阿里云
物联网平台

用户指南

文档版本：20181008

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
	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。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能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，恢复业务所需时间约10分钟。
	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。	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。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 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。	`swich` *`{stand | slave}`*

目录

1 账号与登录

本章将详细介绍IoT控制台账号、登录的操作。

1.1 使用阿里云主账号登录控制台

阿里云主账号具有该账号下所有资源的完全操作权限，并且可以修改账号信息。

使用主账号登录 IoT 控制台

使用阿里云主账号登录物联网控制台，建议您首先完成实名认证，以获取对物联网平台所有操作的完全权限。

1. 访问阿里云官网。
2. 单击控制台。
3. 使用阿里云账号和密码登录。

> 说明：
> 若忘记账号或密码，请单击登录框中**忘记会员名**或**忘记密码**，进入账号或密码找回流程。

4. 在控制台中，单击**产品与服务**，页面显示所有阿里云产品和服务名称。
5. 搜索物联网平台，并单击搜索结果中的**物联网平台**产品名，进入**物联网控制台**。

> 说明：
> 如果您还没有开通物联网平台服务，**物联网控制台**主页会展示相关提示，您只需单击**立即开通**便可快速开通物联网平台服务。

进入**物联网控制台**后，您便可以进行产品管理、设备管理、规则管理等操作。

使用主账号创建访问控制

因为主账号具有账号的完全权限，主账号泄露会带来极严重的安全隐患。因此，若需要授权其他人访问您的阿里云资源，请勿将您的阿里云账号及密码直接泄露出去。应该通过访问控制 RAM 创建子账号，并给子账号授予需要的访问权限。非账号所有者或管理员的其他人通过子账号访问资源。有关子账号访问的具体方法，请参见子账号访问和自定义权限。

1.2 RAM授权管理

本章节将详细介绍物联网平台账号权限控制相关事宜。

1.2.1 RAM 和 STS 介绍

RAM 和 STS 是阿里云提供的权限管理系统。了解 RAM 和 STS 的详情，请参见访问控制产品文档。

RAM 的主要作用是控制账号系统的权限。通过使用 RAM，创建、管理子账号，并通过给子账号授予不同的权限，控制子账号对资源的操作权限。

STS 是一个安全凭证（Token）的管理系统，为阿里云子账号（RAM 用户）提供短期访问权限管理。通过 STS 来完成对临时用户的访问授权。

背景介绍

RAM 和 STS 解决的一个核心问题是如何在不暴露主账号的 AccessKey 的情况下，安全地授权他人访问。因为一旦主账号的 AccessKey 被泄露，会带来极大的安全风险：获得该账号 AccessKey 的人可任意操作该账号下所有的资源，盗取重要信息等。

RAM 提供的是一种长期有效的权限控制机制。通过创建子账号，并授予子账号相应的权限，将不同的权限分给不同的用户。子账号的 AccessKey 也不能泄露。即使子账号泄露也不会造成全局的信息泄露。一般情况下，子账号长期有效。

相对于 RAM 提供的长效控制机制，STS 提供的是一种临时访问授权。通过调用 STS，获得临时的 AccessKey 和 Token。可以将临时 AccessKey 和 Token 发给临时用户，用来访问相应的资源。从 STS 获取的权限会受到更加严格的限制，并且具有时间限制。因此，即使出现信息泄露的情况，影响相对较小。

使用场景示例，请参见使用示例。

基本概念

使用 RAM 和 STS 涉及以下基本概念：

- 子账号：在 RAM 控制台中，创建的用户，每个用户即一个子账号。创建时或创建成功后，均可为子账号生成独立的 AccessKey。创建后，需为子账号配置密码和权限。使用子账号，可以进行已获授权的操作。子账号可以理解为具有某种权限的用户，可以被认为是一个具有某些权限的操作发起者。
- 角色（Role）：表示某种操作权限的虚拟概念，但是没有独立的登录密码和 AccessKey。子账号可以扮演角色。扮演角色时，子账号拥有的权限是该角色的权限。
- 授权策略（Policy）：用来定义权限的规则，比如允许子账号用户读取或者写入某些资源。

- 资源（Resource）：代表子账号用户可访问的云资源，比如表格存储所有的 Instance、某个 Instance 或者某个 Instance 下面的某个 Table 等。

子账号和角色可以类比为个人和其身份的关系。如，某人在公司的角色是员工，在家里的角色是父亲。同一人在不同的场景扮演不同的角色。在扮演不同角色的时候，拥有对应角色的权限。角色本身并不是一个操作的实体，只有用户扮演了该角色之后才是一个完整的操作实体。并且，一个角色可以被多个不同的用户同时扮演。

使用示例

为避免阿里云账号的 AccessKey 泄露而导致安全风险，某阿里云账号管理员使用 RAM 创建了两个子账号，分别命名为 A 和 B ，并为 A 和 B 生成独立的 AccessKey。A 拥有读权限，B 拥有写权限。 管理员可以随时在 RAM 控制台取消子账号用户的权限。

现在因为某些原因，需要授权给其他人临时访问物联网平台接口的权限。这种情况下，不能直接把 A 的 AccessKey 透露出去，而应该新建一个角色 C，并给这个角色授予读取物联网平台接口的权限。但请注意，目前角色 C 还无法直接使用。因为并不存在对应角色 C 的 AccessKey，角色 C 仅是一个拥有访问物联网平台接口权限的虚拟实体。

需调用 STS 的 AssumeRole 接口，获取访问物联网平台接口的临时授权。在调用 STS 的请求中，RoleArn 的值需为角色 C 的 Arn。如果调用成功，STS 会返回临时的 AccessKeyId、AccessKeySecret和 SecurityToken 作为访问凭证（凭证的过期时间，在调用 AssumeRole 的请求中指定）。将这个凭证发给需要访问的用户，该用户就可以获得访问物联网平台接口的临时权限。

为什么 **RAM** 和 **STS** 的使用这么复杂？

虽然 RAM 和 STS 的概念和使用比较复杂，但这是为了账号的安全性和权限控制的灵活性而牺牲了部分易用性。

将子账号和角色分开，主要是为了将执行操作的实体和代表权限集合的虚拟实体分开。如果某用户需要使用多种权限，比如读/写权限，但是实际上每次操作只需要其中的一部分权限，那么就可以创建两个角色。这两个角色分别具有读或写权限。然后，创建一个可以扮演这两个角色的用户子账号。当用户需要读权限的时候，就可以扮演其中拥有读权限的角色；使用写权限的时候同理。这样可以降低每次操作中权限泄露的风险。而且，通过扮演角色，可以将角色权限授予其他用户，更加方便了协同使用。

STS 对权限的控制更加灵活。如按照实际需求设置有效时长。但是，如果需要一个长期有效的临时访问凭证，则可以只适用 RAM 子账号管理功能，而无需使用 STS。

在后面的章节中，我们将提供一些 RAM 和 STS 的使用指南和使用示例。如果您需要了解更多 RAM 和 STS 的代码详情，请参见 *API 参考#RAM#*和 *API 参考#STS#* 。

1.2.2 自定义权限

权限指在某种条件下 ，允许 (Allow) 或 拒绝 (Deny) 对某些资源执行某些操作。

权限的载体是授权策略。自定义权限，即在自定义授权策略时定义某些权限。在 RAM 控制台，策略管理页面，单击新建授权策略开始创建自定义授权策略。创建自定义授权策略时，请选择模板为空白模板。

授权策略是 JSON 格式的字符串，需包含以下参数：

- **Action**：表示要授权的操作。IoT 操作都以*iot:* 开头。定义方式和示例，请参见本文档中Action 定义。
- **Effect** ： 表示授权类型，取值：Allow、Deny。
- **Resource** ：物联网平台目前暂不支持资源粒度的授权，请填写*。
- **Condition** ：表示鉴权条件。详细信息，请参见本文档中 Condition 定义。

Action 定义

Action是 API 的名称。在创建 IoT 的授权策略时，每个 Action 前缀均为*iot:*，多个 Action 以逗号分隔。并且，支持使用星号通配符。IoT API 名称定义，请参见*IoT API 授权映射表* 。

下面介绍一些典型的 Action 定义示例。

- 定义单个 API。

```
"Action": "iot:CreateProduct"
```

- 定义多个API。

```
"Action": [
"iot:UpdateProduct",
"iot:QueryProduct"
]
```

- 定义所有只读 API。

```
{
"Version": "1",
"Statement": [
{
"Action": [
"rds:DescribeDBInstances",
"rds:DescribeDatabases",
"rds:DescribeAccounts",
"rds:DescribeDBInstanceNetInfo"
```

```
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": "ram:ListRoles",
"Effect": "Allow",
"Resource": "*"
},
{
"Action": [
"mns:ListTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"dhs:ListProject",
"dhs:ListTopic",
"dhs:GetTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"ots:ListInstance",
"ots:ListTable",
"ots:DescribeTable"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"log:ListShards",
"log:ListLogStores",
"log:ListProject"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Effect": "Allow",
"Action": [
"iot:Query*",
"iot:List*",
"iot:Get*",
"iot:BatchGet*"
],
"Resource": "*"
}
]
}
```

- 定义所有读写 API。

```
{
"Version": "1",
"Statement": [
{
```

```
"Action": [
"rds:DescribeDBInstances",
"rds:DescribeDatabases",
"rds:DescribeAccounts",
"rds:DescribeDBInstanceNetInfo"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": "ram:ListRoles",
"Effect": "Allow",
"Resource": "*"
},
{
"Action": [
"mns:ListTopic",
"mns:CreateQueue"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"dhs:ListProject",
"dhs:ListTopic",
"dhs:GetTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"ots:ListInstance",
"ots:ListTable",
"ots:DescribeTable"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"log:ListShards",
"log:ListLogStores",
"log:ListProject"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*"
}
]
```

```
}
```

Condition 定义

目前 RAM 授权策略支持访问 IP 限制、是否通过 HTTPS 访问、是否通过 MFA（多因素认证）访问、访问时间限制等多种鉴权条件。物联网平台的所有 API 均支持这些条件。

访问 IP 限制

访问控制可以限制访问 IoT 的源 IP 地址，并且支持根据网段进行过滤。以下是典型的使用场景示例。

- 限制单个 IP 地址和 IP 网段。例如，只允许 IP 地址为 10.101.168.111 或 10.101.169.111/24 网段的请求访问。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"IpAddress": {
"acs:SourceIp": [
"10.101.168.111",
"10.101.169.111/24"
]
}
}
}
],
"Version": "1"
}
```

- 限制多个 IP 地址。例如，只允许 IP 地址为 10.101.168.111 和 10.101.169.111 的请求访问。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"IpAddress": {
"acs:SourceIp": [
"10.101.168.111",
"10.101.169.111"
]
}
}
}
],
"Version": "1"
}
```

HTTPS 访问限制

访问控制可以限制是否通过 HTTPS 访问。

示例：限制必须通过 HTTPS 请求访问。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"Bool": {
"acs:SecureTransport": "true"
}
}
}
],
"Version": "1"
}
```

MFA 访问限制

访问控制可以限制是否通过 MFA (多因素认证) 访问。

示例：限制必须通过 MFA 请求访问。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"Bool": {
"acs:MFAPresent ": "true"
}
}
}
],
"Version": "1"
}
```

访问时间限制

访问控制可以限制请求的访问时间，即只允许或拒绝在某个时间点范围之前的请求。

示例：用户可以在北京时间 2019 年 1 月 1 号凌晨之前访问，之后则不能访问。

```
{
"Statement": [
{
"Effect": "Allow",
```

```
"Action": "iot:*",
"Resource": "*",
"Condition": {
"DateLessThan": {
"acs:CurrentTime": "2019-01-01T00:00:00+08:00"
}
}
}
],
"Version": "1"
}
```

典型使用场景

结合以上对 Action、Resource 和 Condition 的定义，下面介绍一些典型使用场景的授权策略定义和授权方法。

允许访问的授权策略示例

场景：定义访问 IP 地址为 10.101.168.111/24 网段的用户访问 IoT 的权限，且要求只能在 2019-01-01 00:00:00 之前访问和通过 HTTPS 访问。

```
{
"Statement": [
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*",
"Condition": {
"IpAddress": {
"acs:SourceIp": [
"10.101.168.111/24"
]
},
"DateLessThan": {
"acs:CurrentTime": "2019-01-01T00:00:00+08:00"
},
"Bool": {
"acs:SecureTransport": "true"
}
}
}
],
"Version": "1"
}
```

拒绝访问的授权策略示例

场景：拒绝访问 IP 地址为 10.101.169.111 的用户对 IoT 执行读操作。

```
{
"Statement": [
{
"Effect": "Deny",
"Action": [
"iot:Query*",
"iot:List*",
```

```
"iot:Get*",
"iot:BatchGet*"
],
"Resource": "*",
"Condition": {
"IpAddress": {
"acs:SourceIp": [
"10.101.169.111"
]
}
}
}
],
"Version": "1"
}
```

授权策略创建成功后，在访问控制 RAM 控制台，用户管理页面，将此权限授予子账号用户。获得授权的子账号用户就可以进行权限中定义的操作。创建子账号和授权操作帮助，请参见子账号访问。

1.2.3 IoT API 授权映射表

下表中IoT API名称，即您在创建IoT相关授权策略时，参数Action的可选值。

IoT API	RAM 授权操作（Action)	资源（Resource）	接口说明
CreateProduct	iot:CreateProduct	*	创建产品
UpdateProduct	iot:UpdateProduct	*	修改产品
QueryProduct	iot:QueryProduct	*	查询产品
QueryProductList	iot:QueryProductList	*	查询产品列表
GetCats	iot:GetCats	*	获取产品类型信息
QueryProductByUserId	iot:QueryProductByUserId	*	根据用户id查询产品
RegisterDevice	iot:RegisterDevice	*	注册设备
QueryDevice	iot:QueryDevice	*	批量查询设备
QueryByDeviceId	iot:QueryByDeviceId	*	根据deviceId查询设备
QueryDeviceByName	iot:QueryDeviceByName	*	根据deviceName查询设备
DeleteDevice	iot:DeleteDevice	*	删除设备
ApplyDeviceWithNamesRequest	iot:ApplyDeviceWithNamesRequest	*	批量创建设备

IoT API	RAM 授权操作（Action)	资源（Resource）	接口说明
QueryApplyStatus	iot:QueryApplyStatus	*	查询申请状态
QueryPageByApplyId	iot:QueryPageByApplyId	*	根据申请id分页查询
BatchGetDeviceState	iot:BatchGetDeviceState	*	批量获取设备状态
BatchRegisterDeviceWithApplyId	iot:BatchRegisterDeviceWithApplyId	*	根据ApplyId批量申请设备
BatchRegisterDevice	iot:BatchRegisterDevice	*	批次申请特定数量设备
QueryBatchRegisterDeviceStatus	iot:QueryBatchRegisterDeviceStatus	*	查询批量注册设备状态
BatchCheck DeviceNames	iot:BatchCheck DeviceNames	*	批量检查设备名称
GetDeviceStatus	iot:GetDeviceStatus	*	获取设备的运行状态
QueryDeviceStatistics	iot:QueryDeviceStatistics	*	获取设备的统计数量
QueryDeviceEventData	iot:QueryDeviceEventData	*	获取设备的事件历史数据
QueryDeviceServiceData	iot:QueryDeviceServiceData	*	获取设备的服务记录历史数据
SetDeviceProperty	iot:SetDeviceProperty	*	设置设备的属性
InvokeThingService	iot:InvokeThingService	*	设备的服务调用
QueryDevicePropertyStatus	iot:QueryDevicePropertyStatus	*	查询设备的属性快照
QueryDeviceDetail	iot:QueryDeviceDetail	*	查询设备详情
DisableThing	iot:DisableThing	*	设备的禁用
EnableThing	iot:EnableThing	*	设备的解禁
GetThingTopo	iot:GetThingTopo	*	查询设备拓扑关系
RemoveThingTopo	iot:RemoveThingTopo	*	移除设备拓扑关系
NotifyAddThingTopo	iot:NotifyAddThingTopo	*	通知云端增加设备拓扑关系
QueryDevicePropertyData	iot:QueryDevicePropertyData	*	获取设备的属性历史数据
QueryTopic	iot:QueryTopic	*	查询Topic

IoT API	RAM 授权操作（Action)	资源（Resource）	接口说明
StartRule	iot:StartRule	*	启动规则
StopRule	iot:StopRule	*	暂停规则
ListRule	iot:ListRule	*	查询规则列表
GetRule	iot:GetRule	*	查询规则详情
CreateRule	iot:CreateRule	*	创建规则
UpdateRule	iot:UpdateRule	*	修改规则
DeleteRule	iot:DeleteRule	*	删除规则
CreateRuleAction	iot:CreateRuleAction	*	创建规则中的转发数据方法
UpdateRuleAction	iot:UpdateRuleAction	*	修改规则中的转发数据方法
DeleteRuleAction	iot:DeleteRuleAction	*	删除规则中的转发数据方法
GetRuleAction	iot:GetRuleAction	*	获取规则中的转发数据方法
ListRuleActions	iot:ListRuleActions	*	获取规则中的转发数据方法列表
Pub	iot:Pub	*	发布消息
Sub	iot:Sub	*	订阅消息
Unsub	iot:Unsub	*	取消订阅
DeviceRevokeById	iot:DeviceRevokeById	*	通过ID撤销设备权限
DevicePermitModify	iot:DevicePermitModify	*	修改设备权限
ListPermits	iot:ListPermits	*	列出设备的权限
RRpc	iot:RRpc	*	发送消息给设备并得到设备响应
CreateProductTopic	iot:CreateProductTopic	*	创建产品Topic类
DeleteProductTopic	iot:DeleteProductTopic	*	删除产品Topic类
QueryProductTopic	iot:QueryProductTopic	*	查询产品Topic类列表
UpdateProductTopic	iot:UpdateProductTopic	*	修改产品Topic类
QueryDeviceTopic	iot:QueryDeviceTopic	*	查询设备Topic列表

IoT API	RAM 授权操作（Action)	资源（Resource）	接口说明
DebugRuleSql	iot:DebugRuleSql	*	规则引擎SQL调试

1.2.4 子账号访问

子账号用户可以登录物联网控制台管理您的 IoT 资源，和使用子账号的 AccessKeyId 和AccessKeySecret 调用 IoT API。

您需先创建子账号，并通过授权策略授予子账号访问物联网平台的权限。创建自定义授权策略的方法，请参见自定义权限。

创建子账号

如果您已有子账号，请忽略此操作。

1. 用主账号登录访问控制 *RAM* 控制台。
2. 在左侧导航栏中，单击用户管理。
3. 单击新建用户。
4. 输入用户信息，并勾选为该用户自动生成 **AccessKey**前的复选框，再单击确定。

   说明：
   单击确定后，弹出保存 AccessKey 的提示。这是下载该子账号的 AccessKey 的唯一机会。请及时保存该 AccessKey 信息，并妥善保管。子账号用户调用 API 时，需传入该 AccessKey 信息。
5. 设置初始登录密码。
   a. 在用户管理页面，找到刚创建的用户名，单击管理，进入用户详情页面。
   b. 单击启用控制台登录。
   c. 为该子账号设置一个初始密码，并勾选要求该账号下次登录成功后重置密码前的复选框，再单击确定。
6. 启用多因素认证设备（可选）。

   在用户详情页面，单击启用虚拟**MFA**设备。

子账号创建完成后，子账号用户便可通过RAM 用户登录链接登录阿里云官网和控制台。RAM 用户的登录链接，请在访问控制 **RAM** 控制台的概览页面查看。

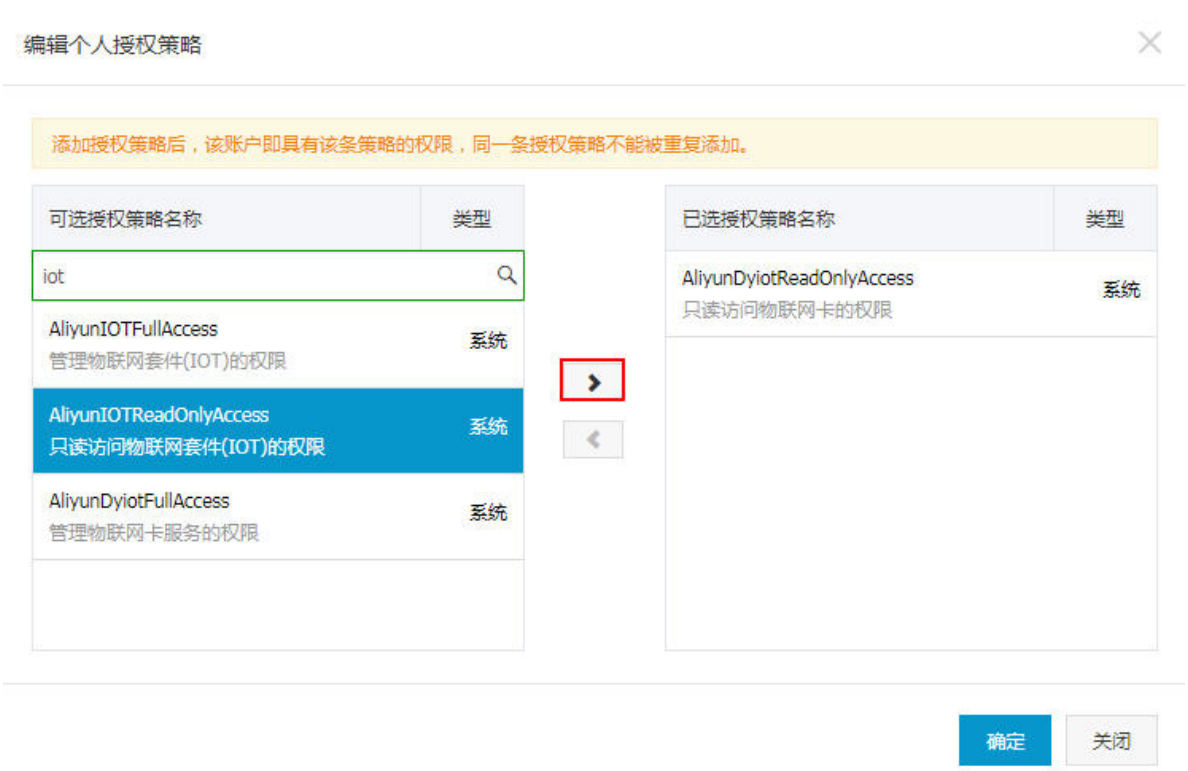
但是，在获得授权之前，该子账号无法访问您的阿里云资源。下一步，为子账号授予物联网平台的访问权限。

授权子账号访问 IoT

在访问控制 RAM 控制台中，您可以在用户管理页面，为单个子账号进行授权；也可以在群组管理页面，为整个群组授予相同的权限。下面我们以为单个子账号授权为例，介绍授权操作流程。

1. 用主账号登录访问控制 *RAM* 控制台。
2. 在左侧导航栏中，单击用户管理。
3. 找到要授权的子账号的用户名，单击授权操作按钮。
4. 在授权对话框中，选择您要授予该子账号的 IoT 授权策略，单击页面中间向右箭头将选中权限移入已选授权策略名称，再单击确定。

> **说明：**
> 如果您要为子账号授予自定义权限，请先创建授权策略。授权策略的创建方法，请参见自定义权限。



授权成功后，子账号用户便可访问授权策略中定义的资源，和进行授权策略中定义的操作。

子账号登录控制台

阿里云主账号登录是从阿里云官网主页直接登录，但是子账号需从**RAM 用户登录**页面登录。

1. 获取**RAM 用户登录**页面的链接地址。

   用主账号登录**访问控制 RAM 控制台**，在**概览**页面查看**RAM 用户登录链接**，并将链接地址分发给子账号的用户。

2. 子账号用户访问**RAM 用户登录**页面，并使用子账号和子账号密码登录。

   > **说明：**
   > 子账号登录格式为：子用户名称@企业别名，如：username@company-alias。并且，首次登录成功后，需修改登录密码。

3. 单击页面右上角**控制台**按钮，进入**管理控制台**。
4. 单击**产品与服务**，选择**物联网平台**，即可进入**物联网控制台**。

子账号用户登录**物联网控制台**后，便可在控制台中，进行已获授权的操作。

1.2.5 进阶使用 STS

STS 权限管理系统是比访问控制（RAM）更为严格的权限管理系统。使用 STS 权限管理系统进行资源访问控制，需通过复杂的授权流程，授予子账号用户临时访问资源的权限。

子账号和授予子账号的权限均长期有效。删除子账号或解除子账号权限，均需手动操作。发生子账号信息泄露后，如果无法及时删除该子账号或解除权限，可能给您的阿里云资源和重要信息带来危险。所以，对于关键性权限或子账号无需长期使用的权限，您可以通过 STS 权限管理系统来进行控制。

图 **1-1:** 子账号获得临时访问权限的操作流程

步骤一：创建角色

RAM 角色是一种虚拟用户，是承载操作权限的虚拟概念。

1. 使用阿里云主账号登录访问控制 *RAM* 控制台。

2. 单击角色管理 > 新建角色，进入角色创建流程。
3. 选择用户角色。
4. 填写信息类型步骤中，直接使用默认的当前云账号信息，单击下一步。
5. 输入角色名称和备注后，单击创建。
6. 单击关闭或授权。

   如果您已创建要授予该角色的授权策略，则单击 授权，并对角色进行授权。

   如果您还没有创建授权策略，则单击关闭。然后，在策略管理中为该角色新建授权策略。

步骤二：创建角色授权策略

角色授权策略，即定义要授予角色的资源访问权限。

1. 在访问控制 *RAM* 控制台主页，单击策略管理 > 新建授权策略。
2. 选择空白模板。
3. 输入授权策略名称和策略内容，再单击新建授权策略。

   授权策略内容的编写方法参考，请单击授权策略格式定义查看。

   授权策略内容示例：IoT 资源只读权限。

```
{
"Version": "1",
"Statement": [
{
"Action": [
"rds:DescribeDBInstances",
"rds:DescribeDatabases",
"rds:DescribeAccounts",
"rds:DescribeDBInstanceNetInfo"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": "ram:ListRoles",
"Effect": "Allow",
"Resource": "*"
},
{
"Action":[
"mns:ListTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"dhs:ListProject",
"dhs:ListTopic",
```

```
"dhs:GetTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"ots:ListInstance",
"ots:ListTable",
"ots:DescribeTable"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action":[
"log:ListShards",
"log:ListLogStores",
"log:ListProject"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Effect": "Allow",
"Action": [
"iot:Query*",
"iot:List*",
"iot:Get*",
"iot:BatchGet*"
],
"Resource": "*"
}
]
}
```

授权策略内容示例：IoT 资源读写权限。

```
{
"Version": "1",
"Statement": [
{
"Action": [
"rds:DescribeDBInstances",
"rds:DescribeDatabases",
"rds:DescribeAccounts",
"rds:DescribeDBInstanceNetInfo"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": "ram:ListRoles",
"Effect": "Allow",
"Resource": "*"
},
{
"Action":[
"mns:ListTopic"
],
```

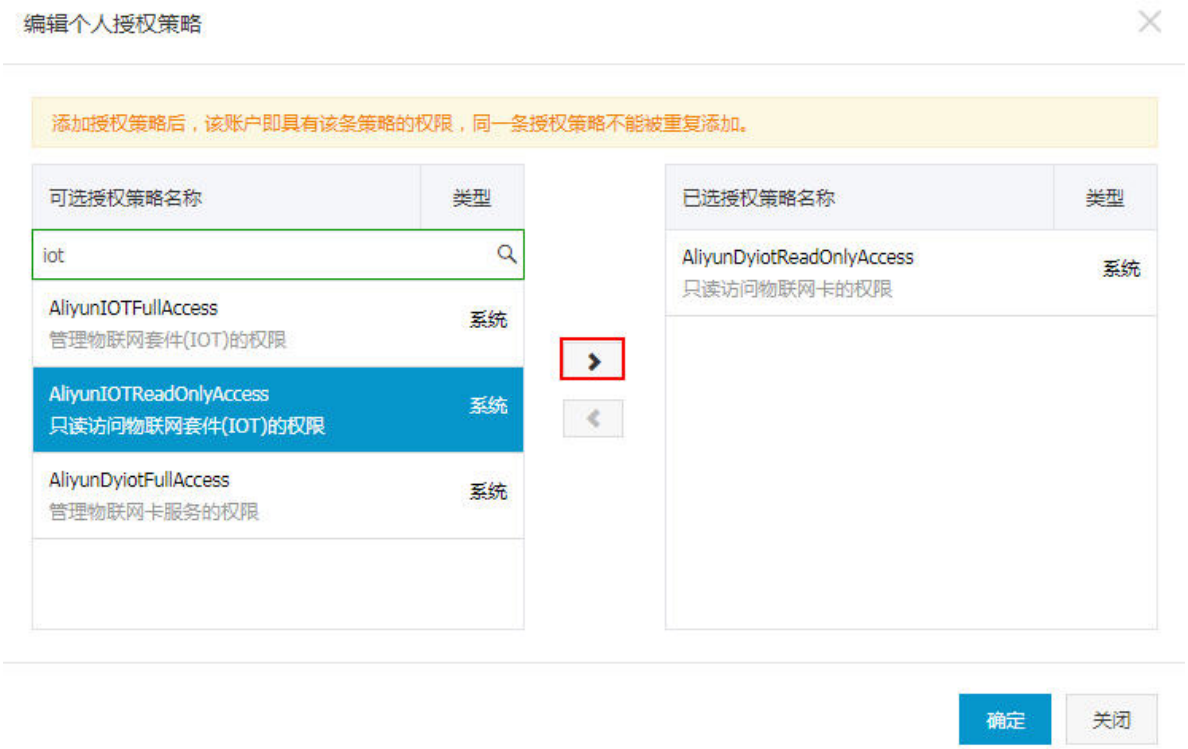
```
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"dhs:ListProject",
"dhs:ListTopic",
"dhs:GetTopic"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action": [
"ots:ListInstance",
"ots:ListTable",
"ots:DescribeTable"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Action":[
"log:ListShards",
"log:ListLogStores",
"log:ListProject"
],
"Resource": "*",
"Effect": "Allow"
},
{
"Effect": "Allow",
"Action": "iot:*",
"Resource": "*"
}
]
}
```

授权策略创建成功后，您就可以将该授权策略中定义的权限授予角色。

步骤三：为角色授权

角色获得授权后，才具有资源访问权限。

1. 在访问控制 *RAM* 控制台主页，单击角色管理。
2. 找到要授权的角色，单击授权。
3. 在授权对话框中，选择要授予角色的自定义授权策略，单击中间的向右箭头，将选中的授权策略移至已选授权策略名称下，再单击确定。



授权完成后，该角色就具有了授权策略定义的权限。您可以单击该角色对应的管理操作按钮，进入角色详情页，查看该角色的基本信息和权限信息。

下一步，为子账号授予可以扮演该角色的权限。

步骤四：授予子账号角色扮演权限

虽然经过授权后，该角色已拥有了授权策略定义的访问权限，但角色本身只是虚拟用户，需要子账号用户扮演该角色，才能进行权限允许的操作。若任意子账号都可以扮演该角色，也会带来风险，因此只有获得角色扮演权限的子账号用户才能扮演角色。

授权子账号扮演角色的方法：先新建一个**Resource**参数值为角色 ID 的自定义授权策略，然后用该授权策略为子账号授权。

1. 在访问控制 *RAM* 控制台主页，单击策略管理 > 新建授权策略。
2. 选择空白模板。
3. 输入授权策略名称和策略内容，再单击新建授权策略。

> 说明：
> 授权策略内容中，参数**Resource** 的值需为角色 Arn。在角色管理页面，单击角色对应的管理按钮，进入角色详情页面中，查看角色的 Arn 。

角色授权策略示例：

```
{
"Version": "1",
"Statement": [
{
"Effect": "Allow",
"Action": "iot:QueryProduct",
"Resource": "角色Arn"
}
]
}
```

4. 授权策略创建成功后，返回**访问控制 RAM 控制台**主页。
5. 单击左侧导航栏中的**用户管理**按钮，进入子账号管理页面。
6. 找到要授权的子账号，并单击对应的**授权**按钮。
7. 在授权对话框中，选择刚新建的角色授权策略，单击中间的向右箭头将授权策略移至**已选授权策略名称**下，再单击**确定**。

授权完成后，子账号便有了可以扮演该角色的权限，就可以使用 STS 获取扮演角色的临时身份凭证，和进行资源访问。

步骤五：子账号获取临时身份凭证

获得角色授权的子账号用户，可以通过直接调用 API 或使用 SDK 来获取扮演角色的临时身份凭证：AccessKeyId、AccessKeySecret、和 SecurityToken。STS API 和 STS SDK 详情，请参见访问控制文档中*API 参考#STS#*和*SDK 参考#STS#*。

使用 API 和 SDK 获取扮演角色的临时身份凭证需传入以下参数：

- RoleArn：需要扮演的角色 Arn。
- RoleSessionName：临时凭证的名称（自定义参数）。
- Policy：授权策略，即为角色增加一个权限限制。通过此参数限制生成的Token的权限。不指定此参数，则返回的 Token 将拥有指定角色的所有权限。
- DurationSeconds：临时凭证的有效期。单位是秒，最小为 900，最大为 3600，默认值是 3600。
- id 和 secret：指需要扮演该角色的子账号的 AccessKeyId 和 AccessKeySecret。

获取临时身份凭证示例

API 示例：子账号用户通过调用 STS 的**AssumeRole**接口获得扮演该角色的临时身份凭证。

```
https://sts.aliyuncs.com?Action=AssumeRole
&RoleArn=acs:ram::1234567890123456:role/iotstsrole
&RoleSessionName=iotreadonlyrole
&DurationSeconds=3600
&Policy=<url_encoded_policy>
&<公共请求参数>
```

SDK 示例：子账号用户使用 STS 的 Python 命令行工具接口获得扮演该角色的临时身份凭证。

```
$python ./sts.py AssumeRole RoleArn=acs:ram::1234567890123456:role/
iotstsrole RoleSessionName=iotreadonlyrole Policy='{"Version":"1","
Statement":[{"Effect":"Allow","Action":"iot:*","Resource":"*"}]}'
DurationSeconds=3600 --id=id --secret=secret
```

请求成功后，将返回扮演该角色的临时身份凭证：AccessKeyId、AccessKeySecret、和 SecurityToken。

步骤六：子账号临时访问资源

获得扮演角色的临时身份凭证后，子账号用户便可以在调用 SDK 的请求中传入该临时身份凭证信息，扮演角色。

Java SDK 示例：子账号用户在调用请求中，传入临时身份凭证的 AccessKeyId、AccessKeySecret 和 SecurityToken 参数，创建 IAcsClient 对象。

```
IClientProfile  profile = DefaultProfile.getProfile("cn-hangzhou",
AccessKeyId,AccessSecret);
RpcAcsRequest request.putQueryParameter("SecurityToken", Token);
IAcsClient client = new DefaultAcsClient(profile);
AcsResponse response = client.getAcsResponse(request);
```

2 产品与设备

本章节将介绍如何使用控制台创建、管理产品与设备。

2.1 创建产品(基础版)

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。

背景信息

物联网平台有两个规格，基础版和高级版。本文档介绍如何在控制台上创建基础版产品。

操作步骤

1. 登录物联网平台控制台。
2. 在产品列表页面，单击创建产品。
3. 选择版本为基础版，按照界面提示设置参数，再单击确认。


物联网平台
产品管理
设备管理
边缘管理
规则引擎
扩展服务
我的服务
产品文档
产品管理
全部(89)
基础版(49
产品列表
请输入产品名称查询
产品名称
basic_001
1111
lidandan
Modbus111

页面参数设置如下：

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。
节点类型	设备或网关。 • 设备：指不能挂载子设备的设备。这种设备可以直连 IoT Hub，也可以作为网关的子设备连接IoT Hub。 • 网关：指可以挂载子设备的直连设备。网关具有子设备管理模块，维持子设备的拓扑关系，并且可以将拓扑关系同步到云端。 网关与子设备的关系，请参见网关与子设备。
使用ID²认证	如果您的产品没有使用ID²，请保持默认选项否。
产品描述	输入文字，用以描述产品。

预期结果

产品创建成功后，页面自动跳转回**产品列表**页面。您可以查看或编辑产品信息。

2.2 创建产品(高级版)

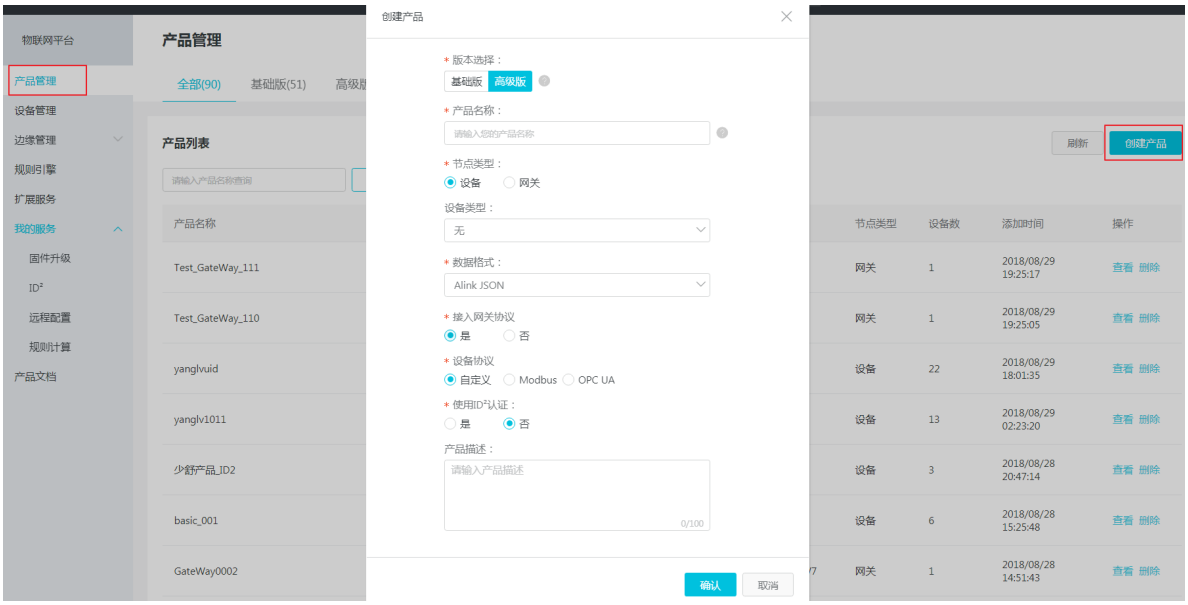
使用物联网平台的第一步：在控制台创建产品。产品是设备的集合，通常是一组具有相同功能定义的设备集合。例如：产品指同一个型号的产品，设备就是该型号下的某个设备。

背景信息

物联网平台有两个规格，基础版和高级版。本文档介绍如何在控制台上创建高级版产品。

操作步骤

1. 登录物联网平台控制台。
2. 在**产品管理**页面，单击**创建产品**。
3. 选择版本为**高级版**，按照页面填写信息，然后单击**确定**。



页面参数设置如下：

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。
节点类型	• 设备：指不能挂载子设备的设备。这种设备可以直连 IoT Hub，也可以作为网关的子设备连接IoT Hub。 • 网关：指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，并且将拓扑关系同步到云端。 网关与子设备的关系，请参见网关与子设备。
设备类型	选择预定义好的标准产品模板；或者设置为无，自定义该产品的功能。 **说明：** 阿里云物联网平台提供多种产品模板，并为对应产品预定义了相关功能。如，电表设备类型中，已预定义用电量、电压、电流、总累积量等电表标准功能。选择产品模板后，您可以在该模板基础上，编辑、修改、新增功能。
数据格式	设备上下行的数据格式，可选择**Alink JSON**或透传/自定义。 • Alink JSON：是物联网平台高级版为开发者提供的设备与云端的数据交换协议，采用 JSON 格式。

参数	描述
	• 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**透传/自定义**。此时，您需将自定义格式的数据通过数据解析转换为 Alink JSON 格式，才能与云端进行通信。
是否接入网关 **说明：** 当节点类型为**设备**时出现的参数。	该产品是否会接入网关产品，成为网关产品的子设备。 • 是：接入网关。 需选择接入网关协议，即该产品作为子设备时与网关的通讯协议类型。 • 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。 • 自定义：表示子设备和网关之间是其它标准或私有协议。 • 否：不接入网关。
产品描述	可输入文字，用来描述产品信息。

4. 参数设置完成，单击**确认**。

   产品创建成功后，页面自动跳转回**产品列表**页面。

后续操作

1. 在产品列表中，单击该产品的**查看**按钮，设置消息通信、物模型#功能定义#、服务端订阅等功能。
2. 参考设备开发指南手册完成设备开发。

3. 在该产品的详情页中，单击**发布**按钮，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**。

2.3 物模型

2.3.1 什么是物模型

物模型指将物理空间中的实体数字化，并在云端构建该实体的数据模型。在物联网平台中，定义物模型即定义产品功能。完成功能定义后，系统将自动生成该产品的物模型。物模型描述产品是什么，能做什么，可以对外提供哪些服务。

物模型，简称TSL，即Thing Specification Language。是一个JSON格式的文件。它是物理空间中的实体，如传感器、车载装置、楼宇、工厂等在云端的数字化表示，从属性、服务和事件三个维度，分别描述了该实体是什么，能做什么，可以对外提供哪些信息。定义了这三个维度，即完成了产品功能的定义。

物模型将产品功能类型分为三类：属性、服务、和事件。定义了这三类功能，即完成了物模型的定义。

功能类型	说明
属性（Property）	一般用于描述设备运行时的状态，如环境监测设备所读取的当前环境温度等。属性支持 GET 和 SET 请求方式。应用系统可发起对属性的读取和设置请求。
服务（Service）	设备可被外部调用的能力或方法，可设置输入参数和输出参数。相比于属性，服务可通过一条指令实现更复杂的业务逻辑，如执行某项特定的任务。
事件（Event）	设备运行时的事件。事件一般包含需要被外部感知和处理的通知信息，可包含多个输出参数。如，某项任务完成的信息，或者设备发生故障或告警时的温度等，事件可以被订阅和推送。

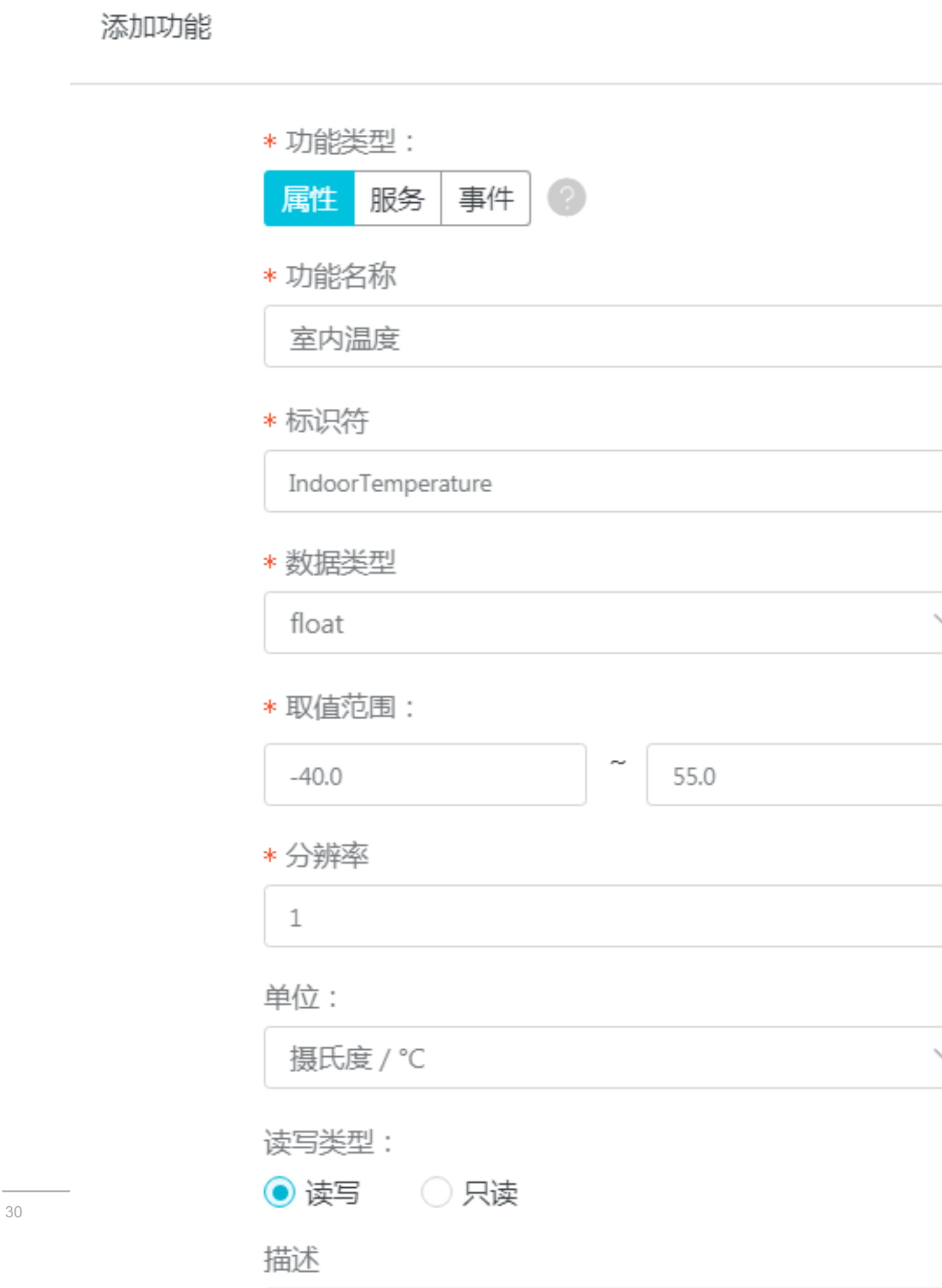
2.3.2 新增物模型

本文讲述如何通过控制台操作，定义物模型。

操作步骤

1. 登录物联网平台控制台。
2. 在产品管理页面，在产品列表中，单击产品所对应的查看操作按钮。
3. 单击功能定义 > 新增为产品新增物模型。

4. 在添加功能页面，选择功能类型为属性。

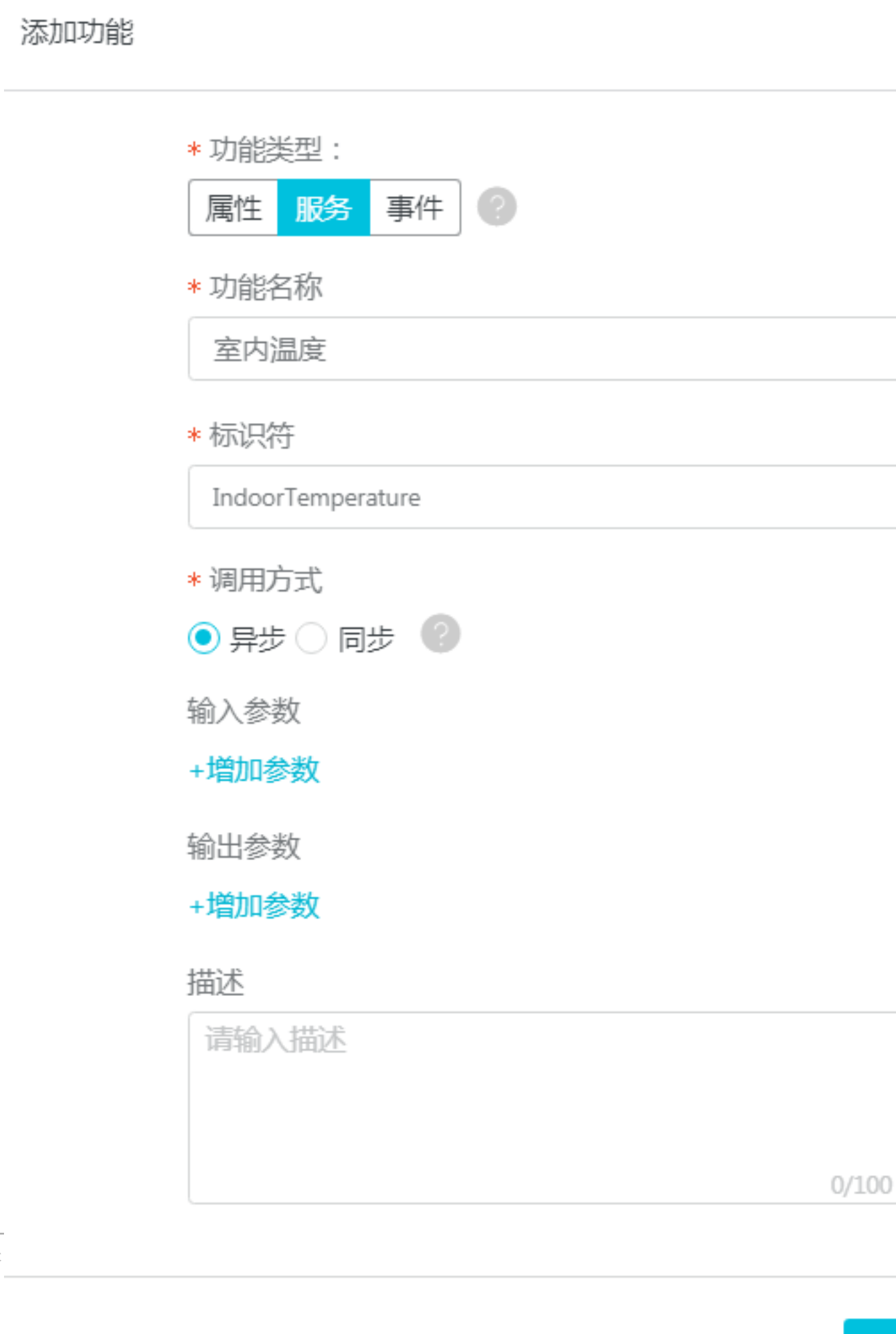

添加功能
* 功能类型：
属性
服务
事件
* 功能名称
室内温度
* 标识符
IndoorTemperature
* 数据类型
float
* 取值范围：
-40.0
~
55.0
* 分辨率
1
单位：
摄氏度 / °C
读写类型：
读写
只读
描述

属性参数设置如下：

参数	描述
功能名称	属性的名称，如用电量。同一产品下功能名称不能重复。如果您创建产品时选择了功能模板，输入功能名称时，将自动从标准功能库中筛选匹配的标准属性，供您选择。 **说明：** 当接入网关协议为Modbus时，不支持标准属性，仅支持自定义属性。
标识符	属性唯一标识符，在产品中具有唯一性。即 Alink JSON 格式中的**identifier**的值，作为设备上报该属性数据的 Key，云端根据该标识符校验是否接收数据。可包含英文、数字、下划线，如 PowerComsuption。
数据类型	• int32：32位整型。需定义取值范围、分辨率和单位符号。 • float：单精度浮点型。需定义取值范围、分辨率和单位符号。 • double：双精度浮点型。需定义取值范围、分辨率和单位符号。 • enum：枚举型。定义枚举项的参数值和参数描述，如 1-加热模式、2-制冷模式。 • bool：布尔型。采用 0 或 1 来定义布尔值，如 0-关、1-开。 • text：字符串。需定义字符串的数据长度，最长支持 2048 字节。 • date：时间戳。格式为 string 类型的 UTC 时间戳，单位：毫秒。 • struct：JSON对象。定义一个 JSON 结构体，新增 JSON 参数项，如定义灯的颜色是由 Red、Green、Blue 三个参数组成的结构体。不支持结构体嵌套。 • array：数组。需声明数组内元素的数据类型，可选择**int32**、**float**、**double**、或**text**。需确保同一个数组元素类型相同，数组长度最长不超过 128 个元素。 **说明：** 当设备协议为Modbus时，无需设置该参数。
读写类型	• 读写：请求读写的方法支持 GET（获取）和 SET（设置）。 • 只读：请求只读的方法仅支持 GET（获取）。 **说明：** 当接入网关协议为Modbus时，无需设置该参数。
描述	输入文字，对该功能进行说明或备注。

参数	描述
扩展描述 说明： 当接入网关协议为Modbus时	• 操作类型：包括输入状态（只读）、线圈状态（读写）、保持寄存器（读写）、输入寄存器（只读）。 • 寄存器地址：十六进制，必须以0x开头，如0xFE，限制范围是0x0~0xFFFF。 • 原始数据类型：支持int16、uint16、int32、uint32、int64、uint64、float、double、string、bool、自定义（原始数据）多种数据类型。 • 寄存器数据个数：操作类型为输入状态/线圈状态时限制为1~2000，为保持寄存器/输入寄存器时限制为1~125。 • 交换寄存器内高低字节：把寄存器内16位数据的前后8个bits互换。 • 交换寄存器顺序：把原始数据32位数据的bits互换。 • 缩放因子：不能为0，默认为1，可以为负数。 • 采集间隔：数据采集间隔，单位ms，不能小于10。 • 数据上报方式：分为按时上报和变更上报。
扩展描述 说明： 当接入网关协议为OPC UA时	节点名称：需保证属性维度下唯一。

5. 在添加功能页面，选择功能类型为服务。


添加功能
* 功能类型：
属性
服务
事件
* 功能名称
室内温度
* 标识符
IndoorTemperature
* 调用方式
异步
同步
输入参数
+增加参数
输出参数
+增加参数
描述
请输入描述
0/100

服务参数设置如下：

参数	描述
功能名称	服务名称。如果您创建产品时选择了功能模板，输入功能名称时，将自动从标准功能库中筛选匹配的标准服务，供您选择。 **说明：** 当接入网关协议为Modbus时，不支持自定义服务。
标识符	服务唯一标识符，在产品下具有唯一性。即 Alink JSON 格式中该服务的**identifier**的值，作为设备上报该属性数据的 Key。可包含英文、数字、和下划线。
调用方式	• 异步：服务为异步调用时，云端执行调用后直接返回结果，不会等待设备的回复消息。 • 同步：服务为同步调用时，云端会等待设备回复；若设备没有回复，则调用超时。
输入参数	设置该服务的入参，可选。 单击**新增参数**，在弹窗对话框中添加一个服务入参。该参数设置，可参考*第4步的设置*。 当接入网关协议为OPC UA时，需设置参数索引，用于标记参数的顺序。 **说明：** • 您可以直接选择某个属性作为入参，也可以自定义参数。如在定义**自动喷灌**服务功能时，将已定义的属性**喷灌时间**和**喷灌量**作为自动喷灌服务的入参，则调用该参数时传入这两个参数，喷灌设备将按照设定的喷灌时间和喷灌量自动进行精准灌溉。 • 一个服务最多支持定义 10 个入参。
输出参数	设置该服务的出参，可选。 单击**新增参数**，在弹窗对话框中添加一个服务出参。该参数设置，可参考*第4步的设置*。 当接入网关协议为OPC UA时，需设置参数索引，用于标记参数的顺序。 **说明：** • 您可以直接选择某个属性作为出参，也可以自定义参数，如将已定义的属性**土壤湿度**作为出参，则云端调用自动喷灌服务时，将返回当前土壤湿度的数据。 • 一个服务最多支持定义10个出参。

参数	描述
描述	输入文字，对该服务功能进行说明或备注。
扩展描述 说明：当接入网关协议为OPC UA时	节点名称：需保证服务维度下唯一。

6. 在添加功能页面，选择功能类型为事件。


添加功能
* 功能类型：
属性
服务
事件
* 功能名称
故障上报
* 标识符
Error
* 事件类型
信息
告警
故障
输出参数
参数名称：：故障代码
+增加参数
描述
请输入描述
0/100

事件参数设置如下：

参数	描述
功能名称	事件的名称。 说明： 当接入网关协议为Modbus时，不支持自定义事件。
标识符	事件唯一标识符，在产品下具有唯一性。即 Alink JSON 格式中该事件的**identifier**的值，作为设备上报该属性数据的 Key，如 ErrorCode。
事件类型	• 信息：指设备上报的一般性通知，如完成某项任务等。 • 告警：设备运行过程中主动上报的突发或异常情况，告警类信息，优先级高。您可以针对不同的事件类型进行业务逻辑处理和统计分析。 • 故障：设备运行过程中主动上报的突发或异常情况，故障类信息，优先级高。您可以针对不同的事件类型进行业务逻辑处理和统计分析。
输出参数	该事件的出参。单击**新增参数**，在弹窗对话框中添加一个服务出参。您可以直接选择某个属性作为出参，也可以自定义参数。如，将已定义的属性电压作为出参，则设备上报该故障事件时，将携带当前设备的电压值，用于进一步判断故障原因。 当接入网关协议为OPC UA时，需设置参数索引，用于标记参数的顺序。 说明： 一个事件最多支持定义10个出参。
描述	输入文字，对该事件功能进行说明或备注。
扩展描述 说明： 当接入网关协议为OPC UA时	节点名称：需保证事件维度下唯一。

2.3.3 导入物模型

本文讲述如何将已有物模型导入新创建的产品。

操作步骤

1. 登录物联网平台控制台。
2. 在产品管理页面，在产品列表中，单击产品所对应的**查看**操作按钮。

3. 单击**功能定义 > 导入物模型**为产品导入物模型。

说明：

- 导入物模型后，会覆盖该产品原有的功能定义。请谨慎使用。
- 设备协议为Modbus的产品，不支持导入物模型。


产品管理 > 产品详情
空调1
高级版
ProductKey：
复制
产品信息
消息通信
功能定义
根据产品的设备类型，我们已
功能类型
属性
属性
属性
属性
事件
导入物模型
注：导入的物
拷贝产品
* 选择
请选

- 复用已有产品的物模型

  1. 在**拷贝产品**页面，选择已有产品，单击**确定**，即可将已有产品的物模型导入到此产品中。
  2. 单击某功能对应的**编辑**按钮，即可修改该功能。

- 如果您已写好物模型脚本，可单击**导入物模型**，然后将物模型脚本粘贴进编辑框中。

2.3.4 物模型格式

物模型以JSON格式表达，简称为TSL（Thing Specification Language）。本文提供物模型的JSON字段说明。

您可以在产品的**功能定义**页面，单击**查看物模型**，查看 JSON 格式的 TSL。

物模型的 JSON 字段说明如下：

```
{
    "schema":"物的TSL描述schema",
    "link":"云端系统级uri,用来调用服务/订阅事件",
    "profile":{
        "productKey":"产品key"
    },
    "properties":[
        {
            "identifier":"属性唯一标识符(产品下唯一)",
            "name":"属性名称",
            "accessMode":"属性读写类型，只读(r)，读写(rw)",
            "required":"标准品类中的必选是否是标准功能的必选属性",
            "dataType":{
                "type":"属性类型: int(原生)，float(原生)，double(原生),
text(原生)，date(String类型UTC毫秒)，bool(0或1的int类型)，enum(int类型),
struct(结构体类型，可包含前面6种类型)，array(数组类型，支持int/double/float/
text)",
                "specs":{
                    "min":"参数最小值(int, float, double类型特有)",
                    "max":"参数最大值(int, float, double类型特有)",
                    "unit":"属性单位",
                    "unitName":"单位名称",
                    "size":"数组大小，默认最大128(数组特有)",
                    "item":{
                        "type":"数组元素的类型"
                    }
                }
            }
        }
    ],
    "events":[
        {
            "identifier":"事件唯一标识符(产品下唯一，其中post是默认生成的属性
上报事件)",
            "name":"事件名称",
            "desc":"事件描述",
```

```
            "type":"事件类型(info,alert,error)",
            "required":"是否是标准功能的必选事件",
            "outputData":[
                {
                    "identifier":"参数唯一标识符",
                    "name":"参数名称",
                    "dataType":{
                        "type":"属性类型: int(原生) , float(原生) , double
(原生), text(原生) , date(String类型UTC毫秒) , bool(0或1的int类型) , enum(int
类型), struct(结构体类型，可包含前面6种类型) , array(数组类型，支持int/double/
float/text)",
                        "specs":{
                            "min":"参数最小值(int, float, double类型特
有)",
                            "max":"参数最大值(int, float, double类型特
有)",
                            "unit":"属性单位",
                            "unitName":"单位名称",
                            "size":"数组大小，默认最大128(数组特有)",
                            "item":{
                                "type":"数组元素的类型"
                            }
                        }
                    }
                }
            ],
            "method":"事件对应的方法名称(根据identifier生成)"
        }
    ],
    "services":[
        {
            "identifier":"服务唯一标识符(产品下唯一，产品下唯一，其中set/get
是根据属性的accessMode默认生成的服务)",
            "name":"服务名称",
            "desc":"服务描述",
            "required":"是否是标准功能的必选服务",
            "inputData":[
                {
                    "identifier":"入参唯一标识符",
                    "name":"入参名称",
                    "dataType":{
                        "type":"属性类型: int(原生) , float(原生) , double
(原生), text(原生) , date(String类型UTC毫秒) , bool(0或1的int类型) , enum(int
类型), struct(结构体类型，可包含前面6种类型) , array(数组类型，支持int/double/
float/text)",
                        "specs":{
                            "min":"参数最小值(int, float, double类型特
有)",
                            "max":"参数最大值(int, float, double类型特
有)",
                            "unit":"属性单位",
                            "unitName":"单位名称",
                            "size":"数组大小，默认最大128(数组特有)",
                            "item":{
                                "type":"数组元素的类型"
```

```
                            }
                        }
                    }
                }
            ],
            "outputData":[
                {
                    "identifier":"出参唯一标识符",
                    "name":"出参名称",
                    "dataType":{
                        "type":"属性类型: int(原生) , float(原生) , double
(原生), text(原生) , date(String类型UTC毫秒) , bool(0或1的int类型) , enum(int
类型), struct(结构体类型，可包含前面6种类型) , array(数组类型，支持int/double/
float/text)",
                        "specs":{
                            "min":"参数最小值(int, float, double类型特
有)",
                            "max":"参数最大值(int, float, double类型特
有)",
                            "unit":"属性单位",
                            "unitName":"单位名称",
                            "size":"数组大小，默认最大128(数组特有)",
                            "item":{
                                "type":"数组元素的类型(数组特有)"
                            }
                        }
                    }
                }
            ],
            "method":"服务对应的方法名称(根据identifier生成)"
        }
    ]
}
```

若该产品是设备，且设备协议为Modbus或OPC UA时，可查看物模型扩展配置。

```
{
  "profile": {
    "productKey": "产品key"
  },
  "properties": [
    {
      "identifier": "属性唯一标识符(产品下唯一)",
      "operateType": "(线圈状态/输入状态/保持寄存器/输入寄存器: coilStatus/
inputStatus/holdingRegister/inputRegister)",
      "registerAddress": "寄存器地址",
      "originalDataType": {
        "type": "属性类型:int16 , uint16, int32 , uint32 , int64 , uint64,
float , double, string , customized data(按大端顺序返回hex data)",
        "specs": {
          "registerCount": "寄存器的数据个数”, string , customized data特
有",
          "swap16": "把寄存器内16位数据的前后8个bits互换 ( byte1byte2 ->
byte2byte10) , 除 string , customized data外，其他数据类型特有",
```

```
          "reverseRegister": "Ex:把原始数据32位数据的bits互换（byte1byte2
byte3byte4 ->byte3byte4byte1byte2”，除 string，customized data外，其他数
据类型特有"
        }
      },
      "scaling": "缩放因子",
      "pollingTime": "采集间隔, 单位是ms",
      "trigger": "数据的上报方式，目前有 按时上报:1和变更上报:2"
    }
  ]
}
```

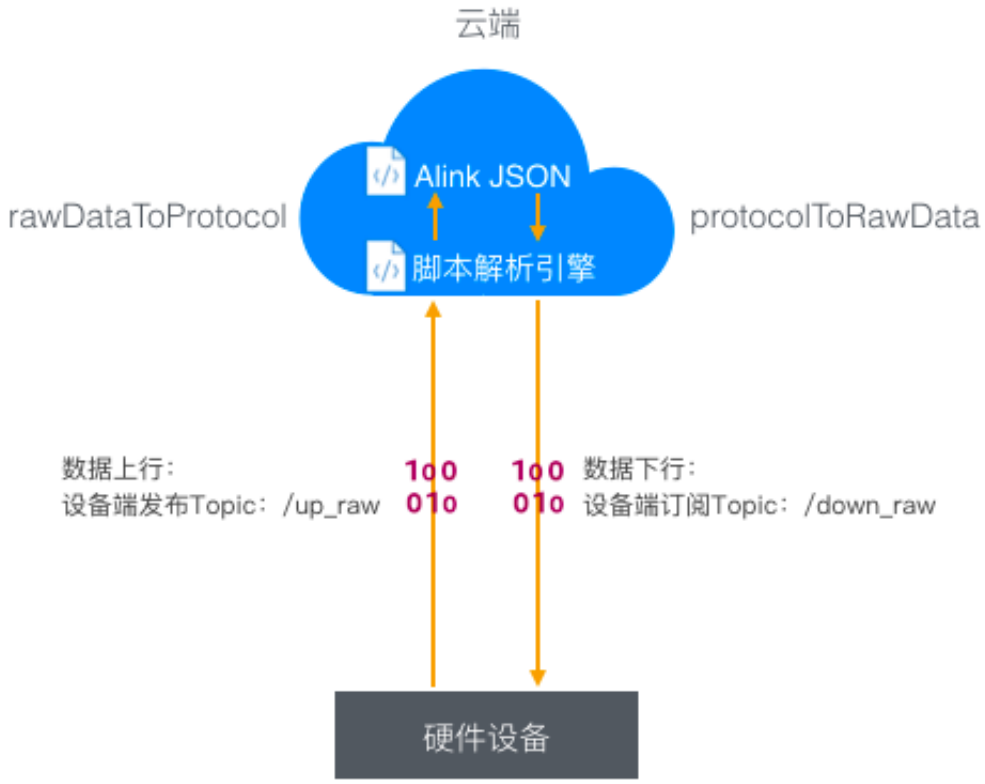
2.4 数据解析

如果您在控制台创建产品时，数据格式选择了**透传/自定义**。您可以在物联网平台控制台上，编写脚本，解析设备数据。

关于数据解析

由于低配置且资源受限或者对网络流量有要求的设备，不适合直接构造JSON数据和云端通信，因此选择将数据透传到云端，由云端运行转换脚本将透传的数据转换成Alink JSON格式的数据。您可以在创建产品时，选择数据格式为**透传/自定义格式**，目前转换脚本通过JavaScript语言开发，需要开发者自行开发转换脚本。物联网平台为开发者提供了用于数据解析的在线脚本编辑器，方便您进行在线的编辑和模拟调试。

数据解析流程：



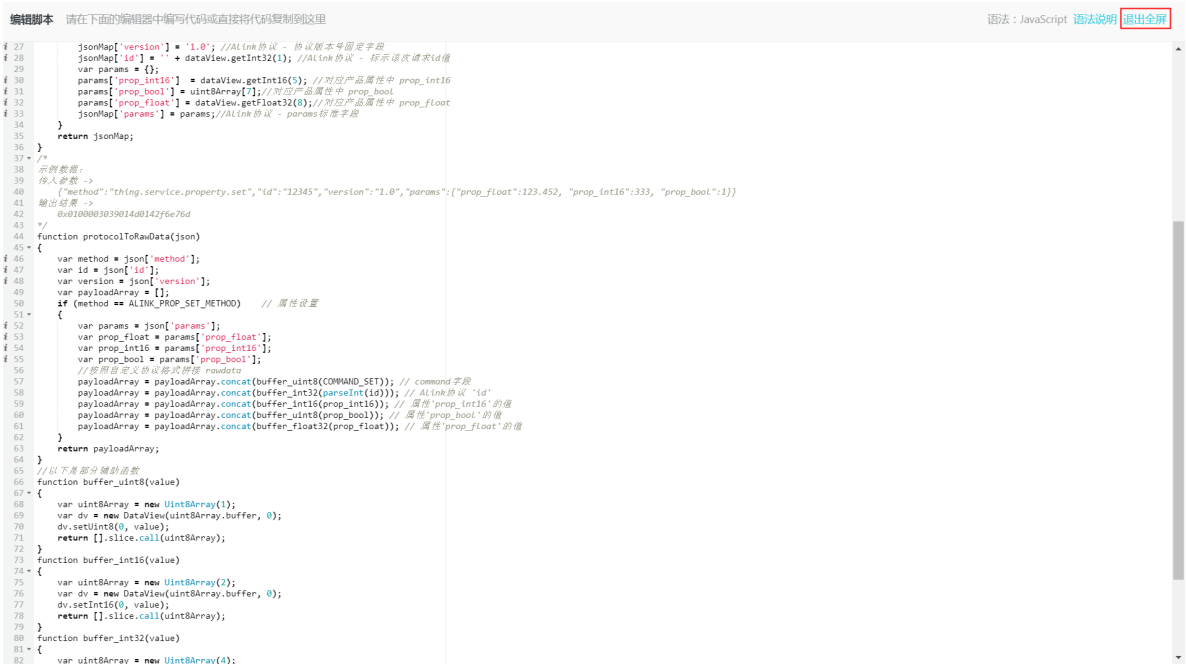
数据解析将为您提供：

- 脚本在线编辑器，支持 JavaScript 语言；
- 支持保存草稿，并从草稿中恢复编辑，支持删除草稿；
- 支持对脚本的模拟数据调试，可输入上下行数据进行模拟转换，查看脚本运行结果；
- 支持对脚本的静态语法检查（JavaScript 语法）；
- 提交脚本到运行环境，在设备上下行时进行调用；

在线编辑脚本

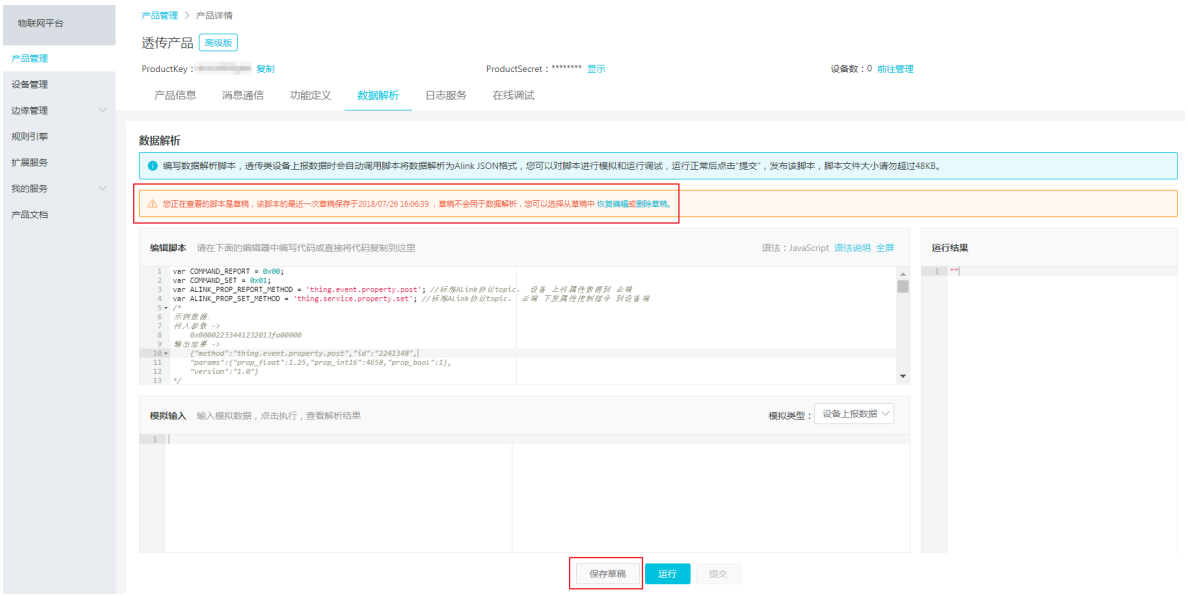
在产品详情页面，单击**数据解析**进入脚本编辑页面，在下方编辑框中编写数据解析脚本，目前支持编写 JavaScript 语言。

- 单击全屏，支持全屏查看或编辑脚本，单击退出全屏即可退出当前编辑界面。




```
        jsonMap['version'] = '1.0'; //Alink协议 - 协议版本号固定字段
        jsonMap['id'] = '' + dataView.getInt32(1); //Alink协议 - 标示该次请求id值
        var params = {};
        params['prop_int16']  = dataView.getInt16(5); //对应产品属性中 prop_int16
        params['prop_bool'] = uint8Array[7];//对应产品属性中 prop_bool
        params['prop_float'] = dataView.getFloat32(8);//对应产品属性中 prop_float
        jsonMap['params'] = params;//Alink协议 - params标准字段
    }
    return jsonMap;
}
/*
示例数据:
传入参数 ->
    {"method":"thing.service.property.set","id":"12345","version":"1.0","params":{"prop_float":123.452, "prop_int16":333, "prop_bool":1}}
输出结果 ->
    0x0100003039014d0142f6e76d
*/
function protocolToRawData(json)
{
    var method = json['method'];
    var id = json['id'];
    var version = json['version'];
    var payloadArray = [];
    if (method == ALINK_PROP_SET_METHOD)    // 属性设置
    {
        var params = json['params'];
        var prop_float = params['prop_float'];
        var prop_int16 = params['prop_int16'];
        var prop_bool = params['prop_bool'];
        //按照自定义协议格式拼接 rawdata
        payloadArray = payloadArray.concat(buffer_uint8(COMMAND_SET)); // command 字段
        payloadArray = payloadArray.concat(buffer_int32(parseInt(id))); // Alink协议 'id'
        payloadArray = payloadArray.concat(buffer_int16(prop_int16)); // 属性'prop_int16'的值
        payloadArray = payloadArray.concat(buffer_uint8(prop_bool)); // 属性'prop_bool'的值
        payloadArray = payloadArray.concat(buffer_float32(prop_float)); // 属性'prop_float'的值
    }
    return payloadArray;
}
//以下是部分辅助函数
function buffer_uint8(value)
{
    var uint8Array = new Uint8Array(1);
    var dv = new DataView(uint8Array.buffer, 0);
    dv.setUint8(0, value);
    return [].slice.call(uint8Array);
}
function buffer_int16(value)
{
    var uint8Array = new Uint8Array(2);
    var dv = new DataView(uint8Array.buffer, 0);
    dv.setInt16(0, value);
    return [].slice.call(uint8Array);
}
function buffer_int32(value)
{
    var uint8Array = new Uint8Array(4);
```

- 单击页面底部的保存草稿，系统将保存本次编辑的结果，您下次再进入平台时，将提示您最近一次保存的草稿，您可以选择恢复编辑或删除草稿。
  - 草稿不会进入到脚本解析的运行环境中，保存草稿不会影响已经提交的正式脚本；
  - 每次保存草稿将覆盖上一次保存的草稿；

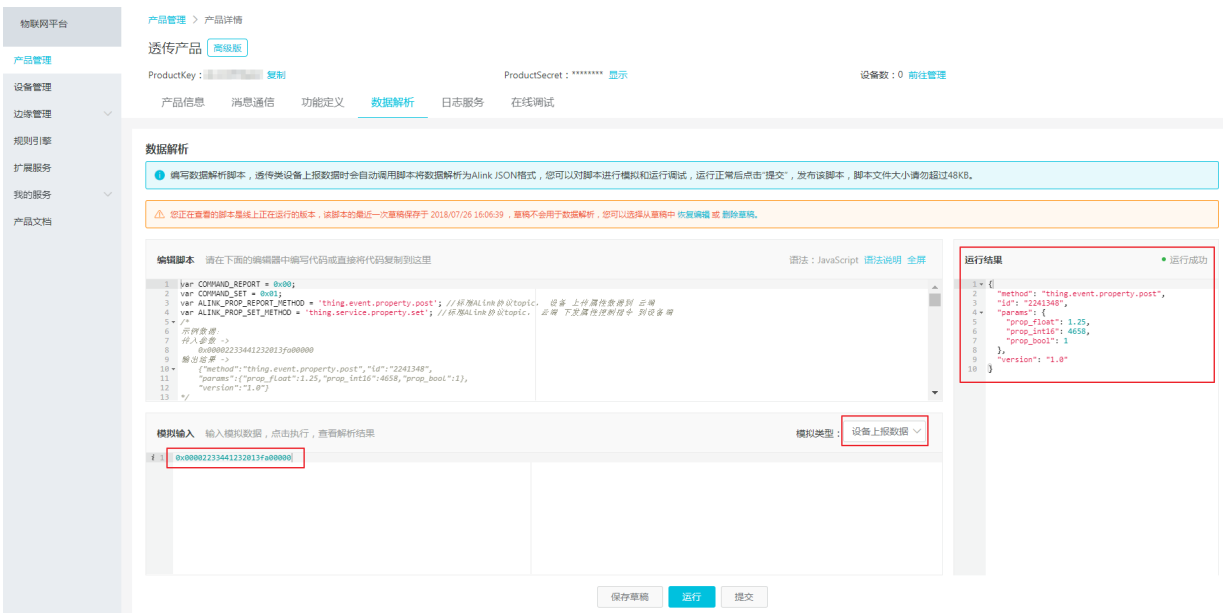


模拟运行

脚本编辑完成后，您可以在编辑器下方输入模拟数据，验证脚本的运行情况，输入模拟数据后，单击运行，系统将调用该脚本模拟进行数据解析，运行结果将显示在右侧的运行结果区域中，若脚本存在错误，将提示运行不通过，请重新修改脚本代码。

模拟上行数据解析

选择模拟类型为**设备上报数据**，输入透传二进制数据，单击**运行**，会将二进制透传数据按照脚本规则转换为JSON格式数据，您可以在运行结果区域看到透传数据的解析结果。

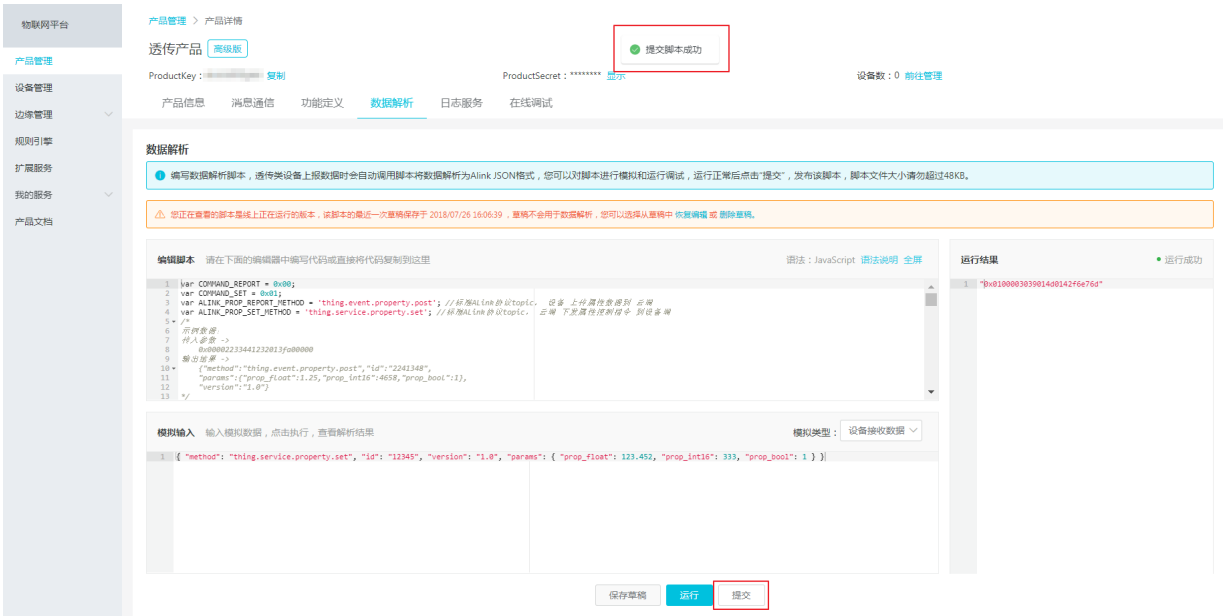


模拟下行数据解析

选择模拟类型为设备接收数据，输入JSON格式数据，单击运行，会将JSON数据按照脚本规则转换为二进制数据，您可以在运行结果区域看到解析后的数据。

提交脚本

为避免错误提交脚本造成数据无法上报，脚本运行通过后，您才可以将脚本提交到运行环境中，设备上下行时将自动调用脚本，实现数据的解析和转换。



说明：

提交脚本之前需至少运行过一次脚本并且运行通过才可提交。

脚本开发框架说明

概述

脚本只需要支持以下两个方法即可接入到物联网平台：

- Alink JSON格式数据转二进制数据：`protocolToRawData`
- 二进制数据转Alink JSON格式数据：`rawDataToProtocol`

脚本语言

目前脚本仅支持符合ECMAScript 5.1的JavaScript语法。

方法定义

- Alink JSON格式数据转二进制数据，方法如下：

```
//解析服务端发送的AlinkJson格式数据，并转换为二进制数据返回
function protocolToRawData(jsonObj){
    return rawdata;
}
```

参数定义：输入参数jsonObj ：符合产品 TSL 定义的 Alink JSON格式数据， 例如：

```
{
    "method": "thing.service.property.set",
    "id": "12345",
    "version": "1.0",
    "params": {
        "prop_float": 123.452,
        "prop_int16": 333,
        "prop_bool": 1
    }
}
```

返回参数 ：二进制byte数组， 如：

```
0x0100003039014d0142f6e76d
```

- 二进制数据转Alink JSON格式数据，方法如下：

```
//解析设备端发送的二进制数据，并转换为Alink JSON格式数据返回
function rawDataToProtocol(rawData){
    return jsonObj;
```

```
}
```

参数定义：输入参数rawData ：二进制byte数组， 例如：

```
0x00002233441232013fa00000
```

返回参数 ：符合产品 TSL 定义的 Alink JSON格式数据，如：

```
{
    "method": "thing.event.property.post",
    "id": "2241348",
    "params": {
        "prop_float": 1.25,
        "prop_int16": 4658,
        "prop_bool": 1
    },
    "version": "1.0"
}
```

脚本Demo示例

1. 创建产品并完成功能定义。

   a. 创建一款高级版产品，数据格式选择为**透传/自定义格式**；

   b. 定义产品功能，创建属性、服务和事件，在本示例中将创建以下三个属性：

标识符	数据类型
prop_float	浮点单精度 float
prop_int16	整数型 int32
prop_bool	布尔型 bool

2. 串口协议格式示例。

帧类型	ID	prop_int16	prop_bool	prop_float
1字节 0-上报；1-下发	请求序号	2字节 prop_int16属性值	1字节 prop_bool属性值	4字节 prop_float属性值

3. 拷贝脚本Demo代码。

   请将参考代码复制到如下输入框中：

```
var COMMAND_REPORT = 0x00;
var COMMAND_SET = 0x01;
var ALINK_PROP_REPORT_METHOD = 'thing.event.property.post'; //标准
ALink JSON格式topic， 设备 上传属性数据到 云端
```

```
var ALINK_PROP_SET_METHOD = 'thing.service.property.set'; //标准ALink
 JSON格式topic， 云端 下发属性控制指令 到设备端
/*
示例数据：
传入参数 ->
    0x000022334412320l3fa00000
输出结果 ->
    {"method":"thing.event.property.post","id":"2241348",
    "params":{"prop_float":1.25,"prop_int16":4658,"prop_bool":1},
    "version":"1.0"}
*/
function rawDataToProtocol(bytes) {
    var uint8Array = new Uint8Array(bytes.length);
    for (var i = 0; i < bytes.length; i++) {
        uint8Array[i] = bytes[i] & 0xff;
    }
    var dataView = new DataView(uint8Array.buffer, 0);
    var jsonMap = new Object();
    var fHead = uint8Array[0]; // command
    if (fHead == COMMAND_REPORT) {
        jsonMap['method'] = ALINK_PROP_REPORT_METHOD; //ALink JSON格
式 - 属性上报topic
        jsonMap['version'] = '1.0'; //ALink JSON格式 - 协议版本号固定字
段
        jsonMap['id'] = '' + dataView.getInt32(1); //ALink JSON格式
 - 标示该次请求id值
        var params = {};
        params['prop_int16'] = dataView.getInt16(5); //对应产品属性中
prop_int16
        params['prop_bool'] = uint8Array[7]; //对应产品属性中
prop_bool
        params['prop_float'] = dataView.getFloat32(8); //对应产品属性
中 prop_float
        jsonMap['params'] = params; //ALink JSON格式 - params标准字段
    }
    return jsonMap;
}
/*
示例数据：
传入参数 ->
    {"method":"thing.service.property.set","id":"12345","version":"
1.0","params":{"prop_float":123.452, "prop_int16":333, "prop_bool":1
}}
输出结果 ->
    0x0100003039014d0142f6e76d
*/
function protocolToRawData(json) {
    var method = json['method'];
    var id = json['id'];
    var version = json['version'];
    var payloadArray = [];
    if (method == ALINK_PROP_SET_METHOD) // 属性设置
    {
        var params = json['params'];
        var prop_float = params['prop_float'];
        var prop_int16 = params['prop_int16'];
        var prop_bool = params['prop_bool'];
        //按照自定义协议格式拼接 rawdata
```

```
        payloadArray = payloadArray.concat(buffer_uint8(COMMAND_SET
)); // command字段
        payloadArray = payloadArray.concat(buffer_int32(parseInt(id
))); // ALink JSON格式 'id'
        payloadArray = payloadArray.concat(buffer_int16(prop_int16
)); // 属性'prop_int16'的值
        payloadArray = payloadArray.concat(buffer_uint8(prop_bool
)); // 属性'prop_bool'的值
        payloadArray = payloadArray.concat(buffer_float32(prop_float
)); // 属性'prop_float'的值
    }
    return payloadArray;
}
//以下是部分辅助函数
function buffer_uint8(value) {
    var uint8Array = new Uint8Array(1);
    var dv = new DataView(uint8Array.buffer, 0);
    dv.setUint8(0, value);
    return [].slice.call(uint8Array);
}
function buffer_int16(value) {
    var uint8Array = new Uint8Array(2);
    var dv = new DataView(uint8Array.buffer, 0);
    dv.setInt16(0, value);
    return [].slice.call(uint8Array);
}
function buffer_int32(value) {
    var uint8Array = new Uint8Array(4);
    var dv = new DataView(uint8Array.buffer, 0);
    dv.setInt32(0, value);
    return [].slice.call(uint8Array);
}
function buffer_float32(value) {
    var uint8Array = new Uint8Array(4);
    var dv = new DataView(uint8Array.buffer, 0);
    dv.setFloat32(0, value);
    return [].slice.call(uint8Array);
}
```

4. 模拟数据解析。

   - 模拟设备上报数据

     模拟类型选择**设备上报数据**，填写测试数据：

     ```
     0x00002233441232013fa00000
     ```

     点击运行，查看上报数据输出结果：

     ```
     {
         "method": "thing.event.property.post",
         "id": "2241348",
         "params": {
             "prop_float": 1.25,
             "prop_int16": 4658,
             "prop_bool": 1
         },
         "version": "1.0"
     ```

```
}
```

- 模拟设备接收数据模拟类型选择**设备接收数据**，填写测试数据：

```
{
    "method": "thing.service.property.set",
    "id": "12345",
    "version": "1.0",
    "params": {
        "prop_float": 123.452,
        "prop_int16": 333,
        "prop_bool": 1
    }
}
```

点击运行，查看接收数据输出结果：

```
0x0100003039014d0142f6e76d
```

附录：本地环境调试脚本的方法

目前物联网平台数据解析不支持debug调试，建议开发过程先在本地开发完成后再将脚本拷贝到物联网控制台的脚本编辑器中。可参考如下方式调用。

```
// Test Demo
function Test()
{
    //0x001232013fa00000
    var rawdata_report_prop = new Buffer([
        0x00, //固定command头, 0代表是上报属性
        0x00, 0x22, 0x33, 0x44, //对应id字段， 标记请求的序号
        0x12, 0x32, //两字节 int16, 对应属性 prop_int16
        0x01, //一字节 bool, 对应属性 prop_bool
        0x3f, 0xa0, 0x00, 0x00 //四字节 float, 对应属性 prop_float
    ]);
    rawDataToProtocol(rawdata_report_prop);
    var setString = new String('{"method":"thing.service.property.
set","id":"12345","version":"1.0","params":{"prop_float":123.452, "
prop_int16":333, "prop_bool":1}}');
    protocolToRawData(JSON.parse(setString));
}
Test();
```

2.5 虚拟设备

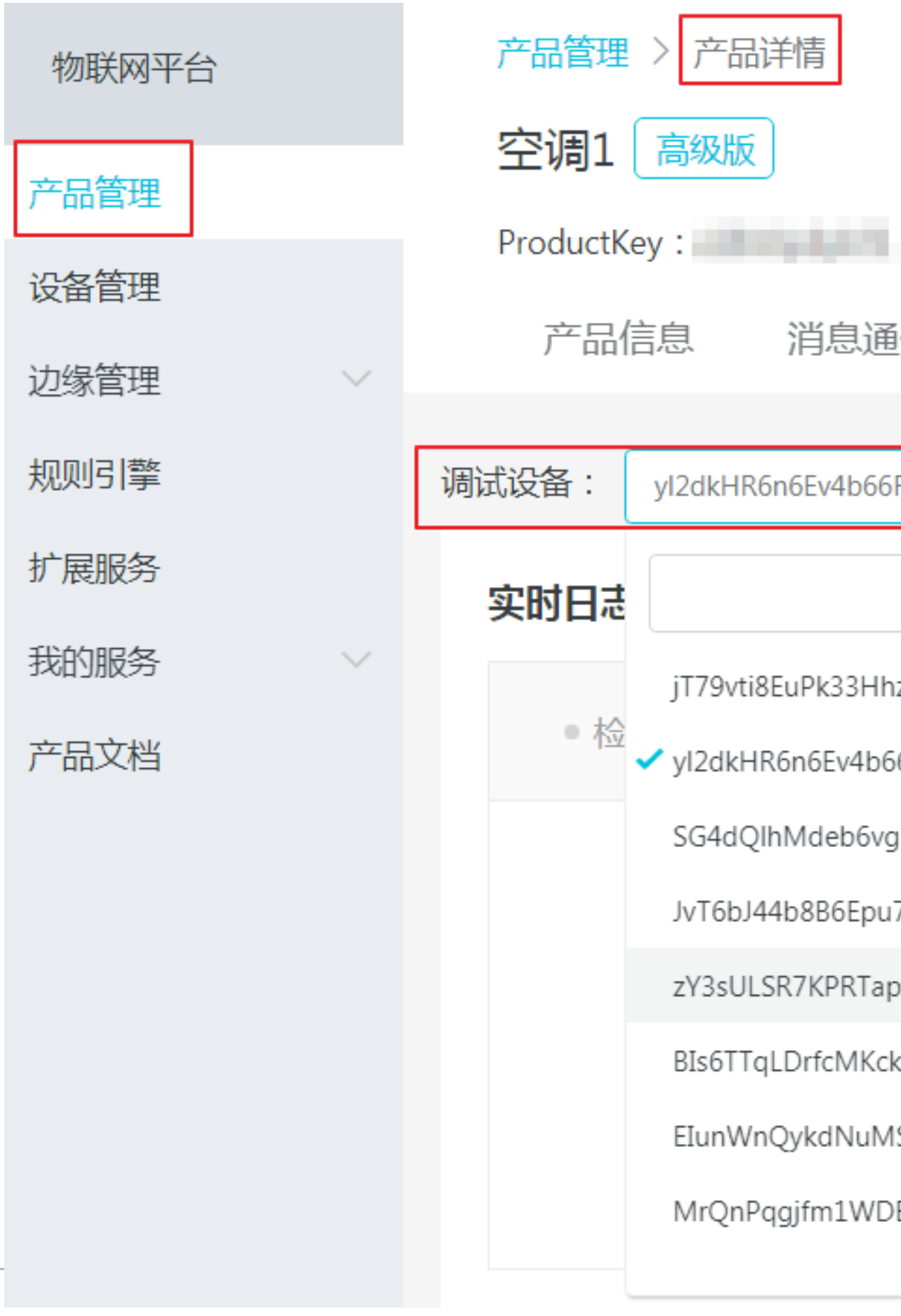
物联网平台提供虚拟设备，供云端开发测试使用。目前仅高级版支持该功能。

物联网正常开发流程是：设备端开发完成，设备上报数据，云端接收数据，云端开始开发工作。这样的开发流程战线较长，耗时较久。物联网平台提供虚拟设备功能，虚拟设备模拟真实设备与云端建立连接，上报定义的属性及事件处理。此时，云端可根据虚拟设备的数据，完成应用的开发调试。当而真实设备上线后，虚拟设备也会自动下线。

操作步骤

1. 登录物联网平台控制台。
2. 单击产品管理，在产品列表中，选中某个产品，单击查看。
3. 在产品详情页面上，单击在线调试。

4. 选中虚拟设备的设备名称。

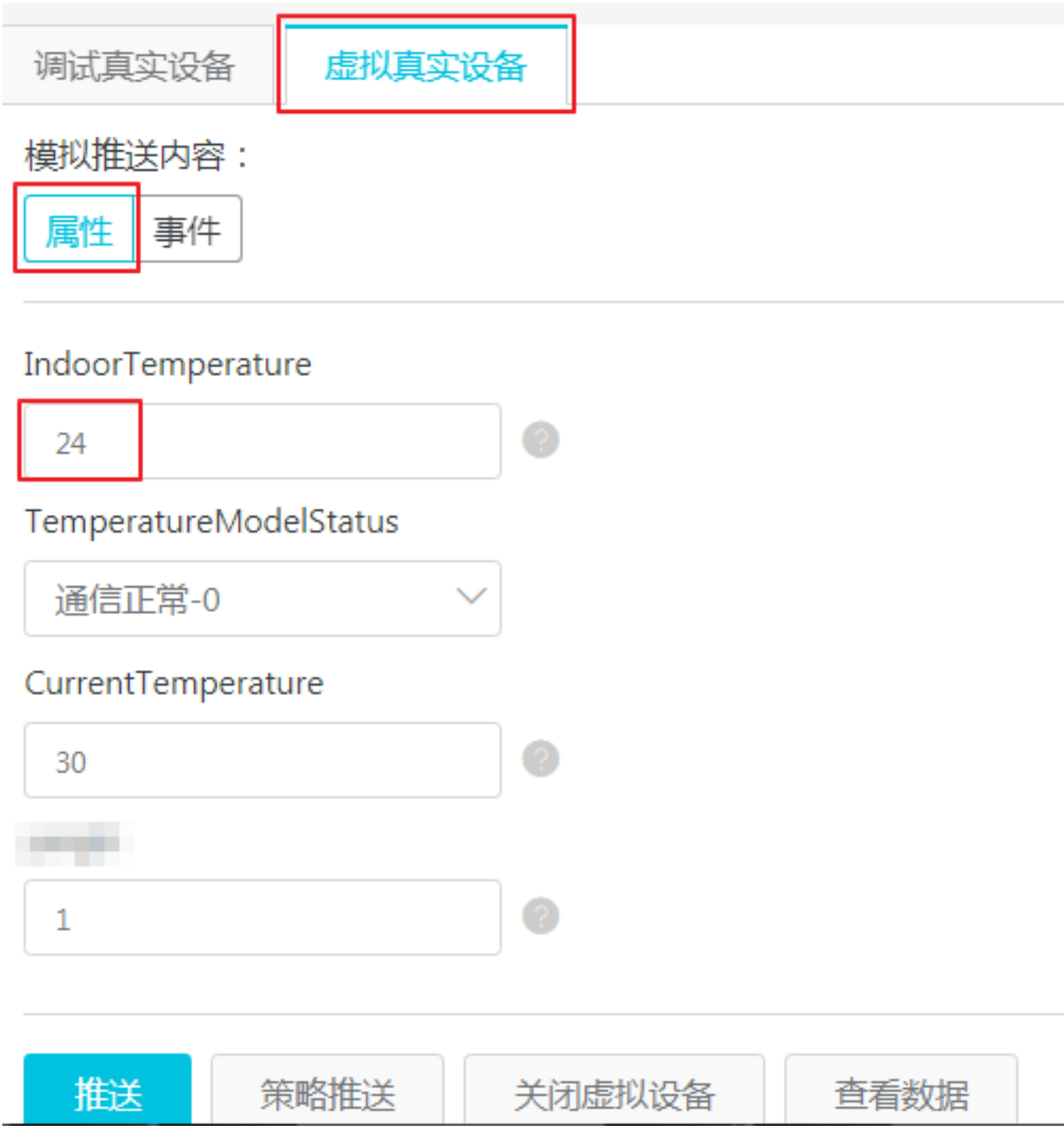


5. 单击虚拟真实设备 > 启动虚拟设备。

> **说明：**
> 当真实设备在线或被禁用时，虚拟设备将会启动失败。

6. 设置模拟推送的内容。

   例如，此处推送属性，室内温度24摄氏度。



7. 选择推送方式。

   - 推送：仅推送一次。
   - 策略推送：
     - 定时推送：在设置好的时间推送数据。
     - 连续推送：在设置好的时间段内，按照固定时间间隔，推送数据。时间间隔单位为秒。

   您可以单击**查看数据**，进入该设备的详情页，查看设备的运行状态。

   虚拟设备使用完成，您可以单击**关闭虚拟设备**，关闭此设备。

使用限制

- 连续推送的最小时间间隔为1秒。
- 最多连续推送1000条消息。
- 单次推送，每天最多推送100次。

2.6 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。

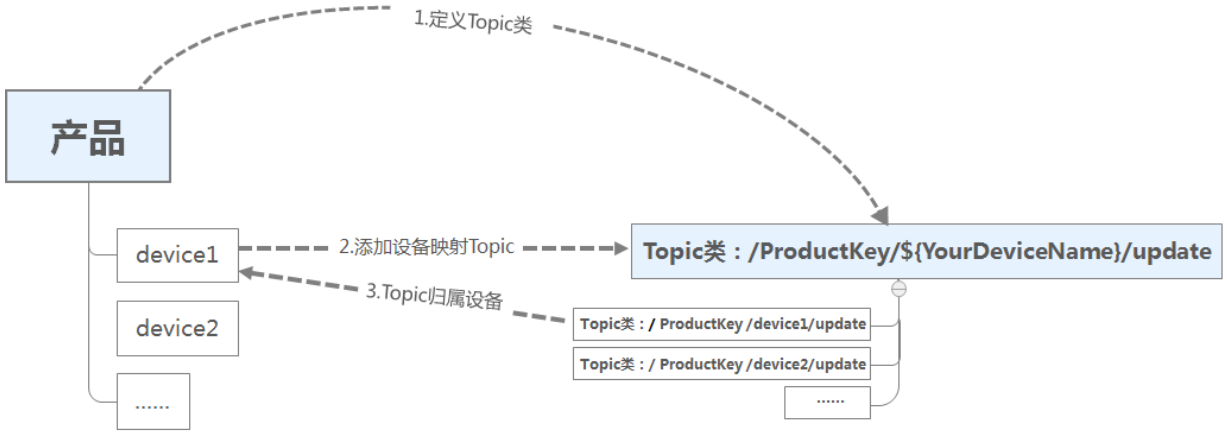
2.6.1 什么是Topic

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。

什么是Topic类？

为了方便海量设备基于海量 Topic 进行通信，简化授权操作，物联网平台增加了 Topic 类的概念。您创建产品后，物联网平台会为该产品自动创建默认的 Topic 类。并且，在您创建设备后，会自动将产品 Topic 类映射到设备上。您无需单独为每个设备授权 Topic。

图 2-1: Topic 自动生成示意图



在您创建产品后，物联网平台会自动为您的产品生成一些标准的 Topic 类。您可以在产品的**消息通信**页面，查看该产品的所有 Topic 类。

关于 Topic 类的说明：

- Topic类是一类 Topic 的集合。例如，Topic 类：`/${YourProductKey}/${YourDeviceName}/update`是具体 Topic：`/${YourProductKey}/device1/update`和`/${YourProductKey}/device2/update`的集合。
- Topic类中必须以正斜线(/)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。
- 类目命名只能包含字母，数字和下划线(_)。每级类目不能为空。
- 设备操作权限：**发布**表示设备可以往 Topic 发布消息；**订阅**表示设备可以从 Topic 订阅消息。
- 基础版产品和高级版产品都支持自定义 Topic 类。您可以根据业务需求，通过自定义 Topic 类灵活地进行消息通信。
- 系统 Topic 类是由系统预定义的 Topic 类，不支持用户自定义，不采用`/${YourProductKey}`开头。例如，高级版中，针对物模型所提供的 Topic 类一般以`/sys/`开头；固件升级相关的Topic类以`/ota/`开头；设备影子的 Topic 类以`/shadow/`开头。

什么是**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 Topic。

- Topic 格式和Topic 类格式一致。区别在于 Topic 类中的变量`${YourDeviceName}`，在 Topic 中则是具体的设备名称。

- 设备对应的 Topic 是从产品 Topic 类映射出来，根据设备名称而动态创建的。设备的具体 Topic 中带有设备名称（即`DeviceName`），只能被该设备用于 Pub/Sub 通信。例如，Topic：`/${YourProductKey}/device1/update`归属于设备名为**`device1`**的设备，所以只能被设备 **`device1`** 用于发布、订阅消息，而不能被设备 **`device2`** 用于发布订阅消息。
- 在配置规则引擎时，配置的 Topic 中可使用通配符，且同一个类目中只能出现一个通配符。

表 **2-1: Topic** 通配符

通配符	描述
#	这个通配符必须出现在 Topic 的最后一个类目，代表本级及下级所有类目。例如， Topic：`/YourProductKey/device1/#`，可以代表`/YourProductKey/device1/update`和`YourProductKey/device1/update/error`。
+	代表本级所有类目。例如，Topic：`/YourProductKey/+/update`，可以代表`/${YourProductKey}/device1/update`和`/${YourProductKey}/device2/update`。

2.6.2 Topic列表

物联网平台的Topic分为系统定义的Topic和自定义的Topic，您可以直接使用系统定义的Topic或参见自定义*Topic*添加更多自定义Topic类。本文主要列出了系统定义的Topic。

基础版默认Topic类

创建基础版产品之后，系统自动创建三个默认的用户Topic类。

- `/${YourProductKey}/${YourDeviceName}/update`：用于设备上报数据。设备操作权限：发布。
- `/${YourProductKey}/${YourDeviceName}/update/error`：用于设备上报错误。设备操作权限：发布。
- `/${YourProductKey}/${YourDeviceName}/get`：用于设备获取云端数据。设备操作权限：订阅。

高级版默认Topic类

高级版产品创建后，您可以使用 SDK 实现消息通信，或者使用Pub 或 Sub 相关的系统 Topic 实现设备数据的上下行通信。

高级版默认Topic类分为 Alink Topic 类和 透传 Topic 类。Alink Topic 类是设备采用 Alink JSON 进行消息通信时使用的 Topic 类。透传 Topic 类是设备采用透传/自定义格式进行消息通信时使用的Topic类。

Alink Topic 类：

- `/sys/${YourProductKey}/${YourDeviceName}/thing/event/property/post`：用于设备上报属性。设备操作权限：发布。
- `/sys/${YourProductKey}/${YourDeviceName}/thing/service/property/set`：用于设置设备属性。设备操作权限：订阅。
- `/sys/${YourProductKey}/${YourDeviceName}/thing/event/{tsl.event.identifer}/post`：用于设备上报事件。设备操作权限：发布。
- `/sys/${YourProductKey}/${YourDeviceName}/thing/service/{tsl.service.identifer}`：用于调用设备服务。设备操作权限：订阅。
- `/sys/${YourProductKey}/${YourDeviceName}/thing/deviceinfo/update`：用于设备上报标签。设备操作权限：发布。

透传 Topic 类：

- `/sys/${YourProductKey}/${YourDeviceName}/thing/model/up_raw`：设备透传数据上报，用于设备上报属性、事件和扩展信息的数据。设备操作权限：发布。
- `/sys/${YourProductKey}/${YourDeviceName}/thing/model/down_raw`：设备透传数据下行，用于获取设备属性、设置设备属性和调用设备服务。设备操作权限：订阅。

设备影子相关Topic类

设备影子提供系统 Topic 类，主要用于基础版产品的影子更新。

- `/shadow/update/${YourProductKey}/${YourDeviceName}`：用于更新设备影子。设备和应用程序操作权限：发布。
- `/shadow/get/${YourProductKey}/${YourDeviceName}`：用于获取设备影子。设备操作权限：订阅。

远程配置相关Topic类

远程配置提供系统Topic类，用于给设备下发配置文件。

- `/sys/${YourProductKey}/${YourDeviceName}/thing/config/push`：用于设备接收云端推送的配置信息。设备操作权限：订阅。

- `/sys/${YourProductKey}/${YourDeviceName}/thing/config/get`：用于设备主动请求更新配置。设备操作权限：发布。
- `/sys/${YourProductKey}/${YourDeviceName}/thing/config/get_reply`：用于设备主动请求更新配置，接收云端返回的配置信息。设备操作权限：订阅。

固件升级相关Topic类

固件升级提供系统Topic类，用于设备上报固件版本和接收升级通知。

- `/ota/device/inform/${YourProductKey}/${YourDeviceName}`：用于设备端上报固件版本给云端。设备操作权限：发布。
- `/ota/device/upgrade/${YourProductKey}/${YourDeviceName}`：用于设备端接收云端固件升级通知。设备操作权限：订阅。
- `/ota/device/progress/${YourProductKey}/${YourDeviceName}`：用于设备端上报固件升级进度。设备操作权限：发布。
- `/ota/device/request/${YourProductKey}/${YourDeviceName}`：设备端请求是否固件升级。设备操作权限：发布。

设备广播Topic类

设备广播提供系统Topic类，但是可以自定义广播设备范围。

`/broadcast/${YourProductKey}/+`：用于设备接收广播消息。设备操作权限：订阅。通配符（+）可自定义为广播的设备范围。

RRPC通信相关Topic类

IoT Hub基于开源协议MQTT封装了同步的通信模式，服务端下发指令给设备可以同步得到设备端的响应。

- `/sys/${YourProductKey}/${YourDeviceName}/rrpc/response/${messageId}`：用于设备响应RRPC请求。设备操作权限：发布。
- `/sys/${YourProductKey}/${YourDeviceName}/rrpc/request/+`：用于设备订阅RRPC请求。设备操作权限：订阅。

2.6.3 自定义Topic

本文介绍基础版产品和高级版产品如何自定义Topic。

1. 在产品列表页面，选择需要自定义Topic的产品，单击右侧的查看，进入产品详情页面。

2. 选择消息通信，单击定义**Topic**类，系统显示定义**Topic**类窗口。
3. 自定义Topic类。

- **Topic**类：根据界面上**Topic**规则设置Topic类。
- 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。

- **描述**：对自定义的Topic进行描述，可以为空。

4. 物联网平台也支持自定义带通配符的Topic类。如果您需要订阅通配符，则首先需要自定义带通配符的Topic，再订阅。

   创建Topic类时，参数设置如下：

   - **Topic类**：您可以创建带有通配符#或者+的Topic类。通配符说明可参见*表 1*。

     > **说明**：
     > 通配符#只能出现在Topic类的最后面。

   - **设备操作权限**：设备对该Topic的操作权限，只有选择订阅，才能支持填写通配符。
   - **描述**：对自定义的Topic进行描述，可以为空。

   已创建的带通配符的Topic，在设备的**Topic列表**页面不支持执行**发布消息**的操作。

   > **说明**：
   > 当前基础版产品的**设备详情**页有**Topic列表**，高级版产品的**设备详情**页，暂无**Topic列表**。

5. 单击**确认**，完成自定义Topic类。

2.7 标签

物联网平台的标签是您给产品或设备自定义的标识。您可以使用标签功能来灵活管理产品与设备。

物联网往往涉及量级产品与设备的管理。如何区分不同批次的产品与设备，如何实现批量管理，成为一大挑战。阿里云物联网平台为解决这一问题提供了标签功能。您可以为不同产品或设备贴上不同标签，然后根据标签实现分类统一管理。

标签包括产品标签与设备标签。产品标签是设备标签的母版。创建产品标签后，该产品下新增的设备将自动继承产品标签。您也可以单独为单个设备添加设备标签。标签的结构为`Key:Value`。

本文将详细讲解产品标签与设备标签的创建。

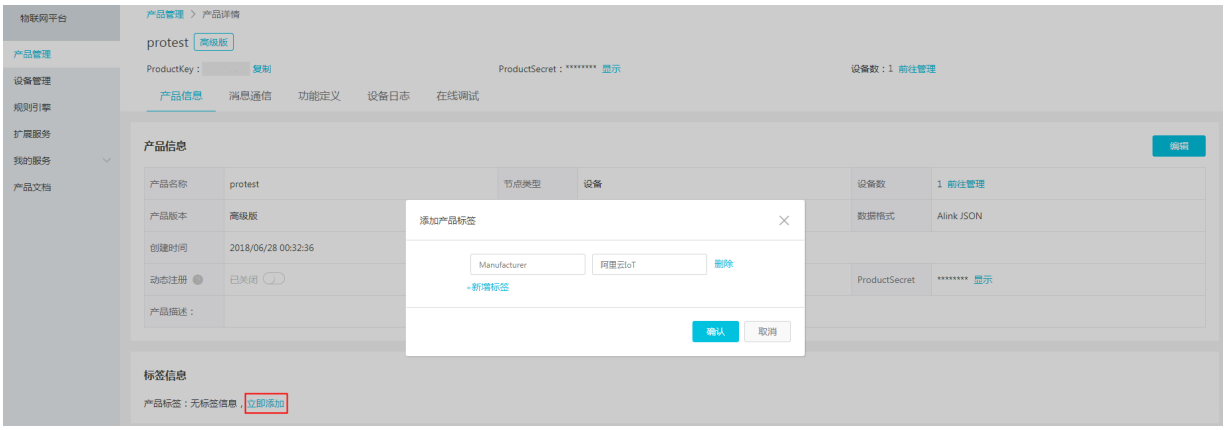
产品标签

产品标签通常描述的是对一个产品下所有设备所具有的共性信息。如产品的制造商、所属单位、外观尺寸、操作系统等。需在创建产品后，再为该产品添加产品标签。

添加产品标签操作步骤：

1. 登录物联网平台控制台。

2. 在产品管理页面，单击要添加标签的产品所对应的查看操作按钮，进入产品详情页面。
3. 在标签信息部分，单击立即添加按钮。
4. 在弹出的对话框中，输入标签的 标签Key 和 标签Value，然后单击确认。



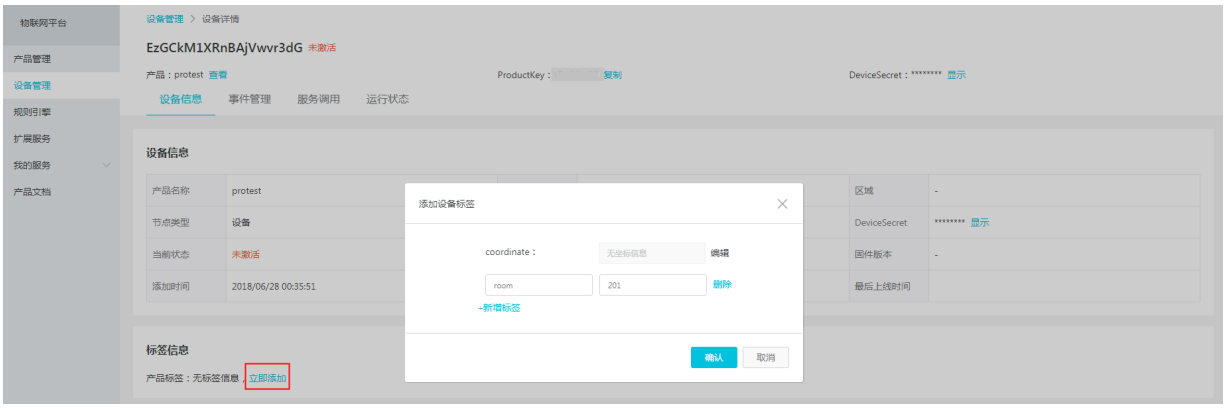
产品标签创建后，该产品下的新增设备将自动继承产品的标签。

设备标签

在产品下添加设备后，设备将自动继承所属产品的标签。但是，从产品中继承的标签一般只定义了产品下所有设备的共性。您可以根据设备的特性为设备添加特有的标签，方便对设备进行管理。例如，为房间 201 的智能电表定义一个标签为`room:201`。您可以在控制台管理设备标签，也可以通过 API 管理设备标签。

在控制台添加设备标签的操作步骤：

1. 登录物联网平台控制台。
2. 单击设备管理。
3. 在设备管理页面，单击要添加标签的设备所对应的查看操作按钮，进入设备详情页面。
4. 在标签信息部分，单击立即添加按钮。
5. 在弹出的对话框中，输入标签的 标签Key 和 标签Value，然后单击确认。



设备标签信息会跟随设备在系统内部流转。并且，物联网平台可以基于规则引擎，将设备标签发给阿里云其他云服务。

2.8 网关与子设备

2.8.1 网关与子设备

物联网平台支持设备直连，也支持设备挂载在网关上，作为网关的子设备，由网关直连。

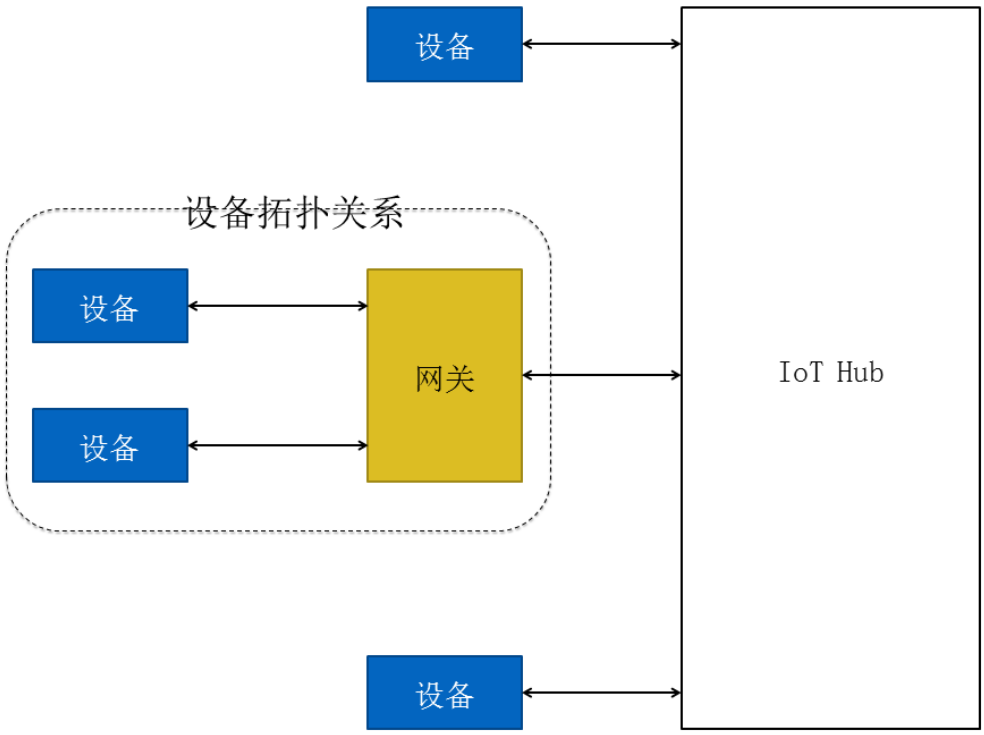
网关与设备

创建产品与设备时，需要选择节点类型。平台目前支持两种节点类型：设备和网关。

- 设备：指不能挂载子设备的设备。设备可以直连IoT Hub，也可以作为网关的子设备，由网关代理连接IoT Hub。
- 网关：指可以挂载子设备的直连设备。网关可以管理子设备、可以维持与子设备的拓扑关系，并将该拓扑关系同步到云端。

网关与子设备的拓扑关系

图 2-2: 设备拓扑关系



产品与设备创建完成后，您可以：

- 将网关连接上云，由网关将拓扑关系同步至云端，代替子设备完成设备认证、消息上传、指令接收等与平台的通信。而子设备将由网关统一管理。具体请参考设备开发指南和子设备接入。
- 在控制台配置子设备通道，管理拓扑关系，并将配置下发至设备端。具体请参考子设备通道管理和子设备管理。

2.8.2 子设备通道管理

当您使用高级版产品时，可以为网关设备添加子设备通道，添加完成后，该网关设备可以使用这些通道管理子设备。当前支持添加Modbus协议、OPC UA协议和自定义协议三种通道。

1. 在设备管理页面，找到网关设备，单击查看，进入设备详情页。
2. 单击子设备通道管理，并添加不同协议的管理通道。

- Modbus

  在**Modbus**页面，单击添加**Modbus**通道，并根据页面提示，输入参数。

参数	描述
通道名称	网关下需要唯一
传输模式	支持RTU和TCP两种

参数	描述
当传输模式为RTU时，需设置以下参数：	
选择串口	如/dev/tty0、/dev/tty1
波特率	从下拉列表中选择
数据位	支持5、6、7、8四种
校验位	支持无校验、奇校验、偶校验三种
停止位	支持1、1.5、2三种
当传输模式为TCP时，需设置以下参数：	
IP地址	输入点分十进制格式的地址
端口号	输入0~65535范围的整数

- OPC UA

  单击**OPC UA > 添加OPC UA通道**，并根据页面提示，输入参数。

参数	描述
通道名称	网关下需要唯一
连接地址	如opc.tcp://localhost:4840
用户名	非必填
密码	非必填
方法调用超时时间	单位为秒

- 自定义

  1. 单击**自定义 > 添加自定义通道**。
  2. 在弹出界面上，设置通道名称。
  3. 添加自定义配置。

  > **说明：**
  > 自定义配置仅支持JSON格式。建议您在本地写好后，粘贴进来。

2.8.3 子设备管理

您可以在网关下面关联子设备，并将配置下发至子设备。

前提条件

- 若子设备接入网关协议为Modbus或OPC UA。子设备接入网关前，该网关下应配置好对应的子设备通道，具体请参考子设备通道管理文档。
- 2018年9月4日之前创建的产品及设备，也可以作为子设备添加进来，仅支持拓扑关系添加，不支持关联子设备通道和自定义配置。

操作步骤

1. 在**设备管理**页面，找到网关设备，单击**查看**，进入**设备详情**页。
2. 单击**子设备管理 > 添加子设备**。


物联网平台
产品管理
设备管理
边缘管理
规则引擎
扩展服务
我的服务
产品文档
设备管理 > 设备详情
产品：
查看
设备信息
Topic列表
子设备管理(0)
显示设备拓扑关
请输入DeviceName
DeviceName
批量删除
批量禁用

3. 在弹出页面上，设置要关联的子设备相关信息。

参数	描述
产品	选择子设备对应的产品名称。
设备	选择子设备对应的设备名称。
如果该设备使用Modbus协议	
关联通道	必选，从网关下的子设备管理通道中，选择该子设备使用的关联通道。
从站号	输入1~247之间的整数值。
如果该设备使用OPC UA协议	
关联通道	必选，从网关下的子设备管理通道中，选择该子设备使用的关联通道。
节点路径	如Objects/Device1，Objects是固定根节点，后面是到设备节点路径上的所有节点名称，以/分隔。
如果该设备使用自定义协议	
关联通道	非必选，从网关下的子设备管理通道中，选择该子设备使用的关联通道。
自定义配置	若选择了关联通道，需自定义配置。只支持JSON格式。

4. 子设备信息配置完成后，可回到网关设备详情页，单击**配置下发**，将子设备所属产品的物模型及扩展配置下发至设备端。

5. 子设备信息配置完成后，可在对应子设备详情页，查看相应信息。此处可单击**编辑**，修改相关信息。

后续操作

- 您可以使用*Alink*协议自行开发设备，并参考网关配置下发章节，将此处网关与子设备的配置下发至设备端。
- 若您使用了边缘计算节点，可参考驱动概述进行配置。

2.9 服务端订阅

2.9.1 什么是服务端订阅

服务端可以直接订阅产品下配置的所有类型的消息。

目前，新版物联网平台通过HTTP/2通道进行消息流转。配置HTTP/2服务端订阅后，物联网平台会将消息通过HTTP/2通道推送至服务端。通过接入HTTP/2 SDK，企业服务器可以直接从物联网平台接收消息。HTTP/2 SDK提供身份认证、Topic订阅、消息发送和消息接收能力，并支持设备接入和

云端接入能力。HTTP/2 SDK适用于物联网平台与企业服务器之间的大量消息流转，也支持设备与物联网平台之间的消息收发。

说明：

旧版物联网平台用户使用阿里云消息服务（MNS）进行消息流转，您可以将服务端订阅方式升级为HTTP/2方式。如果您继续使用在MNS这种方式，物联网平台将设备消息推送至MNS，服务端应用通过监听MNS队列接收设备消息。

2.9.2 开发指南

本文档介绍如何配置服务端订阅、接入HTTP/2 SDK、进行身份认证和设置消息接收接口。

以下简单介绍服务端订阅的开发流程。有关配置的具体信息，请参考服务端接收消息 *SDK demo*。

配置服务端订阅

1. 登录物联网平台控制台。
2. 单击左侧导航栏产品管理。
3. 在产品列表中，搜索到要配置服务端订阅的产品，并单击该产品对应的查看按钮，进入产品详情页。
4. 单击服务端订阅 > 设置。
5. 选择推送的消息类型：设备上报消息或设备状态变化通知。

- 设备上报消息：指产品下所有设备 Topic 列表中，具有发布权限的 Topic 中的消息。勾选后，可以通过 HTTP/2 SDK 接收这些消息。

  例如，有一个高级版产品，有3个Topic类，分别是：

  - `/${YourProductKey}/${YourDeviceName}/user/get` ，具有订阅权限。
  - `/${YourProductKey}/${YourDeviceName}/user/update`，具有发布权限。
  - `/sys/${YourProductKey}/${YourDeviceName}/thing/event/property/post`，具有发布权限。

  那么，服务端订阅会推送具有发布权限的Topic类中的消息，即`/${YourProductKey}/${YourDeviceName}/user/update`和`/sys/${YourProductKey}/${YourDeviceName}/thing/event/property/post`中的消息。其中，`/sys/${YourProductKey}/${YourDeviceName}/thing/event/property/post`中的数据已经过系统处理。
- 设备状态变化通知：指一旦该产品下的设备状态变化时通知的消息，例如设备上线、下线的消息。设备状态消息的发送 Topic 为 `/as/mqtt/status/${YourProductKey}/${YourDeviceName}`。勾选后，可以通过 HTTP/2 SDK 接收设备状态变化的通知消息。

接入 SDK

在工程中添加 maven 依赖接入 SDK。

```
<dependency>
    <groupId>com.aliyun.openservices</groupId>
    <artifactId>iot-client-message</artifactId>
    <version>1.1.3</version>
</dependency>

<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>3.7.1</version>
```

```
</dependency>
```

身份认证

使用服务端订阅功能，需要基于您的阿里云 AccessKey 进行身份认证并建立连接。

建立链接示例如下：

```
// 阿里云accessKey
        String accessKey = "xxxxxxxxxxxxxx";
        // 阿里云accessSecret
        String accessSecret = "xxxxxxxxxxxxxxx";
        // regionId
        String regionId = "cn-shanghai";
        // 阿里云uid
        String uid = "xxxxxxxxxxxx";
        // endPoint:  https://${uid}.iot-as-http2.${region}.aliyuncs.
com
        String endPoint = "https://" + uid + ".iot-as-http2." +
regionId + ".aliyuncs.com";

        // 连接配置
        Profile profile = Profile.getAccessKeyProfile(endPoint,
regionId, accessKey, accessSecret);

        // 构造客户端
        MessageClient client = MessageClientFactory.messageClient(
profile);

        // 数据接收
        client.connect(messageToken -> {
            Message m = messageToken.getMessage();
            System.out.println("receive message from " + m);
            return MessageCallback.Action.CommitSuccess;
        });
```

accessKey 即您的账号的 AccessKey ID， **accessSecret** 即 AccessKey ID对应的 AccessKey Secret。请登录阿里云控制台，将光标移至您的账号头像上，在选项框中选择 **accesskeys** 查看您的 AccessKey ID 和 AccessKey Secret；选择安全设置查看您的账号 ID。

regionId为您的物联网平台服务地域。

设置消息接收接口

连接建立后，服务端会立即向 SDK 推送已订阅的消息。因此，建立链接时，需要提供消息接收接口，用于处理未设置回调的消息。建议在`connect`之前，调用 **setMessageListener** 设置消息回调。

您需要通过 **MessageCallback** 接口的`consume`方法，和调用 `messageClient` 的`setMessageListener()`方法来设置消息接收接口。

`consume` 方法的返回值决定 SDK 是否发送 ACK。

设置消息接收接口的方法如下：

```
MessageCallback messageCallback = new MessageCallback() {
    @Override
    public Action consume(MessageToken messageToken) {
        Message m = messageToken.getMessage();
        log.info("receive : " + new String(messageToken.getMessage().
getPayload()));
        return true;
    }
};
messageClient.setMessageListener("/${YourProductKey}/#",messageCal
lback);
```

其中，

- 参数 **`MessageToken`** 指消息回执的消息体。通过`MessageToken.getMessage()`可获取消息体。**`MessageToken`**可以用于手动回复 ACK 。

  消息体包含的内容如下：

  ```
  public class Message {
      // 消息体
      private byte[] payload;
      // Topic
      private String topic;
      // 消息ID
      private String messageId;
      // QoS
      private int qos;
  }
  ```

- 具体请参考消息体格式文档。
- `messageClient.setMessageListener("/${YourProductKey}/#",messageCal lback);`用于设置回调。本示例中，设置为指定 Topic 回调。

  您可以设置为指定 Topic 回调，也可以设置为通用回调。

  - 指定 Topic 回调

    指定 Topic 回调的优先级高于通用回调。一条消息匹配到多个 Topic 时，按字典顺序优先调用，并且仅回调一次。

    设置回调时，可以指定带通配符的 Topic，如 `/${YourProductKey}/${YourDevice Name}/#`。

示例：

```
messageClient.setMessageListener("/alEddfaXXXX/device1/#",
messageCallback);
//当收到消息的Topic，如"/alEddfaXXXX/device1/update"，匹配指定Topic
时，会优先调用该回调
```

- 通用回调

  未指定 Topic 回调的消息，则调用通用回调。

  设置通用回调方法：

```
messageClient.setMessageListener(messageCallback);
//当收到消息topic未匹配到已定指的Topic 回调时，调用该回调
```

- 设置回复 ACK。

  对于 QOS>0 的消息，消费后需要回复 ACK。SDK 支持自动回复 ACK 和手动回复 ACK。默认为自动回复 ACK。本示例中未设置回复 ACK，则默认为自动回复。

  - 自动回复 ACK：设置为自动回复 ACK 后，若 `MessageCallback.consume` 的返回值为 `true` 则 SDK 会发送 ACK；返回 `false` 或抛出异常，则不会返回 ACK。对于 QOS>0 且未回复 ACK 的消息，服务端会重新发送。
  - 手动回复 ACK：通过`MessageClient.setManualAcks`设置手动回复 ACK。

    设置为手动回复 ACK 后，需要调用`MessageClient.ack()`方法回复 ACK，参数为 **`MessageToken`**。**`MessageToken`** 参数值可在接收消息中获取。

    手动回复 ACK 的方法：

```
messageClient.ack(messageToken);
```

2.9.3 使用限制

当您使用服务端订阅时，请注意以下限制。

限制	描述
JDK版本	仅支持JDK8。
认证超时	连接建立之后，需要立刻发送认证请求。如果15秒内没有认证成功，服务器将主动关闭连接。
数据超时	连接建立之后，客户端需要定期发送PING包来维持连接。如果超过60秒没有收到来自客户端的PING包或数据，服务器将主动关闭连接。

限制	描述
推送超时	推送失败重试消息时，每次批量推送10条。若该批次消息在10秒后，仍未收到客户端回复的ACK，则认为推送超时。
失败推送重试策略	每60秒重新推送一次因客户端离线、消息消费慢等原因导致的堆积消息。
消息保存时长	QoS0的消息保存1天，QoS1的消息保存7天。
连接个数	每个阿里云账号最多建立20个连接。

2.10 设备分组

物联网平台提供设备分组功能。您可以通过设备分组来进行跨产品管理设备。本章节介绍如何在物联网平台控制台创建设备分组和管理分组。

操作步骤

1. 登录物联网平台控制台。
2. 单击设备管理 > 分组进入分组管理页面。
3. 单击新建分组，设置分组参数，并单击保存。

> 说明：
> 一个阿里云账号下最多可创建1,000个分组，包括分组和子分组。

参数信息解释如下：

- 父组：选择创建的分组类型。
  - 分组：创建的分组是一个父组。

— 选择指定父组：以指定的分组为父组，创建子分组。

- **分组名称**：给该分组创建名称。分组名称支持中文、英文字母、数字和下划线，长度限制4~30。分组名称必须为账号下唯一，且创建后不能修改。
- **分组描述**：输入文字，描述该分组。可为空。

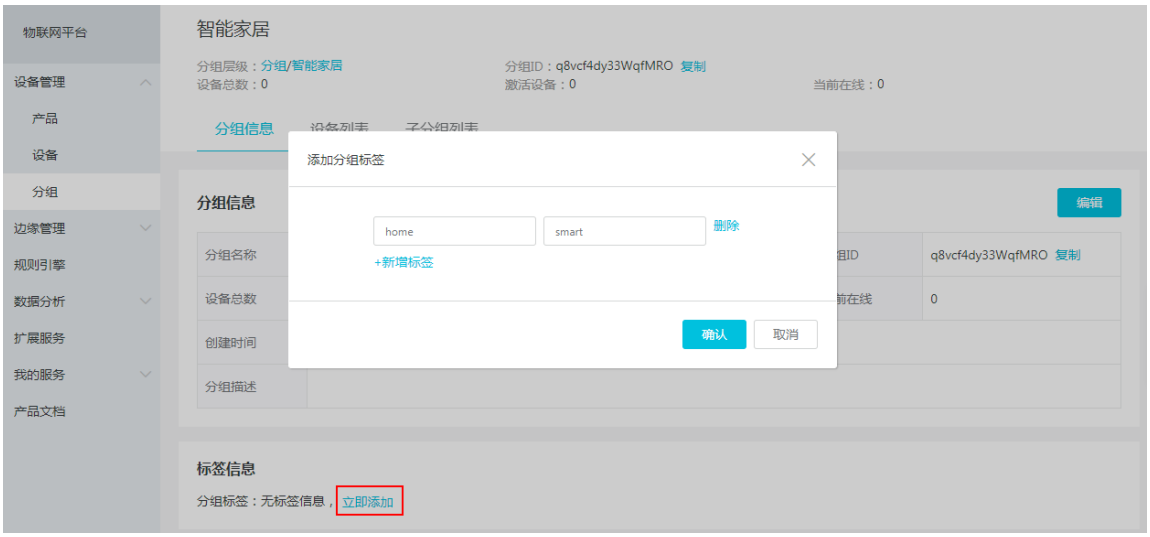
4. 在**分组管理**页面，单击已创建分组对应的**查看**操作按钮，进入**分组详情**页面。
5. （可选）为分组添加标签，即自定义分组标识，以便灵活管理分组。

a) 单击标签信息栏下的**立即添加** ，并输入标签的key和value。

b) 单击**确认**添加标签。

说明：

一个分组最多可添加100个标签。



6. 单击**设备列表 > 添加设备到分组**。勾选设备，将选中设备添加至指定分组。

说明：

- 单次最多添加200个设备。单个分组最多添加20,000个设备。
- 一个设备最多可以被添加到10个分组中。

添加设备到分组页面右上方的全部和已选择按钮说明：

- 单击全部，显示所有设备列表。
- 单击已选择，显示您已勾选的设备列表。

7. （可选）单击子分组列表 > 新建分组，为分组添加子分组。

子分组功能用于细化设备管理。例如，您可以在“智能家居”分组下，创建“智能厨房”、“智能卧室”等子分组，实现厨房设备和卧室设备的分开管理。具体操作如下：

a) 选择该子分组的父组，输入子分组名称和描述，然后单击保存。

b) 在子分组列表页面，单击子分组对应的查看操作按钮，进入该子分组的分组详情页。

c) 单击设备列表 > 添加设备到分组，然后为该子分组添加设备。

创建子分组和添加子分组设备完成后，您可以对该子分组和其设备进行管理。您还可以在子分组下再创建子分组。

说明：

- 一个分组最多可包含100个子分组。
- 分组只支持三级嵌套，即分组>子分组>子子分组。
- 一个子分组只能隶属于一个父组。
- 分组的嵌套关系创建后不能修改，只能删除后重新创建。
- 分组下有子分组时，不能直接删除分组。需子分组全部删除后，才能删除父组。

2.11 高级版

高级版除了具有基础版的所有功能外，还具有更多设备管理功能，进一步降低机器智能化周期和成本投入，让开发者更聚焦于搭建垂直业务系统。

高级版特点：

- 具有基础版的所有功能。
- 具备更加完整的设备全生命周期管理能力。
- 支持设备模型定义、数据解析、在线调试和远程维护。
- 提供原始数据存储。
- 支持实时获取设备运行状态或历史数据。

2.11.1 基础版

物联网平台基础版适用于具有较强软硬件开发实例，并希望灵活组合阿里云各种云产品来搭建业务系统的开发者。

基础版特点：

- 提供设备与云端稳定的双向通信能力。
- 提供规则引擎，处理数据，连通阿里云产品。
- 提供设备认证、传输加密、多重防护，保障设备安全。

2.11.1.1 创建设备

设备需归属于某个产品下。设备可以直接连接物联网平台，也可以作为子设备通过网关连接物联网平台。

背景信息

本文档介绍如何创建基础版设备。

操作步骤

1. 登录物联网控制台。
2. 单击左侧导航栏中设备管理。
3. 在设备管理页面上，选择已创建的基础版产品，再单击添加设备。

4. 在弹出对话框中，输入设备名称，单击确定。

> 说明：
> 设备名称（即 DeviceName）需在产品内具有唯一性。可作为设备的唯一标识符，用于与 IoT Hub 进行通信。

5. 在添加成功页面，单击一键复制复制设备的 ProductKey、DeviceName、和 DeviceSecret，然后粘贴到文件中，并妥善保管。

   ProductKey、DeviceName、 和 DeviceSecret 即设备三元组：

   - ProductKey：物联网平台为您创建的产品颁发的全局唯一标识符。
   - DeviceName：自定义的设备唯一标识符。用于设备认证和通信。
   - DeviceSecret：物联网平台为设备颁发的设备秘钥，用于认证加密，需与DeiceName（或者deviceId）成对使用。

预期结果

设备创建成功，物联网平台会根据设备所属产品的消息通信中的 Topic 类，自动为设备生成对应的 Topic。设备可以基于 Topic 中定义的操作权限进行 Pub/Sub 通信。

3 规则引擎

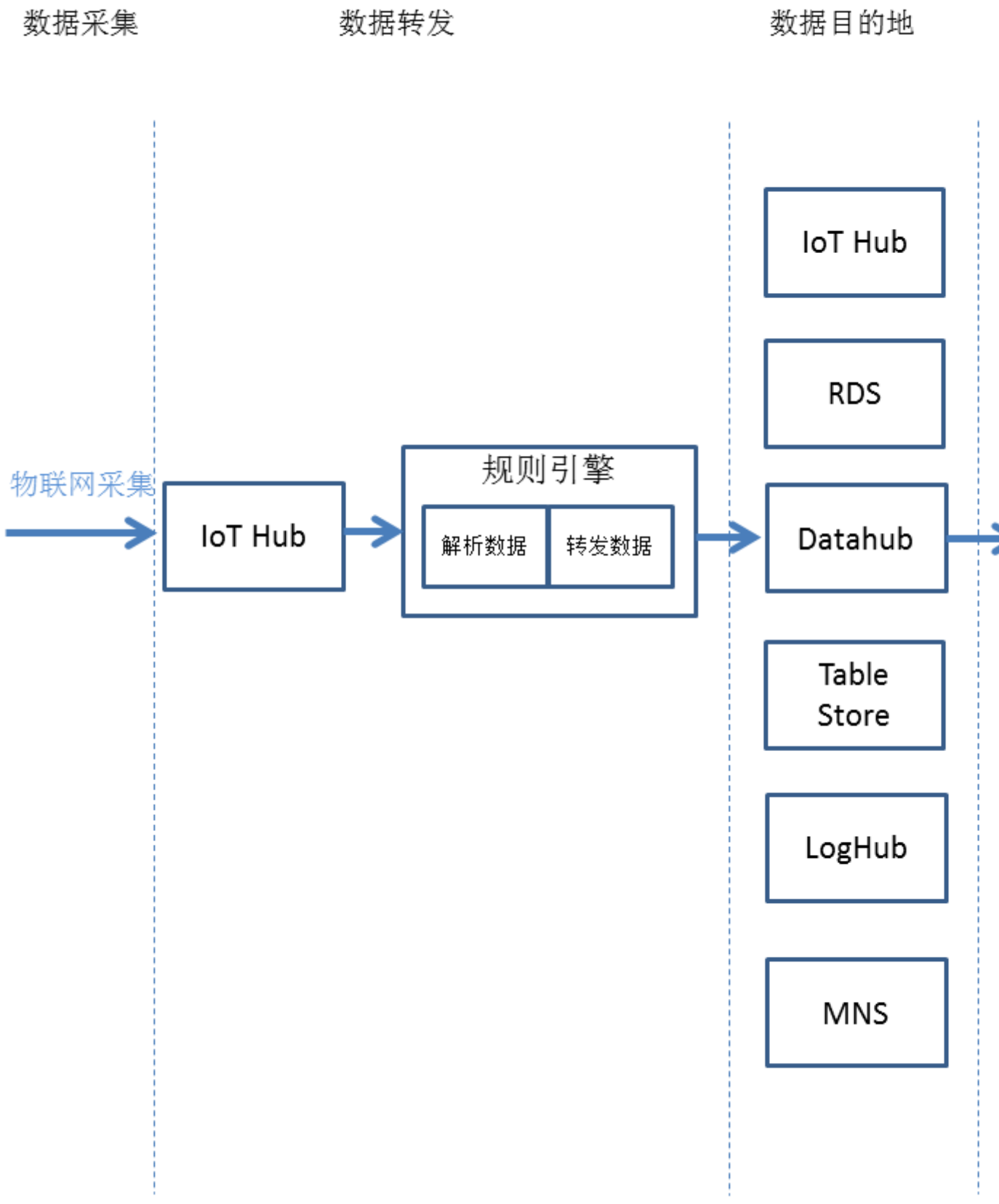
3.1 概览

什么是规则引擎

当设备基于*Topic*进行通信时，您可以使用规则引擎，编写SQL对Topic中的数据进行处理，并配置转发规则将处理后的数据转发到阿里云其他服务。例如：

- 可以转发到*RDS*、表格存储、中进行存储。
- 可以转发到函数计算进行事件计算。
- 可以转发到另一个Topic中实现M2M通信。
- 可以转发到队列*MQ*实现高可靠消费数据·。

使用规则引擎后，您无需购买服务器部署分布式架构，即可实现采集 + 计算 + 存储的全栈服务。


数据采集
数据转发
数据目的地
物联网采集
IoT Hub
规则引擎
解析数据
转发数据
IoT Hub
RDS
Datahub
Table Store
LogHub
MNS

使用路径

- 基础版产品
  1. 设置规则引擎：如何设置一条转发规则。
  2. *SQL*表达式：规则中SQL表达式的写法详解。
  3. 函数列表：规则中SQL表达式支持的函数列表。
  4. 数据流转：使用规则引擎过程中的数据流转格式。
  5. 地域和可用区：确认支持的目的云产品及地域信息。
  6. 使用实例：不同云产品目的地的转发设置详解。
- 高级版产品
  1. 设置规则引擎：如何设置一条转发规则。
  2. *SQL*表达式：规则中SQL表达式的写法详解。
  3. 函数列表：规则中SQL表达式支持的函数列表。
  4. 数据流转：使用规则引擎过程中的数据流转格式。
  5. 数据格式*(高级版)*：高级版产品Topic中的数据格式。
  6. 地域和可用区：确认支持的目的云产品及地域信息。
  7. 使用实例：不同云产品目的地的转发设置详解。

使用限制

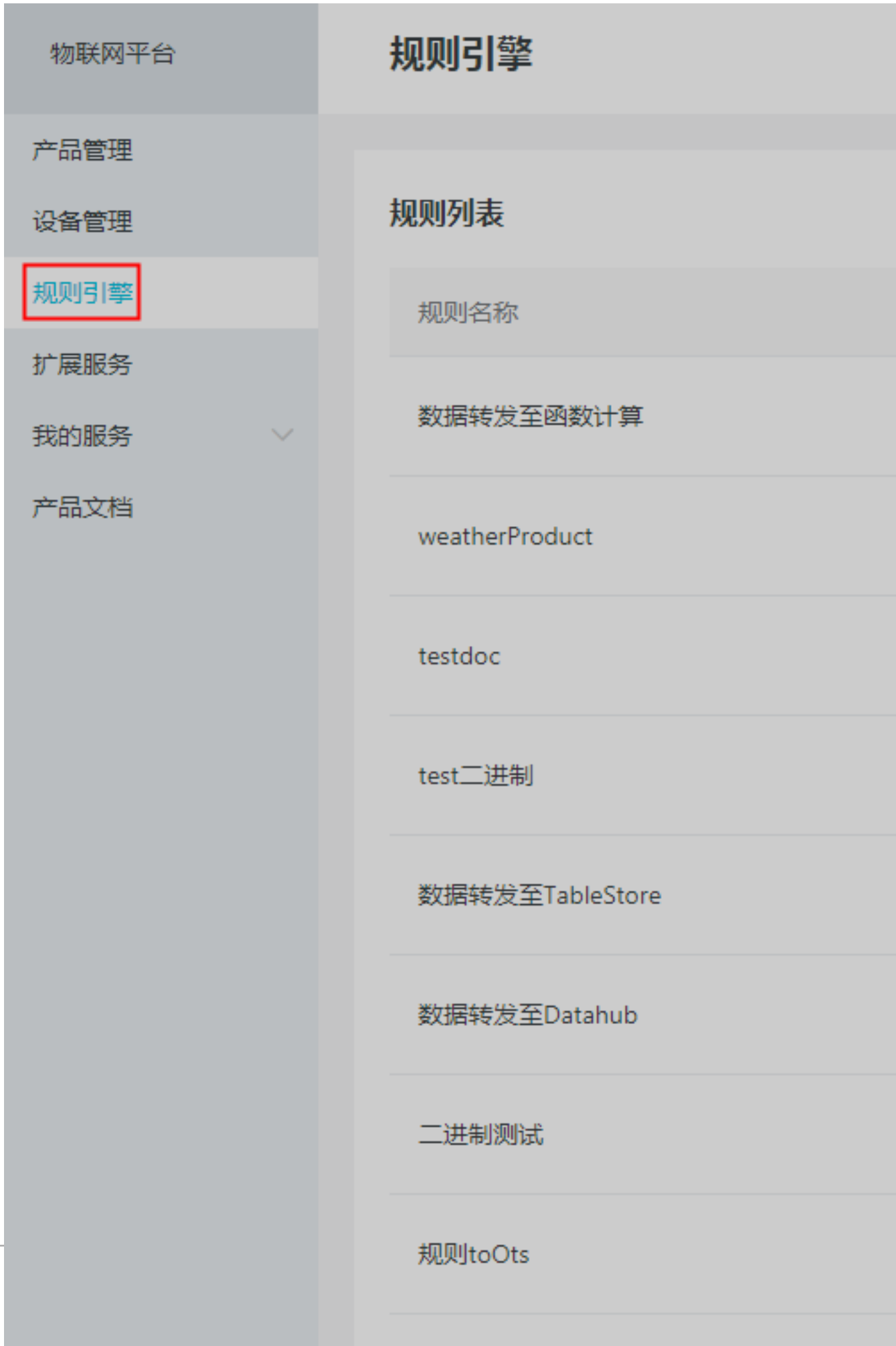
- 规则引擎基于Topic对数据进行处理。只有通过Topic进行通信时，才能使用规则引擎。
- 规则引擎通过SQL对Topic中的数据进行处理。
- SQL语法目前不支持子查询。
- 支持部分函数，比如`deviceName()`获取当前设备名称，具体请参考函数列表。

3.2 设置规则引擎

本文将为您详细讲解如何设置一条完整的规则。

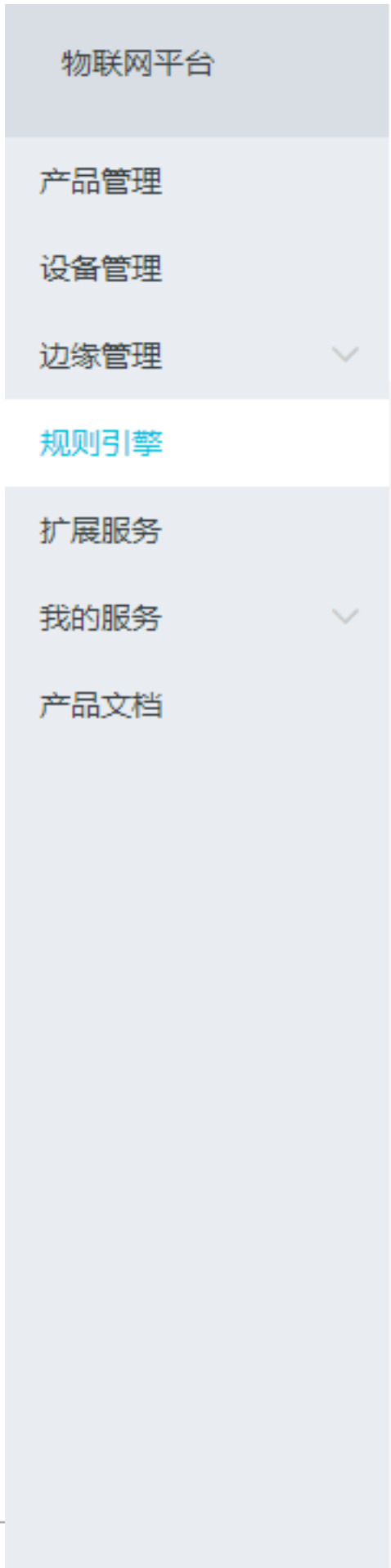
操作步骤

1. 单击规则引擎，单击创建规则，创建一条新的规则。
2. 填写规则名称，选择数据格式。


物联网平台
产品管理
设备管理
规则引擎
扩展服务
我的服务
产品文档
规则引擎
规则列表
规则名称
数据转发至函数计算
weatherProduct
testdoc
test二进制
数据转发至TableStore
数据转发至Datahub
二进制测试
规则toOts

- 规则名称：输入规则名称，用以区别各条规则。
- 数据格式：支持JSON和二进制。因规则引擎基于Topic处理数据，此处的数据格式需与被处理Topic中的数据格式保持一致。

3. 找到刚创建的规则，单击**管理**，进入**规则详情**页，设置规则具体信息。



规则引擎 > 规则详情

数据转发至函数计算

数据格式：JSON

产品描述：

处理数据

转发数据

数据目的地

a) 单击**编写SQL**，编写SQL，设置数据处理的具体规则。

> **说明：**
> 二进制数据可使用`to_base64(*)`将原始数据转换成base64String，同时支持内置函数和条件筛选。

例如：以下SQL可以将deviceName从Basic_Light_001产品下，后缀为data的自定义Topic类中取出。

- **规则查询语句**：此处根据**字段**、**Topic**和**条件**，系统自动补充完整规则查询语句。
- **字段**：指定消息内容字段，此示例填`deviceName() as deviceName`。
- **Topic**：选择需要处理的消息Topic。

- 自定义：选择自定义后，表明该Topic是您自定义的产品Topic。此时您需要在选择产品后，补充完整该Topic。
- sys：选择sys后，表明该Topic是系统定义的产品Topic。此时您需要在选择产品后，还需要选择设备，选择系统定义的某个Topic。

此示例选Basic_Light_001产品自定义的后缀为data的Topic类。

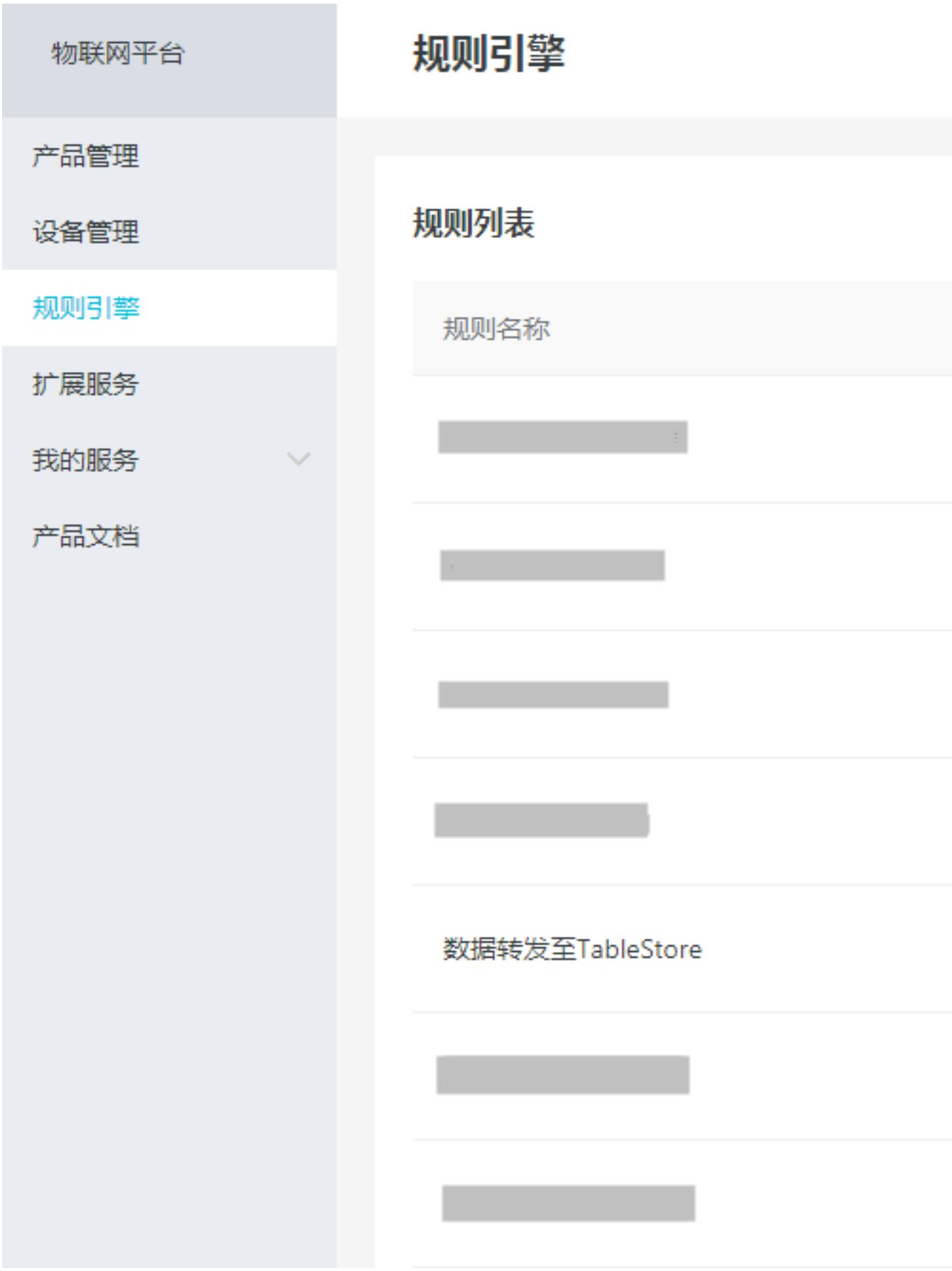
- **条件**：规则触发条件。

具体编写方法请参考*SQL*表达式和函数列表。

b) 单击**转发数据**一栏的**添加操作**，将数据转发至目的云产品。

数据转发的具体实例，请参考使用实例。

4. 返回至**规则引擎**页。单击**启动**，数据即可按照规则进行转发。


物联网平台
规则引擎
产品管理
设备管理
规则引擎
扩展服务
我的服务
产品文档
规则列表
规则名称
数据转发至TableStore

您也可以：

- 单击管理修改规则具体设置。
- 单击删除删除对应规则。其中，启动中的规则不可删除。
- 单击停止停止某条规则转发数据。

3.3 SQL表达式

使用规则引擎时，若您的数据为JSON格式，可以编写SQL来解析和处理数据。规则引擎对二进制格式的数据不做解析，直接透传。本文主要讲解SQL表达式。

SQL表达式

JSON数据可以映射为虚拟的表，其中Key对应表的列，Value对应列值，这样就可以使用SQL处理。为便于理解，我们将规则引擎的一条规则抽象为一条SQL表达（类试MySQL语法）：

例如某环境传感器用于火灾预警，可以采集温度、湿度及气压数据，上报数据内容如下：

```
{
"temperature":25.1
"humidity":65
"pressure":101.5
"location":"xxx,xxx"
}
```

假定温度大于38，湿度小于40时，需要触发报警，可以编写如下的SQL语句：

```
SELECT temperature as t, deviceName() as deviceName, location FROM /ProductA/+/update WHERE temperature > 38 and humidity < 40
```

当上报的数据中，温度大于38且湿度小于40时，会触发该规则，并且解析数据中的温度、设备名称、位置，用于进一步处理。

FROM

FROM 需要填写Topic通配符，用于匹配需要处理的消息Topic。当有符合Topic规则的消息到达时，消息的payload数据以JSON格式解析，并根据SQL语句进行处理（如果消息格式不合法，将忽略此消息）。您可以使用`topic()`函数引用具体的Topic值。

上文例子中，"FROM /ProductA/+/update"语句表示该SQL仅处理符合/ProductA/+/update格式的消息，具体匹配参考 *Topic*。

SELECT

- JSON数据格式

  SELECT语句中的字段，可以使用上报消息的payload解析结果，即JSON中的键值，也可以使用SQL内置的函数，比如`deviceName()`。不支持子SQL查询。

  上报的JSON数据格式，可以是数组或者嵌套的JSON，SQL语句支持使用JSONPath获取其中的属性值，如对于`{a:{key1:v1, key2:v2}}`，可以通过`a.key2` 获取到值v2。使用变量时，需要注意单双引号区别：单引号表示常量，双引号或不加引号表示变量。如使用单引号`'a.key2'`，值为a.key2。

  内置的SQL函数可以参考函数列表。

  例如上文，"SELECT temperature as t, deviceName() as deviceName, location"语句，其中temperature和loaction来自于上报数据中的字段，deviceName()则使用了内置的SQL函数。

- 二进制数据格式

  - 可填`*`直接透传数据。
  - 可使用`to_base64(*)`函数，将原始Payload二进制数据转成base64String，提取出来。同时支持使用内置的函数和条件，如`deviceName()`，提取所需信息。

说明：

SELECT语句中的字段最多支持50个。

WHERE

- JSON数据格式

规则触发条件，条件表达式。不支持子SQL查询。WHERE中可以使用的字段和SELECT语句一致，当接收到对应Topic的消息时，WHERE语句的结果会作为是否触发规则的判断条件。具体条件表达式列表见下方表格。

上文例子中， `"WHERE temperature > 38 and humidity < 40"` 表示温度大于38且湿度小于40时，才会触发该规则，执行配置。

- 二进制数据格式

  目前二进制格式WHERE语句中仅支持内置函数及条件表达式，无法使用payload中的字段。

SQL结果

SQL语句执行完成后，会得到对应的SQL结果，用于下一步转发处理。如果payload数据解析过程中出错会导致规则运行失败。 转发数据动作中的表达式需要使用 `${表达式}` 引用对应的值。

对于上文例子，配置转发动作时，可以`${t}`、`${deviceName}`和`${loaction}`获取SQL解析结果，如果要将数据存储到`TableStore`，配置中可以使用`${t}`、`${deviceName}`和`${loaction}`。

数组使用说明

数组表达式需要使用双引号，比如设备消息为：`{a:[{v:1},{v:2},{v:3}]}`，那么SQL语句中的SELECT写法为：`select "$.a[0]" data1,".a[1].v" data2,".a[2]" data3`，则`data1={v:1}`，`data2=2`，`data3=[{v:3}]`。

条件表达式支持列表

操作符	描述	举例
=	相等	color = ‘red’
<>	不等于	color <> ‘red’
AND	逻辑与	color = ‘red’ AND siren = ‘on’
OR	逻辑或	color = ‘red’ OR siren = ‘on’
( )	括号代表一个整体	color = ‘red’ AND (siren = ‘on’ OR isTest)
+	算术加法	4 + 5
-	算术减	5 - 4
/	除	20 / 4
*	乘	5 * 4

%	取余数	20 % 6
<	小于	5 < 6
<=	小于或等于	5 <= 6
>	大于	6 > 5
>=	大于或等于	6 >= 5
函数调用	支持函数，详细列表请参考函数列表。	deviceId()
JSON属性表达式	可以从消息payload以JSON表达式提取属性。	state.desired.color,a.b.c[0].d
CASE … WHEN … THEN … ELSE … END	Case 表达式	CASE col WHEN 1 THEN ‘Y’ WHEN 0 THEN ‘N’ ELSE ‘’ END as flag
IN	仅支持枚举，不支持子查询。	比如， where a in(1,2,3)。不支持以下形式： where a in(select xxx)
like	匹配某个字符， 仅支持%通配符，代表匹配任意字符串。	比如， where c1 like ‘%abc’, where c1 not like ‘%def%’

3.4 函数列表

规则引擎提供多种函数，您可以在编写SQL时使用它们，实现多样化数据处理。

使用方法

您可以在SQL语句中使用函数获取数据或者对数据做处理。

例如：下面代码中用到了deviceName()，abs(number)，topic(number)三个函数。

```
SELECT case flag when 1 then '开灯' when 2 then '关灯' else '' end flag
 ,deviceName(),abs(temperature) tmr FROM "/topic/#" WHERE temperature>
10 and topic(2)='123'
```

说明：

使用时请注意，通常单引号代表常量，不带引号或双引号代表变量。如select “a” a1, ‘a’ a2, a a3中，a1与a3等效，a2代表常量a。

函数名	函数说明
abs(number)	返回绝对值。

asin(number)	返回 number的反正弦。
attribute(key)	返回key所对应的设备标签。调用依赖设备，使用SQL调试时返回为空。
concat(string1, string2)	字符串连接 示例：concat(field,’a’)
cos(number)	返回number的余弦。
cosh(number)	返回number的双曲余弦（hyperbolic cosine）。
crypto(field,String)	对field的值进行加密。 第二个参数String为算法字符串。可选：MD2，MD5，SHA1，SHA-256，SHA-384，SHA-512。
deviceName()	返回当前设备名称。调用依赖设备，使用SQL调试时返回为空。
endswith(input, suffix)	判断input是否以suffix结尾。
exp(number)	返回指定数字的指定次幂。
floor(number)	返回一个最接近它的整数，它的值小于或等于这个浮点数。
log(n, m)	返回自然对数。 如果不传m，则返回log(n)。
lower(string)	返回小写字符串。
mod(n, m)	n%m 余数。
nanvl(value, default)	返回属性值。 若属性值为null，则返回default。
newuuid()	返回一个随机uuid字符串。
payload(textEncoding)	返回设备发布消息的payload转字符串。 字符编码默认UTF-8，即 payload()默认等价于payload(‘utf-8’)。
power(n,m)	返回n的m次幂。
rand()	返回[0~1)之间随机数。
replace(source, substring, replacement)	对某个目标列值进行替换。 示例：replace(field,’a’,’1’)。
sin(n)	返回 n的正弦。
sinh(n)	返回 n的双曲正弦（hyperbolic sine）。
tan(n)	返回 n的正切。
tanh(n)	返回n的双曲正切（hyperbolic tangent）。

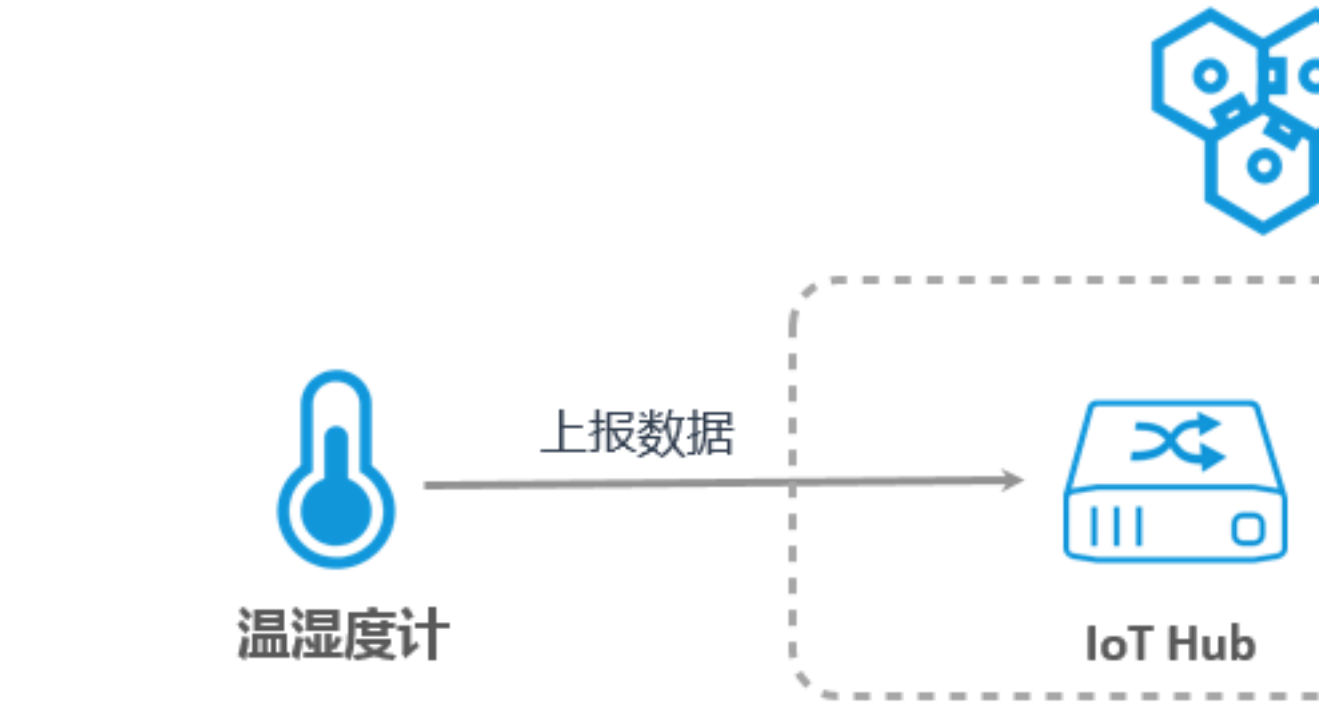
timestamp(format)	返回当前系统时间。 format可选。如果为空则返回当前系统时间戳毫秒值，比如 timestamp() = 1232323233，timestamp('yyyy-MM-dd HH:mm:ss.SSS')=2016-05-30 12:00:00.000。
topic(number)	返回Topic分段信息。 如，有一个Topic： /abcdef/ghi。函数 topic() 返回“ /abcdef/ghi”，topic(1) 返回“abcdef”， topic(2) 返回“ghi”。
upper(string)	返回大写字符。
to_base64(*)	当原始Payload数据为二进制数据时，可使用该函数，将所有二进制数据转换成base64String。

3.5 数据流转

规则引擎仅能处理发送至Topic的数据。本文将为您讲解使用规则引擎时，数据的流转和不同阶段的数据格式。

基础版产品

基础版产品设备数据直接透传至IoT Hub，数据结构不变。数据流转图如下：



设备：D39383

```
topic："/{PK}/{DN}/data"
payload:
{
"temperature":23,
"humidity":63
}
```

设备原始payload数据

```
topic："/{PK}/{DN}/da
payload:
{
"temperature":23,
"humidity":63
}
```

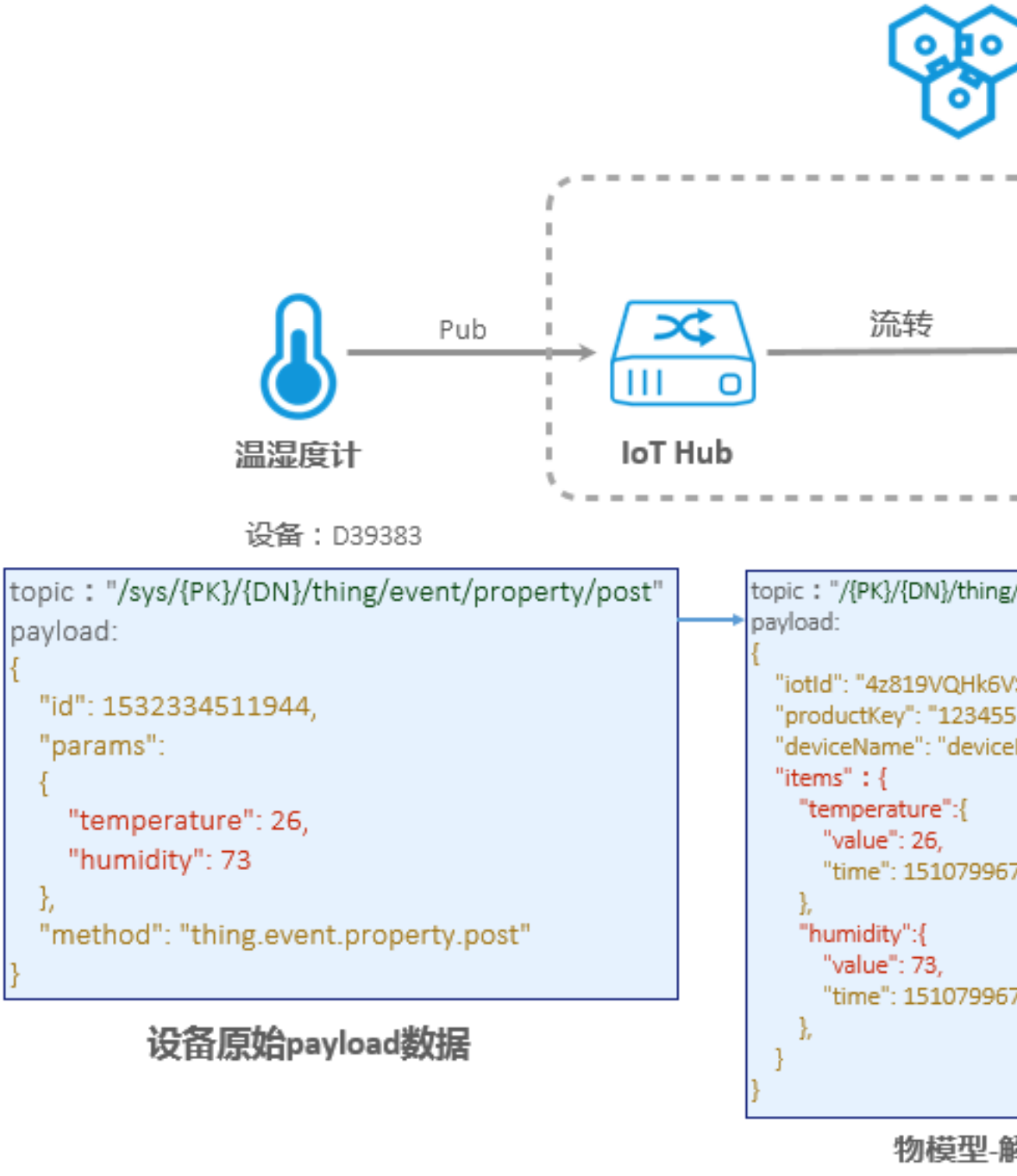
透传结构不

```
SELECT
deviceName() as deviceNa
timestamp('HH:mm') as tir
temperature,
humidity
FROM "/{PK}/+/data"
```

高级版产品

创建高级版产品时，可选择数据格式为Alink JSON格式或者二进制格式。

- 二进制格式数据规则引擎不做处理，直接透传，数据流转请参考基础版产品的数据流转图。
- Alink JSON格式数据会经物模型解析，规则引擎SQL处理解析后的数据，具体请参考数据格式(高级版)。



3.6 数据格式(高级版)

使用规则引擎，您需要基于Topic编写SQL处理数据。基础版Topic和高级版自定义的Topic，数据格式是您自定义的，物联网平台不做处理。高级版系统默认的Topic，物联网平台定义了Topic格

式，此时您需要根据平台定义的数据格式，处理数据。本文将为您讲解高级版系统默认Topic的数据格式。

设备属性上报

通过该Topic获取设备上报的属性信息。

Topic：`/sys/{productKey}/{deviceName}/thing/event/property/post`

数据格式：

```
{
    "iotId":"4z819VQHk6VSLmmBJfrf00107ee200",
    "productKey":"1234556554",
    "deviceName":"deviceName1234",
    "gmtCreate":1510799670074,
    "deviceType":"Ammeter",
    "items":{
        "Power":{
            "value":"on",
            "time":1510799670074
        },
        "Position":{
            "time":1510292697470,
            "value":{
                "latitude":39.9,
                "longitude":116.38
            }
        }
    }
}
```

参数说明：

参数	类型	说明
iotId	String	设备在平台内的唯一标识
productKey	String	产品的唯一标识
deviceName	String	设备名称
deviceType	String	设备类型
items	Object	设备数据
Power	String	属性名称，产品所具有的属性名称请参考TSL描述
Position	String	属性名称，产品所具有的属性名称请参考TSL描述
value	根据TSL定义	属性值

参数	类型	说明
time	Long	属性产生时间，如果设备没有上报默认采用云端生成时间
gmtCreate	Long	数据流转消息产生时间

设备上报事件

通过该topic获取设备上报的事件信息。

Topic：`/sys/{productKey}/{deviceName}/thing/event/{tsl.event.identifier}/post`

数据格式：

```
{
    "identifier":"BrokenInfo",
    "name":"损坏率上报",
    "type":"info",
    "iotId":"4z819VQHk6VSLmmBJfrf00107ee200",
    "productKey":"X5eCzh6fEH7",
    "deviceName":"5gJtxDVeGAkaEztpisjX",
    "gmtCreate":1510799670074,
    "value":{
        "Power":"on",
        "Position":{
            "latitude":39.9,
            "longitude":116.38
        }
    },
    "time":1510799670074
}
```

参数说明：

参数	类型	说明
iotId	String	设备在平台内的唯一标识
productKey	String	产品的唯一标识
deviceName	String	设备名称
type	String	事件类型，事件类型参考TSL描述
value	Object	事件的参数
Power	String	事件参数名称
Position	String	事件参数名称

参数	类型	说明
time	Long	事件产生时间，如果设备没有上报默认采用远端时间
gmtCreate	Long	数据流转消息产生时间

网关发现子设备数据流转

在一些场景中网关能够检测到子设备，并将检测到的子设备信息上报。此时可以通过该Topic获取到上报的信息。

Topic：`/sys/{productKey}/{deviceName}/thing/list/found`

数据格式：

```
{
    "gwIotId":"4z819VQHk6VSLmmBJfrf00107ee200",
    "gwProductKey":"1234556554",
    "gwDeviceName":"deviceName1234",
    "devices":[
        {
            "productKey":"12345565569",
            "deviceName":"deviceName1234",
            "iotId":"4z819VQHk6VSLmmBJfrf00107ee201"
        }
    ]
}
```

参数说明：

参数	类型	说明
gwIotId	String	网关设备在平台内的唯一标识
gwProductKey	String	网关产品的唯一标识
gwDeviceName	String	网关设备名称
iotId	String	子设备在平台内的唯一标识
productKey	String	子设备产品的唯一标识
deviceName	String	子设备名称

设备下行指令结果数据流转

通过该Topic可以获取，通过异步方式下发指令给设备，设备进行处理后返回的结果信息。如果下发指令过程中出现错误，也可以通过该Topic得到指令下发的错误信息。

Topic：`/sys/{productKey}/{deviceName}/thing/downlink/reply/message`

数据格式：

```
{
    "gmtCreate":1510292739881,
    "iotId":"4z819VQHk6VSLmmBJfrf00107ee200",
    "productKey":"1234556554",
    "deviceName":"deviceName1234",
    "requestId":1234,
    "code":200,
    "message":"success",
    "topic":"/sys/1234556554/deviceName1234/thing/service/property/set
",
    "data":{

    }
}
```

参数说明：

参数	类型	说明
gmtCreate	Long	UTC时间戳
iotId	String	设备在平台内的唯一标识
productKey	String	产品Key
deviceName	String	设备名称
requestId	Long	阿里云产生和设备通信的信息id
code	Integer	调用的结果信息
message	String	结果信息说明
data	Object	设备返回的结果，非透传之间返回设备结果，透传则需要经过脚本转换

返回信息：

参数	类型	说明
200	success	请求成功
400	request error	内部服务错误， 处理时发生内部错误
460	request parameter error	请求参数错误， 设备入参校验失败
429	too many requests	请求过于频繁
9200	device not actived	设备没有激活
9201	device offline	设备不在线

参数	类型	说明
403	request forbidden	请求被禁止，由于欠费导致

设备上下线状态数据流转

通过该Topic获取设备的上下线状态。

数据流转Topic：`{productKey}/{deviceName}/mqtt/status`

数据格式：

```
{
    "productKey":"1234556554",
    "deviceName":"deviceName1234",
    "gmtCreate":1510799670074,
    "deviceType":"Ammeter",
    "iotId":"4z819VQHk6VSLmmBJfrf00107ee200",
    "action":"online",
    "status":{
        "value":"1",
        "time":1510292697471
    }
}
```

参数说明：

参数	类型	说明
iotId	String	设备在平台内的唯一标识
productKey	String	产品的唯一标识
deviceName	String	设备名称
status	Object	设备状态
value	String	状态值 1上线，0离线
time	Long	设备上下线时间
gmtCreate	Long	数据流转消息产生时间
action	String	设备状态变更动作，online上线， offline离线

3.7 地域和可用区

物联网平台使用规则引擎将设备数据转发至其他阿里云产品。在转发时，需确认目的云产品已经在该地域和可用区上线，并且支持相应格式数据的转发。

表 3-1: 地域和可用区列表

数据转发目标云产品	华东2（上海）		新加坡		日本（东京）		美国（硅谷）		德国（法兰克福）	
	JSON	二进制	JSON	二进制	JSON	二进制	JSON	二进制	JSON	二进制
表格存储（Table Store）	√	-	√	-	√	-	√	-	√	-
云数据库RDS版（RDS）	√	-	√	-	√	-	√	-	√	-
消息服务（Message Service）	√	√	√	√	√	√	√	√	√	√
函数计算（FC，Function Compute）	√	√	√	√	-	-	-	-	-	-

3.8 使用实例

3.8.1 数据转发到另一Topic

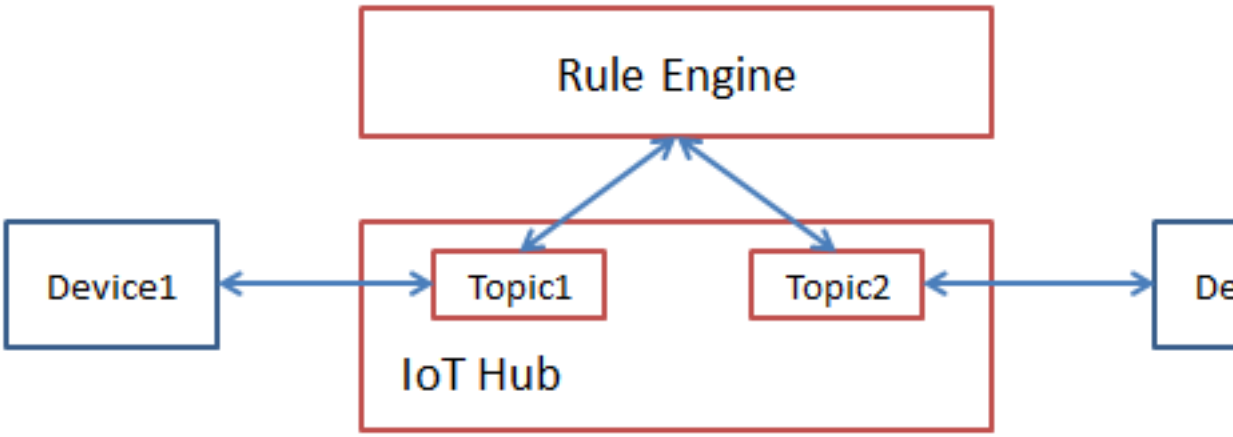
您可以将依据SQL规则处理完的数据，设置转发到另一个Topic中，实现M2M或者更多其他场景。

前提条件

在设置转发之前，您需要参考设置规则引擎编写SQL完成对数据的处理。

背景信息

本文将教您如何设置数据从Topic1中依照规则引擎设置转发到Topic2内：



操作步骤

1. 单击数据转发一栏的添加操作。出现添加操作页面。

2. 按照页面提示，设置参数。
   - 选择操作：此处选择发布到另一个**Topic**。
   - Topic：选择您需要把数据转发到哪一个Topic中。
     - 自定义：填写您自定义的产品Topic。在选择产品后，还需补充完整该Topic。您可以使用`${}`表达式引用上下文值。例如，填写`${devicename}/get`表示从消息中筛选出devicename信息，转发到后缀为get的Topic中。
     - sys：选择系统定义的Topic。在选择产品后，还需要选择设备，选择系统定义的某个Topic。

3.8.2 数据转发到MQ

您可以使用规则引擎，将数据转发到消息队列（以下简称为MQ）中，从而实现消息依次从设备、物联网平台、MQ到应用服务器之间的全链路高可靠传输能力。

前提条件

在设置转发之前，您需要参考设置规则引擎编写SQL完成对数据的处理。

操作步骤

1. 单击数据转发一栏的添加操作，出现添加操作页面。选择发送数据到消息队列**(Message Queue)**中。

2. 按照界面提示，设置参数。
   - 选择操作：选择MQ。
   - 地域：选择MQ Topic所在地域。

- Topic：根据业务选择需要写入数据的Topic。
- 授权：授权平台将数据写入MQ Topic的权限。

> **说明：**
> MQ非铂金实例的Topic不支持跨地域的访问写入，此时需保证规则引擎与MQ Topic在同一地域下。若您已购买并使用铂金实例，则无需关心地域问题。

转入MQ的规则启动之后，平台即可将数据写入MQ的Topic，您可以使用MQ实现消息收发，具体使用方法，请参考*MQ*订阅消息。

3.8.3 数据转发到表格存储

您可以使用规则引擎，将数据转发到表格存储（Table Store）中。

前提条件

在设置转发之前，您需要参考设置规则引擎编写SQL完成对数据的处理。

操作步骤

1. 单击数据转发一栏的添加操作，出现添加操作页面。选择存储到表格存储**(Table Store)**。

添加操作

选择操作：

存储到表格存储(Table Store)

该方法将数据插入到表格存储(Table Store)中, 详情请参考
中, 详情请参考文档

* 地域：

华东 2

* 实例：

ShanghaiRegion

* 数据表：

shanghai_61034

* 主键：

pk1 *值： 12

* 角色：

AliyunIOTAccessingOTSRole

2. 按照界面提示，设置参数。
   - 选择操作：选择Table Store。
   - 地域、实例、数据表：选择Table Store数据表的基本信息，用于数据存储。
   - 主键：Table Store数据表都包含主键，选择好数据表之后，控制台将自动读出该表的主键，此处需要您配置主键的值。
   - 角色：授权平台将数据写入Table Store。此处需要您先创建一个具有Table Store写入权限的角色，然后将该角色赋予给规则引擎。这样，规则引擎才可以将处理完成的数据写入数据表中。

后续操作

示例

使用SQL抽取JSON数据：`{"device":"bike","product":"xxx","data2":[{...}]}`。因业务需要，需将此JSON数据存入Table Store中，主键是`device`，`product`，`id`。

配置及效果：

1. 在控制台配置主键的值，输入`${device}`。当有消息过来并触发规则，主键device就会存入JSON中device的value值。主键product同理。

    说明：

   `${}`是转义符，如果不输入该转义符，存入的将会是一个常量。

2. 规则会自动匹配主键是否为自增列，如果是自增列，将自动返填`AUTO_INCREMENT`，并且不能编辑。
3. 主键配置完成后，平台将自动解析JSON中除主键外的key值，并依据key值自动创建Table Store的数据列。此例中，将会自动创建data1和data2两个列，且在每列下面存入对应的value值。

    说明：

   目前只支持一级JSON解析，不支持嵌套JSON的解析。因此，该示例中data2下面将会以字符串形式存入整个嵌套JSON，而不会对嵌套JSON再次进行解析创建列。

3.8.4 数据转发到DataHub

物联网平台用于接入和管理设备，数据存储和计算将交给阿里云其他产品。比如，您可以使用规则引擎将数据转到*DataHub*上，由DataHub为下游流计算、MaxCompute等提供实时数据，帮助用户实现更多计算场景。

前提条件

在设置转发之前，您需要参考设置规则引擎编写SQL完成对数据的处理。

操作步骤

1. 单击数据转发一栏的添加操作，出现添加操作页面。选择发送数据到**DataHub**中。


添加操作
选择操作：
发送数据到DataHub中
该操作将数据插入到Datahub中, 详情请参考文档
* 地域：
华东 2
* Project：
iotdata
* Topic：
iotdata
* 角色：
AliyunIOTAccessingDataHubRole

2. 按照界面提示，设置参数。

   - 选择操作：选择发送数据到DataHub中。
   - 选择DataHub对应的地域、Project和Topic。选完Topic后，规则引擎会自动获取Topic中的Schema，规则引擎筛选出来的数据将会映射到对应的Schema中。

     说明：

     - 将数据映射到Schema时，需使用`${}`，否则存入表中的将会是一个常量。
     - Schema与规则引擎的的数据类型必须保持一致，不然无法存储。
   - 角色：授权物联网平台将数据写入DataHub。此处需要您先创建一个具有DataHub写入权限的角色，然后将该角色赋予给规则引擎。这样，规则引擎才可以将处理完成的数据写入DataHub中。

3.8.5 数据转发到RDS

您可以设置规则引擎，将处理后的数据转发到云数据库（以下简称RDS）的VPC实例中。

使用须知

- 目前转发至RDS只发布在华东2、硅谷和新加坡三个节点。
- 只支持同region转发，不支持跨区转发。比如，华东2节点的平台数据只能转发到华东2的RDS中。
- 只支持转发到VPC实例。
- 只支持MySQL实例。
- 支持普通数据库和高权限数据库的转发。

准备工作

在设置转发之前，您需要参考设置规则引擎编写SQL完成对数据的处理。

操作步骤

1. 单击数据转发一栏的添加操作，出现添加操作页面。选择存储到云数据库(RDS)中。

添加操作

选择操作：

存储到云数据库(RDS)中

该操作将数据插入到云数据库(RDS)中, 详情请参考云数据库(RDS)中, 详情请参考文档

特别提醒：此操作仅针对专有网络的RDS实例，并且将会在您的RDS白名单中添加一条记录100.104.123.0/24，用于IoT访问您的数据库，请勿删除。

区域：

华东2

* VPC实例：

rm-uf63b1cji3z6xgpk5 创建实

* MySQL数据库：

asdf

* 账号：

请选择 创建账

* 请输入密码：

请输入该账号的密码

* 表名：

请输入已存在的表名

* 键：

RDS表中的字段

2. 按照界面提示，设置参数。

   - 选择操作：选择存储到云数据库(RDS)中。
   - VPC实例、MySQL数据库：根据您的业务选择当前区域下的VPC实例和MySQL数据库。

     
     说明：
     如果是高权限数据库，需要您手动输入数据库名称。

   - 账号、密码：输入数据库的账号、密码。此账号密码应具有该数据库的读写权限，否则规则引擎无法将数据写入RDS。

     
     说明：
     规则引擎获得账号后，仅负责将规则匹配的数据写进数据库中，不会做其他操作。

   - 表名：输入数据库中已建立的数据表名，规则引擎将把数据写入这张表上。
   - 字段：此处输入数据表的字段，规则引擎将把处理后的数据存入该字段中。
   - 值：此处填写输入数据表字段的值。可以使用转义符`$`，格式为`${key}`，即提取Topic中key对应的Value值作为输入值。

     规则引擎的SQL为：`SELECT tem FROM mytopic`；RDS数据库有一张表，表中有`tem`字段，类型是`String`。

     控制台配置时，字段处填入RDS数据表的字段**`tem`**，值处填入规则引擎筛选出来的JSON字段**`${key}`**。

     
     说明：
     - 值处需使用`${}`，否则填入表中的将是一个常量。
     - 字段与值的数据类型需保持一致，否则无法存储成功。

3. 配置完成后，规则引擎为了连接RDS，会在RDS的白名单中添加下列IP。若这些IP未出现，请手动添加。

   - 华东2：100.104.123.0/24
   - 亚太东南1（新加坡）：100.104.106.0/24
   - 美国西部1（硅谷）：100.104.8.0/24

   RDS控制台的白名单示例如下：

3.8.6 数据转发到HiTSDB

您可以配置规则引擎将处理过的数据转发到时间序列数据库（*HiTSDB*）的实例中。

使用须知

- 目前转发至HiTSDB仅发布在华东2节点。
- 只支持专有网络下HiTSDB实例.

- 只支持同region转发。例如：华东2节点的套件数据只能转发到华东2的HiTSDB中。
- 只支持JSON格式数据转发。

准备工作

在设置转发之前，您需要参考设置规则引擎编写SQL完成对数据的处理。

操作步骤

1. 单击**数据转发**一栏的**添加操作**，出现**添加操作**页面。选择**存储到高性能时间序列数据库(HiTSDB)**中。

添加操作


选择操作：
存储到高性能时间序列
该操作将数据插入到 时序数
区域：
华东2
* VPC实例：
ts-uf6x015n22as884
* timestamp：
请输入时间戳标签

2. 按照界面提示，设置参数。

   - 选择操作：此处选择高性能时间序列数据库(HiTSDB)
   - 地域、实例：选择处理后的数据将要存入哪个数据库实例中。
   - timestamp：此处的值必须为Unix时间戳，如1404955893000。有如下两种配置方式：
     - 使用转义符`${}`表达式，例如`${time}`。此时timestamp的值为指定Topic对应数据包中time字段对应的值。建议使用此方式。
     - 使用规则引擎函数`timestamp()`，此时timestamp的值为规则引擎服务器的时间戳。
   - tag：必须使用常量配置，值有三种配置方式：
     - 使用转义符`${}`表达式，例如`${city}`，此时值为指定Topic对应数据包中city字段对应的值。建议使用此方式。
     - 使用规则引擎函数规定的一些函数，例如`deviceName()`，此时值为设备名称。
     - 使用常量配置，例如beijing，此时值为beijing。
   - 授权：勾选同意物联网平台向HiTSDB写数据。此时，规则引擎会向时间序列数据库实例中添加网络白名单，用于IoT访问您的数据库， 请勿删除这些IP段。

示例

规则引擎的SQL：

```
SELECT time,city,power,distance, FROM "/myproduct/myDevice/update"；
```

配置的规则如操作步骤所示。

发送的消息：

```
{
"time": 1513677897,
"city": "beijing",
"distance": 8545,
"power": 93.0
}
```

规则引擎会向高性能时间序列数据库中写入的两条数据：

```
数据：timestamp:1513677897, [metric:distance value:8545]
tag： device=myDevice,product=bikes,cityName=beijing
```

```
数据：timestamp:1513677897, [metric:power value:93.0]
```

```
tag:device=myDevice,product=bikes,cityName=beijing
```

注意

- 发送的消息中除了在规则引擎中配置为timestamp或者tag值的字段外，其他字段都将作为metric写入时间序列数据库。如上例所示除time和city以外，distance和power作为两个metric写入数据库。
- metric的值只能为**数值类型**，否则会导致写入数据库失败。
- 用户要保证在规则引擎中配置的tag-值键值对能够获取到，如果获取不到任意一个tag-值键值对，会导致写入数据库失败。
- tag-值键值对限制最多输入8个。
- metric、tag和tag值只可包含大小写英文字母、中文、数字，以及特殊字符`-_./():,[]=‘`
- 规则引擎在HiTSDB的白名单中添加以下IP段用以访问数据库。若IP未出现，请手动添加。

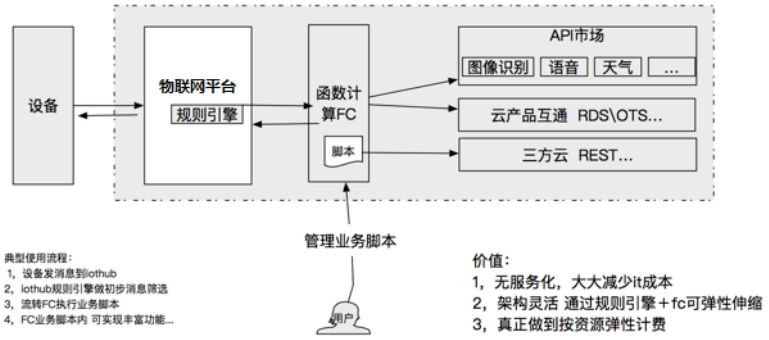
  华东2：100.104.76.0/24

  HiTSDB控制台白名单示例如下：


实例详情
实例详情
实例监控
数据时效
数据清理
时间线清理
数据查询
产品文档
基础信息
实例ID： ts-
地域： 华东 2
专有网络： vpc-
VPC网络地址： ts-
公共网络地址： 申请公共网络
运行状态
运行状态： 运行中
创建时间： 2017-09-14 15:54:20
配置信息
存储容量： 200 GB
最大支持时间线： 1000000
白名单状态
网络地址白名单：
100.104.76.0/24

3.8.7 转发数据到函数计算

您可以使用规则引擎，将处理后的数据转发至函数计算（Function Compute）中。



操作流程概述：

1. 函数计算控制台创建好服务、函数。
2. 创建规则，将IoT数据处理并转发到函数计算中，启动规则。
3. 使用配置了规则的Topic发送一条消息。
4. 查看函数计算服务实时监控大盘查询函数执行情况，或者根据函数的具体业务逻辑查看结果是否正确。

操作步骤

1. 登录函数计算控制台，创建服务与函数。
   a. 创建服务。其中，**服务名称**必须填写，其余参数请根据您的需求设置。


函数计算
服务列表
搜索服务
1/2
计量数据
监控数据每小时更新并尽最大可能
本月执行次数
5.901万
触发器
对象存储
结合OSS事件即时触发函数处
数据
HTTP 触发器
NEW
通过访问HTTP URL来调用函
函数模版
图像分类
您仅需要通过OSS上传图片，
计算服务进行图像识别并对结
类存储的模板。

b. 创建服务成功后，创建函数。

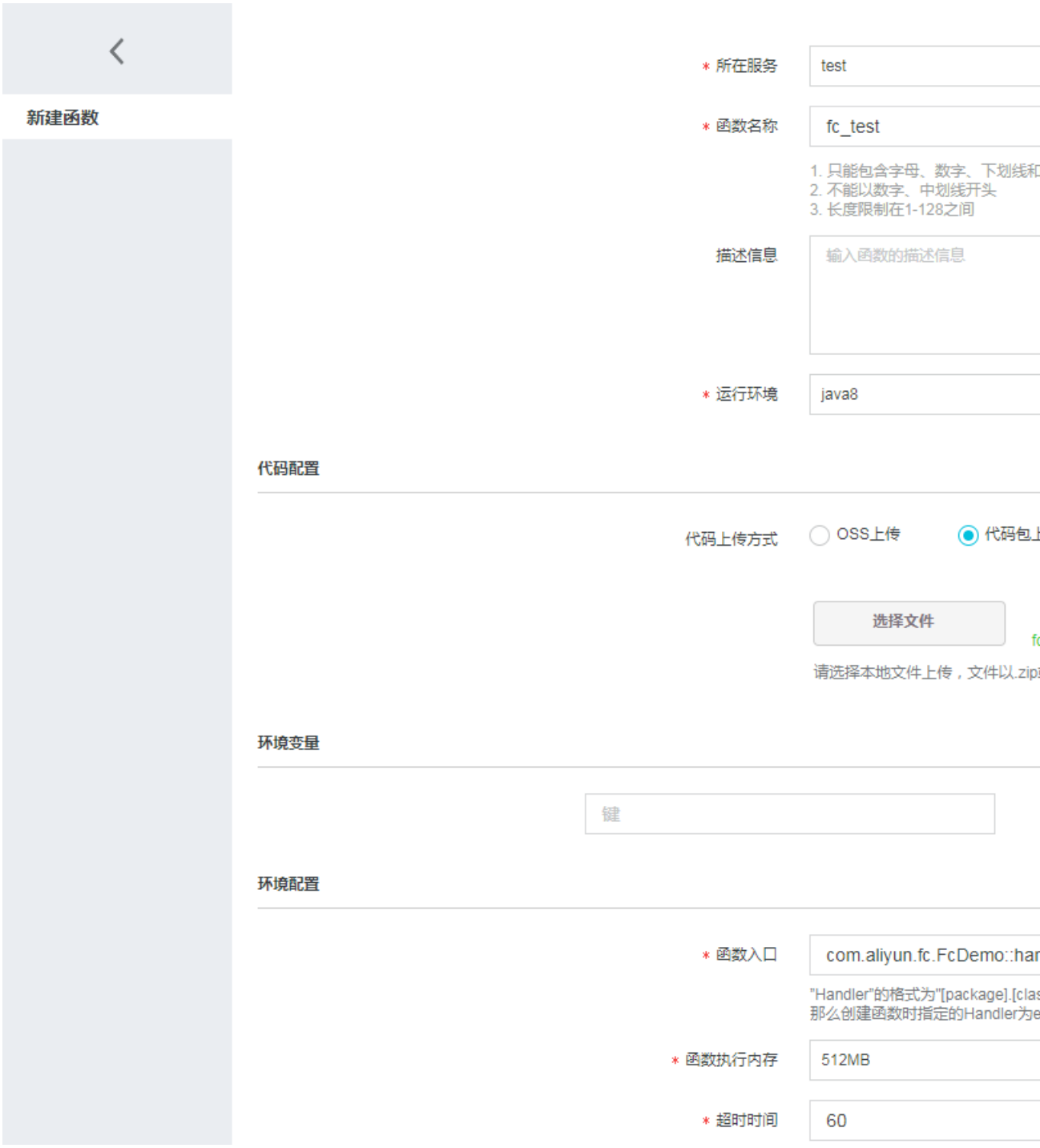

华东2 (上海) > test
服务概览
函数列表
搜索函数
1/1
计量数据
监控数据每小时更新并尽最大可能推送
本月执行次数
0.0
基础配置
服务名称 test
创建时间 06/28/2018, 09:
功能描述
高级配置
日志项目
服务角色

c. 选择函数模板，此处以空白函数模板为示例。


新建函数
新建函数
函数模版
函数模版选择
业务模板提供了示例配置和代码，为您创建函数时
选择全部的语言
空白函数
空白函数模板会创建一个空白函数，通过
置和代码开发，完成函数的创建。
flask-web
python2.7
通过该模板的示例代码，用户可以通过u
serverless.
Previous
1
2
3
Next

d. 设置函数参数。

此处设置函数的逻辑为直接在函数计算中显示获取的数据。



其中，

所在服务：选择*1.a*中创建的服务。

函数名称：设置您的函数名称。

运行环境：设置函数运行的环境，此示例中选择java8。

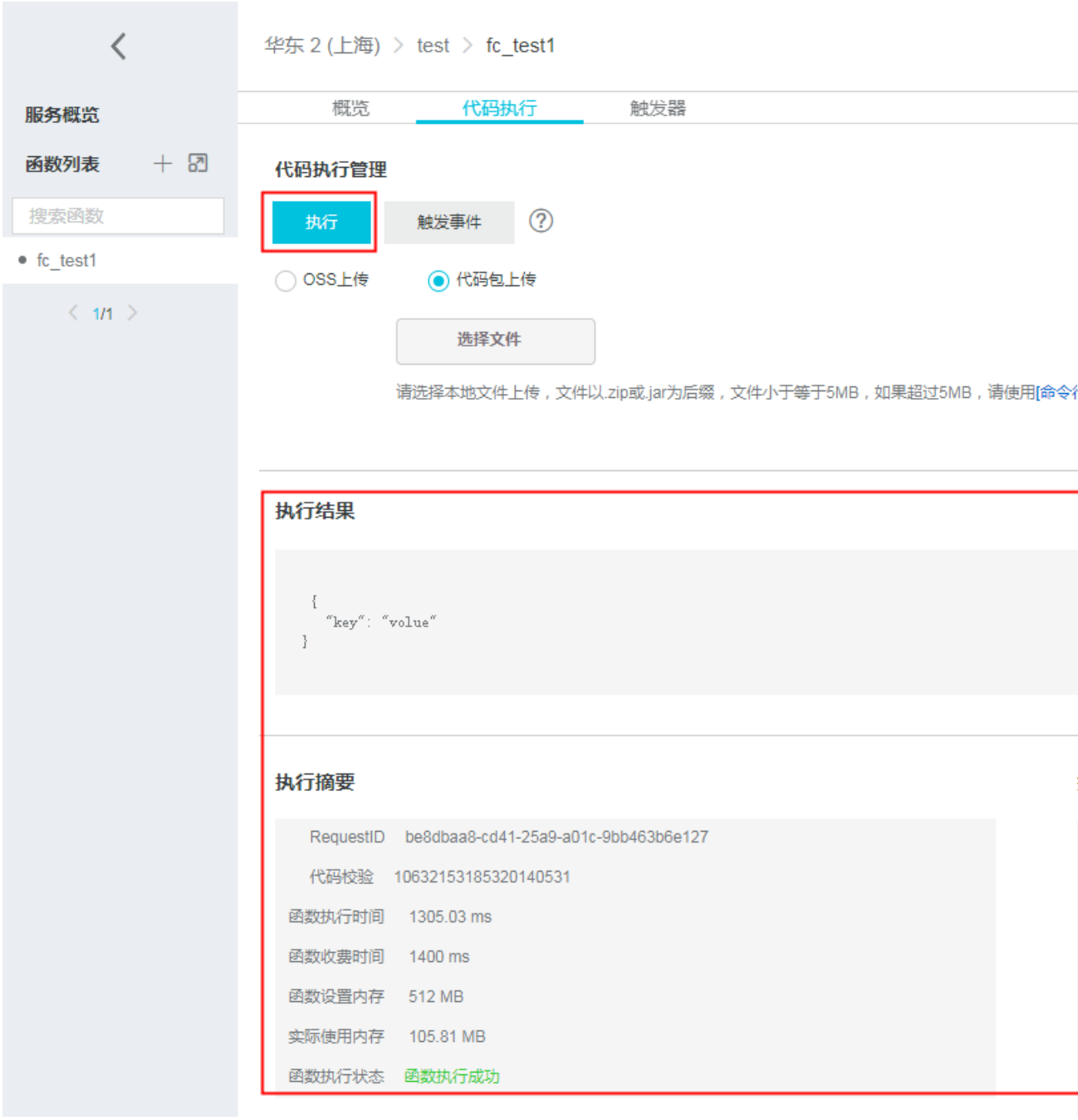
代码配置：上传您的代码。

函数入口：设置您的函数在函数计算系统运行的调用入口，此示例中必须设置为`com.aliyun.fc.FcDemo::handleRequest`。

其余参数的值请根据您的需求，参见函数计算设置。

e. 函数执行验证。

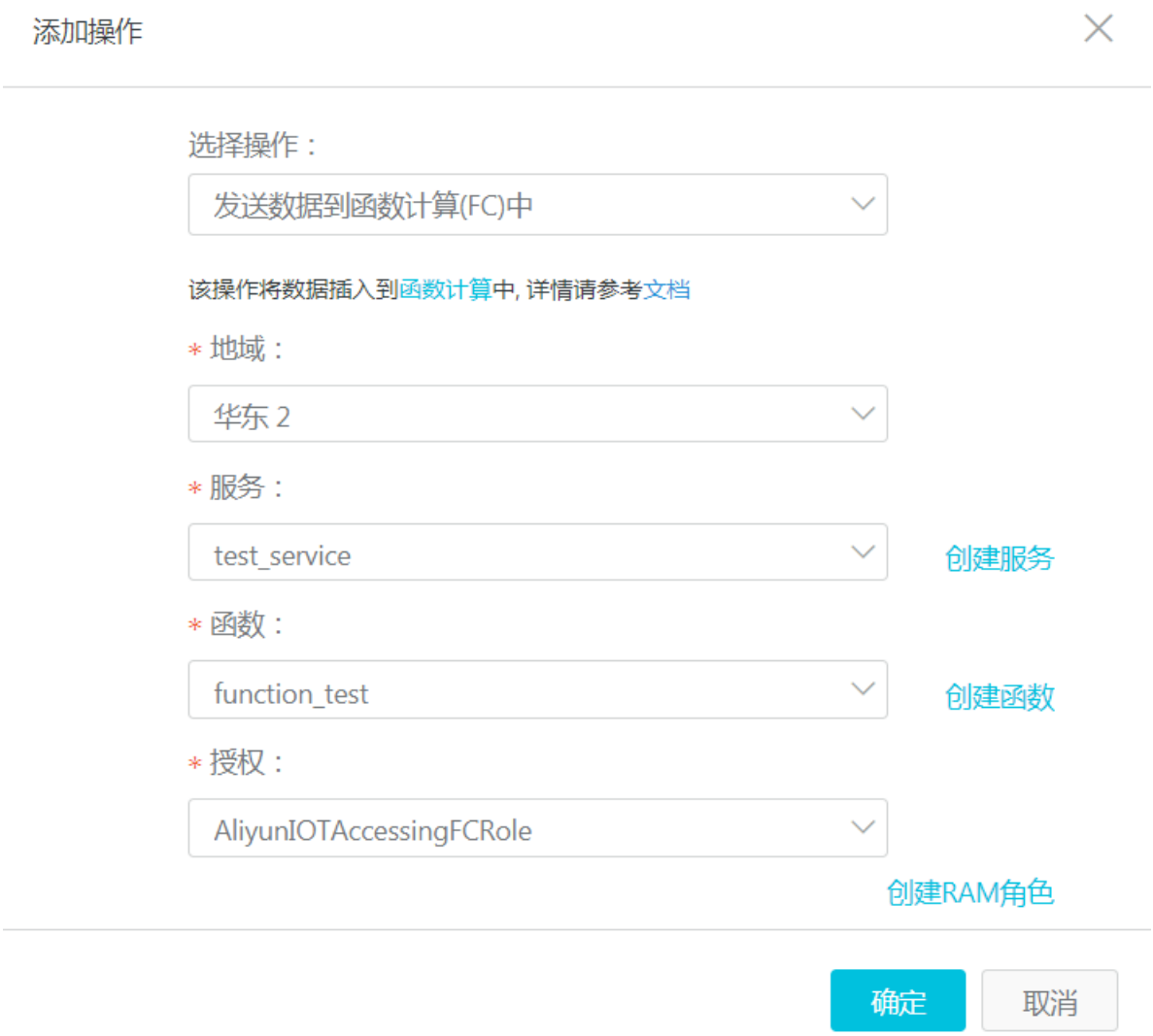
函数成功创建后，可以直接在函数计算的控制台执行，以验证函数执行情况。函数计算会直接将函数的输出和请求的相关信息打印在控制台上。



2. 函数正常执行后，配置IoT规则引擎。
3. 在设置转发之前，您需要参考设置规则引擎编写SQL完成对数据的处理。

> 说明：
> JSON格式和二进制格式都支持转发到函数计算中

4. 单击规则名称，进入规则详情页面。
5. 单击数据转发一栏的添加操作，并在弹出的添加操作页面中设置参数。



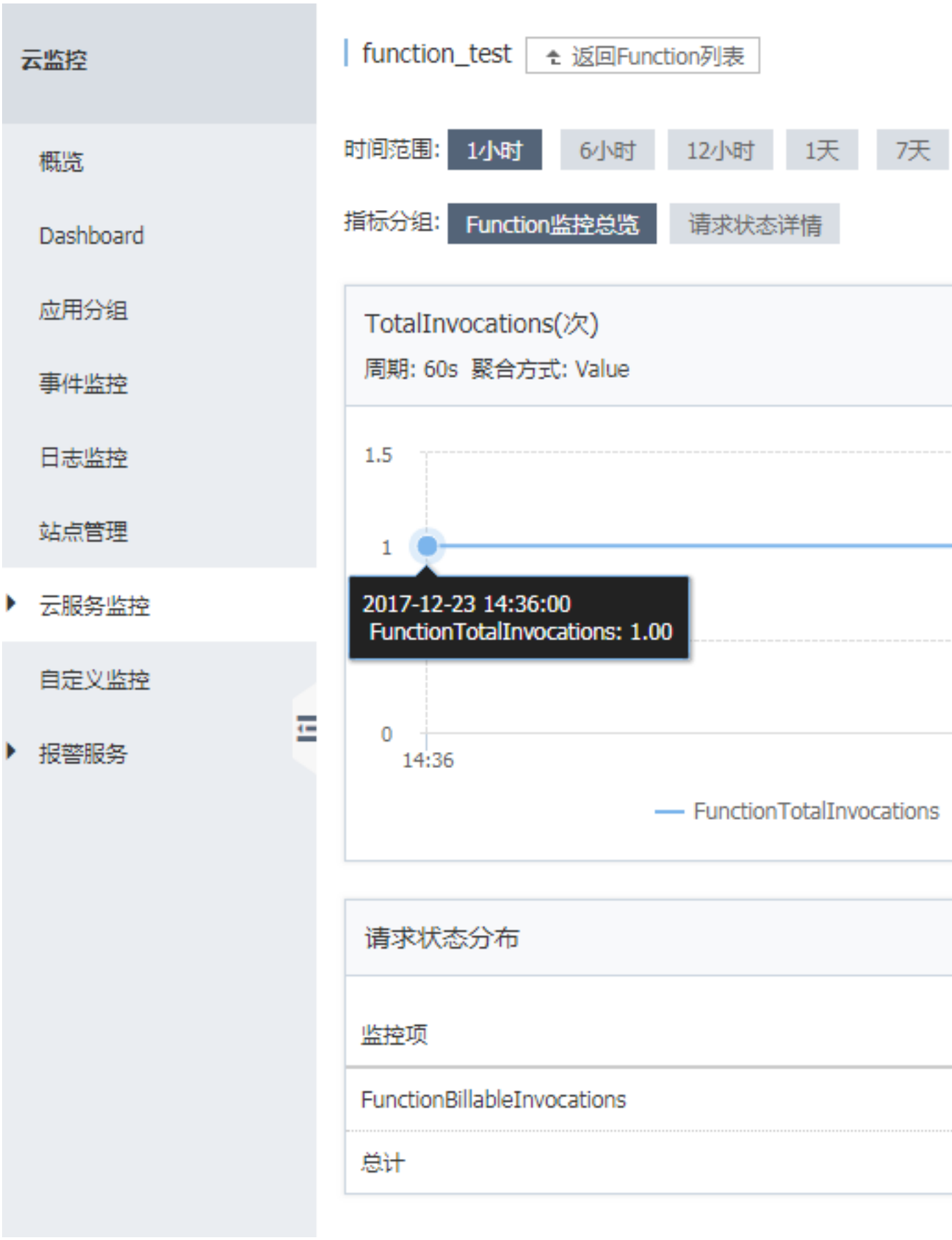
- 选择操作：发送数据到函数计算(FC)中。
- 地域：根据您的业务选择将处理后的数据转发至某个地域。如果该地域没有对应资源，需要到函数计算控制台创建相应的资源。

  说明：
  目前只支持华东2节点
- 服务：根据您的地域选择服务。若没有服务，请单击**创建服务**。
- 函数：根据您的地域选择函数。若没有函数，请单击**创建函数**。
- 角色：授权物联网平台操作函数的权限。此处需要您先创建一个具有执行函数权限的角色，然后将该角色赋予给规则引擎。

6. 启动规则。规则运行后，IoT将根据编写的SQL，将处理后的数据转发到函数计算中。函数计算根据您函数定义的逻辑，直接将接收到的数据显示在函数计算控制台上。

结果验证

函数计算控制台针对函数的执行情况有监控统计。统计有大概5分钟的延时，可以通过监控大盘查询函数的执行情况。


云监控
概览
Dashboard
应用分组
事件监控
日志监控
站点管理
云服务监控
自定义监控
报警服务
function_test
返回Function列表
时间范围:
1小时
6小时
12小时
1天
7天
指标分组:
Function监控总览
请求状态详情
TotalInvocations(次)
周期: 60s 聚合方式: Value
1.5
1
0
2017-12-23 14:36:00
FunctionTotalInvocations: 1.00
14:36
FunctionTotalInvocations
请求状态分布
监控项
FunctionBillableInvocations
总计

4 扩展服务

4.1 固件升级

物联网平台提供固件升级服务。设置设备端支持OTA服务后，您可以在控制台上传待升级的固件，将固件升级消息推送给设备，设备在线升级。本文将为您讲解如何升级固件。

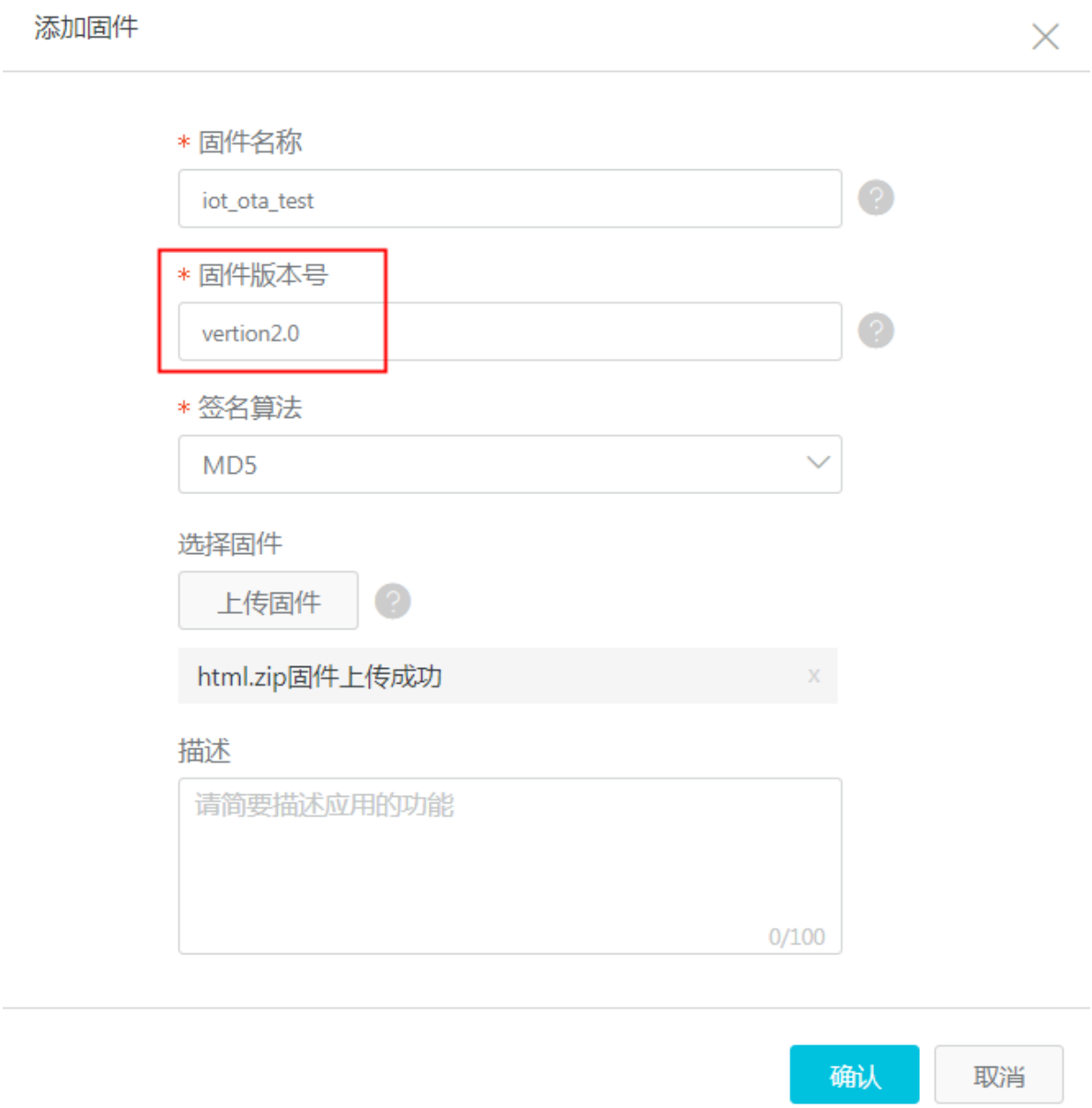
前提条件

使用固件升级功能前，请确保：

- 设备端支持OTA升级服务。
  - 如果您使用设备端SDK，请参考设备*OTA*开发。
  - 如果您使用AliOS Things，请参考*AliOS Things*技术文档。
- 已开通固件升级服务。如果未开通，登录物联网平台的控制台，选择**扩展服务**，单击固件升级下的**使用服务**。

操作步骤

1. 登录物联网平台的控制台。
2. 选择**我的服务** > **固件升级** > **新增固件**，将固件信息上传至云端。



添加固件页面需设置如下参数：

参数	描述
固件名称	设置固件名称。不能包含特殊字符，仅支持中文、英文字母、数字和下划线，长度限制4~32。
固件版本号	输入固件版本号。仅支持英文字母、数字、点、中划线和下划线，长度限制1~64。的固件文件大小不能超过10M。
签名算法	仅支持MD5和SHA256。
上传固件	上传固件文件。文件大小不能超过10M，仅支持bin，tar，gz，zip类型的文件。

参数	描述
	说明： 新增固件最多支持上传100个固件。

3. （可选）若您使用AliOS Things接入平台，可以使用安全升级功能。

   单击**我的服务 > 固件升级 > 安全升级**，将对应AliOS Things待升级产品安全升级按钮置为开。复制公钥用于设备端签名，具体请参考*AliOS Things*技术文档。

   安全升级是保证固件完整性、机密性的一种方式，建议打开。使用安全升级功能，需设备端配合对固件、固件的签名进行验证。

4. 在固件列表选中固件，单击**验证固件**。目的是为了在少量设备上进行测试，验证固件可用后再进行批量升级。

验证固件


为了确保固件批量升级后设备能正常工作，请在批量升
备进行验证固件测试，防止将错误的固件升级到大量设备造
* 升级方式
整包
差分
* 设备所在区域
华东2
* 设备所属产品
水表
* 固件版本号
* DeviceName：
请输入设备

参数	描述
升级方式	• 整包：将整个升级包推送至设备。 • 差分：物联网平台将提取升级包与前一个版本的差异，仅将差异部分推送至设备。差分升级可有效减少升级对设备资源的占用。 **说明：** • 仅AliOS Things设备mk3060，esp8266两个型号支持差分升级功能。 • 差分目标固件大小不能小于20K。
设备所在区域	该设备所在地域。
设备所属产品	选中待升级设备所属的产品。
DeviceName	选中待升级的设备。
选择整包升级时	固件版本号：产品自动上报当前的固件版本号。 固件版本号设置完成后，单击**确定**，开始验证固件。
选择差分升级时	• 待升级版本号：产品自动上报当前固件版本号。此处需选中待升级的固件版本。 • 差分的固件名称：待升级的最新固件。 • 切片大小规格：切片将固件进行切片传输，支持不切片、2M、64K、32K四种规格。 参数设置完成，单击**确定**，平台开始制作差分固件。 差分固件制作完成后，单击**升级**，进行差分升级操作。

说明：

- 设备升级通知：
  - MQTT接入的设备，在线时可以立即接收到升级通知；不在线的设备下次接入时，系统会再次推送升级通知。
  - 其他接入方式（如CoAP或者HTTPS）的设备都是短连接，在线时可以立即收到升级通知；不在线时，无法收到通知。
- 固件上传至云端后，必须使用少量设备验证固件是否可用。固件可用，才允许在大量设备上进行升级。
- 可以反复发起验证固件。

- 只要您进行了固件验证操作，固件状态都会从“未验证”变为“已验证”。与设备的实际升级结果无关。
- 可以单击对应固件的**升级详情**查看固件验证结果，即设备是否升级成功。

5. 固件验证可用后，可单击对应固件的**批量升级**，设置参数，向大批设备定向推送升级通知。

说明：

批量升级前，请确保该固件已验证可用。

批量升级

* 升级方式

整包 差分

* 区域：

华东2

* 设备所属产品：

请选择设备所属产品

* 待升级版本号：

请输入版本号

* 升级策略：

静态升级

* 升级范围：

请选择升级范围

参数	描述
升级方式	• 整包升级：将升级包整包推送到设备端进行升级。需设置待升级的固件版本号。 • 差分升级：将升级包与某一待升级固件间的差异提取出来，仅将差异部分推送至设备端进行升级。需设置待升级的固件版本号、差分的固件名称还有切片大小。
区域	选择设备服务器所在地域。
设备所属产品	选中待升级设备所属的产品。
升级策略	• 静态升级：仅升级满足指定条件的当前设备。 • 动态升级：满足指定条件的设备都将收到升级通知。动态升级将持续维护需升级的设备范围，包括当前已经上报版本号的设备和新激活的设备。
升级范围	• 全部设备：该产品下全部的设备。 • 定向升级：选择定向升级后，仅升级被选中的设备。 • 区域升级：升级实际地理位置在该区域的设备。

说明：

批量升级任务冲突，同一产品下，创建批量升级时，如果选择的待升级版本号，已经处于另一个固件的批量升级过程中，则会提示升级任务冲突。

示例：同一产品下，某一个设备固件版本号为A，用户在控制台创建了两个固件，版本号分别为B和C。用户在控制台创建了固件B的批量升级，升级策略为从版本A到版本B。此时，如果用户想创建固件C的批量升级，升级策略为从版本A到版本C，则云端会触发升级任务冲突。

6. 您可以单击该固件的**升级详情**查看升级状态。

   - 待升级：已选中的待升级设备。
   - 升级中：设备收到升级通知并上传升级进度。
   - 升级成功：本次升级成功的设备。
   - 升级失败：本次升级失败的设备及简要的升级失败原因。以下原因可能造成设备升级失败：
     - 待升级的设备中有些设备还未结束上一次的升级动作。这部分设备升级失败。等设备完成升级动作后，可尝试再次升级。

— 设备在实际升级过程中出现如下载失败、校验失败、解压失败等错误。此时可以尝试再次升级。

4.2 远程配置

前提条件

- 已开通远程配置服务，如果未开通，登录物联网平台的控制台，选择**扩展服务**，单击远程配置下的**开通服务**。
- 设备端SDK已开启支持远程配置服务。需要在设备端SDK中定义 `FEATURE_SERVICE_OTA_ENABLED = y`，SDK提供接口**`linkkit_cota_init`**来初始化远程配置（Config Over The Air，COTA）。

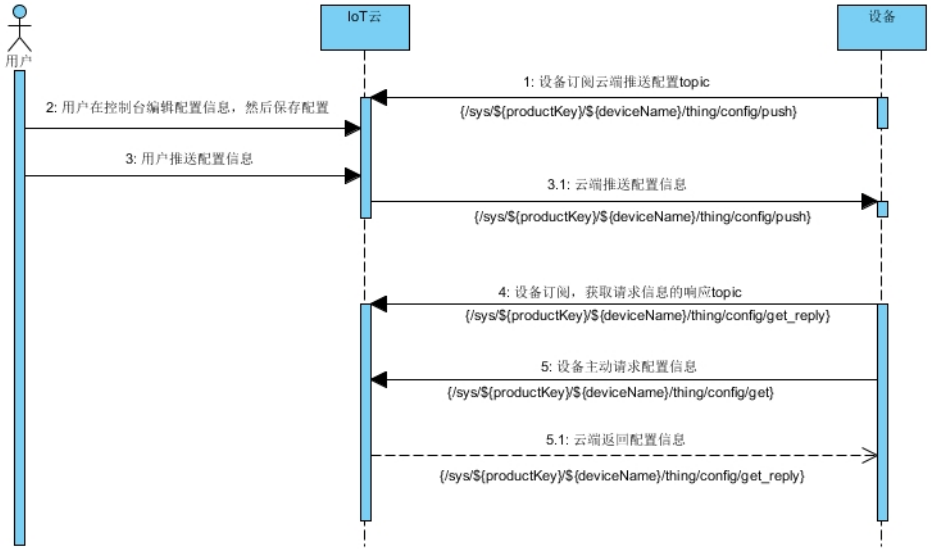
远程配置说明

很多场景下，开发者需要更新设备的配置信息，包括设备的系统参数、网络参数或者本地策略等等。通常情况下更新设备的配置信息是通过固件升级的方式完成的。这将加大固件版本的维护工作，并且需要设备中断运行以完成更新。为了解决上述问题，物联网平台为您提供了远程配置更新的功能，设备无需重启或中断运行即可在线完成配置信息的更新。

物联网平台提供的远程配置功能，可支持开发者：

- 开启/关闭远程配置；
- 在线编辑配置文件并支持版本管理；
- 向设备批量更新配置信息；
- 支持设备主动请求更新配置信息。

远程配置流程图如下所示：



远程配置大致分为三部分：

- 配置生成，您在IoT控制台编辑并保存配置信息；
- 配置使用，您在IoT控制台批量推送配置信息给设备，设备接收后修改本地配置文件；
- 主动请求，设备主动向云端请求新的配置文件并进行更新。

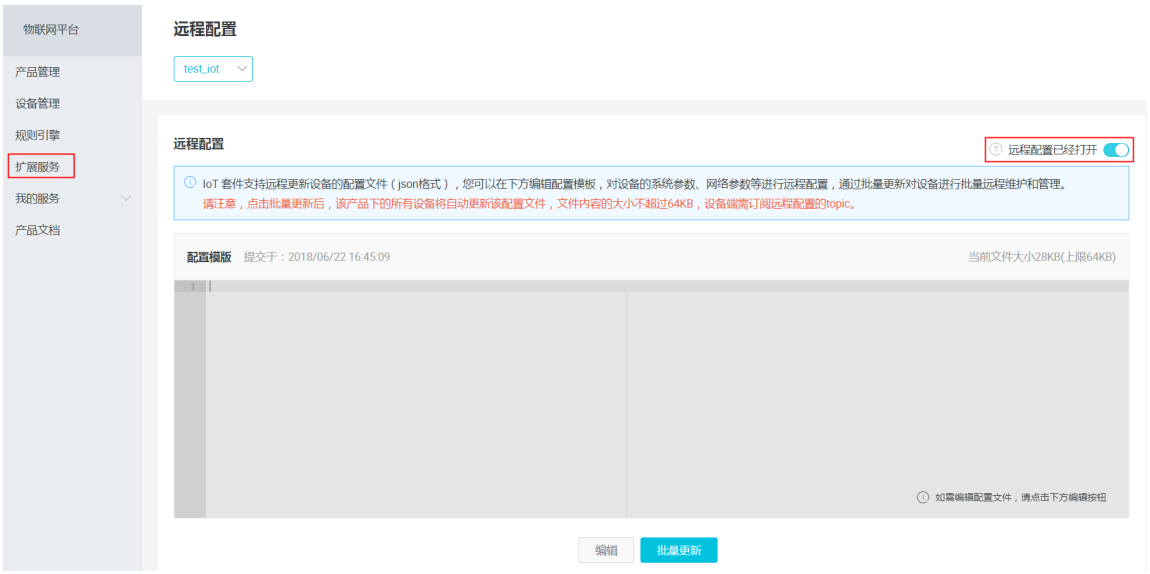
开启远程配置

开启远程配置分为两种场景，一种是云端下发配置信息给设备端的场景，一种是设备端主动查询配置信息的场景。根据场景的不同，开启远程配置的步骤也有所区别。

场景一

设备接收云端的配置信息，可按如下步骤进行远程配置。

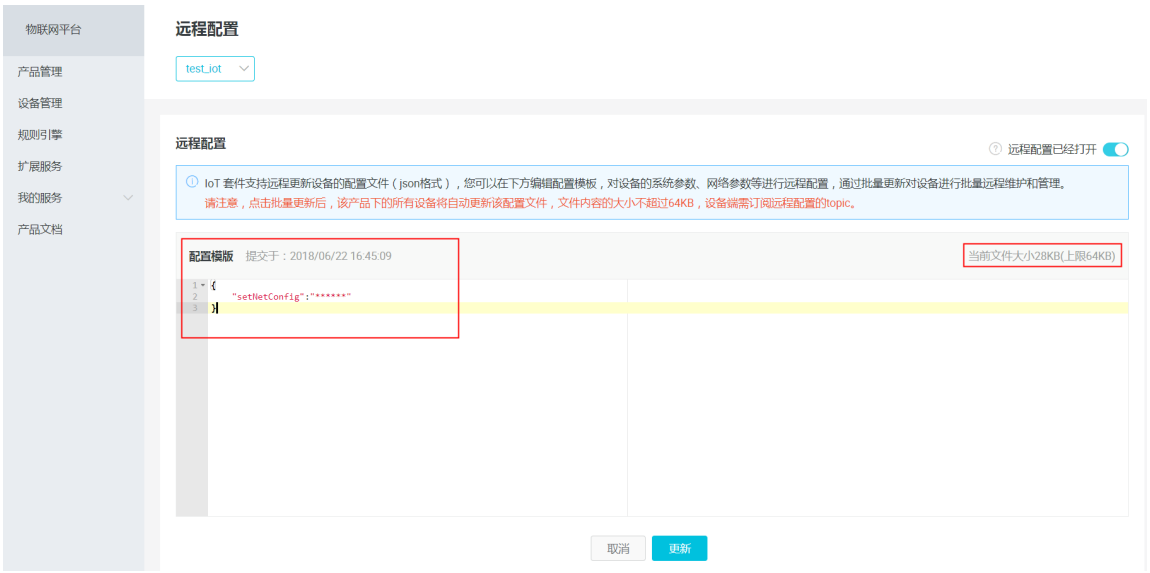
1. 设备上线，订阅推送配置信息的topic(/sys/${productKey}/${deviceName}/thing/config/push)。
2. 在IoT控制台中开启远程配置。
   a. 登录物联网平台的控制台，选择**扩展服务**。
   b. 单击**远程配置**，进入服务详情页面，单击**使用服务**。
   c. 选择产品，打开**远程配置**后的远程配置开关。



说明：

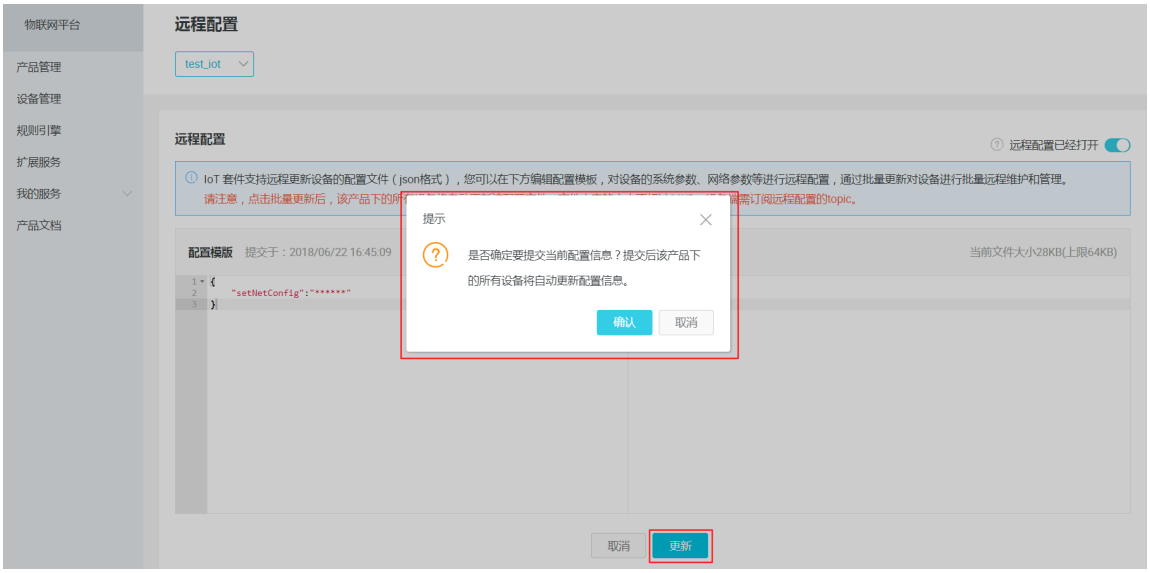
- 必须开启远程配置功能后，才可以编辑配置信息。
- 切换为关闭状态，即可关闭远程配置。

d. 在编辑区，单击**编辑**直接编写配置信息或拷贝配置信息到编辑区。产品配置模板适用于该产品下的所有设备，目前不支持对单个设备进行配置更新。

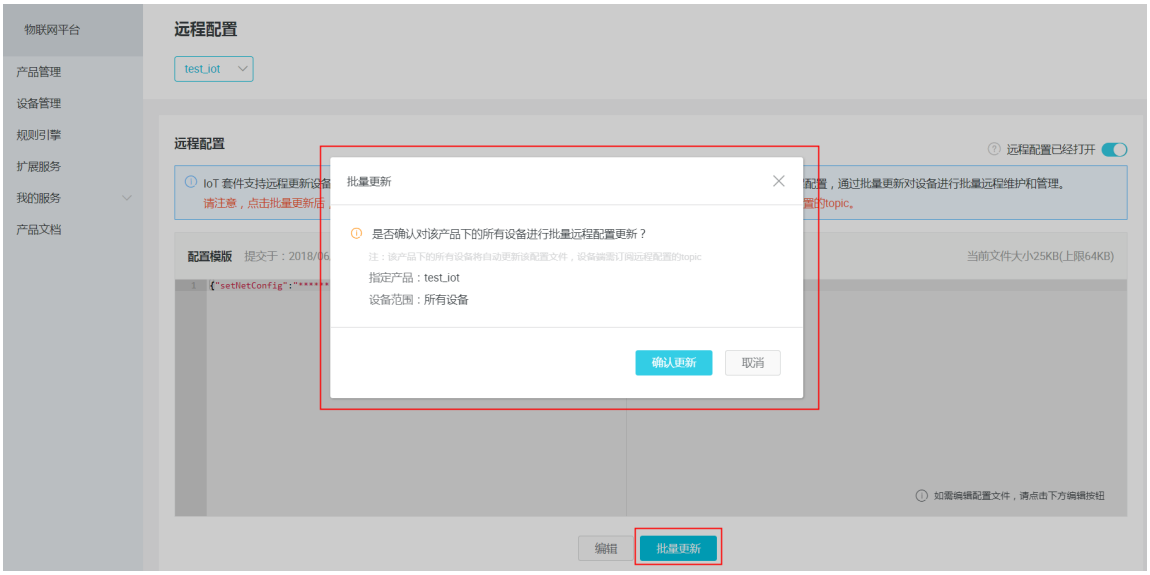


- 远程配置采用JSON格式，物联网套件对配置内容没有特殊要求，但系统会在提交时对内容进行JSON格式校验，避免错误格式引起配置异常；
- 配置文件最大支持64KB，编辑框右上角将实时显示当前文件的大小，超过64KB的配置文件无法提交。

e. 编辑完成配置信息后，需要单击更新，此时将生成正式的配置文件，允许设备主动请求更新该配置信息。



f. 配置文件提交后，云端不会立即向设备推送更新，需要单击批量更新，此时系统会向该产品下的所有设备批量推送配置文件更新。

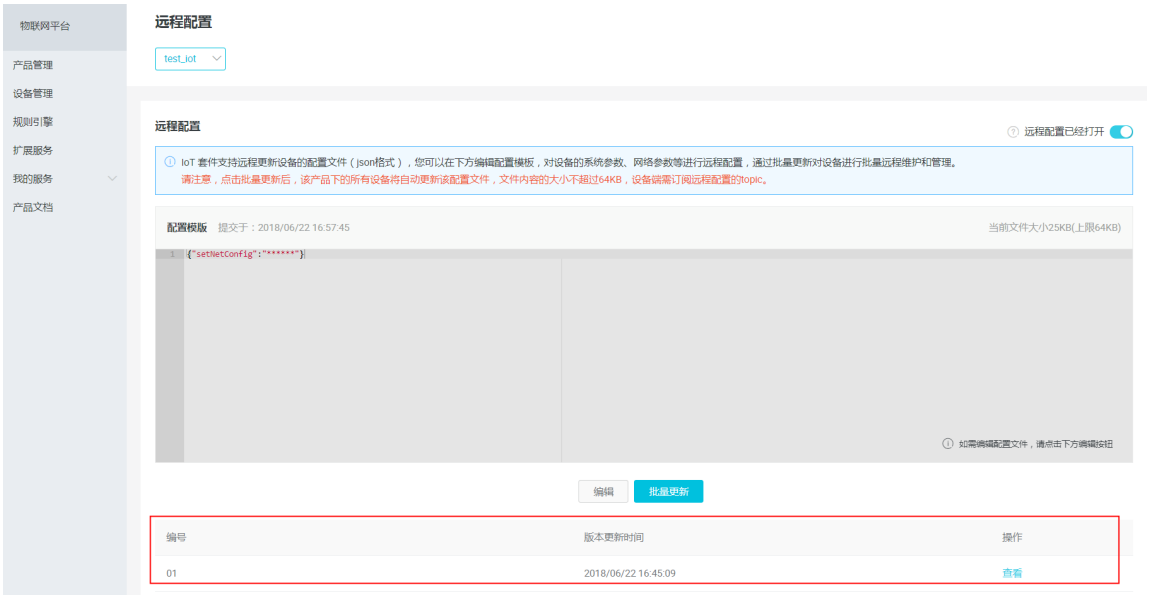


说明：

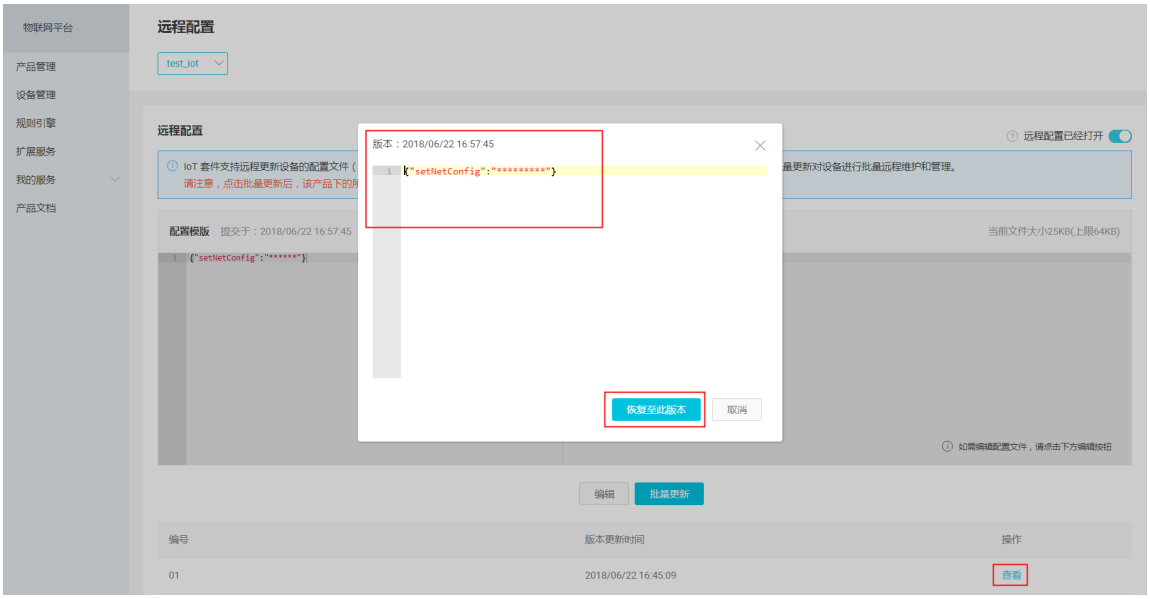
- 批量更新一小时内仅允许操作一次 ，请勿频繁操作；
- 如果您希望停止批量推送更新，单击远程配置开关，关闭远程配置，系统将停止所有更新推送，并且拒绝设备主动请求更新。

g. 可查看远程配置修改记录。

远程配置默认保存最近5次的配置修改记录，每次编辑并提交后，上一次配置将显示在版本记录列表中，您可以查看版本更新时间和配置内容，方便追溯。



在列表中单击查看，将显示该版本的配置内容，单击恢复至此版本，自动将所选版本中的内容恢复至编辑区中，您可以直接编辑修改并更新。



3. 设备端接收云端下发的配置信息后，自动更新配置。

场景二

在一些场景中，设备需要主动查询配置信息，可按如下步骤进行远程配置。

1. 设备端订阅topic(/sys/${productKey}/${deviceName}/thing/config/get_reply)。
2. 在IoT控制台中开启远程配置。详细步骤请参见2。

3. 设备端用户使用接口**linkkit_invoke_cota_get_config**来触发云端远程配置的请求。
4. 设备通过topic(/sys/${productKey}/${deviceName}/thing/config/get)，主动查询最新的配置信息。
5. 接收到设备的请求后云端会返回最新的配置信息到设备中。
6. 设备端用户在回调函数**cota_callback**中去处理远程配置下发的配置文件。

4.3 三维数据可视化

三维数据可视化服务通过空间建模展示设备实时状态，方便您查看设备状态并进行管理。

三维数据可视化主要功能包括：

- 支持拖拽建模
- 支持IoT设备联动
- 支持单空间搜索

目前仅支持高级版产品使用三维数据可视化服务。

开通服务

三维数据可视化为扩展服务，需开通后方能使用。

登录物联网平台的控制台，选择**扩展服务 > 三维数据可视化**，在服务详情页面单击**开通服务**。

创建场景

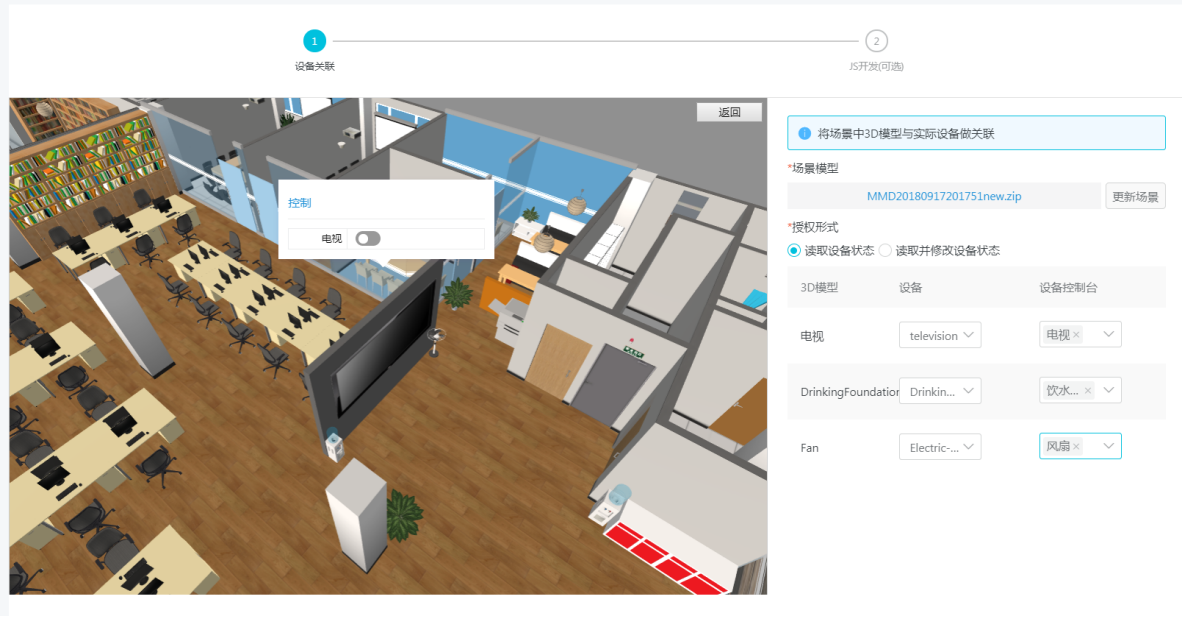
1. 登录物联网平台控制台。
2. 单击**我的服务 > 三维数据可视化**。
3. 在**我的场景**下，单击**“+”**号图标。
4. 根据参数说明设置场景参数，并单击**确定**。

参数	说明
场景名称	设置您的场景名称。
目标产品或设备组	选择部署场景的目标产品或设备组。 目标产品或设备组中包含的设备，与您使用的场景模型关联。创建设备组的操作请参见设备分组。
场景模型	上传场景模型。 **说明：**

参数	说明
	您可以下载IoT提供的模型编辑器，参考搭建工具手册，制作您自己的场景模型。

5. 配置设备关联参数。

下图以智能家居为例，展示设备关联内容。



参数	说明
场景模型	您可以更新已上传的场景模型。
授权形式	授权形式，可分为**读取设备状态**与**读取并修改设备状态**。 • 选择**读取设备状态**表示您授权第三方读取您的设备状态，展示在页面上。 • 选择**读取并修改设备状态**表示您授权第三方读取您的设备状态，并且操作您的设备来改变设备状态，主要用于对设备的远程控制。
3D模型、设备、设备控制台	将您3D场景模型中的设备与IoT平台创建的设备关联起来，并将设备属性展示出来。 • 3D模型：模型中的设备 • 设备：IoT控制台创建的设备 • 设备控制台：IoT控制台创建设备时，设置的设备属性。 **说明：** • 设备属性具体信息请参考新增物模型和导入物模型。

参数	说明
	• 设备控制台处也支持展示设备的位置信息，位置信息需要通过设备标签来配置。

6. 单击保存，可保存您设置的模型与设备的关联关系。
7. 单击下一步，对已创建好的场景模型进行JS补充，具体操作方法请参见搭建工具手册。
8. 单击保存，可保存您修改的代码。
9. 单击发布 > 确定，将您创建的场景模型发布到三维数据可视化服务中。

5 日志服务

本文主要介绍日志服务的三种类型及日志格式。

使用说明

日志类型分为三类：

- 设备行为分析
- 上行消息分析
- 下行消息分析

物联网平台支持按如下方式搜索日志：

搜索日志方式	说明
DeviceName	设备名称，该产品下设备唯一标识。可以根据DeviceName搜索出该设备的相关日志。
MessageID	消息ID，某条消息在物联网平台里的唯一标识。可以用MessageID追踪到这条消息全链路的流转状况。
状态	日志有两个状态：成功和失败。
时间范围	可以筛选某一时间段打印出的日志。

说明：

- {}是变量，日志根据实际运行打印相应结果
- 日志提供的是英文，不会打印中文
- 一旦出现失败状态的日志内容，非system error的原因都是由于用户使用不当或者产品限制导致的，请仔细排查。

设备行为分析

设备行为主要有设备上线（online），设备下线(offline)的日志。

可按DeviceName，时间范围来筛选日志，操作界面如下图所示：

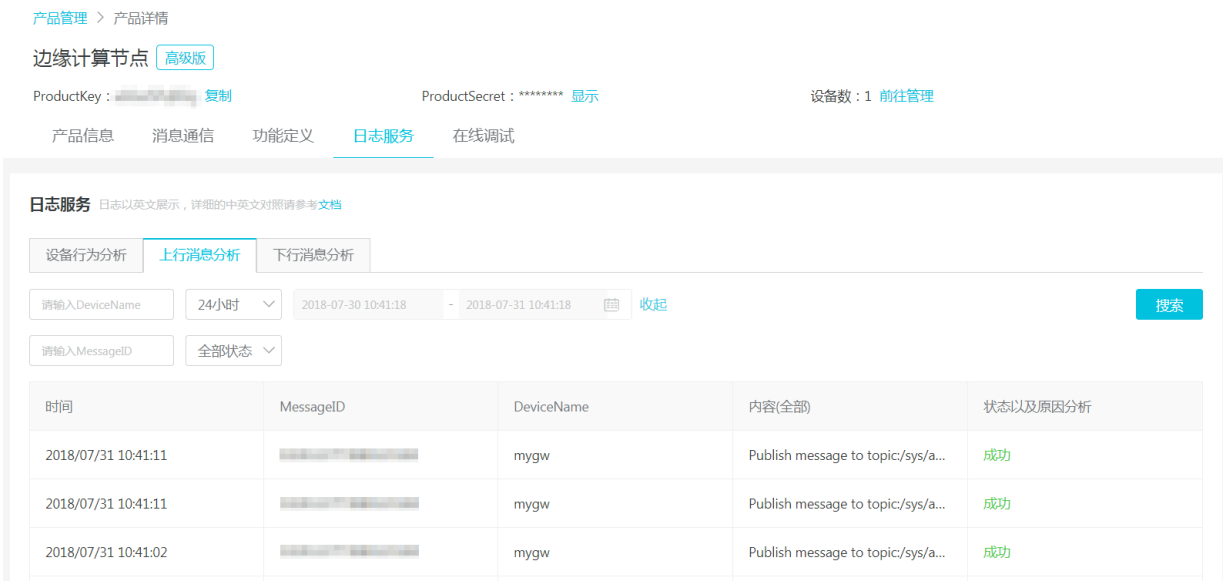
设备离线原因中英文对查表

英文	中文
Kicked by the same device	被相同设备踢下线
Connection reset by peer	TCP连接被对端重置
Connection occurs exception	通信异常,IoT服务端主动关闭连接
Device disconnect	设备主动发送MQTT断开连接请求
Keepalive timeout	心跳超时，IoT服务端关闭连接

上行消息分析

上行消息主要包括设备发送消息到topic，消息流转到规则引擎，规则引擎转发到云产品的日志。

可按DeviceName、MessageID、状态、时间范围来筛选日志，操作界面如下图所示：



上行消息中英文对查表

说明：

其中包括context（打印的英文日志+中文注释），error reason（打印的英文日志），失败的原因（中文注释）。

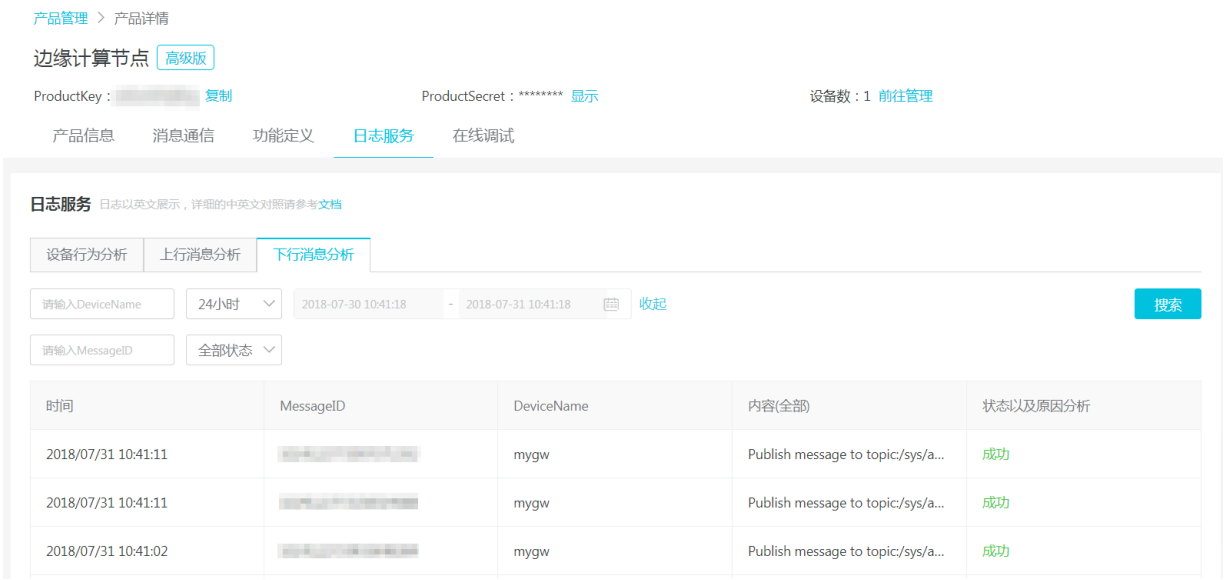
context	error reason	失败原因
Device publish message to topic:{},QoS={},protocolMessageId:{}设备发送消息到topic:{},QoS={},protocolMessageId:{}	Rate limit:{maxQps},current qps:{}	限流{最大流量},当前qps:{}
	No authorization	未授权
	System error	系统错误
send message to RuleEngine，topic:{} protocolMessageId:{}IoT Hub发送消息给规则引擎，topic:{} protocolMessageId:{}	{eg，too many requests}	失败信息，例如太多请求被限流导致失败
	System error	系统错误
Transmit data to DataHub, project:{},topic:{},from IoT topic:{}规则引擎发送数据到Datahub,project:{}，topic:{},来自IoT的topic:{}	DataHub Schema:{} is invalid!	DataHub Schema:{}无效，类型不匹配
	DataHub IllegalArgumentException:{}	Datahub 参数异常:{}
	Write record to dataHub occurs error! errors:[code:{},message:{}]	写数据到datahub出错，错误:[code:{},message{}]
	Datahub ServiceException:{}	Datahub服务异常:{}

	System error	系统错误
Transmit data to MNS,queue:{},theme:{},from IoT topic:{}发送数据到MNS，queue:{},topic:{},来自IoT的topic:{}	MNS IllegalArgumentException:{}	MNS参数异常:{}
	MNS ServiceException:{}	MNS服务异常:{}
	MNS ClientException:{}	MNS客户端异常:{}
	System error	系统错误
Transmit data to MQ,topic:{},from IoT topic:{}规则引擎发送数据到MQ,topic:{},来自IoT的topic:{}	MQ IllegalArgumentException:{}	MQ参数异常:{}
	MQ ClientException:{}	MQ客户端异常:{}
	System error	系统错误
Transmit data to TableStore,instance:{},tableName:{},from IoT topic:{}规则引擎发送数据到TableStore，实例名:{},表名:{},来自IoT的topic:{}	TableStore IllegalArgumentException:{}	TableStore参数异常:{}
	TableStore ServiceException:{}	TableStore服务异常:{}
	TableStore ClientException:{}	TableStore客户端异常:{}
	System error	系统错误
Transmit data to RDS,instance:{},databaseName:{},tableName:{},from IoT topic:{}规则引擎发送数据到RDS,实例名:{},数据库名:{},表名:{}	RDS IllegalArgumentException:{}	RDS参数异常:{}
	RDS CannotGetConnectionException:{}	RDS无法连接:{}
	RDS SQLException:{}	RDS SQL语句异常
	System error	系统错误
Republish topic, from topic:{} to target topic:{}规则引擎转发topic,从topic:{}到目标topic:{}	System error	系统错误
RuleEngine receive message from IoT topic:{}规则引擎接收消息，来自IoT的topic:{}	Rate limit:{maxQps},current qps:{}	限流{最大流量},当前qps:{}
	System error	系统错误
Check payload, payload:{}检测payload,payload:{}	Payload is not json	Payload的json格式不合法

下行消息分析

下行消息主要是云端发送消息到设备的日志。

可按DeviceName、MessageID、执行状态、时间范围来筛选日志，操作界面如下图所示：



下行消息中英文对查表

说明：

其中包括context（打印的英文日志+中文注释），error reason（打印的英文日志），失败的原因（中文注释）。

context	error reason	失败原因
Publish message to topic:{}, protocolMessageId:{}推送消息给topic:{},protocolMessageId:{}	No authorization	未授权
Publish message to device, QoS={}IoT Hub发送消息给设备	IoT hub cannot publish messages	因为服务端没有得到设备的puback，会一直发消息，直到超过50条的阈值然后IoT Hub就会无法发送消息
	Device cannot receive messages	设备端接受消息的通道阻塞，可能由于网络慢，或者设备端消息能力不足导致了服务端发送消息失败
	Rate limit:{maxQps},current qps:{}	限流{最大流量},当前qps:{}
Publish RRPC message to deviceIoT发送RRPC消息给设备	IoT hub cannot publish messages	设备端一直没有回复response，而且服务端一直发送消息超出阈值导致发送失败

	Response timeout	设备响应超时
	System error	系统错误
RRPC finishedRRPC结束	{e.g rrpcCode}	错误信息，会打出相应的RRPCCode,比如UNKNOW,TIMEOUT,OFFLINE,HALFCONN
Publish offline message to device IoT Hub发送离线消息给设备	Device cannot receive messages	设备端接受消息的通道阻塞，可能由于网络慢，或者设备端消息能力不足导致了服务端发送消息失败