618afe74b3eab12b94042****"}
```

   除 Port参数外，其他参数及Payload内容说明，可参见使用对称加密自主接入。

   返回结果示例：

```
response：{"token":"f13102810756432e85dfd351eeb4****"}
```

返回码说明

Code	描述	Payload	备注
2.05	Content	认证通过：Token对象	正确请求。
4.00	Bad Request	no payload	请求发送的Payload非法。
4.01	Unauthorized	no payload	未授权的请求。
4.03	Forbidden	no payload	禁止的请求。
4.04	Not Found	no payload	请求的路径不存在。
4.05	Method Not Allowed	no payload	请求方法不是指定值。

Code	描述	Payload	备注
4.06	Not Acceptable	no payload	Accept不是指定的类型。
4.15	Unsupported Content-Format	no payload	请求的content不是指定类型。
5.00	Internal Server Error	no payload	auth服务器超时或错误。

4. 上行数据。

   设备发送数据到某个Topic，只支持发布权限的Topic，支持自定义Topic。

   例如：Topic为 `/${YourProductKey}/${YourDeviceName}/pub` ，假设当前设备名称为device，所属产品的ProductKey为a1GFjLP****，那么您可以使用 `a1GFjLP****.coap.cn-shanghai.link.aliyuncs.com:5684/topic/a1GFjLP****/device/pub` 地址来上报数据。

   上报数据请求：

```
POST /topic/${topic}
Host: ${YourEndpoint}
Port: 5684
Accept: application/json or application/cbor
Content-Format: application/json or application/cbor
payload: ${your_data}
CustomOptions: number:2088(标识token)
```

上报数据请求参数说明

参数	是否必需	说明
Method	是	请求方法。支持POST方法。
URL	是	`/topic/${topic}` 。其中，变量 `${topic}` 需替换为当前设备对应的Topic。
Host	是	Endpoint地址。
Port	是	端口，取值：5684。
Accept	是	设备接收的数据编码方式。目前，支持两种方式：*application/json*和*application/cbor*。
Content-Format	是	上行数据的编码格式，服务端对此不做校验。目前，支持两种方式：*application/json*和*application/cbor*。

参数	是否必需	说明
CustomOptions	是	○ number取值：2088。 ○ token为设备认证（auth）返回的token值。 说明 每次上报数据都需要携带token信息。如果token失效，需要重新进行设备认证，获取token。

7.3. HTTP协议接入

7.3.1. HTTP协议规范

物联网平台支持HTTPS协议。本文介绍物联网平台支持的HTTP协议规范。

HTTP协议版本

- 支持 Hypertext Transfer Protocol — HTTP/1.0 协议，具体请参见：RFC 1945
- 支持 Hypertext Transfer Protocol — HTTP/1.1 协议，具体请参见：RFC 2616

通道安全

使用HTTPS（Hypertext Transfer Protocol Secure协议）保证通道安全。

说明
- 支持TLS协议1.0、1.1和1.2版本，强烈建议您的设备使用TLS 1.2加密。因TLS 1.0、1.1版本较老，可能有安全风险。
- 设备端Link SDK已配置V1.2版本的TLS协议，您无需自行配置。

限制

- 仅支持HTTPS。
- 不支持以问号（?）形式传参数。
- 暂时不支持资源发现。

说明

- URI规范，HTTP的URI资源和MQTT Topic保持一致，请参见MQTT协议规范。
- 如设备在10分钟内使用HTTP协议上报过数据，则设备在物联网平台控制台显示为在线状态。

7.3.2. HTTP连接通信

物联网平台支持使用HTTP接入，目前仅支持HTTPS协议。下面介绍使用HTTP连接通信的接入流程。

限制说明

- 仅华东2（上海）、华北2（北京）、华南1（深圳）地域支持HTTP通信。
- 仅支持HTTPS协议。

- 适合单纯的数据上报场景，数据上行接口传输的数据大小限制为128 KB。
- Topic规范和MQTT的Topic规范一致，可以复用MQTT连接通信的Topic。使用HTTP协议连接，上报数据请求： `${endpoint}/topic/${topic}` 。不支持以 `?query_String=xxx` 格式传参。
- HTTP请求只支持POST方式。
- 设备认证返回的token会在一定周期后失效。目前token有效期是7天，请务必考虑token失效逻辑的处理。

接入流程

接入流程主要包含进行设备认证以获取设备token和采用获取的token进行持续地数据上报。

1. 认证设备，获取设备的token。 Endpoint地址：
   - 对于您购买的实例，接入域名请在物联网平台控制台，找到对应的实例，单击实例进入**实例详情**查看。
   - 华东2（上海）地域，公共实例的Endpoint地址为 `https://iot-as-http.cn-shanghai.aliyuncs.com` 。

   认证设备请求：

```
POST /auth HTTP/1.1
Host: ${YourEndpoint}
Content-Type: application/json
body: {"version":"default","clientId":"mylight1000002","signmethod":"hmacsha1","sign":"4870141D4
067227128CBB4377906C3731CAC221C","productKey":"ZG1EvTE****","deviceName":"NlwaSPXsCpTQu
h8FxBGH","timestamp":"1501668289957"}
```

参数说明

参数	说明
Method	请求方法，只支持POST方法。
URL	URL地址，只支持HTTPS，取值：/auth。
Host	Endpoint地址。
Content-Type	设备发送给物联网平台的上行数据的编码格式，目前只支持application/json。若使用其他编码格式，会返回参数错误。
body	设备认证信息。JSON数据格式。具体信息，请参见下表body参数。

body参数

字段名称	是否必需	说明
productKey	是	设备所属产品的ProductKey。可从物联网平台控制台对应实例下的**设备详情**页获取。
deviceName	是	设备名称。可从物联网平台控制台对应实例下的**设备详情**页获取。

字段名称	是否必需	说明
clientId	是	客户端ID。长度为64字符内，建议以MAC地址或SN码作为clientId。
timestamp	否	时间戳。校验时间戳15分钟内的请求有效。时间戳格式为数值，值为自GMT 1970年1月1日0时0分到当前时间点所经过的毫秒数。
sign	是	签名。 签名计算格式为 `hmacmd5(DeviceSecret,content)` 。 其中，content为将所有提交给服务器的参数（除version、sign和signmethod外），按照英文字母升序，依次拼接排序（无拼接符号）的结果。 签名示例： 假设clientId = 127.0.0.1，deviceName = http_test，productKey = a1FHTWxQ****，timestamp = 1567003778853，signmethod = hmacmd5，deviceSecret = 89VTJylyMRFuy2T3sywQGbm5Hmk1****，签名计算为： `hmacmd5("89VTJylyMRFuy2T3sywQGbm5Hmk1****","clientId127.0.0.1deviceNamehttp_testproductKeya1FHTWxQ****timestamp1567003778853").toHexString();` 其中， `toHexString()` 是将计算结果二进制数据的每个byte按4 bit转化为十六进制字符串，大小写不敏感。例如，计算结果byte数组是：[60 68 -67 -7 -17 99 30 69 117 -54 -58 -58 103 -23 113 71]，转换后得到的字符串为：3C44BDF9EF631E4575CAC6C667E97147。
signmethod	否	算法类型，支持hmacmd5和hmacsha1。 若不传入此参数，则默认为hmacmd5。
version	否	版本号。若不传入此参数，则默认default。

设备认证返回结果示例：

```
body:
{
 "code": 0,
 "message": "success",
 "info": {
  "token": "6944e5bfb92e4d4ea3918d1eda39****"
 }
}
```

> 说明
> - 请将返回的token值缓存到本地。
> - 每次上报数据时，都需要携带token信息。如果token失效，需要重新认证设备获取token。

错误码说明

code	message	备注
10000	common error	未知错误。
10001	param error	请求的参数异常。
20000	auth check error	设备鉴权失败。
20004	update session error	更新失败。
40000	request too many	请求次数过多，流控限制。

2. 上报数据。

设备发送数据到某个Topic，只支持发布权限的Topic，支持自定义Topic。

例如：Topic为 `/${YourProductKey}/${YourDeviceName}/pub` ，假设当前设备名称为device123，产品的ProductKey为a1GFjLP****，那么您可以调用 `https://iot-as-http.cn-shanghai.aliyuncs.com/topic/a1GFjLP****/device123/pub` 地址来上报数据。

上报数据请求：

```
POST /topic/${topic} HTTP/1.1
Host: ${YourEndpoint}
password:${token}
Content-Type: application/octet-stream
body: ${your_data}
```

上报数据参数说明

参数	说明
Method	请求方法，只支持POST方法。
URL	`/topic/${topic}` 。其中，变量 `${topic}` 需替换为数据发往的目标Topic。只支持HTTPS。
Host	Endpoint地址。
password	放在Header中的参数，取值为调用设备认证接口auth返回的token值。

参数	说明
Content-Type	设备发送给物联网平台的上行数据的编码格式，目前仅支持application/octet-stream。若使用其他编码格式，会返回参数错误。
body	发往${topic}的数据内容。

返回结果示例：

```
body:
{
 "code": 0,
 "message": "success",
 "info": {
  "messageId": 892687627916247040,
 }
}
```

错误码说明

code	message	备注
10000	common error	未知错误。
10001	param error	请求的参数异常。
20001	token is expired	token失效。需重新调用auth进行鉴权，获取token。
20002	token is null	请求header中无token信息。
20003	check token error	根据token获取identify信息失败。需重新调用auth进行鉴权，获取token。
30001	publish message error	数据上行失败。
40000	request too many	请求次数过多，流控限制。

8.泛化协议

8.1. 什么是泛化协议SDK

阿里云物联网平台支持基于MQTT、CoAP和HTTP协议的通信，其他类型协议，如消防协议GB/T 26875.3-2011、Modbus、JT808等暂未接入。在特定场景下，有些设备可能无法直接接入物联网平台。此时，您需要使用泛化协议SDK，快速构建桥接服务，搭建设备或平台与阿里云物联网平台的双向数据通道。

说明

支持泛化协议SDK的地域包括：华东2（上海）、华北2（北京）、华南1（深圳）、德国（法兰克福）和美国（弗吉尼亚）。

服务架构

泛化协议SDK是协议自适应的框架，用以构建与阿里云物联网平台进行高效双向通信的桥接服务。

服务架构如下图所示。

适用场景

泛化协议SDK面向的目标场景包括：

- 由于网络环境或者硬件限制，设备无法直接接入物联网平台。
- 设备只支持某种类型协议，而这种协议目前物联网平台不支持。
- 设备与您的设备接入服务器（网桥Server）之间已有通信网络，您希望在不修改设备和协议的情况下，将设备接入物联网平台。
- 设备直接接入到您的服务器，且需要做一些其他的处理逻辑。

主要功能

泛化协议SDK使得网桥Server具备与物联网平台进行通信的能力。

基础功能：

- 提供基于配置文件的静态配置管理能力。
- 提供设备连接管理能力。
- 提供上行通信能力。
- 提供下行通信能力。

进阶功能如下：

- 提供基于接口的动态配置管理能力。
- 已封装属性、事件、标签数据上报接口供您调用。

名词解释

名词	描述
设备	您的真实物联网场景设备，该设备无法直接使用物联网平台所支持的协议直接与云端通信。

名词	描述
网桥Server	您的设备接入服务器。该服务器使用特定类型协议与设备通信，使用泛化协议SDK与物联网平台通信。
原始协议	设备与网桥Server之间使用的特定类型协议。泛化协议SDK不关心原始协议的具体定义和实现。
原始身份标识符	设备与网桥Server使用原始协议通信时的唯一标识符。泛化协议SDK接口参数中，用originalIdentity表示设备的原始身份标识符。
设备证书	在物联网平台注册设备后，获得的设备证书信息，包括ProductKey、DeviceName、DeviceSecret。使用泛化协议的场景下，不将设备证书烧录到设备上；而是配置泛化协议SDK文件*devices.conf*，由网桥将设备原始身份标识符originalIdentity映射到设备证书信息。
网桥证书	在物联网平台注册网桥设备后，获得的网桥设备证书信息，包括ProductKey、DeviceName、DeviceSecret，用于在云端标识网桥的身份。

开发和部署

1. 创建产品与设备。

   在物联网平台控制台，创建产品和设备。请参见创建产品和单个创建设备或批量创建设备。

   获取网桥设备证书信息。在泛化协议SDK配置时，需配置网桥设备证书信息。

   > **说明** 网桥是个虚拟概念，您可以使用任意设备的证书信息作为网桥的证书信息。

2. 配置泛化协议SDK。

   目前，仅提供Java语言的泛化协议SDK，支持JDK 1.8及以上版本。

   泛化协议SDK配置细节，请参见基础用法和进阶用法。

3. 部署服务。

   已开发完成的桥接服务，可以使用阿里云ECS和SLB等服务，以高度可扩展的方式部署至阿里云上；也可以直接部署到本地环境中，以保证可信通信环境。

   以基于阿里云云服务器ECS为例，上线流程如下。

   

8.2. 基础用法

基于泛化协议SDK，通过桥接服务，您的设备可以接入阿里云物联网平台，与物联网平台通信。本文介绍如何配置泛化协议SDK，实现设备上下线和消息上下行等基础能力。

物联网平台提供泛化协议SDK Demo，请访问泛化协议SDK Demo GitHub地址查看。

流程图

使用泛化协议SDK，桥接设备与物联网平台的整体流程图如下。

部署开发环境

部署Java SDK开发环境，并添加泛化协议SDK的项目Maven依赖。

```
<dependency>
 <groupId>com.aliyun.openservices</groupId>
 <artifactId>iot-as-bridge-sdk-core</artifactId>
 <version>2.1.3</version>
</dependency>
```

初始化

- 初始化SDK。

  需要创建一个BridgeBootstrap对象实例，并调用bootstrap方法。泛化协议SDK初始化工作完成后，读取网桥信息，并向云端发起网桥设备上线请求等。

  此外，可以在调用bootstrap方法的同时，向泛化协议SDK注册一个DownlinkChannelHandler回调，用于接收云端下行消息。

  代码示例如下。

  ```
  BridgeBootstrap bridgeBootstrap = new BridgeBootstrap();
  bridgeBootstrap.bootstrap(new DownlinkChannelHandler() {
     @Override
     public boolean pushToDevice(Session session, String topic, byte[] payload) {
        //接收云端下行消息。
        String content = new String(bytes);
        log.info("Get DownLink message, session:{}, {}, {}", session, topic, content);
        return true;
     }

     @Override
     public boolean broadcast(String topic, byte[] payload) {
        return false;
     }
  });
  ```

- 配置网桥信息。

  网桥配置默认使用配置文件方式。默认从Java工程默认资源文件路径（一般是*src/main/resources/*）下的*application.conf*中读取配置文件，格式支持HOCON（JSON超集）。泛化协议SDK使用typesafe.config解析配置文件。

  支持两种网桥配置方法：指定网桥设备和动态注册网桥设备。本文中仅提供指定网桥设备的配置示例；动态注册网桥设备的具体方法，请参见动态创建网桥设备。

  指定网桥配置参数说明如下表所示。

参数	是否必需	说明
productKey	是	网桥所属产品的ProductKey。
deviceName	是	网桥的DeviceName。
deviceSecret	是	网桥的DeviceSecret。
subDeviceConnectMode	否	网桥挂载设备模式： ○ 传入该参数，且取值为3，则为大型网桥，单网桥下最大支持挂载200,000个设备。 ○ 不传入该参数，则为小型网桥，单网桥下最大支持挂载1,5000个设备。 大型网桥、小型网桥的挂载设备下线策略不同，请参见设备下线。
http2Endpoint	是	HTTP2网关服务地址。网桥和云端通过HTTP2协议建立长连接通道。 ○ 对于公共实例，HTTP2网关服务地址的结构为 `https://${productKey}.iot-as-http2.${RegionId}.aliyuncs.com:443` 。 其中，变量*${productKey}*需替换成您的网桥所属产品的ProductKey。 变量*${RegionId}*需替换成您的服务所在地域代码。RegionId的表达方法，请参见地域和可用区。 例如：某用户的网桥设备的productKey为a1abcab****，地域为上海，HTTP2网关服务地址为 `https://a1abcab****.iot-as-http2.cn-shanghai.aliyuncs.com:443` 。 ○ 对于您购买的实例，HTTP2网关服务地址的结构为 `https://${IotInstanceId}.http2.iothub.aliyuncs.com:443` 。 其中，变量*${IotInstanceId}*需替换成实例ID。 例如：某用户的实例的实例ID为iot-cn-g06kwb****，HTTP2网关服务地址为 `https://iot-cn-g06kwb****.http2.iothub.aliyuncs.com:443` 。

参数	是否必需	说明
authEndpoint	是	设备认证服务地址。 ○ 对于公共实例，设备认证服务地址结构为 `https://iot-auth.${RegionId}.aliyuncs.com/auth/bridge` 。 其中，变量*${RegionId}*需替换成您的服务所在地域代码。RegionId的表达方法，请参见地域和可用区。 例如：地域为上海，则认证服务地址为 `https://iot-auth.cn-shanghai.aliyuncs.com/auth/bridge` 。 ○ 对于您购买的实例，设备认证服务地址结构为 `https://${IotInstanceId}.auth.iothub.aliyuncs.com/auth/bridge` 。 其中，变量*${IotInstanceId}*需替换成实例ID。 例如：某用户的实例的实例ID为iot-cn-g06kwb****，则认证服务地址为 `https://iot-cn-g06kwb****.auth.iothub.aliyuncs.com/auth/bridge` 。

以公共云实例为例，指定小型网桥设备证书信息的配置示例如下。

```
# 服务地址
http2Endpoint = "https://a1tN7OBmTcd.iot-as-http2.cn-shanghai.aliyuncs.com:443"
authEndpoint = "https://iot-auth.cn-shanghai.aliyuncs.com/auth/bridge"

# 网桥设备信息
productKey = ${bridge-ProductKey-in-Iot-Plaform}
deviceName = ${bridge-DeviceName-in-Iot-Plaform}
deviceSecret = ${bridge-DeviceSecret-in-Iot-Plaform}
```

设备认证并上线

- 配置设备上线。

  泛化协议SDK中的设备上线接口设置如下。

```
/**
 * 设备认证。
 * @param newSession 设备Session信息，下行回调的时候会将这个Session传递回来。
 * @param originalIdentity 设备原始身份标识符。
 * @return
 */
public boolean doOnline(Session newSession, String originalIdentity);
```

设备上线时，需要传Session。下行消息回调时，会把Session回调给网桥。Session中包含设备的原始身份标识符字段，以便网桥判断消息属于哪个设备。

此外，Session中还有一个可选的channel字段，设计上可以用来存放设备的连接信息。例如，您的网桥Server是基于Netty构建的，这里可以存放设备长连接对应的channel对象，消息下行的时候就可以直接从Session中获取channel进行操作。channel的数据类型是Object。泛化协议SDK不会对channel数据做任何处理。您也可以根据使用场景，在channel中存放任何设备相关的信息。

设备上线代码示例如下。

```
UplinkChannelHandler uplinkHandler = new UplinkChannelHandler();
//创建Session。
Object channel = new Object();
Session session = Session.newInstance(originalIdentity, channel);
//设备上线。
boolean success = uplinkHandler.doOnline(session, originalIdentity);
if (success) {
   //设备上线成功，网桥接受后续设备通信请求。
} else {
   //设备上线失败，网桥可以拒绝后续设备通信请求，如断开连接。
}
```

- 配置映射设备证书信息。

  配置设备原始身份标识符和设备证书信息的映射关系。默认使用配置文件方式，默认从Java工程的默认资源文件路径（一般是*src/main/resources/*）下的*devices.conf*中读取配置文件，格式支持HOCON（JSON超集）。泛化协议SDK使用typesafe.config解析配置文件。

  设备证书信息配置文件内容格式如下。

```
${device-originalIdentity} {
  productKey : ${device-ProductKey-in-Iot-Plaform}
  deviceName : ${device-DeviceName-in-Iot-Platform}
  deviceSecret : ${device-DeviceSceret-in-Iot-Platform}
}
```

参数	是否必需	说明
productKey	是	设备所属产品的ProductKey。
deviceName	是	设备的DeviceName。
deviceSecret	是	设备的DeviceSecret。

设备发送上行数据

泛化协议SDK的设备上报消息接口设置如下。

```
/**
 * 发送设备上行消息，同步调用接口。
 * @param originalIdentity 设备原始身份标识符。
 * @param protocolMsg 待发送消息，包含Topic、消息体、QoS等信息。
 * @param timeout 超时时间，单位秒。
 * @return 超时时间内是否发送成功。
 */
boolean doPublish(String originalIdentity, ProtocolMessage protocolMsg, int timeout);
/**
 * 发送设备上行消息，异步调用接口。
 * @param originalIdentity 设备原始身份标识符。
 * @param protocolMsg 待发送消息，包含Topic、消息体、QoS等信息。
 * @return 调用后立即返回CompletableFuture，调用者可进一步处理该future。
 */
CompletableFuture<ProtocolMessage> doPublishAsync(String originalIdentity,
                                    ProtocolMessage protocolMsg);
```

接口调用代码示例如下。

```
DeviceIdentity deviceIdentity =
   ConfigFactory.getDeviceConfigManager().getDeviceIdentity(originalIdentity);
ProtocolMessage protocolMessage = new ProtocolMessage();
protocolMessage.setPayload("Hello world".getBytes());
protocolMessage.setQos(0);
protocolMessage.setTopic(String.format("/%s/%s/update",
      deviceIdentity.getProductKey(), deviceIdentity.getDeviceName()));
//同步发送。
int timeoutSeconds = 3;
boolean success = upLinkHandler.doPublish(originalIdentity, protocolMessage, timeoutSeconds);
//异步发送。
upLinkHandler.doPublishAsync(originalIdentity, protocolMessage);
```

网桥推送下行数据给设备

网桥在调用bootstrap方法时，向泛化协议SDK注册了DownlinkChannelHandler。当有下行消息的时候，泛化协议SDK就会回调DownlinkChannelHandler的pushToDevice方法。可以在pushToDevice中配置网桥处理下行消息。

> 说明 pushToDevice方法中不要做耗时逻辑，否则会阻塞下行消息接收的线程。如果有耗时或者IO逻辑，如收到云端下行消息后通过网络长连接发给子设备，请采用异步处理。

代码示例如下。

```
private static ExecutorService executorService  = new ThreadPoolExecutor(
    Runtime.getRuntime().availableProcessors(),
    Runtime.getRuntime().availableProcessors() * 2,
    60, TimeUnit.SECONDS,
    new LinkedBlockingQueue<>(1000),
    new ThreadFactoryBuilder().setDaemon(true).setNameFormat("bridge-downlink-handle-%d").build()
,
    new ThreadPoolExecutor.AbortPolicy());
public static void main(String args[]) {
    //默认使用application.conf和devices.conf。
    bridgeBootstrap = new BridgeBootstrap();
    bridgeBootstrap.bootstrap(new DownlinkChannelHandler() {
        @Override
        public boolean pushToDevice(Session session, String topic, byte[] payload) {
            //接收云端下行消息。
            executorService.submit(() -> handleDownLinkMessage(session, topic, payload));
            return true;
        }
        @Override
        public boolean broadcast(String s, byte[] bytes) {
            return false;
        }
    });
}
private static void handleDownLinkMessage(Session session, String topic, byte[] payload) {
    String content = new String(payload);
    log.info("Get DownLink message, session:{}, topic:{}, content:{}", session, topic, content);
    Object channel = session.getChannel();
    String originalIdentity = session.getOriginalIdentity();
}
```

参数	说明
Session	Session是设备在doOnline的时候传递进来的，可以用来区分下行消息是发给哪个设备的。
topic	下行消息的Topic。
payload	二进制格式的下行消息的消息体数据。

设备下线

设备下线分为三种情况：

- 对于小型网桥，网桥与云端之间的连接断开时，所有的设备会自动从云端离线。
- 对于大型网桥，网桥与云端之间的连接断开时，设备不会自动从云端离线，网桥重连后可以通过设备下线接口主动更新设备的状态。

  子设备的状态表示子设备接入网关的状态，由网关上报到物联网平台进行状态的刷新。如果网关不能正常上报子设备的状态信息到物联网平台，则展示的子设备状态不会刷新。

  例如：某子设备通过网关接入到物联网平台，子设备状态为在线状态，如果此时网关与物联网平台断开连接，则网关不能上报子设备的状态到物联网平台，该子设备的状态会一直显示在线。
- 对于小型网桥和大型网桥，网桥与云端之间的连接未断开时，网桥可以主动向云端上报某个设备下线的消息。

  网桥上报设备下线的接口定义如下：

```
/**
 * 向云端上报某个设备下线。
 * @param originalIdentity 设备原始身份标识符。
 * @return 是否成功上报。
 */
boolean doOffline(String originalIdentity);
```

  调用下线接口代码示例如下：

```
upLinkHandler.doOffline(originalIdentity);
```

8.3. 进阶用法

本文介绍泛化协议SDK的一些进阶能力的用法，包括自定义配置文件路径、配置动态创建网桥设备、调用泛化协议SDK中封装的数据上报接口上报属性、事件和标签。

自定义配置管理

默认情况下，网桥的配置文件和设备证书的映射关系配置文件，都是从固定路径的固定文件名（分别是*application.conf*和*devices.conf*）中读取的。泛化协议SDK提供了自定义配置管理的能力，您只需要在调用bootstrap方法之前，先调用*ConfigFactory.init*方法，自定义配置文件的路径，也可以自定义实例实现对应的接口。

自定义配置代码示例：

```
ConfigFactory.init(
    ConfigFactory.getBridgeConfigManager("application-self-define.conf"),
    selfDefineDeviceConfigManager);
bridgeBootstrap.bootstrap();

private static DeviceConfigManager selfDefineDeviceConfigManager = new DeviceConfigManager() {
    @Override
    public DeviceIdentity getDeviceIdentity(String originalIdentity) {
        return devicesMap.get(originalIdentity);
    }

    @Override
    public String getOriginalIdentity(String productKey, String deviceName) {
        return null;
    }
};
```

动态创建网桥设备

当您需要在大量的服务器上部署网桥应用，如果为每一个网桥服务器指定不同的网桥设备信息会比较繁琐。您可以配置网桥信息文件*application.conf*动态创建网桥设备。您需在配置文件中，传入参数productKey和popClientProfile，泛化协议SDK将调用物联网平台开放API，以服务器MAC地址作为设备名称，新创建一个网桥设备。

> 说明
>
> - 采用动态创建网桥设备方法，仅需要修改网桥配置文件，调用代码与基础用法一致。
> - 网桥配置文件中，如果已经指定了网桥设备信息，不会再动态创建设备。仅当配置文件中deviceName和deviceSecret配置为空，且popClientprofile所有配置齐全的情况下，泛化协议SDK才会尝试调用物联网平台API，以服务器MAC地址作为设备名称动态创建设备。如果当前MAC地址已经创建过设备，则会直接使用这个设备作为网桥设备。
> - 如果采用这种方式配置网桥，不建议您使用生产环境的配置直接在本地机器上调试。因为在多个本地PC上调试程序，每次都会将当前机器的MAC地址注册为网桥，并将设备信息配置文件*devices.conf*中的所有设备与该网桥关联。建议在调试阶段使用专门用于测试的设备，以免干扰生产环境。

配置参数说明

参数	是否必需	说明
productKey	是	网桥所属产品的ProductKey。

参数	是否必需	说明
subDeviceConnectMode	否	网桥挂载设备模式： • 传入该参数，且取值为3，则为大型网桥，单网桥下最大支持挂载200,000个设备。 • 不传入该参数，则为小型网桥，单网桥下最大支持挂载1,5000个设备。 大型网桥、小型网桥的挂载设备下线策略不同，请参见设备下线。
http2Endpoint	是	HTTP2网关服务地址。 网桥和云端通过HTTP2协议建立长连接通道。 • 对于公共实例，HTTP2网关服务地址的结构为 `https://${productKey}.iot-as-http2.${RegionId}.aliyuncs.com:443` 。 其中，变量*${productKey}*需替换成您的网桥所属产品的ProductKey。 变量*${RegionId}*需替换成您的服务所在地域ID。RegionId的表达方法，请参见地域和可用区。 例如：某用户的网桥设备的productKey为a1abcab****，地域为上海，HTTP2网关服务地址为 `https://a1abcab****.iot-as-http2.cn-shanghai.aliyuncs.com:443` 。 • 对于您购买的实例，HTTP2网关服务地址的结构为 `https://${IotInstanceId}.http2.iothub.aliyuncs.com:443` 。 其中，变量*${IotInstanceId}*需替换成实例ID。 例如：某用户的实例的实例ID为iot-cn-g06kwb****，HTTP2网关服务地址为 `https://iot-cn-g06kwb****.http2.iothub.aliyuncs.com:443` 。
authEndpoint	是	设备认证服务地址。 • 对于公共实例，设备认证服务地址结构为 `https://iot-auth.${RegionId}.aliyuncs.com/auth/bridge` 。 其中，变量*${RegionId}*需替换成您的服务所在地域ID。RegionId的表达方法，请参见地域和可用区。 例如：地域为上海，则认证服务地址为 `https://iot-auth.cn-shanghai.aliyuncs.com/auth/bridge` 。 • 对于您购买的实例，设备认证服务地址结构为 `https://${IotInstanceId}.auth.iothub.aliyuncs.com/auth/bridge` 。 其中，变量*${IotInstanceId}*需替换成实例ID。 例如：某用户的实例的实例ID为iot-cn-g06kwb****，则认证服务地址为 `https://iot-cn-g06kwb****.auth.iothub.aliyuncs.com/auth/bridge` 。

参数	是否必需	说明
popClientProfile	是	配置此参数，泛化协议SDK将调用阿里云端开放接口自动创建一个网桥设备。 具体参数配置见下表popClientProfile。

popClientProfile

参数	是否必需	描述
accessKey	是	您的阿里云账号的AccessKey ID。 在物联网平台控制台，鼠标移动到您的账号头像上，然后单击AccessKey管理，创建或查看AccessKey。
accessSecret	是	您的阿里云账号的AccessKey Secret。
name	是	将要创建网桥设备的所在地域ID。 地域的表达方法，请参见地域和可用区。
region	是	将要创建网桥设备的所在地域ID。 表达方法同name。
product	是	产品名称，固定为Iot。
endpoint	是	调用指定地域API的节点地址。节点地址结构为 `iot.${RegionId}.aliyuncs.com` 。 其中，变量*${RegionId}*需替换成您的服务所在地域ID。RegionId的表达方法，请参见地域和可用区。 如上海节点的endpoint为 `iot.cn-shanghai.aliyuncs.com` 。

以公共云实例为例，动态创建小型网桥设备的配置示例如下。

```
# 服务地址
http2Endpoint = "https://${YourProductKey}.iot-as-http2.cn-shanghai.aliyuncs.com:443"
authEndpoint = "https://iot-auth.cn-shanghai.aliyuncs.com/auth/bridge"

# 网桥设备信息
productKey = ${YourProductKey}

popClientProfile = {
  accessKey = ${YourAliyunAccessKey}
  accessSecret = ${YourAliyunAccessSecret}
  name = cn-shanghai
  region = cn-shanghai
  product = Iot
  endpoint = iot.cn-shanghai.aliyuncs.com
}
```

调用物模型数据上报接口

为了方便使用，减少您的封装操作，泛化协议SDK中已封装部分数据上报接口，包括属性上报接口reportProperty、事件上报接口fireEvent和更新设备标签接口updateDeviceTag。设备可以通过这些接口向物联网平台上报相应消息。

接口使用前提和说明：

- 调用reportProperty和fireEvent上报属性值和事件前，您需先在物联网平台控制台设备所属产品的**产品详情**页的**功能定义**页签下，定义属性和事件。请参见单个添加物模型。
- 调用updateDeviceTag接口上报的设备标签，如果您已经在物联网平台控制台设备对应的**设备详情**页，添加了该标签，则更新标签值（value）；若没有对应的标签，则新建标签。

接口调用示例：

```
TslUplinkHandler tslUplinkHandler = new TslUplinkHandler();
//上报属性。
//已定义testProp属性。
String requestId = String.valueOf(random.nextInt(1000));
tslUplinkHandler.reportProperty(requestId, originalIdentity, "testProp", random.nextInt(100));

//上报事件。
//已定义testEvent事件。
requestId = String.valueOf(random.nextInt(1000));
HashMap<String, Object> params = new HashMap<String, Object>();
params.put("testEventParam", 123);
tslUplinkHandler.fireEvent(originalIdentity, "testEvent", ThingEventTypes.INFO, params);

//上报设备标签。
//已定义设备标签key为testDeviceTag。
requestId = String.valueOf(random.nextInt(1000));
tslUplinkHandler.updateDeviceTag(requestId, originalIdentity, "testDeviceTag", String.valueOf(random
.nextInt(1000)));
```

以上示例中，接口调用的参数说明：

参数	说明
requestId	请求消息ID。
originalIdentity	设备的原始身份标识符。
testProp	属性的identifier。本示例的前提条件是：为产品定义功能时，定义了一个identifier为testProp的属性。本示例代码为上报属性testProp的值。
random.nextInt(100)	上报的属性值。属性值的取值范围也在定义属性时定义。在本示例中，使用 `random.nextInt(100)` 表示取小于100的整形随机值。
testEvent	事件的identifier。本示例的前提条件是：为产品定义功能时，定义了一个identifier为testEvent的事件。本示例代码为上报事件testEvent。
ThingEventTypes.INFO	事件类型。ThingEventTypes参数表示事件类型，*INFO*表示事件类型取值为INFO（信息）。 本示例的前提条件是：定义事件testEvent时，选择的事件类型为信息（即INFO）。 如果事件类型定义为故障，则该参数为 `ThingEventTypes.ERROR` 。
params	事件的输出参数。事件输出参数的identifier、数据类型、取值范围等也在定义事件时定义。本示例中，上报事件的出参identifier是testEventParam，参数值是123。

参数	说明
testDeviceTag	设备标签键（key），String类型。本示例中为testDeviceTag。 实际使用时，请根据设备标签键规范和您的需求设置。请参见设备标签。
String.valueOf(random.nextInt(1000))	设备标签值（value），String类型。本示例中，用 String.valueOf(random.nextInt(1000)) 表示取值为一个小于1000的随机值。实际使用时，请根据设备标签值规范和您的需求设置，请参见设备标签。

8.4. OTA升级

泛化协议SDK 2.1.3及以上版本支持设备固件升级。下面介绍使用泛化协议SDK调用OTA升级接口，实现设备固件升级。

背景信息

> 说明 仅泛化协议SDK 2.1.3及以上版本支持固件OTA升级。

设备OTA升级流程如下图所示。

云端推送升级包操作说明，请参见推送升级包到设备端。

设备OTA升级流程说明，以及使用的Topic和数据格式等信息，请参见设备端OTA升级。

调用OTA接口

泛化协议SDK中已封装OTA升级相关接口。您需配置SDK调用以下三个接口实现OTA升级。

- 设备上报固件版本接口

  设备启动时和OTA升级后，上报固件当前版本到Topic： `/ota/device/inform/${YourProductKey}/${YourDeviceName}` 。

  泛化协议SDK调用TslUplinkHandler.reportOtaVersion接口，上报版本信息。接口设置如下：

```
/**
 * 设备上报固件版本。
 * @param requestId 请求ID。
 * @param originalIdentity 设备身份原始标识符。
 * @param version 待上报的固件版本。
 * @return 上报成功则返回true。
*/
boolean reportOtaVersion(String requestId, String originalIdentity, String version)
```

  调用示例：

```
TslUplinkHandler tslUplinkHandler = new TslUplinkHandler();
tslUplinkHandler.doOnline(session, originalIdentity);
tslUplinkHandler.reportOtaVersion("12345", originalIdentity, "1.0.1");
```

- 监听云端下发升级通知接口

  您在物联网平台上添加升级包，并启动批量升级后，物联网平台向设备下发升级通知。设备通过Topic：`/ota/device/upgrade/${YourProductKey}/${YourDeviceName}` 获取升级通知。

  泛化协议SDK调用BridgeBootStrap.setOtaUpgradeHandler()接口，并设置一个回调，接收云端推送的OTA升级的信息。接口设置如下：

```
/**
 * 设置回调，接收云端推送的OTA升级的信息。
 * @param otaUpgradeHandler 设置的回调。
*/
public void setOtaUpgradeHandler(OtaUpgradeHandler otaUpgradeHandler) {
    this.callback.setOtaUpgradeHandler(otaUpgradeHandler);
}

public interface OtaUpgradeHandler {

    /**
     * 云端推送OTA升级消息。
     * @param requestId 云端推送OTA升级消息的ID。
     * @param firmwareInfo OTA升级信息。
     * @param session 当前Session。
     * @return
     */
    boolean onUpgrade(String requestId, OtaFirmwareInfo firmwareInfo, Session session);
}

public class OtaFirmwareInfo {

    /**
     * 升级包大小。
     */
    private long size;

    /**
     * 签名方法：MD5、SHA256。
     */
    private String signMethod;
```

```
    /**
     * 升级包签名。
     */
    private String sign;

    /**
     * 升级包版本。
     */
    private String version;

    /**
     * 升级包下载URL。
     */
    private String url;
}
```

调用示例：

```
bridgeBootstrap.setOtaUpgradeHandler(new OtaUpgradeHandler() {
    @Override
    public boolean onUpgrade(String requestId, OtaFirmwareInfo firmwareInfo, Session session) {
        log.info("ota onUpgrade, requestId:{}, firmware:{}, identity:{}",
            requestId, firmwareInfo, session.getOriginalIdentity());
        //处理OTA升级。
        return true;
    }
});
```

- 上报升级进度接口

  设备开始升级，并上报升级进度到Topic：`/ota/device/progress/${YourProductKey}/${YourDeviceName}` 。

  泛化协议SDK调用TslUplinkHandler.reportOtaProgress接口，上报升级进度。接口设置如下：

```
/**
 * 上报OTA升级进度。
 * @param requestId 请求消息ID。
 * @param originalIdentity 设备原始身份标识符。
 * @param step 当前进度。
 * @param desc 描述。
 * @return
*/
boolean reportOtaProgress(String requestId, String originalIdentity, String step, String desc)
```

调用示例：

```
tslUplinkHandler.reportOtaProgress("7979", session.getOriginalIdentity(), "100", "ota success");
```