阿里云物联网套件

控制台使用手册

为了无法计算的价值 | 阿里云

控制台使用手册

产品管理

概述

物联网套件为开发者提供基础版和高级版两种规格选择，基础版主要提供连接通道能力，高级版在连接通道能力上提供了更为完整的设备功能模型（物模型）、协议解析、在线调试、数据存储等能力。本文档主要介绍产品管理（高级版）的控制台使用指南，基础版的使用指南请参考文档。

开通物联网套件

开通产品，您可以根据实际需要，在创建产品时选择相应的版本规格进行创建。更多关于版本间的对比，请参考产品规格说明。

高级版接入引导

设备通过高级版接入的流程如下：

1. 开通物联网套件；
2. 创建产品；
3. 定义产品功能，包括属性、服务、事件；
4. 配置数据解析脚本（可选，针对数据格式为透传/自定义的产品）；
5. 添加设备；
6. 在线调试；
7. 管理设备；

常见问题

1. 高级版是否提供设备端的SDK，与基础版的SDK区别是什么？

设备端SDK中支持基础版和高级版的接入，对于高级版，SDK中已经实现了设备消息通信的Topic，并将物模型封装为相应的上下行接口，可自动完成数据转换，您无需理解ALink协议即可完成设备的数据上报和指令接收。

2. 高级版支持哪些连接协议，MQTT的Topic与基础版有何区别？

基础版支持的连接协议，在高级版中同样适用，包括HTTP、MQTT和CoAP协议，在Topic上，高级版不支持自定义Topic类，设备需采用系统默认的Topic进行消息通信。

3. 为何使用高级版需要定义产品功能？

功能定义本质上是对物的一种数据建模，高级版中将根据产品功能定义生成对应的物的描述语言TSL（Thing Specification Language，JSON格式）,用于接收设备的消息并进行相应的存储。

4. 使用高级版是否还需要另外购买数据库进行数据存储？

高级版默认包含数据存储服务，将保存设备上报的所有原始数据（JSON格式），您可以根据实际的业务需求，选择通过服务端API查询设备历史数据，也可以通过规则引擎将数据流转到您自己的数据库中存储。

5. 高级版是否支持规则引擎流转？

支持，规则引擎使用与基础版相同。

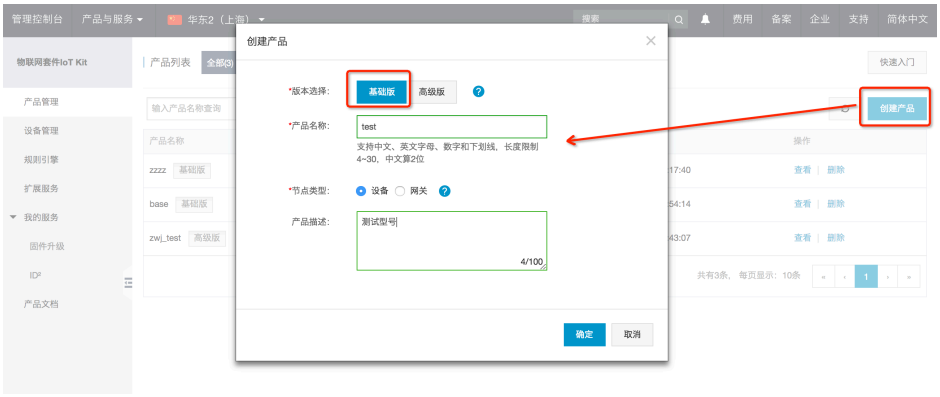
6. 高级版是否支持固件升级？

支持，固件升级使用与基础版相同。

创建产品

创建产品

产品相当于某一类设备的集合，用户可以根据产品管理其设备等。进入产品管理页面，点击创建产品，在弹窗



中选择创建产品。

选择版本进行创建，创建后该产品即可使用该版本所提供的各项服务。

通用

- 产品名称：对产品命名，例如可以填写产品型号，产品名称在账号内保持唯一；
- 节点类型：设备和网关
  - 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；

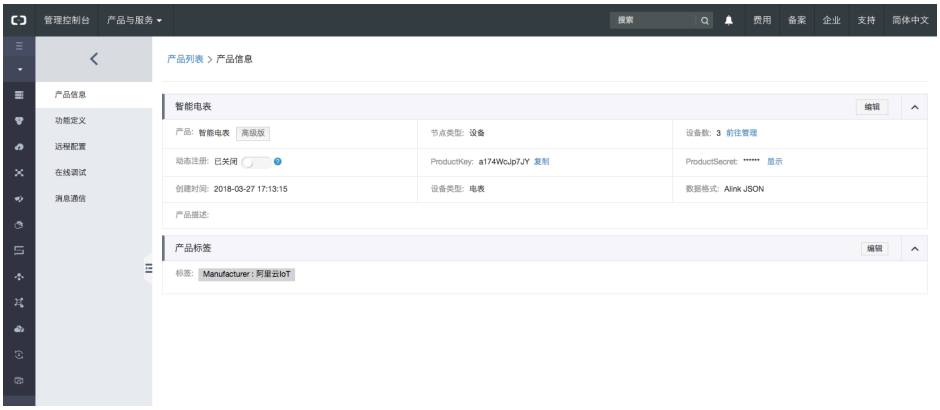
高级版特有

- 设备类型：设备类型是一组预定义的标准功能模板，例如我们为“智能电表”预定义了“用电量”、“电压”、“电流”和“总累计量”等标准功能，选择“智能电表”后，将自动为您创建好以上标准功能，您可以在标准功能模板的基础上编辑修改，也可以添加更多自定义功能，如果设备类型选择“无”，将不会创建任何标准功能，您可以完全自定义该产品的功能；
- 数据格式：设备上下行的数据格式，支持选择Alink JSON和透传/自定义格式；
  - Alink JSON 是套件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；
  - 如果您希望使用自定义的串口数据格式，可以选择“透传/自定义格式”，并在云端配置数据解析脚本，将透传/自定义格式的数据转换为Alink JSON进行解析，请参考文档数据解析脚本。

正确填写以上信息后，点击确定，完成产品的创建。

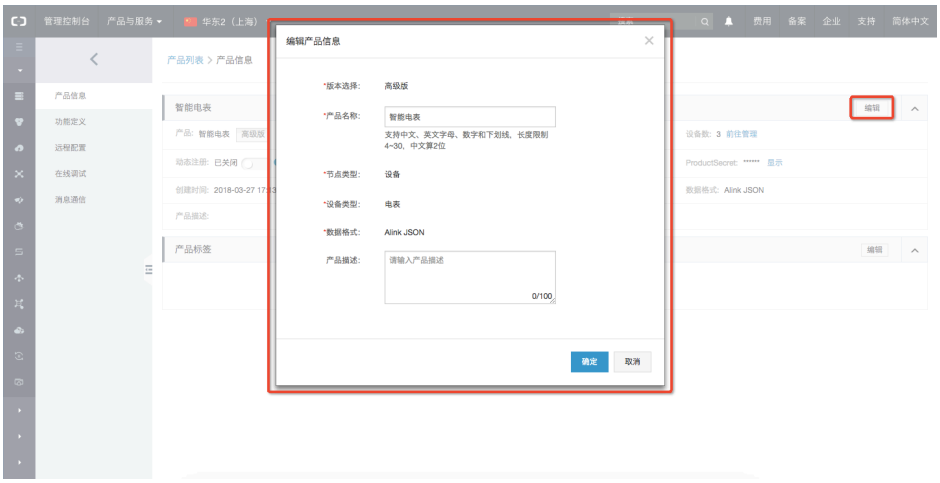
查看产品信息

进入产品管理页面，在产品列表中，选择所要查看的产品，点击查看，进入产品详情页面查看产品信息。



编辑产品信息

在产品信息页面，点击编辑，在弹窗中编辑产品的基本信息。



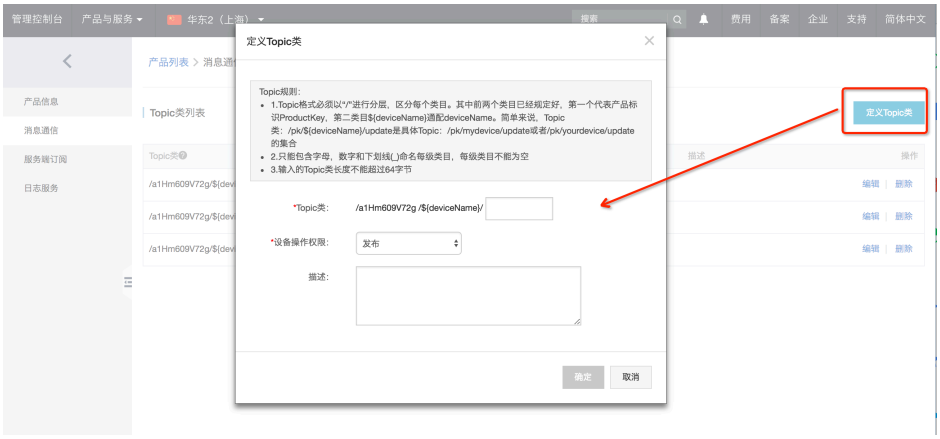
消息通信

用户在创建产品之后，可以在消息通信页面里配置或者查看Topic让设备使用Pub/Sub这种通信方式，详情请参考文档通信方式。

- 基础版的产品，Topic类可以自定义，方便客户灵活使用多种Topic通信。
- 高级版为了简便客户使用，自定义一些系统Topic，以/sys开头，客户只要基于这些系统的Topic即可使用高级版的所有服务。

基础版

用户可以在这里定义Topic类，Topic类请参考文档Topic。



Topic类的特性：

- Topic类是一类Topic的集合，举个例子，Topic类：/pk/${deviceName}/update是具体Topic：/pk/device1/update或者/pk/device2/update的集合。
- Topic类格式必须以“/”进行分层，区分每个类目。其中前两个类目已经规定好，第一个代表产品标识ProductKey，第二类目${deviceName}通配deviceName
- 其余类目命名只能包含字母，数字和下划线(_)命名每级类目，每级类目不能为空
- 设备操作权限：发布就意味着设备可以往这个Topic发布消息，订阅就意味着设备可以从这个Topic订阅消息。

注意：Topic类不能用于通信，pub/sub是基于具体的Topic通信。举个例子，用户不能使用/pk/${deviceName}/update进行通信，只能使用/pk/device1/update或者/pk/device2/update通信。

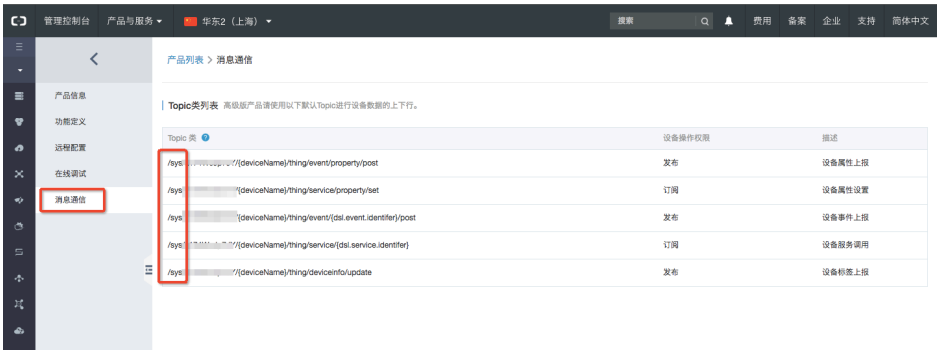
定义好之后该产品下面的所有设备都会拥有对应的Topic

高级版

根据高级版产品的数据格式，Topic类分为Alink Topic和透传Topic。在设备端开发中，您需要Pub或者Sub相关的Topic，实现高级版设备的消息通信，请参考文档设备模型开发。

Alink Topic 类

设备采用Alink JSON进行消息通信时，Topic类包括：



- 设备属性上报Topic：

```
/sys/{ProductKey}/{deviceName}/thing/event/property/post
```

- 设备属性设置Topic：

```
/sys/{ProductKey}/{deviceName}/thing/service/property/set
```

- 设备事件上报Topic：

```
/sys/{ProductKey}/{deviceName}/thing/event/{dsl.event.identifer}/post
```

- 设备服务调用Topic：

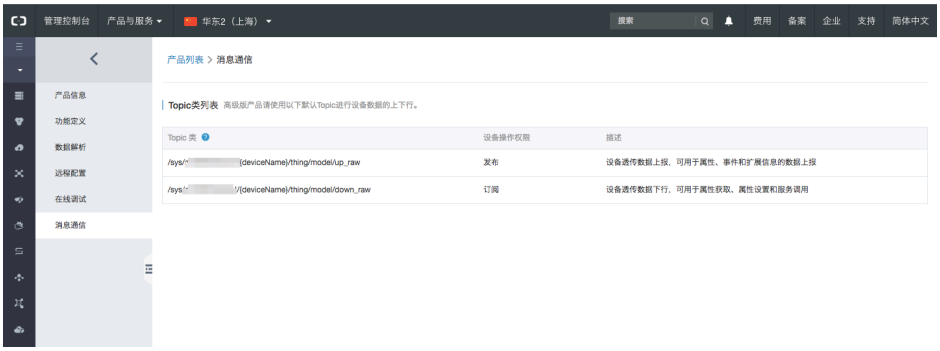
```
/sys/{ProductKey}/{deviceName}/thing/service/{dsl.service.identifer}
```

- 设备标签上报Topic：

```
/sys/{ProductKey}/{deviceName}/thing/deviceinfo/update
```

透传 Topic 类

设备采用透传/自定义格式进行消息通信时，Topic类包括：



- 透传数据上行Topic：

```
/sys/{ProductKey}/{deviceName}/thing/model/up_raw
```

透传数据下行Topic：

```
/sys/{ProductKey}/{deviceName}/thing/model/down_raw
```

功能定义

物模型

物联网（Internet of Things）中的“物”（Thing）可以是各类传感器，也可以是由传感器组成的设备，甚至可以是任一种物理实体，比如“楼宇”、“房间”、“汽车”或者“工厂”。物模型实际上是对物理空间中的实体进行数字化，在云端构建其数据模型。

物联网套件中，对产品的功能定义就是将设备（Device）抽象为“物”，通过物模型对设备是什么、能做什么、可以对外提供哪些服务进行描述。一般情况下，产品的功能包括至少一个“属性”（Property）或“服务”（Service）或“事件”（Event）。完成产品的功能定义后，系统将自动生成该产品的“物模型”，以JSON格式表述，称之为 TSL（即 Thing Specification Language）。

功能类型说明

功能类型说明如下：

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，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，包含输入参数和输出参数，相比于下发指令设置属性值，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，相比于属性状态，事件一般而言包含设备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/告警时的温度等，事件可以被订阅和推送。

标准功能模板

创建高级版产品时，您可以选择设备类型，以自动创建该类型设备的标准功能模板。例如选择设备类型为“智能电表”，将为您自动创建电表的标准功能模板：

请注意：

1. 功能定义仅针对使用高级版的产品，基础版产品无需定义产品功能；
2. 创建产品功能后需要设备端实现高级版的相关Topic，包括属性上报、属性设置、事件上报、服务调用、标签信息上报等，参见高级版的消息通信；

新增产品功能

在功能列表右上方点击新增，在弹窗中选择功能类型，新增一个产品功能。

新增属性

选择功能类型为属性，新增一个属性类型的功能。

- 功能名称：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，供您直接选择，您也可以直接创建自定义属性，功能名称一般为中文，如“用电量”，同一个产品下功能名称请勿重复；
- 标识符：即Alink JSON格式中的identifier，作为设备上报该属性数据的Key，如“PowerComsuption”，云端根据该标识符校验是否接收该数据；
- 数据类型：支持选择int32（32位整型）、float（单精度浮点型）、double（双精度浮点型）、enum（枚举型）、bool（布尔型）、text（字符串）、date（时间戳）、struct（JSON对象）、array（数组）；
    - int32/float/double：需定义取值范围和单位符号；
    - enum：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：采用0/1来定义布尔值，如0-关/1-开;
    - text：定义字符串的数据长度，最长支持1024字节；
    - date：格式为string类型的UTC时间戳，支持毫秒；
    - struct：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：需声明数组内元素的数据类型，支持int32、float、double、text，同一个数组需确保元素类型相同，数组长度为不定长，最长不超过128个元素；
- 读写类型：支持选择读写和只读，读写支持get（获取）和set（设置）方法，只读仅支持get（获取）的方法；
- 描述：对该功能进行说明或备注；

新增服务

选择功能类型为服务，新增一个服务类型的功能。

- 功能名称：为该服务命名，一般为中文，支持从标准功能库中搜索标准服务，如“自动灌溉”服务；
- 标识符：该服务的identifier，作为该服务被调用时的Key，如“AutoSprinkle”；
- 调用方式：支持选择异步和同步，异步调用是指云端执行调用后直接返回，不会关心设备的回复消息，如果服务为同步调用，云端会等待设备回复，否则会调用超时。
- 输入参数（可选）：该服务的入参，点击新增参数，在弹窗中添加一个服务入参，您可以直接选择某个属性作为入参，也可以自定义参数，如将“喷灌时间”和“喷灌量”作为“自动喷灌”服务的入参，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（可选）：该服务的出参，点击新增参数，在弹窗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“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；

- 描述：对该功能进行说明或备注；

新增事件

选择功能类型为事件，新增一个事件类型的功能。

- 功能名称：为该事件命名，一般为中文，支持从标准功能库中搜索标准事件，如“故障上报”事件；
- 标识符：该事件的identifier，作为设备上报该事件时的Key，如“ErrorCode”；
- 事件类型：“信息”是设备上报的一般性通知，如完成某项任务等，“告警”和“故障”是设备运行过程中主动上报的突发或异常情况，优先级高，您可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（可选）：该事件的出参，点击新增参数，在弹窗中添加一个事件出参，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“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；

一个事件最多支持定义10个出参；

- 描述：对该功能进行说明或备注；

编辑功能

在功能列表中，选择需要编辑的功能，点击编辑，在弹窗中编辑该功能的相关信息。

删除功能

在功能列表中，选择需要删除的功能，点击删除，在弹窗中确认删除该功能。

查看物模型

完成产品的功能定义后，系统将自动生成该产品的物模型（即TSL），采用JSON格式，您可以根据物模型组装设备上报的数据。

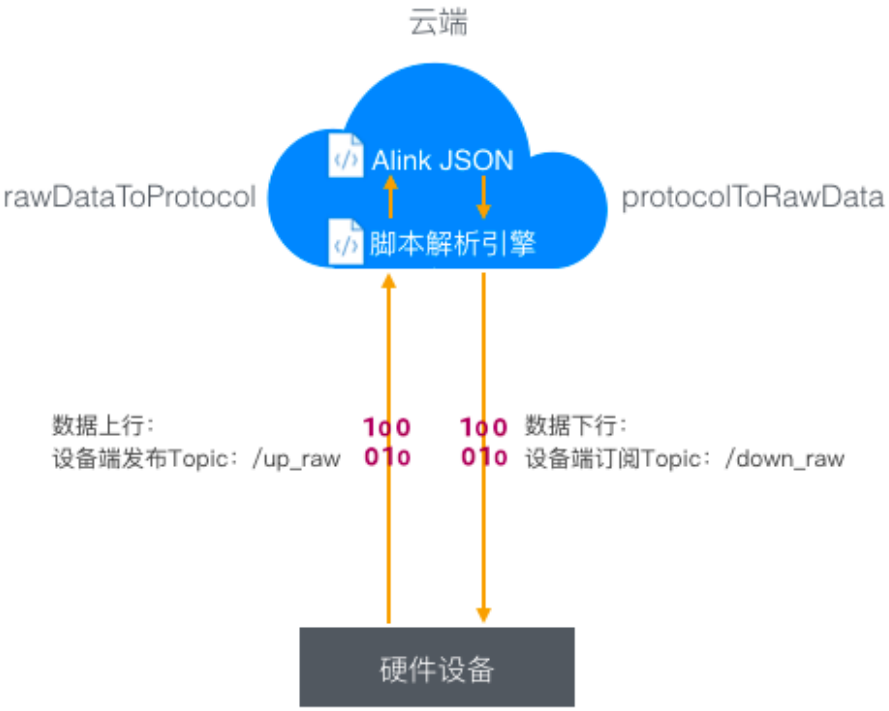
物模型支持导出为JSON文件，点击导出模型文件，选择文件保存路径。请参考文档物模型TSL字段描述说明。

脚本解析

脚本解析说明

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，因此选择通过二进制方式透传到云端，由云端运行转换脚本将透传的数据转换成Alink格式的JSON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JS开发，需要开发者自行开发转换脚本。物联网套件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（js语法）；
- 提交脚本到运行环境，在设备上下行时进行调用；

编辑脚本内容

在线编辑脚本

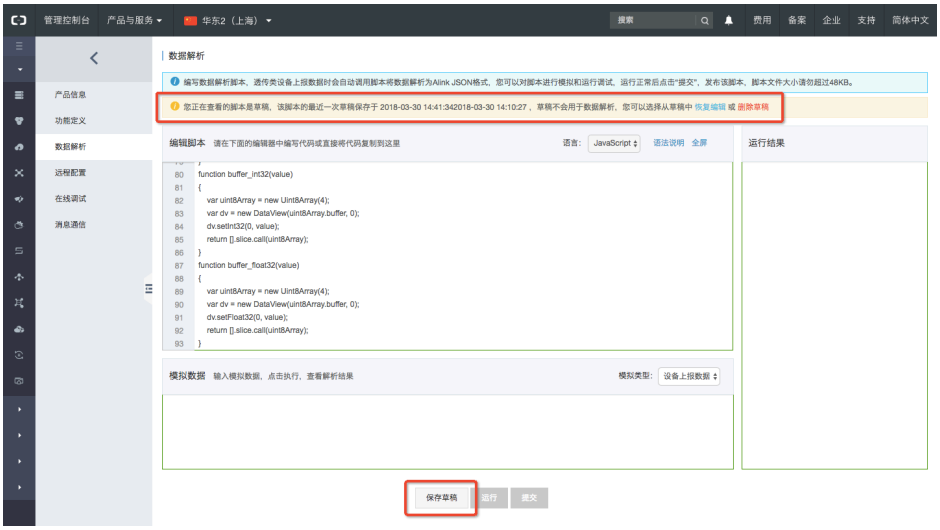
在产品详情页面，点击数据解析进入脚本编辑页面，在下方编辑框中编写数据解析脚本，目前支持编写JavaScript 语言。

全屏编辑

点击全屏按钮，支持全屏查看或编辑脚本，点击退出全屏即可退出当前编辑界面。

保存草稿

点击页面底部的保存草稿按钮，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。



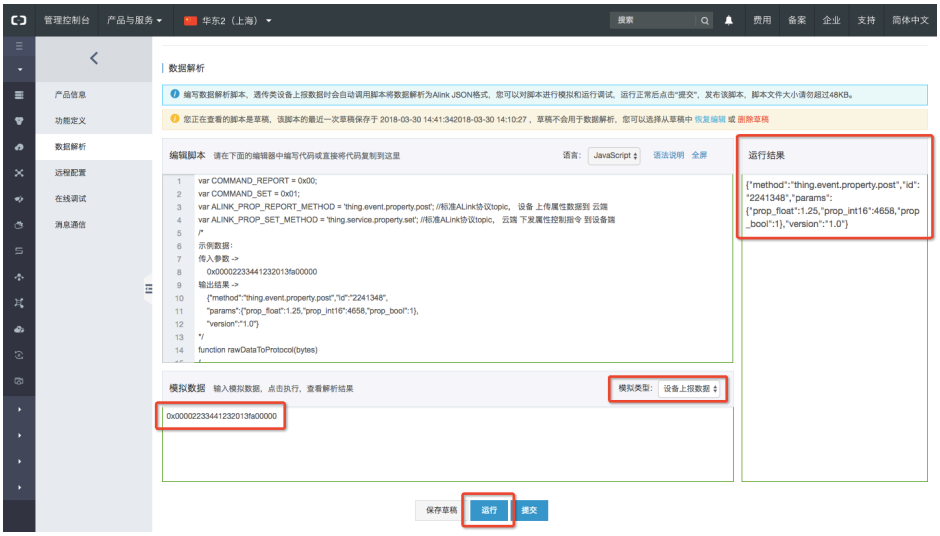
- 草稿不会进入到脚本解析的运行环境中，保存草稿不会影响已经提交的正式脚本；
- 每次保存草稿将覆盖上一次保存的草稿；

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，点击运行按钮，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。

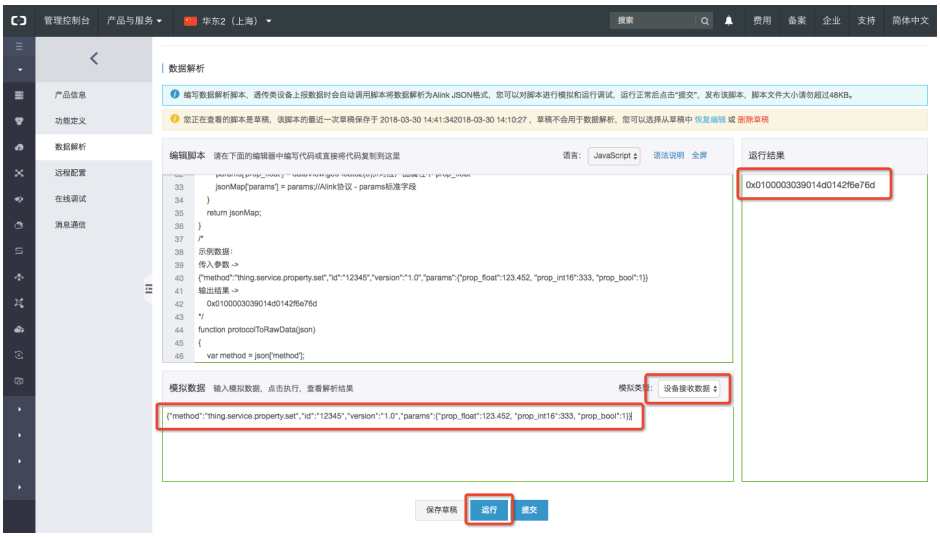
模拟上行数据解析

选择模拟类型为设备上报数据，输入透传二进制数据，点击“运行”按钮，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。



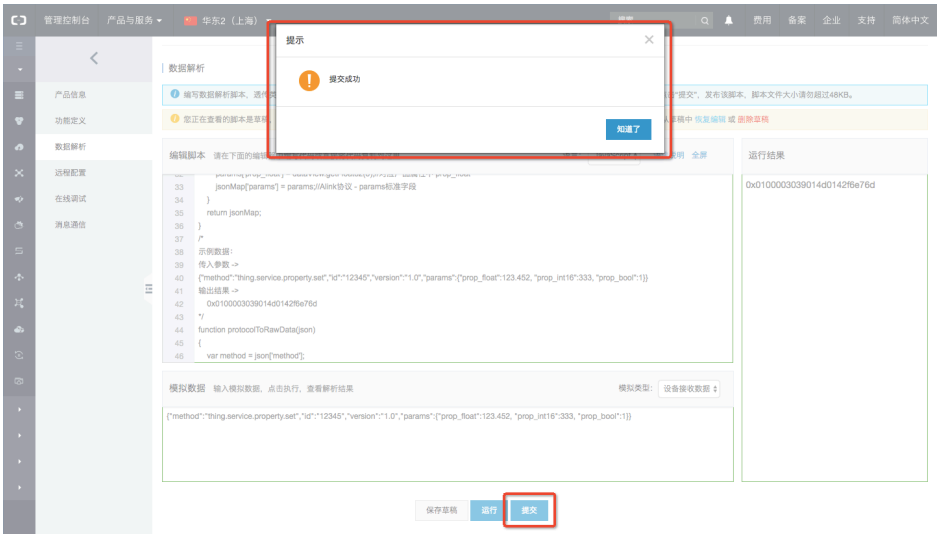
模拟下行数据解析

选择模拟类型为设备接收数据，输入JSON格式数据，点击运行按钮，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。



请注意，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套件：

- Alink协议数据转二进制数据：protocalToRawData
- 二进制数据转Alink协议数据：rawDataToProtocal

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协议数据转二进制数据，方法如下：

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

参数定义：输入参数jsonObj ：符合产品 TSL 定义的 Alink 协议格式数据， 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协议数据，方法如下：

```
//解析设备端发送的二进制数据，并转换为AlinkJson数据返回
function rawDataToProtocal(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协议格式 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

Step1：创建产品并完成功能定义

1. 创建一款高级版产品，数据格式选择为透传/自定义格式；
2. 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：prop_float，数据类型：浮点单精度float
标识符：prop_int16，数据类型：整数型int32
标识符：prop_bool，数据类型：布尔型bool
```

Step2：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

Step3：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协议topic， 设备 上传属性数据到 云端
var ALINK_PROP_SET_METHOD = 'thing.service.property.set'; //标准ALink协议topic， 云端 下发属性控制指令 到设备端

/*
示例数据：
传入参数 ->
0x00002233441232013fa00000
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协议 - 属性上报topic
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);
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);
}
```

Step4：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
```

在本地环境调试脚本

目前以上方式不支持脚本的在线调试，建议开发过程先在本地开发完成后在将脚本拷贝到在线的脚本编辑器中。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();
```

在线调试

概述

在线调试可用于在开发和远程维护中对设备的功能进行调试和验证，仅面向使用高级版接入的设备提供该功能。在线调试主要由实时日志和功能调试两部分组成，支持调试的功能包括：

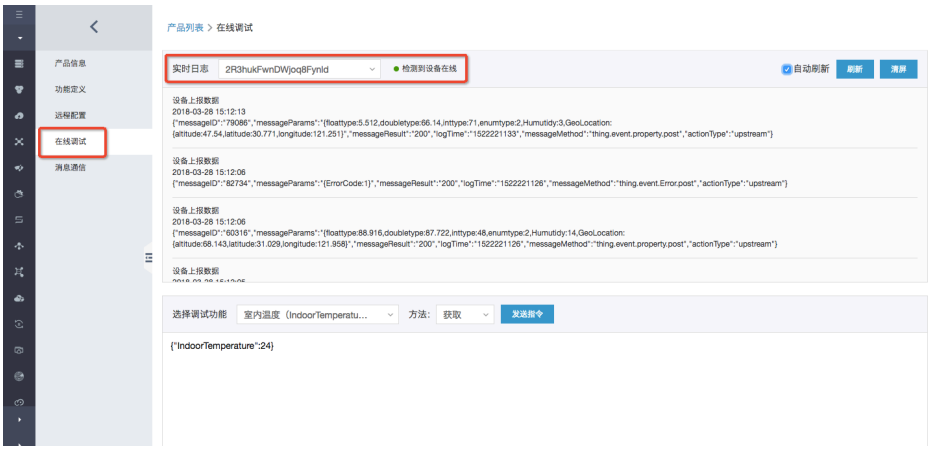
- 获取设备最新一条属性数据；
- 设置属性目标值，并发送给设备；
- 调用设备的服务，并发送给设备；
- 获取设备最新一条事件数据；

前置操作

使用在线调试前，请确保您已经：

1. 创建一款高级版产品；
2. 定义产品的功能，包括属性、服务、事件；
3. 数据格式为透传/自定义格式的产品需配置数据解析脚本，并成功提交；
4. 在该产品下添加至少一个设备；
5. 完成设备端程序开发，设备可正常接入阿里云物联网套件；
6. 待调试设备处于在线状态；

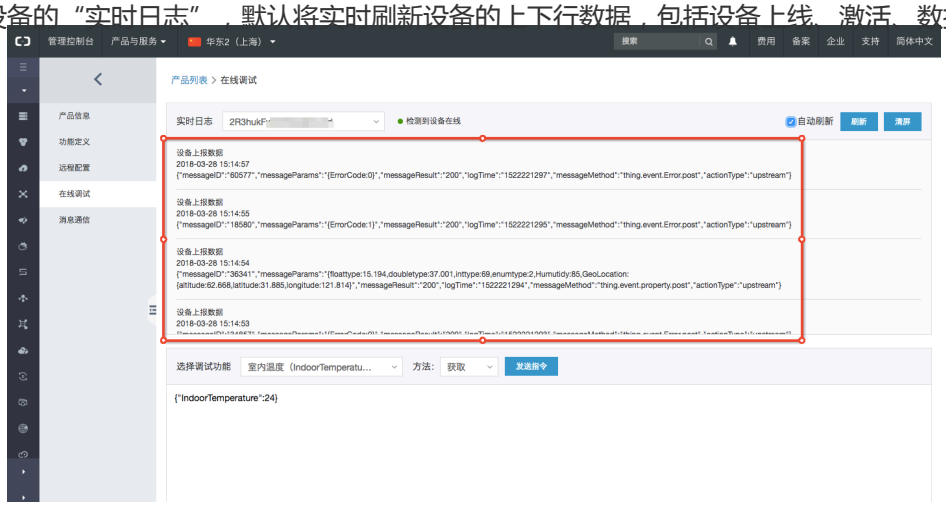
在产品详情页面，点击在线调试，开始进行设备调试。



在线调试仅支持在线设备调试，如果设备处于离线状态将无法进行调试。

实时日志

页面上方为设备的“实时日志”，默认将实时刷新设备的上下行数据，包括设备上线、激活、数据上报、云端下发等日志。



选择调试功能

在线调试当前仅支持对单个功能进行调试，您可以在下方选择一个功能进行在线实时调试，支持分别对设备的

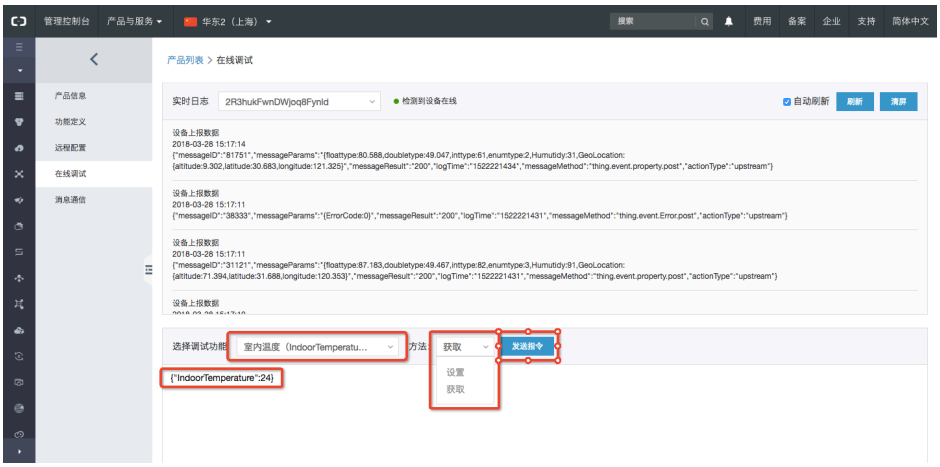
属性、服务和事件进行调试，根据功能类型和读写类型，可以选择“设置”或“获取”方法进行调试。

调试属性

- 获取设备的属性数据：

选择该设备的某个属性类功能，选择方法为获取，点击发送指令，将触发一个 Get 请求下发到设备端，下方编辑区中将以 JSON 格式显示该属性的最新数据值。

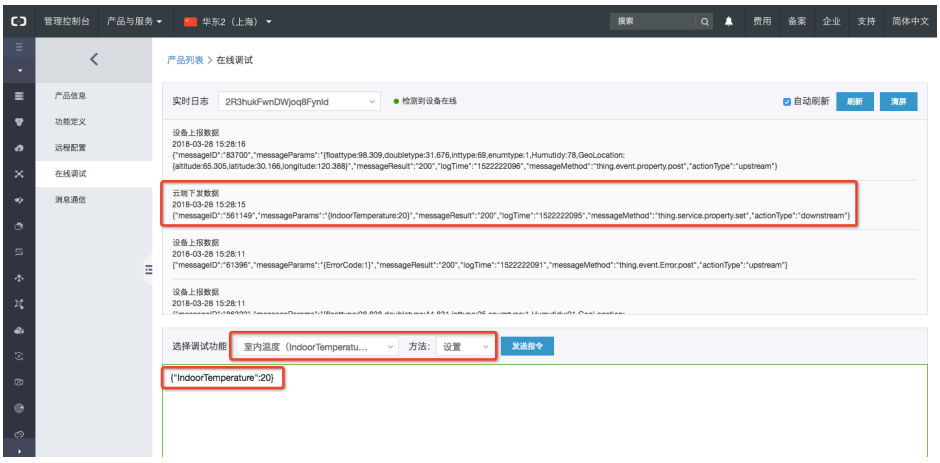
在如下示例中，通过在线调试获取室内温度（IndoorTemperature）的最新一条数据：



- 设置设备的属性值：

选择该设备的某个属性类功能，该属性读写类型需支持读写，选择方法为设置，编辑区将默认以 JSON 格式显示该属性的最新数据，修改为期望的目标值，点击发送指令，平台将下发一个 Set 指令到设备，设备执行该指令，并且在实时日志区域显示本次下发的调试日志。

在如下示例中，通过在线调试设置室内温度（IndoorTemperature）的目标值为20：



请注意，设置设备的属性值时需确保属性值与功能定义中的数据类型和相关定义一致，否则可能会提示错误，无法下发该指令。

调试服务

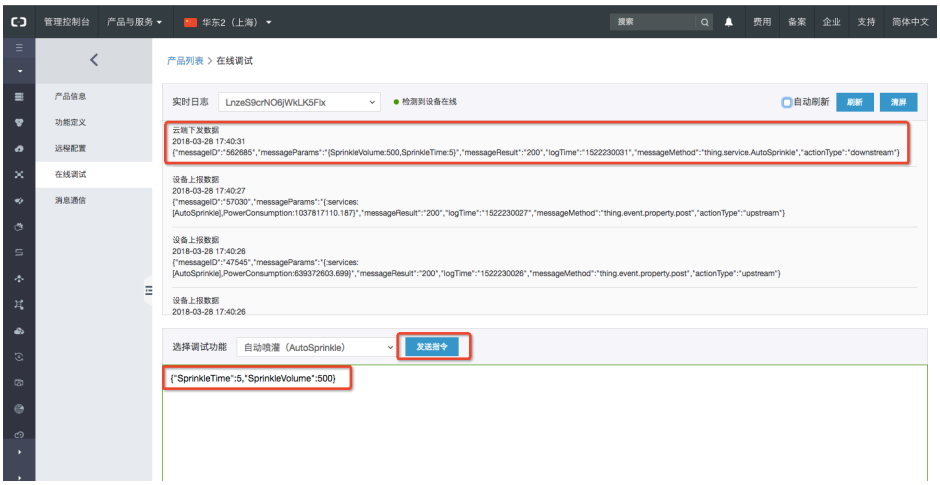
- 调用设备服务

选择该设备的某个服务类功能，编辑区中将自动显示出该服务所定义的入参，您可以构造一个服务的调用，向设备传入指定参数，如果入参为空，将仅显示大括号{}。

在如下示例中，通过在线调试调用设备的服务自动喷灌（AutoSprinkle），其中需要传入两个参数，分别是：

- 喷灌时间（SprinkleTime）：数据类型为int32；
- 灌溉量（SprinkleVolume）：数据类型为float；

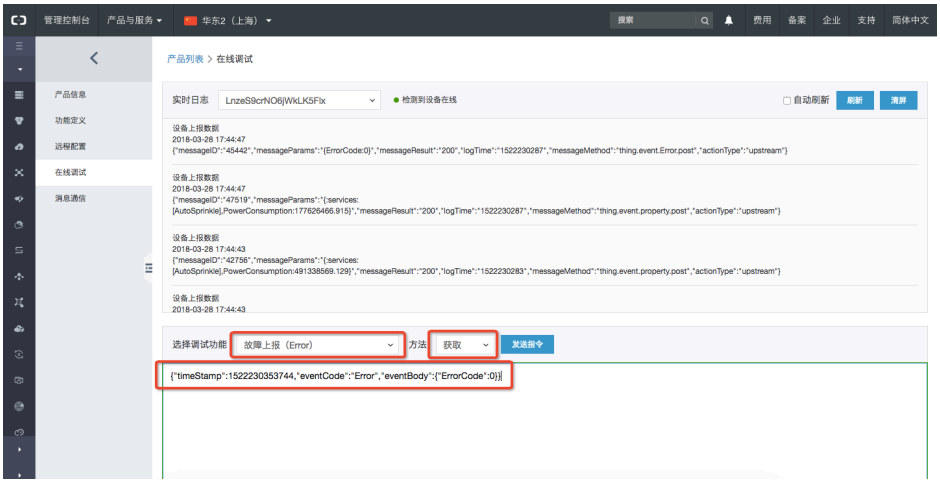
选择该服务，对这两个入参的期望值进行编辑，构造入参为{"SprinkleTime":5,"SprinkleVolume":500}，代表喷灌时间为5分钟，本次灌溉量为500ml，点击发送指令，云端将下发该服务的远程指令，由设备端执行灌溉任务。



调试事件

选择该设备的某个事件类功能，选择方法为获取，点击发送指令，将触发一个 Get 请求下发到设备端，下方编辑区中将以 JSON 格式显示该事件的最新数据值。

在如下示例中，通过在线调试获取故障上报（Error）的最新一条数据：



远程配置

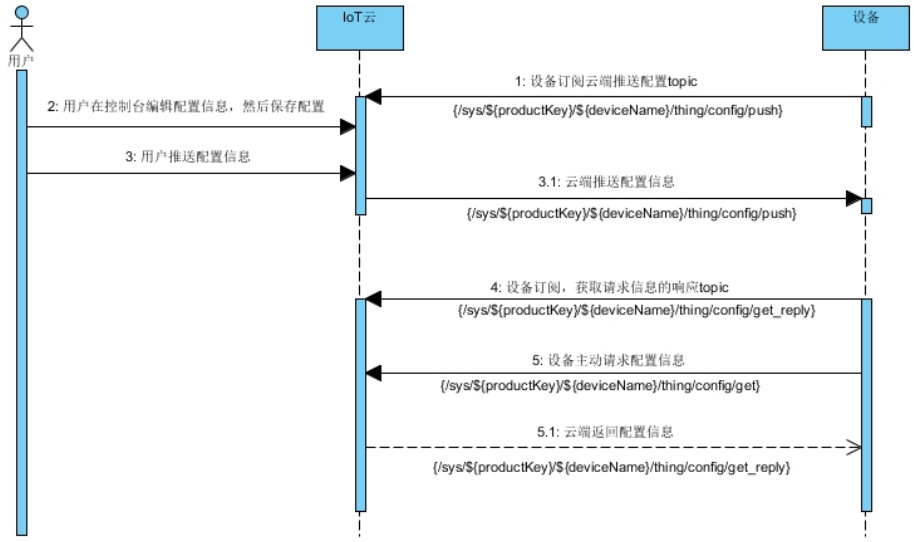
远程配置说明

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，但不可能每次通过固件升级的方式更新配置信息，这将导致固件版本的维护成本大大增加，并且需要设备中断运行以完成更新。

基于以上考虑，物联网套件为用户提供了远程配置更新的能力，设备无需重启或中断运行即可在线完成配置的更新，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



远程配置分为三部分：

- 配置生成，用户在套件控制台编辑并保存配置信息；
- 配置使用，用户在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；

1. 设备上线，订阅配置推送的topic(/sys/${productKey}/${deviceName}/thing/config/push)
2. 用户在套件管理控制台编辑并保存配置，然后推送配置信息，在线的设备可以收到配置信息
3. 在一些场景中，设备需要主动查询配置信息。 设备先订阅 topic(/sys/${productKey}/${deviceName}/thing/config/get_reply)
4. 然后设备通过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设备最新的配置信息。

设备端开发说明

- 参考以下文档，在物联网套件控制台中开启远程配置；
- 设备端SDK中需要定义 FEATURE_SERVICE_OTA_ENABLED = y就可以使用远程配置功能。
- SDK提供接口linkkit_cota_init来初始化COTA，然后使用接口linkkit_invoke_cota_get_config来触发请求云端远程配置。在回调函数cota_callback中去处理远程配置下发的配置文件。

```
    linkkit_cota_init(cota_callback);
linkkit_invoke_cota_get_config("product","file","",NULL);
```

远程配置Topic

远程配置中，设备端需订阅的Topic为：

- 设备接收云端的配置信息推送：

```
/sys/${productKey}/${deviceName}/thing/config/push
```

- 设备主动请求更新配置：

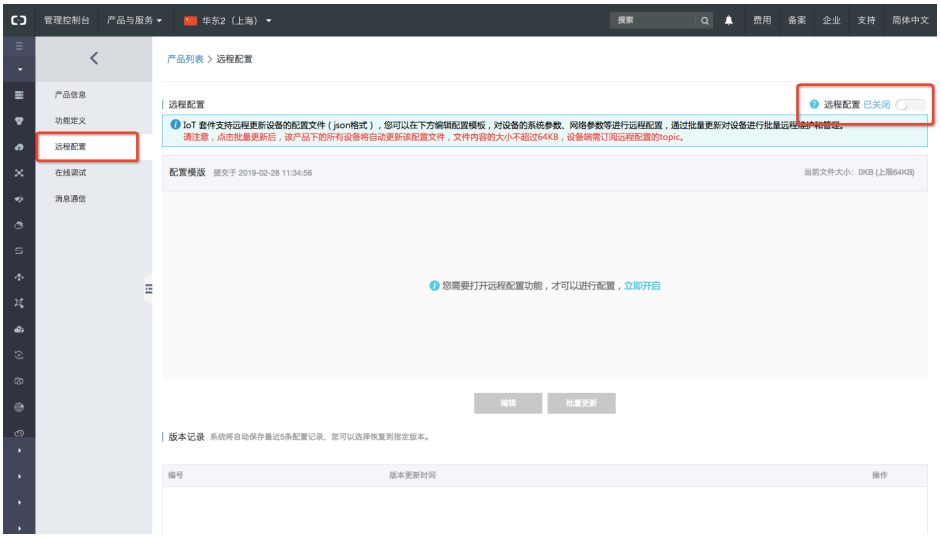
```
/sys/${productKey}/${deviceName}/thing/config/get
```

- 设备主动请求更新配置，接收云端返回的配置信息：

```
/sys/${productKey}/${deviceName}/thing/config/get_reply
```

开启远程配置

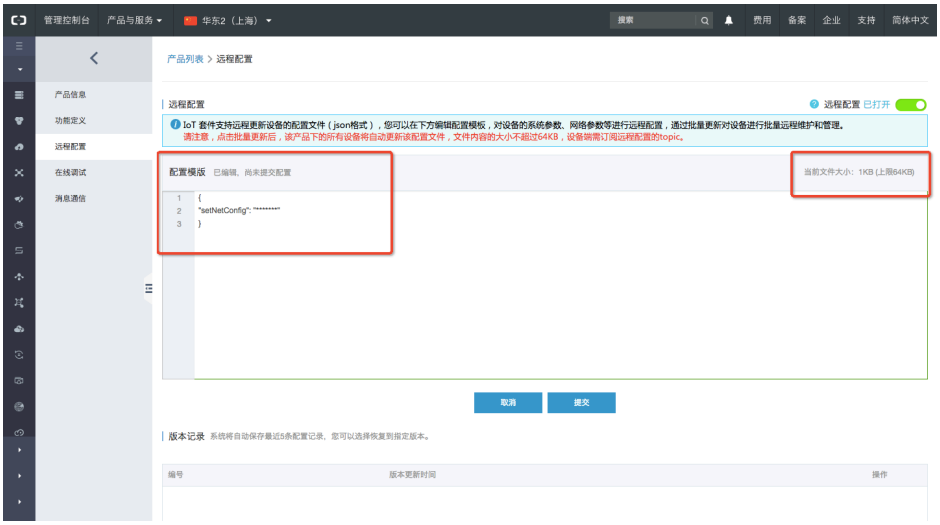
远程配置功能默认处于关闭状态，在产品详情页面，点击远程配置开关，置为打开状态，即可开启远程配置功能，切换为关闭状态，即可关闭远程配置。



编辑配置信息之前必须首先开启远程配置功能。

编辑配置模板

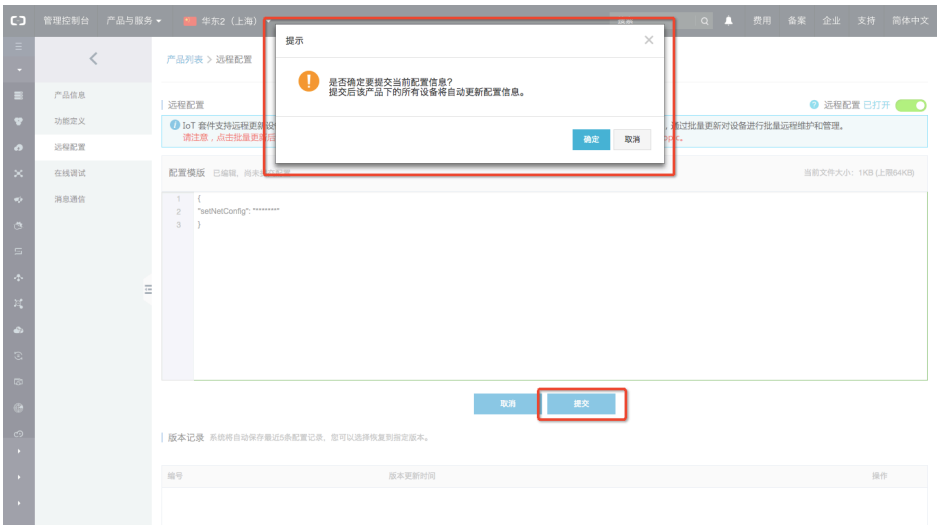
在编辑区直接编写配置信息或拷贝配置信息到编辑区。产品配置模板适用于该产品下的所有设备，目前不支持对单个设备进行配置更新。



- 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；

提交配置文件

编辑完成配置信息后，需要点击提交，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。

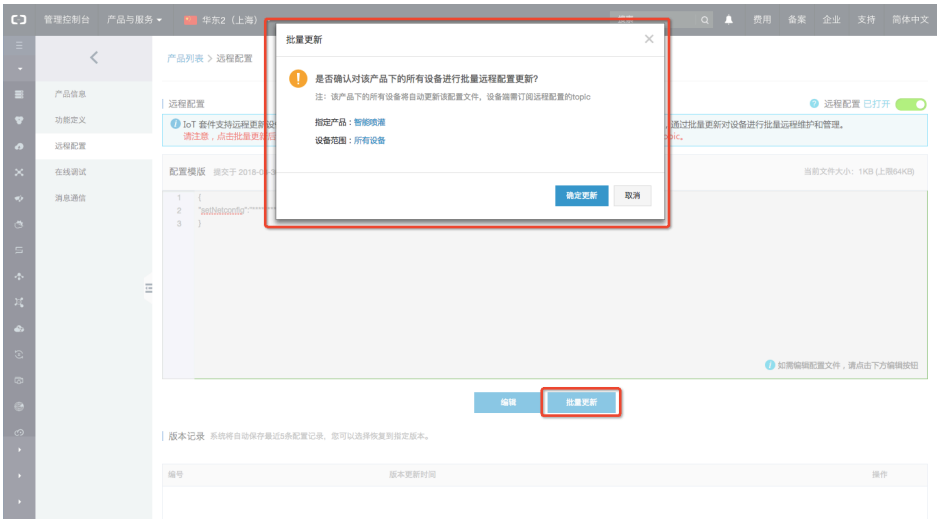


批量更新

配置文件提交后，云端不会立即向设备推送更新，需要点击批量更新，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。

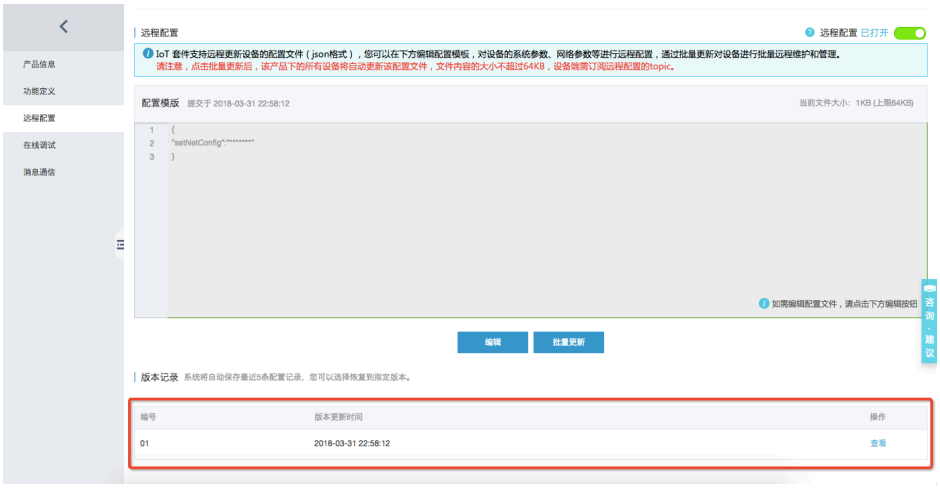
- 请注意，批量更新一小时仅允许操作一次 ，请勿频繁操作；

- 如果您希望停止批量推送更新，可以点击远程配置开关，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；

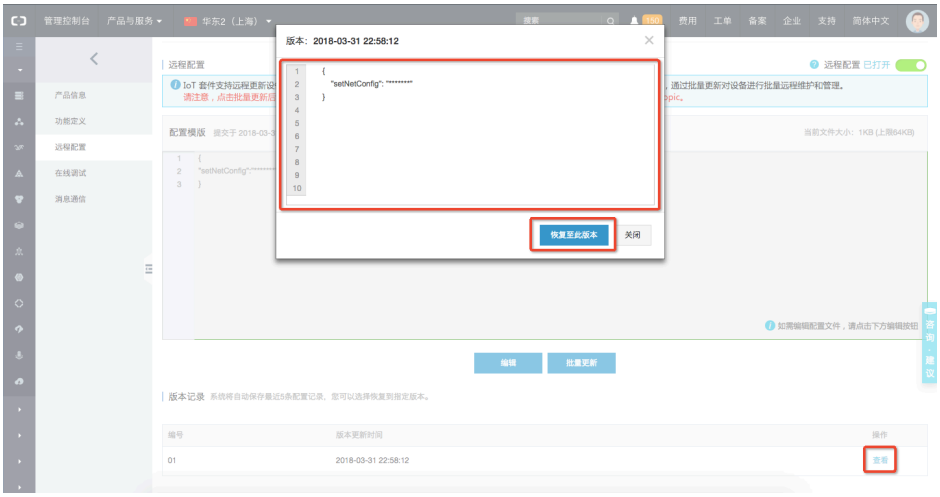


版本记录

远程配置将默认保存最近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。



在列表中点击查看，将显示该版本的配置内容，点击恢复至此版本，将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提交。



日志服务

目前该功能基础版具备，高级版待上线。

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；支持按设备名称（deviceName）,消息ID（messageId）,状态（成功、失败、全部），时间范围来筛选日志，方便查看上下文内容，失败原因等。

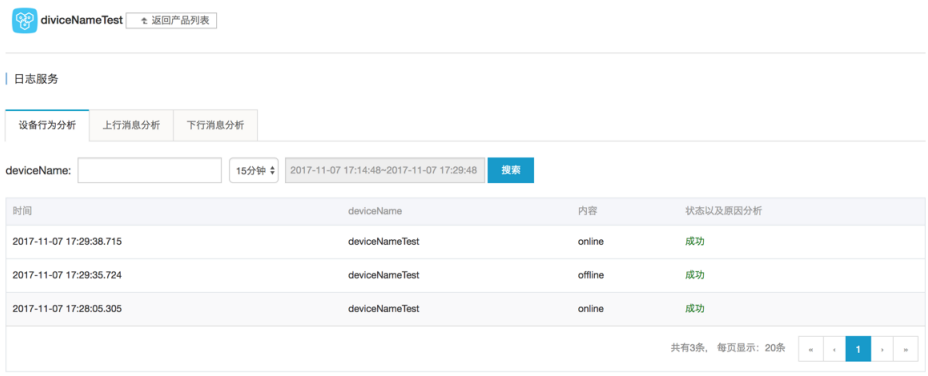
特别注意：

1. deviceName：该产品下设备唯一标识，客户可以根据deviceName搜索出该设备的相关日志。
2. messageId：某条消息在套件里的唯一标识，可以用messageId追踪到这条消息全链路的流转状况。
3. 状态：有两个状态：成功、失败。
4. {}是变量，日志根据实际运行打印相应结果
5. 日志提供的是英文，不会打印中文
6. 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：

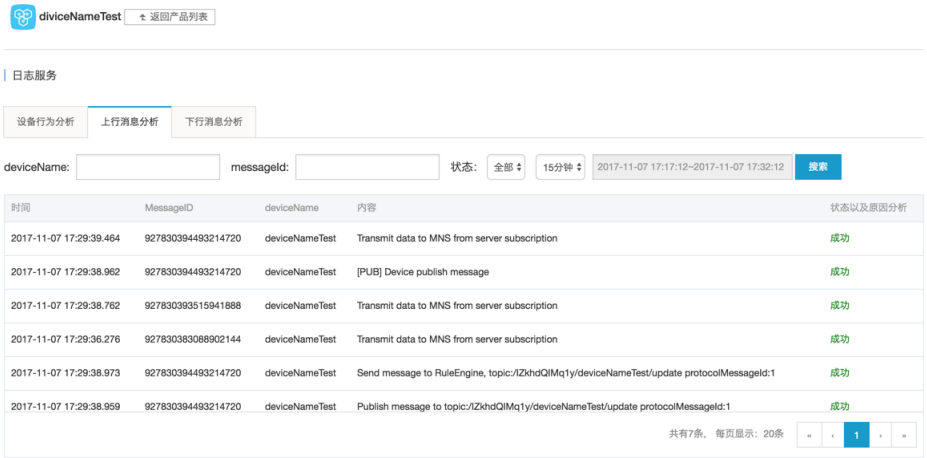


设备离线原因中英文对查表

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，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）

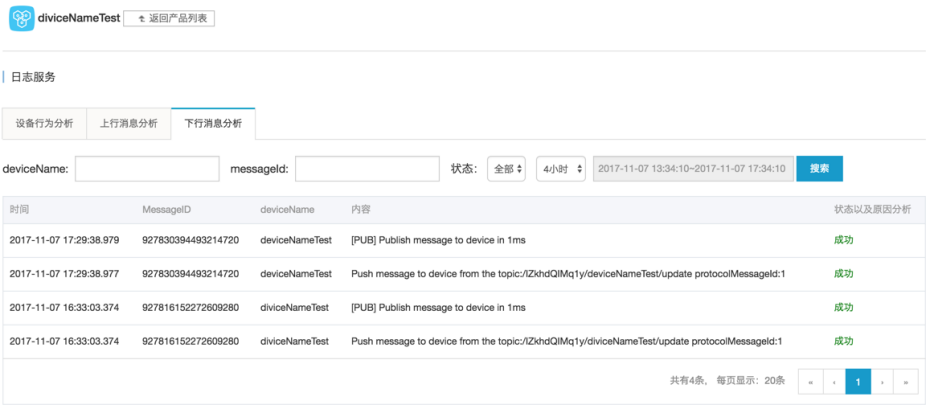
context	error reason	失败原因
Device publish message to topic:{},QoS={},protocolMessageId:{}	Rate limit:{maxQps},current qps:{}	限流{最大流量},当前qps:{}

设备发送消息到topic:{},QoS={},protocolMessageId:{}	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，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 device IoT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 finished RRPC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

服务端订阅

目前该功能只有基础版具备。

用户在创建产品之后，可以在服务端订阅页面里面配置，选择设备消息类型推送到MNS队列中，用户服务端从队列里获得设备数据。这样简化服务端订阅设备数据的流程，让客户的服务端能够简单方便并高可靠的获得设备数据。

配置前提：

- 创建产品
- 开通MNS服务
- 授权IoT写入MNS队列的权限

配置

名词解释：

- 订阅的消息类型:

1. 设备上报消息:指的是产品下所有设备Topic列表中具有发布权限的Topic中的消息，例如产品下面有三个Topic类，其中有/pk/${deviceName}/get:订阅、/pk/${deviceName}/update:发布、/pk/${deviceName}/update/error:发布。那么设备上报消息指的是，/pk/${deviceName}/update和/pk/${deviceName}/update/error对应的所有Topic中的消息。选中后保存，系统会把这些Topic中的消息转发到上面默认创建的MNS队列里
2. 设备状态变化通知:指的是一旦该产品下的设备状态变化时，例如上线，下线，套件产生的消息。选中后保存，系统会推送设备上下线消息到上面默认创建的MNS队列里

- 区域:IoT默认在该区域创建Queue,下面在使用MNS SDK获取Endpoint 需要在MNS控制台选择该区域（region）
- 队列: IoT会自动到MNS华东2下创建aliyun-iot-${productKey}队列，并将选择的消息写入该队列，用户可以通过监听这个队列来获取设备上报的消息。具体可以到MNS控制台查看。
- 角色名称：用户授权IoT访问用户MNS系统的角色，IoT系统根据这个授权角色写入消息到用户的MNS消息队列，否则消息无权写入。

配置保存后IoT会自动在MNS华东2区域下创建aliyun-iot-${productKey}队列，**60s之后**将选择的消息写入到该队列。

特别说明：IoT创建的队列命名规则是aliyun-iot-${productkey}，${productkey}是该产品的productkey。

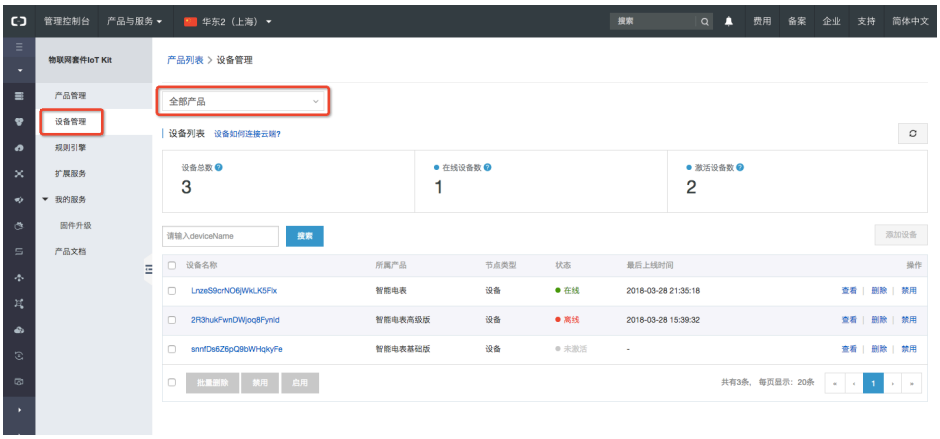
消息写入队列之后，如何从队列中获取数据，请参考文档云端接收设备数据

设备管理

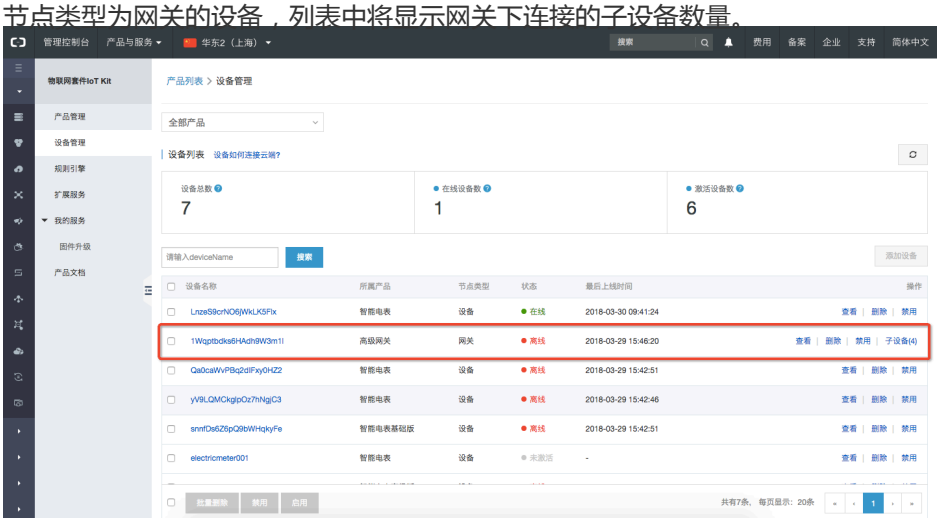
添加设备

设备列表

进入设备管理页面，默认显示您账号下全部产品的所有设备。



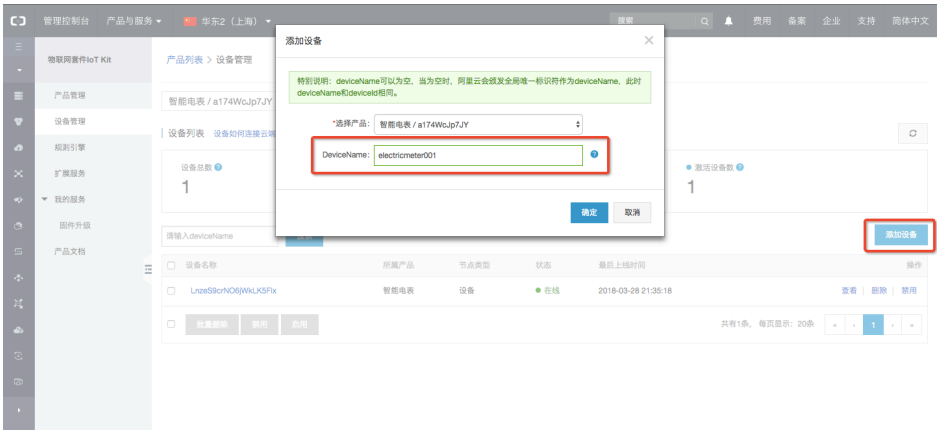
节点类型为网关的设备，列表中将显示网关下连接的子设备数量。



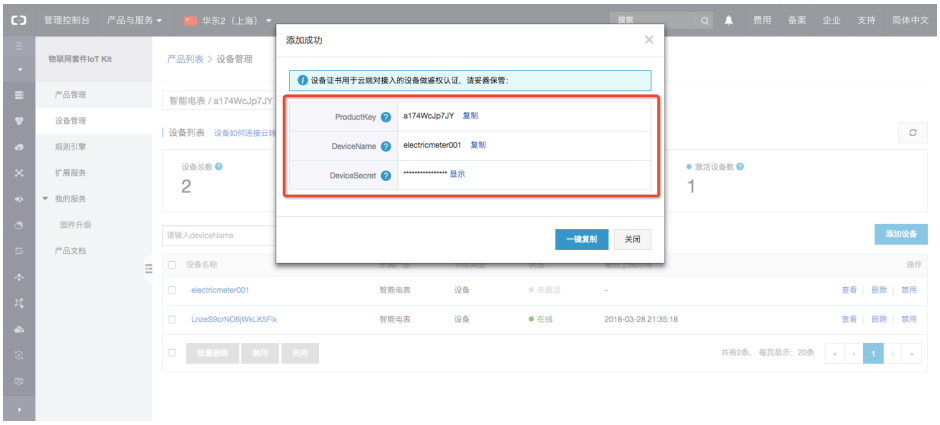
- 设备总数：统计您账号下的设备总数；
- 在线设备数：统计当前在线的设备数，设备在线/离线为动态变化数据，显示可能存在延迟；
- 激活设备数：统计当前已激活的设备数，设备激活为动态变化数据，显示可能存在延迟；
- 子设备：针对网关类设备，统计该网关下连接的子设备数量；

添加设备

选择一款产品，点击添加设备，在输入框中填写设备的DeviceName，请确保该Devicename在产品下唯一。输入框为空时，将由系统为您默认颁发一个全局唯一的Devicename。



点击确定后，即可完成该设备的添加，系统将返回该设备的设备证书。

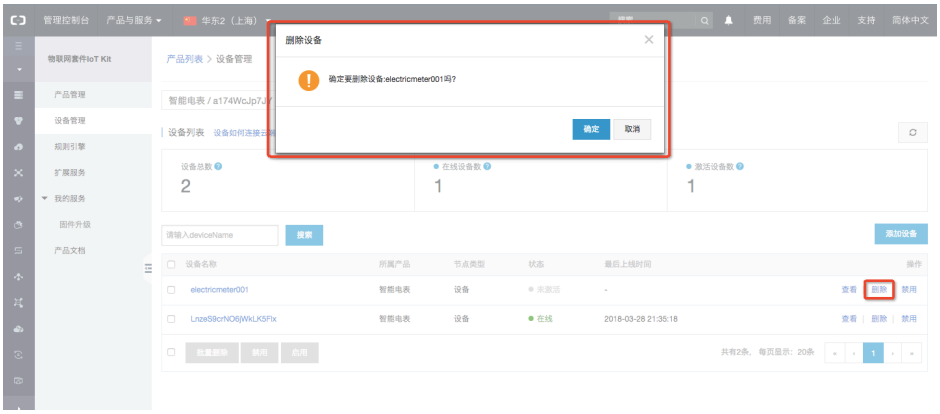


- 如果设备采用一机一密进行云端认证，需将设备证书烧录至设备中，包括ProductKey、Devicename和DeviceSecret，用于设备联网激活；
- 如果设备采用一型一密进行云端认证，您可以无需关心设备证书，设备中仅需烧录ProductKey和ProductSecret，设备DeviceSecret将在设备首次联网激活时动态下发，设备本地进行安全存储后再与云端连接进行激活；

请注意，您可以通过控制台添加单个设备，如果需要批量申请设备，可通过服务端API接口进行申请。

删除设备

在设备列表中，选择需要删除的设备，点击删除，确认删除后，系统将删除该设备的相关数据。



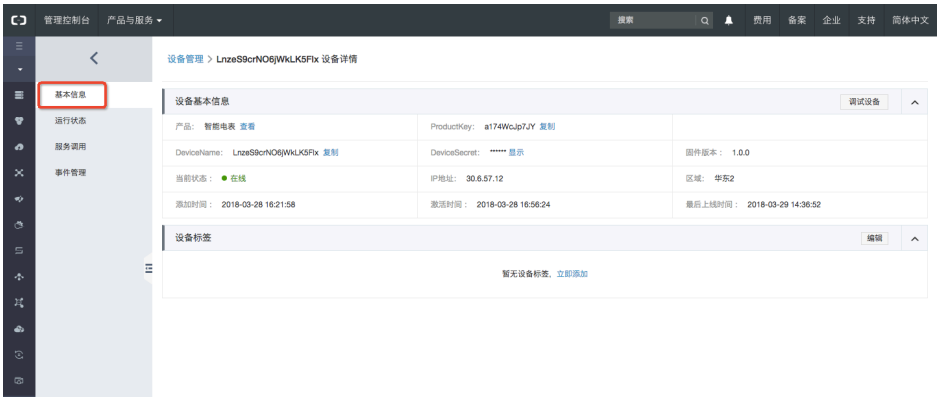
如果您需要删除多个设备，可以勾选设备列表中的复选框，点击列表下方的批量删除，确认后批量删除所选设备。

> 请注意，删除设备操作无法恢复，重新添加设备的Devicename将颁发新的DeviceSecret，可能导致原有设备无法正常接入云端，需要重新烧录设备证书，请谨慎操作。

基本信息

设备基本信息

进入设备详情后，可查看该设备的基本信息，包括设备身份信息、连接状态、固件版本、ip地址等。



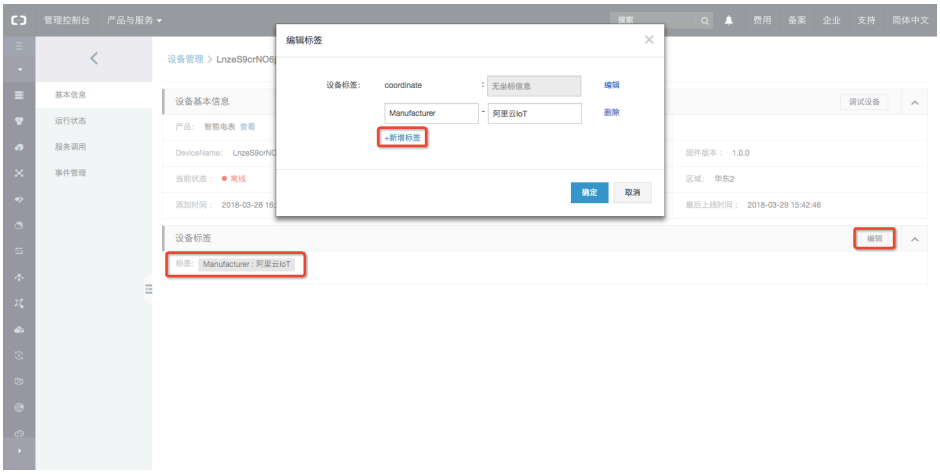
- IP地址：当设备激活连接到套件时，套件会获取到设备入网的出口IP地址。
- 固件版本：需要设备连接套件时，通过设备端OTA SDK将版本号上报上来。

设备标签

设备标签采用Key-Value键值对，可作为设备扩展性信息的描述和补充，如设备制造商、所属单位、规格尺寸

等。设备标签的创建支持以下三种方式：

- 如果产品定义了标签，则该产品下的所有设备将自动继承产品的标签；
- 支持为单个设备自定义标签，支持对设备的标签进行编辑和删除；
- 设备可以通过标签上报Topic主动更新自身的标签信息，便于云端进行管理；



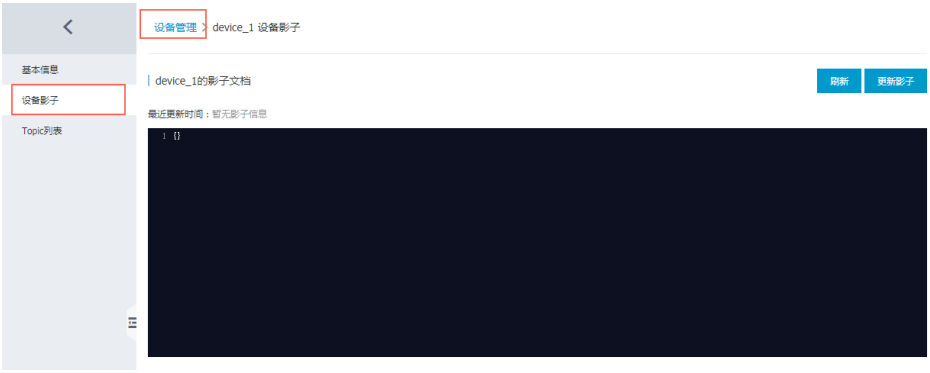
- 提供一个默认属性：坐标，并提供地图服务让用户可以方便为设备确定坐标。
- 设备标签支持规则引擎流转，可使用函数attribute(key)，函数中的key对应设备标签的Key，这个函数可以将系统中设备定义的标签名称对应的值提取出来，在规则引擎中流转，帮助客户实现丰富的应用场景。
- 设备标签支持在控制台添加也可以通过API添加。

设备影子

该功能基础版具备，高级版可以直接查看运行状态

影子初始状态：

- 创建好产品和设备，查看影子信息，内容为空



设备端更新影子信息：

- 设备端上报状态到IoT套件后。

应用程序更新影子状态：

repored字段可以为空，desired字段不能为空，输入正确的json格式数据即可。

更新完后影子状态变为：

desired内容可以是嵌套的json，也可以是数组。

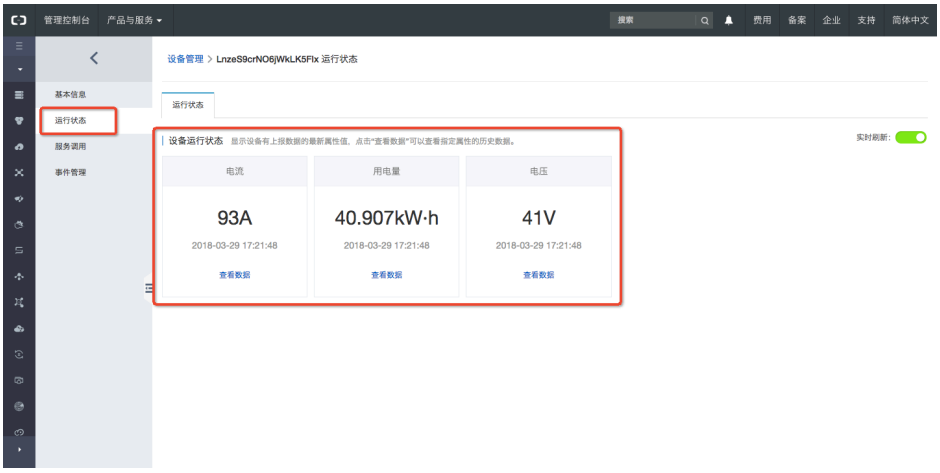
```
{
"colors" : ["RED", "GREEN", "BLUE" ]
}

{
"colors" : {"RED":125,"GREEN":125,"BLUE":125}
}
```

运行状态

运行状态仅面向使用高级版接入的设备提供。

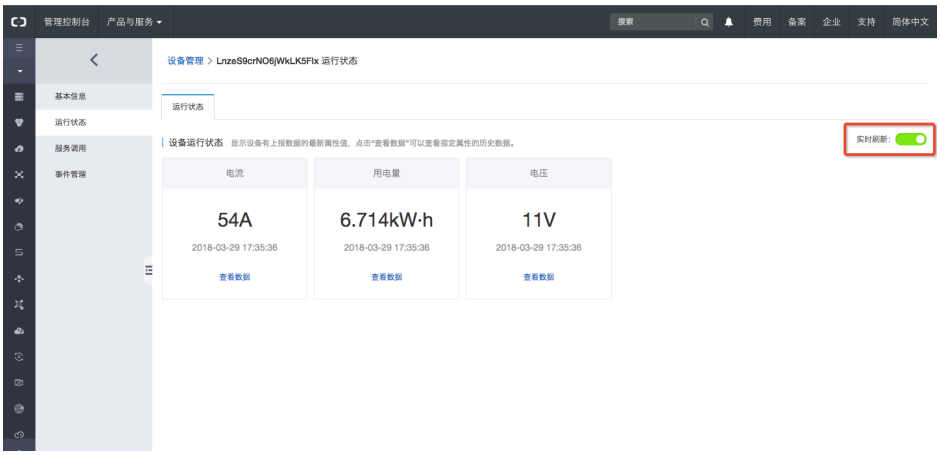
设备运行快照

在设备详情页面，点击运行状态，将以卡片形式显示该设备的运行快照。状态快照是设备最近一次上报的属性数据，如果该设备定义了某一属性但从未上报过数据，则此处不做显示。



实时刷新

打开卡片右侧的实时刷新开关，卡片中将实时刷新设备的运行数据，页面每5s自动更新一次，数据显示与设备的上报频率有关，可能存在时间上的延迟。

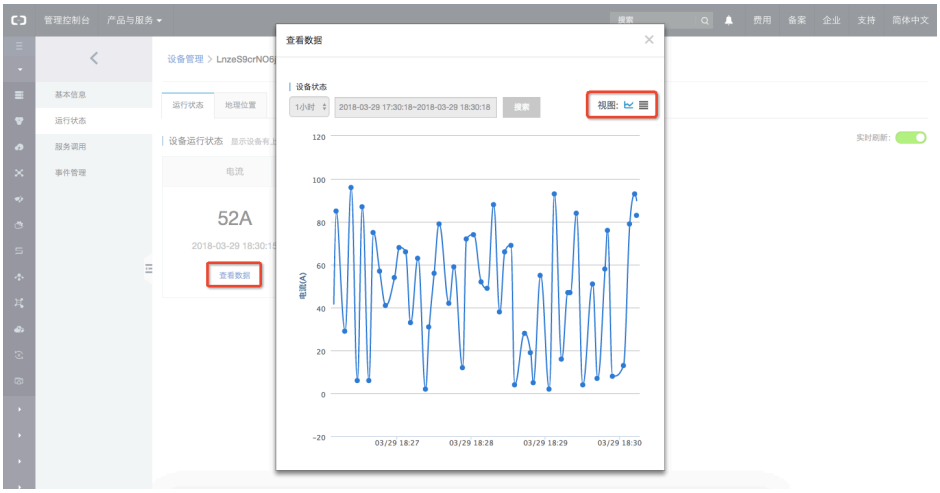


查看历史数据

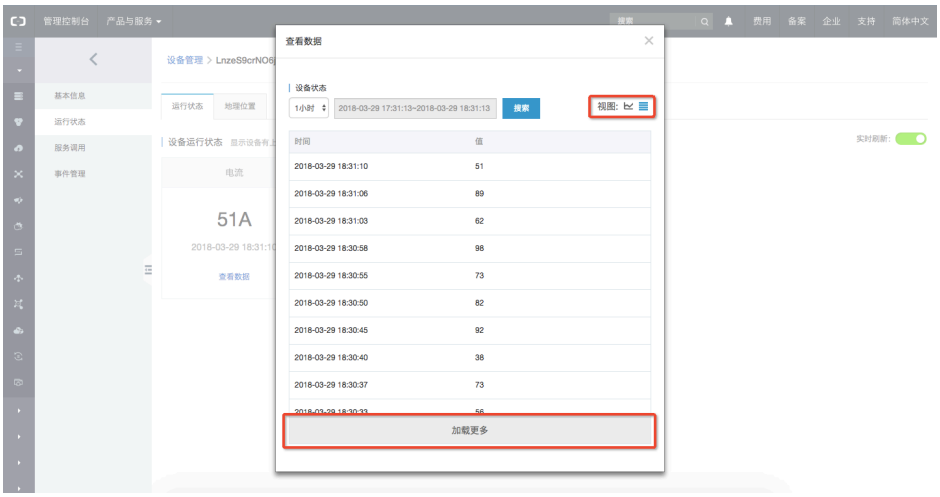
点击卡片下方的查看数据，可以在弹窗中查看当前属性的历史数据，默认显示最近1小时内的数据，如果提示“暂无数据”，请选择其他时间范围进行查看。

针对数值型数据（int32/float/double），支持图表和表格两种视图，图表视图下将显示最近一小时的数据趋势，表格视图将可查询其他时间范围内的更多数据，非数值型数据将默认仅支持表格视图。

图表视图：



表格视图：

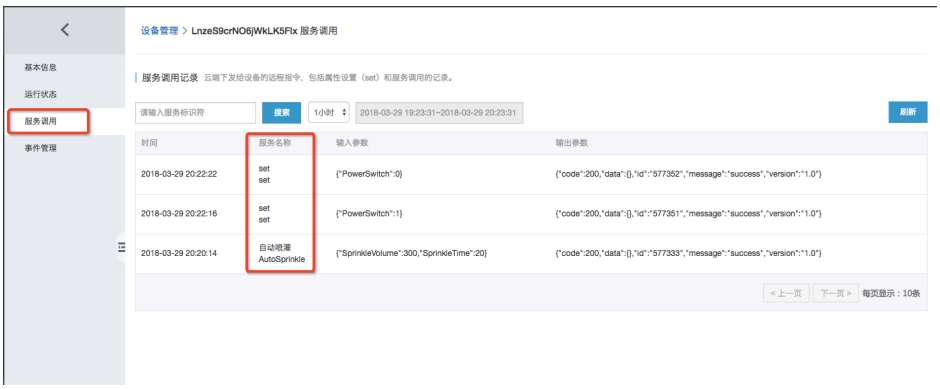


服务调用

设备服务调用仅面向使用高级版接入的设备提供。

设备服务调用

在设备详情页面，点击服务调用查看云端下发给设备的指令记录，列表中显示该服务被创建的时间以及调用时的出入参信息。



设备的服务包括两类：

- set服务：系统默认服务，对设备属性调用set方法，设置属性的目标值，如将电源开关置为“关”，前提是已经产品中定义了该设备的属性，并且该属性支持“读写”；
- 自定义服务：用户定义一个更为复杂的设备服务并进行远程调用，如农机设备的“自动喷灌”服务，自定义服务可以引入设备的属性作为出入参，也可以自定义出入参，不受限于设备的属性；

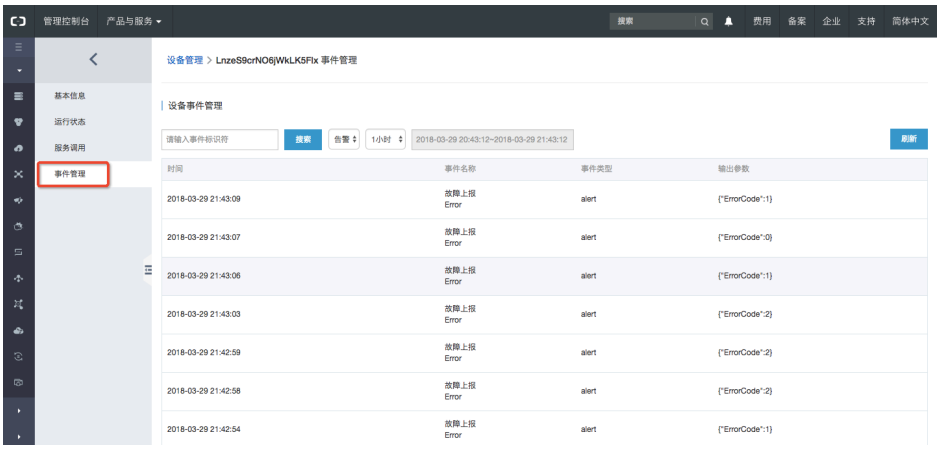
服务调用记录可通过服务的标识符（identifier）搜索，默认显示最近1小时的记录，您也可以选择其他时间范围（24小时/7天/自定义）查看。

事件管理

事件管理仅面向使用高级版接入的设备提供。

设备事件管理

在设备详情页面，点击事件管理查看设备上报的事件记录，列表中显示事件上报的时间以及携带的出参。



事件记录可通过事件的标识符（identifier）搜索，默认显示最近1小时的记录，您也可以选择其他时

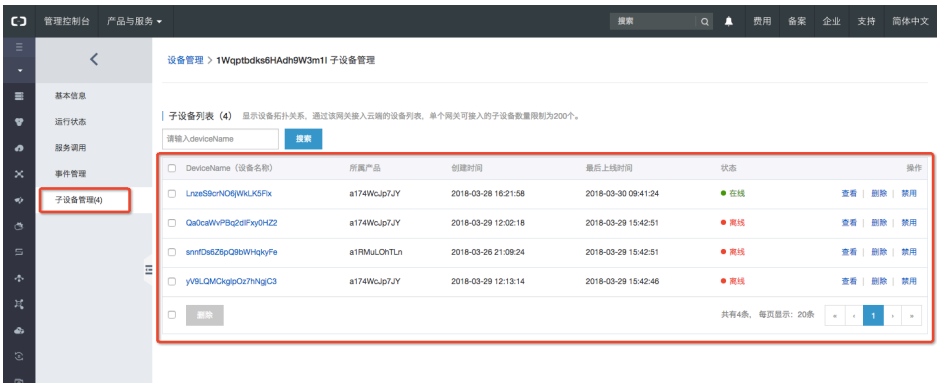
间范围（24小时/7天/自定义）查看。

事件记录可筛选不同的类型进行查看，支持查看“全部类型”、“信息 / info”、“告警 / alert”和“故障 / error”。

子设备管理

子设备列表

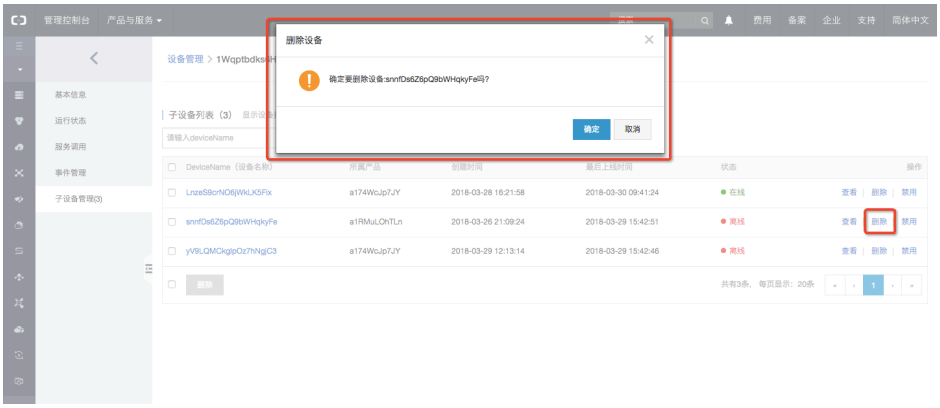
对于网关类节点的设备，支持管理网关和子设备的拓扑关系，您可以直接查看和管理网关下连接的子设备。点击查看，可跳转到子设备的详情页面，子设备详情页面与普通直连设备一致。



网关与子设备之间的拓扑关系由云端动态维护，只有通过该网关接入云端的子设备才会在这个网关的子设备管理页面显示；

删除子设备

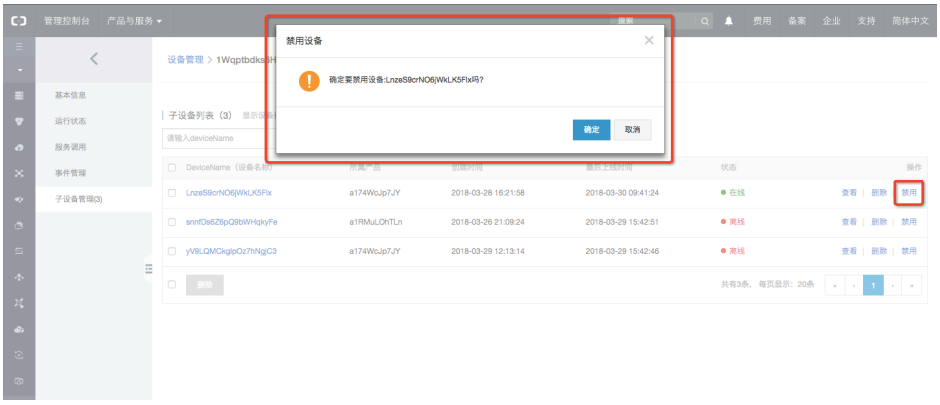
点击删除，可删除子设备。



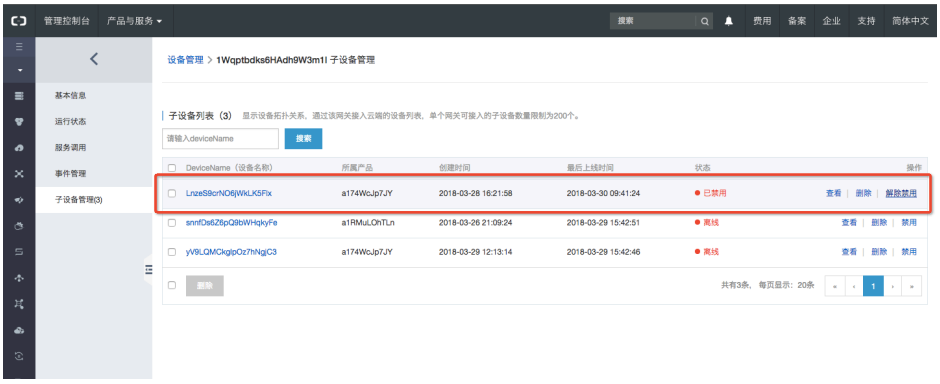
- 删除子设备将解除子设备与网关之间的拓扑关系，同时删除该子设备的相关数据，删除操作不可恢复。
- 如果您删除的是网关设备，将自动解除该网关设备与下挂子设备的拓扑关系，如果子设备在线将导致子设备的状态变为离线，您可以添加新网关与原有子设备建立联系，以恢复子设备与云端的通信，删除网关不会直接删除子设备。

禁用子设备

点击禁用，弹窗确认禁用子设备后，云端将断开网关与子设备之间的消息通信，禁止子设备接入云端。



您可以点击解除禁用，恢复子设备与云端之间的消息通道。

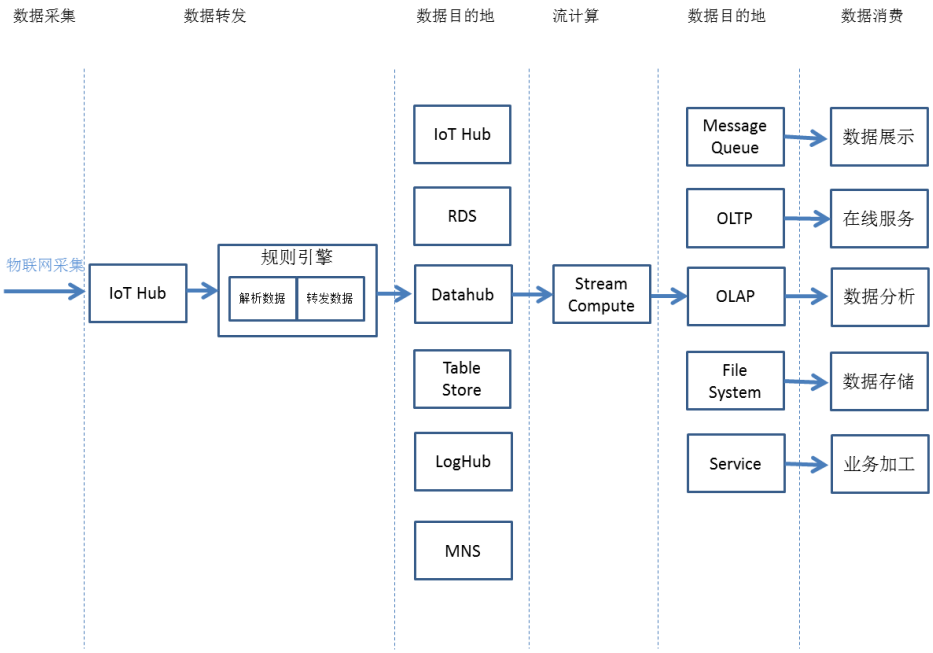


规则引擎

概览

当用户是基于Topic进行通信时，则可以使用规则引擎对Topic中的数据进行处理，然后转发到阿里云其他服务上，例如可以转发到RDS、Table Store中进行存储，例如可以转发到流式计算中进行实时计算，例如可以转发到消息中间件DataHub、LogHub上进而与流计算配合提供服务，等等。当然也可以使用规则引擎将Topic中的数据处理转发到另一个Topic中实现M2M通信。

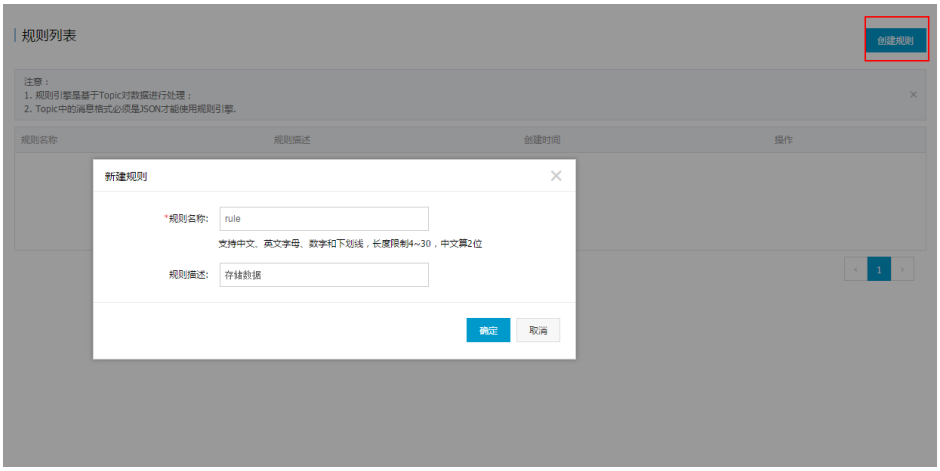
数据从采集到计算到存储，用户无需购买服务器部署分布式架构，用户通过规则引擎只需在web上配置规则即可实现采集+计算+存储等全栈服务。



特别注意：

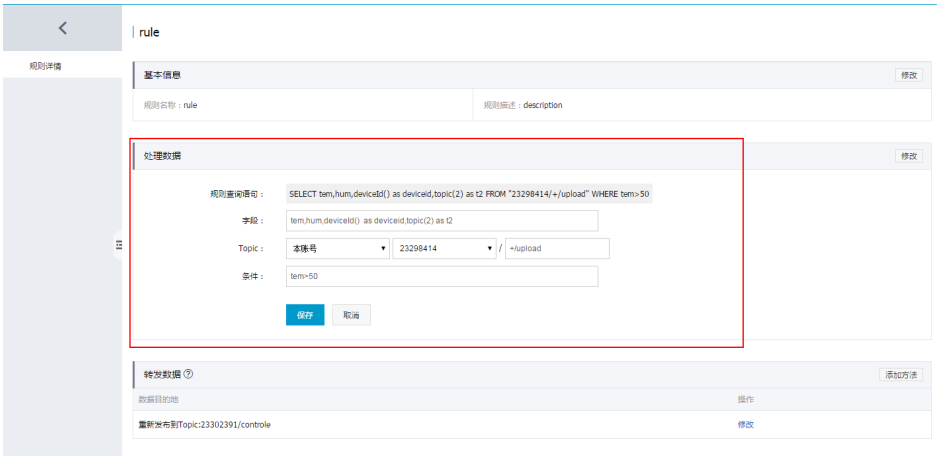
1. 规则引擎是基于Topic对数据进行处理
2. Topic中的消息格式必须是JSON才能使用规则引擎
3. 规则引擎是通过SQL对Topic中的数据进行处理
4. SQL语法目前不支持子查询、不支持like操作符，
5. 支持部分函数，比如deviceId()获取当前设备Id，请参考函数列表

创建规则



处理数据

创建完规则，即可编写SQL对某一Topic中的数据进行处理，详细SQL规则请参考文档表达式



图中例子表达的规则：将key为tem和hum对应的数据从Topic:/232*****/+/update中的JSON消息中提取出来，而且利用规则引擎规定的函数deviceId()和topic()将特定的数据提取出来（规则引擎支持的函数详情请参考文档函数），再通过条件tem>50对数据进行过滤，最后得到处理过后的数据，以便进行下一步转发该数据。

特别注意：

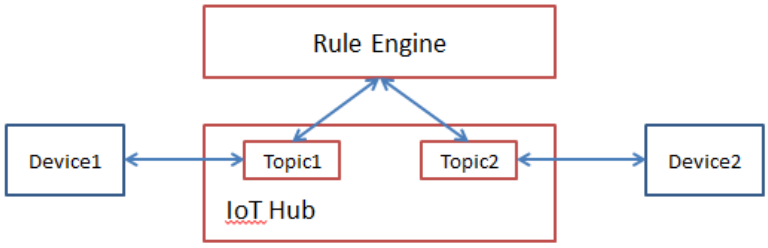
1. Topic支持通配符
2. 规则引擎中的SQL表达式支持标准SQL语法，而且会提供额外的函数方便实现丰富的场景。

数据转发到另一Topic

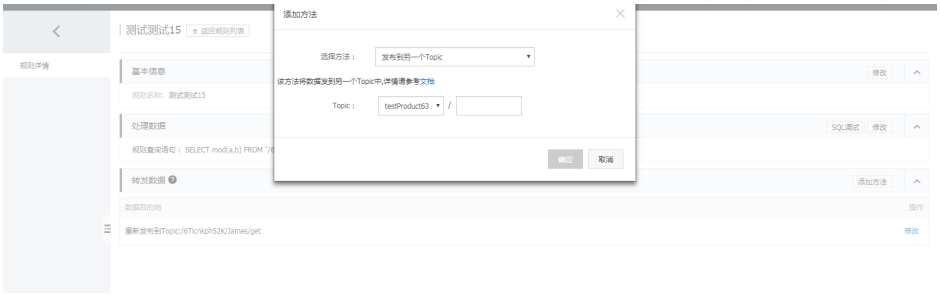
转发数据

通过编写SQL对Topic中的消息进行处理，然后可以添加方法对处理过后的消息进行转发操作。下面将详细介绍目前已经上线的转发操作，而且我们的方法会不断增加，提供更多更完善的服务给用户。

发布消息到另一个Topic



将Topic中的消息处理过后，发送到另一个Topic，可以实现M2M场景，但是又不局限于M2M，帮助节省服务端的开发。

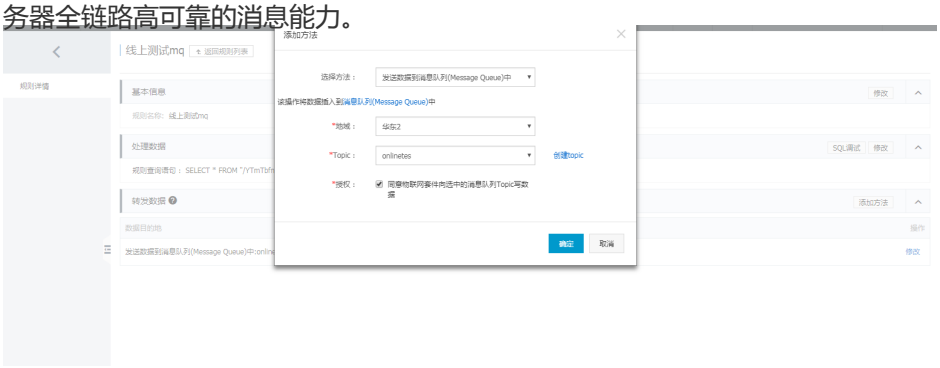


特别注意：

重新发送的Topic不支持通配符, 但可以使用${}表达式引用上下文值，比如${a}将解析为 select xxx as a 对应的值。

数据转发到MQ

用户通过物联网套件的规则引擎将数据转发到消息队列MQ中，从而具备了从设备到IoT套件到MQ再到应用服务器全链路高可靠的消息能力。



- 用户选择消息队列Topic所在地域。
- 用户根据业务选择需要写入数据的Topic
- 用户授权套件具有写入消息队列Topic的权限

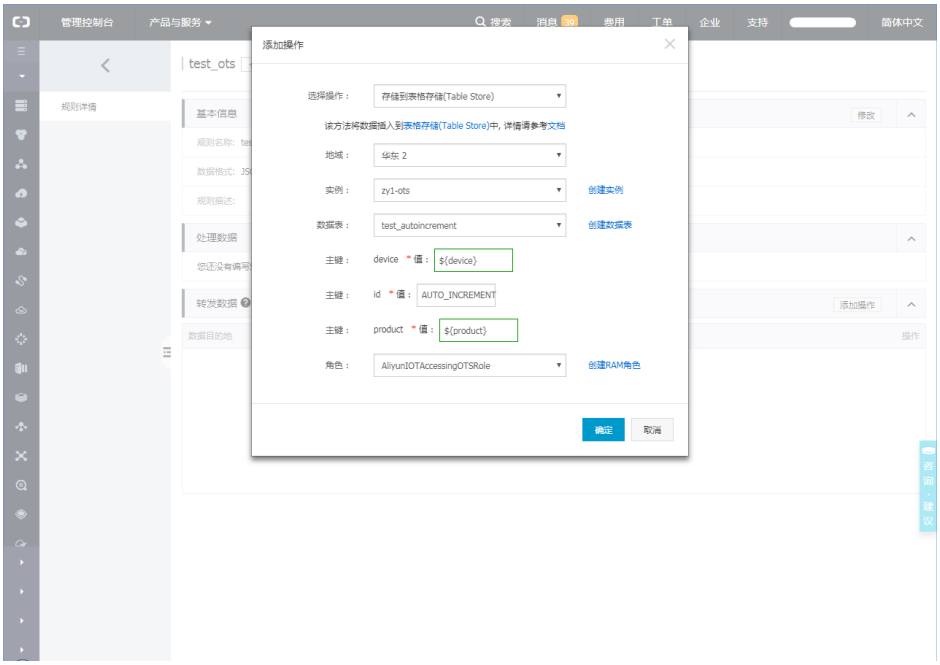
在此需要强调一下，MQ非铂金实例的Topic不支持跨地域的访问写入。所以用户如果没有创建铂金实例的Topic，那就必须在套件规则引擎所在的地域下创建Topic，否则不能写入，例如图中规则引擎所在地域是华东2，那么消息队列的Topic也需要在华东2创建；如果用户需要购买MQ的铂金实例，那可以不用关心在哪个地域购买铂金实例，因为铂金实例的Topic，套件都可以有权限写入数据。

启动规则之后，套件就会将数据写入消息队列的Topic中，然后用户可以基于MQ的实现消息的收发，具体请参考文档MQ订阅消息

数据转发到表格存储中

存储表格存储(Table Store)

可以将处理过后的消息，通过配置方法存储到表格存储（Table Store）中。想了解更多表格存储的信息，请参考表格存储（Table Store）



操作注意事项：

- 用户需要在控制台上选择Table Store数据表，用于数据存储。如果没有资源，则需要用户创建数据表
- 创建Table Store数据表必须创建主键，当用户选择好数据表之后，控制台会自动读出该表的主键，用户需要配置主键的值。
- **规则引擎不能操作用户的Table Store数据表，必须经过用户的授权才能对用户的数据表进行写数据。**所以，用户需要创建一个具有Table Store写入权限的角色，然后将该角色赋予给规则引擎，这样规则引擎才能将处理过后的数据写入数据表中。

示例：

经过SQL抽取出来的JSON数据：{"device":"bike","product":"xxx","data2":[{...}]}。业务上需要把这个JSON数据存入Table Store中，并且主键是device, product, id。

那么用户只需在控制台配置主键的值，输入${device}，这就意味着当有消息过来并触发规则，主键device就会存入JSON中device的value值，主键product同理。这里要特别强调一下，${}是转义符，如果不输入该转义符，存入的将会是一个常量。

规则会自动匹配主键是否为自增列，如果主键是自增列，则自动返填 AUTO_INCREMENT，并且不能编辑。

配置完主键之后，当有消息过来，套件会自动解析JSON中的除了主键之外的key值，然后根据key自动创建Table Store的数据列。例如，该示例中，就会创建两列：data1和data2，并且会在每列下面存入对应的value值。这里要特别强调一下，目前只支持一级JSON的解析，不支持嵌套JSON的解析，那么在该示例，data2下面就会以字符串的形式存入整个嵌套JSON，而不能再次对嵌套JSON进行解析创建列。

数据转发到DataHub

数据转发到DataHub中

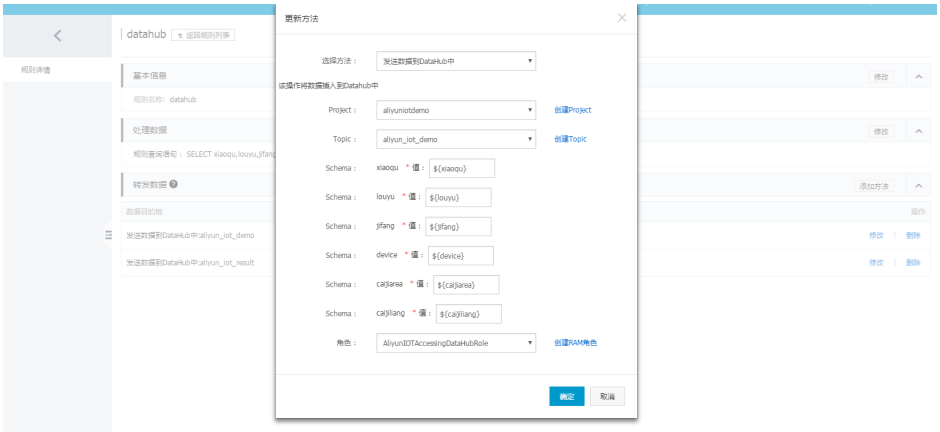
物联网套件的定位在于设备接入和设备管理，对于数据存储和计算，物联网套件会将这部分工作交给阿里云其他云产品。

在很多物联网场景中，流计算是刚需。阿里云流计算平台的数据采集模块，均是围绕DataHub作为流式数据采集的目的Pub/Sub系统。

> Q: DataHub是什么？
>
> A: 流计算是一种事件触发的模型，即一旦有新的事件(数据)达到，流计算系统将完成一次计算，并继续转为等待下一次事件到来。源源不断的数据流将为下游的流计算提供触发，阿里云流计算触发的数据流就存放在DataHub，DataHub产品即可为下游的流式计算提供事件触发机制，触发流计算的运行。因此用户只需要将驱动流计算运行的流式数据写入DataHub，使用了该DataHub Topic的下游流计算任务即可被触发进行一次运算。
>
> DataHub定义为大数据Pub/Sub系统，为下游的流计算、MaxCompute等提供了实时数据的入口。

规则引擎将设备数据实时转发到Datahub,进而和流式计算打通，帮助用户实现对设备数据进行实时计算的场景。详细请参考流计算文档

下图是在控制台上配置转发规则，将数据转发到DataHub中。



操作说明：

- 用户需要先选择DataHub中的Project，然后根据Project选择Topic。如果没有资源，那就需要去DataHub控制台创建相应的资源。
- 选择完DataHub中的Topic后，规则引擎会自动获取Topic中的Schema，接下来需要将规则引擎筛选出来的数据映射到对应的Schema中。
- **规则引擎不能操作用户的DataHub资源，必须经过用户的授权才能对用户的DataHub进行写数据。**所以，用户需要创建一个具有DataHub写入权限的角色，然后将该角色赋予给规则引擎，这样规则引擎才能将处理过后的数据写入DataHub中。

特别注意：将规则引擎筛选出来的数据映射到对应的Schema，需要使用${}，如果不使用的话，存到表中的将

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。

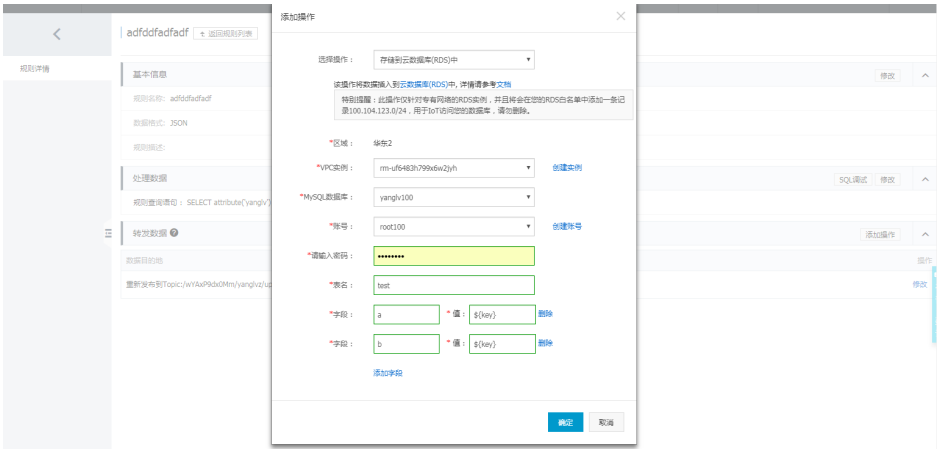
数据转发到RDS中

存储到云数据库（RDS）中

用户可以通过在控制台上配置规则引擎将物联网套件中的数据路由转发到云数据库（RDS）的VPC实例中。想要了解更多RDS信息的，请参考文档云数据库（RDS）。

- 发布的节点有华东2、硅谷、新加坡
- 只支持转发到VPC实例
- 只支持MySQL实例
- 只支持同region转发，不支持跨区转发，举个例子，华东2节点的套件数据只能转发到华东2的RDS中
- 支持普通数据库和高权限数据库

具体操作



操作说明：

- 首先用户需要根据自己的业务选择数据库进行数据存储。用户当前实例下的VPC实例，最后根据实例选择数据库，**如果是高权限数据库，需要用户手动输入数据库名称**。如果没有资源，那就需要去RDS控制台创建相应的资源。
- 选择完数据库之后，需要选择对该数据库**具有读写权限**的账号，如果没有，则需要去RDS控制台创建账号。输入该账号密码，让规则引擎去连接RDS进而写数据。
- 输入数据库中已经建立的数据表名，确定之后，数据将会写进这张表中。
- 确定数据表之后，需要将规则引擎筛选出来的数据对应存到数据表中的字段中。可以使用转义符$，格式为${key}，将Topic中key对应的value提取出作为输入值。举个例子

假如规则引擎的SQL：SELECT tem FROM mytopic.

假如RDS数据库有一张表，表中有tem字段，类型是String。

存储可以在控制台上配置，字段填入的是RDS数据表中的字段，例如tem，值填入的是规则引擎筛选出来的JSON字段，例如${tem},这里要强调两点，需要使用${}，如果不使用的话，存到表中的将会是一个常量，例如填入tem，那数据表存入就是tem这个常量；字段与值的数据类型必须保持一致，不然无法存储。

特别注意：

- 目前规则引擎支持MySQL，所以您购买RDS实例请选择MySQL类型
- 规则引擎为了连接RDS,会在RDS的白名单中添加。
  - 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

这些IP段不能删除，不然物联网套件就无法连接RDS，进而也就无法将数据写进RDS数据库中。如果没有此记录请手工添加。

下图展示的就是RDS控制台的白名单，当用户使用规则引擎将数据写进RDS某个数据库实例时，就会出现在白名单中出现。

- 规则引擎为了连接RDS数据库，需要使用用户的账号去连接，这就需要用户提供账号和密码给规则引擎。**这里面需要强调的是规则引擎取得用户的账号后，只是负责将规则匹配的数据写进数据库中，不会做其他操作。**
- 规则引擎只是负责将数据写进RDS数据表中，不会去帮用户去创建数据表，所以用户需要在RDS中自行创建数据表。
- **存储需要用户自行保证字段与值的数据类型一致，不然会导致写入不成功；值应该使用函数${}，这样才能存入JSON数据中的value，不然的话将会存入一个常量。**

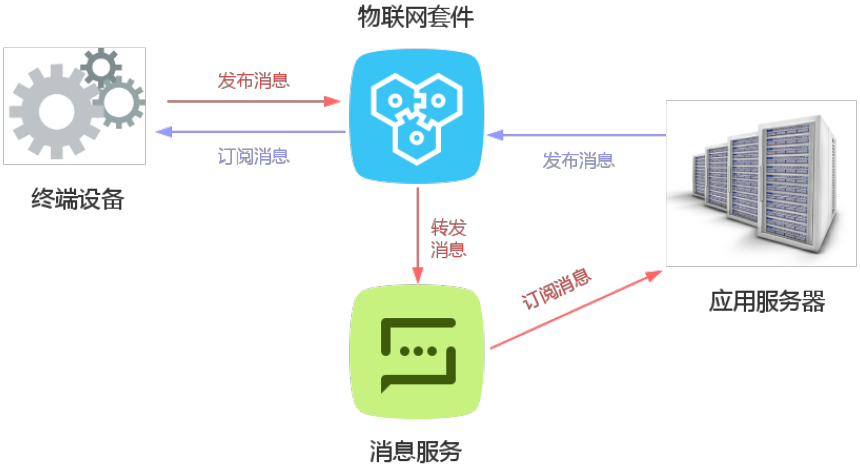
数据转发到MNS中

数据转发到消息服务（Message service）中

规则引擎可以将IoT Hub中的数据转发到消息服务(MNS)中。有关消息服务的详情，请戳这里。

物联网套件与消息服务的结合，可以实现设备端与服务端之间**高性能**的消息闭环传输。

- **设备发送数据到服务端**：设备发布消息到物联网套件中，物联网套件通过规则引擎将消息进行处理并转发到MNS的主题中，最后客户的应用服务器调用消息服务的接口订阅消息。这种方式优势是MNS可以保证消息的可靠性，避免了服务端不可用时的消息丢失，同时MNS在处理大量消息并发时有削峰填谷的作用，保证服务端不会因为突然的并发压力导致服务不可用。
- **服务端发送数据到设备**：客户的应用服务器调用物联网套件的OpenAPI发布数据到物联网套件中，然后设备从物联网套件中订阅消息。



使用步骤

1. mns控制台创建主题
2. 创建规则，将iot数据处理并转发到mns主题中
3. 您的服务器接入mns的sdk（推荐使用queue模式订阅）
4. 发送一条设备消息，查看您的服务器是否收到

详细参考以下截图：

MNS控制台操作

MNS控制台

1.创建主题

这个主题是为了给规则引擎推送数据使用的。

2.创建订阅

主题创建后，还需要给这个主题创建订阅者，这样您的服务器以某个订阅者身份去订阅数据。

主题的消息可以被多个订阅者消费，目前几种方式：

1. 队列：把topic里消息转到某个队列，这样您的服务器可以基于某个队列监听数据
2. http：把topic里消息主动通知到您的http地址（需要您部署http webserver）
3. 邮件：参考mns文档 邮件推送
4. 短信：参考mns文档短信推送

用户根据自己的业务创建订阅者，可以有多个订阅者。

规则引擎转发

如下图，添加方法，将数据转发到创建的主题中

操作说明：

- 在方法中选择发送消息到消息服务（Message service）中
- 首先用户需要根据自己的业务选择消息服务中主题作为数据转发目的地。用户需要先选择地域，然后根据地域选择主题。如果没有资源，那就需要去消息服务控制台创建相应的资源。
- 规则引擎不能操作用户的消息服务中的主题，必须经过用户的授权才能对用户的主题进行写数据。所以，用户需要创建一个具有消息服务写入数据权限的角色，然后将该角色赋予给规则引擎，这样规则引擎才能将处理过后的数据写入数据表中。如果存在该角色，选择该角色，如果不存在，创建该角色。角色具体信息请到RAM控制台查看。

配置好方法之后，运行该规则，就可以将经过SQL语法处理过后的数据转发到消息服务的主题中。

如何从MNS主题获取数据，请参考文档主题使用手册

数据转发到HiTSDB

存储到时间序列数据库（HiTSDB）中

用户可以通过在控制台上配置规则引擎将物联网套件中的数据转发到时间序列数据库（HiTSDB）的实例中。想要了解更多HiTSDB信息的，请参考文档时序数据库HiTSDB。

- 发布的节点有华东2
- 只支持专有网络下HiTSDB实例
- 只支持同region转发，不支持跨region转发，举个例子，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

具体操作

操作说明：

- 选择存储到高性能时间序列数据库(HiTSDB)操作。选择时间序列数据库实例。如果没有资源，需要去

HiTSDB控制台创建相应的资源。

timestamp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Topic中的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Topic中的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deviceName;
- 使用常量配置，例如beijing，此时值为beijing;

授权: 同意物联网套件向HiTSDB写数据, 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;

举例:

假如规则引擎的SQL:

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

配置的规则如上图所示；

```
发送的消息:
{
"time": 1513677897,
"city": "beijing",
"distance": 8545,
"power": 93.0
}
```

结果:规则引擎会向高性能时间序列数据库中写入两条数据:

```
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
```

特别注意：

- 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。如上例所示除了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-_./():,[]= ‘
- 规则引擎为了连接HiTSDB,会在HiTSDB的白名单中添加,不同节点会添加不同白名单：
    - 华东2：100.104.76.0/24

这些IP段不能删除，不然物联网套件就无法连接HiTSDB，进而也就无法将数据写进HiTSDB数据库中。如果没有此记录请手工添加。

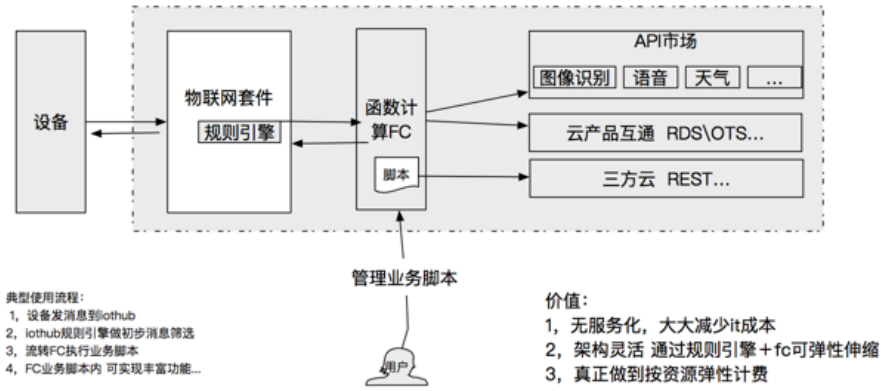
下图展示的就是HiTSDB控制台的白名单，当用户使用规则引擎将数据写进HiTSDB某个数据库实例时，就会在白名单中出现。

- 规则引擎只是负责将Topic中的数据写进HiTSDB实例中，不会做其他操作。

转发数据到函数计算

数据转发到函数计算（Function Computer）中

规则引擎可以将IoT Hub中的数据转发到函数计算(FC)中。有关函数计算的详情，请戳这里



使用步骤

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是否结果是否正确

详细参考一下截图：

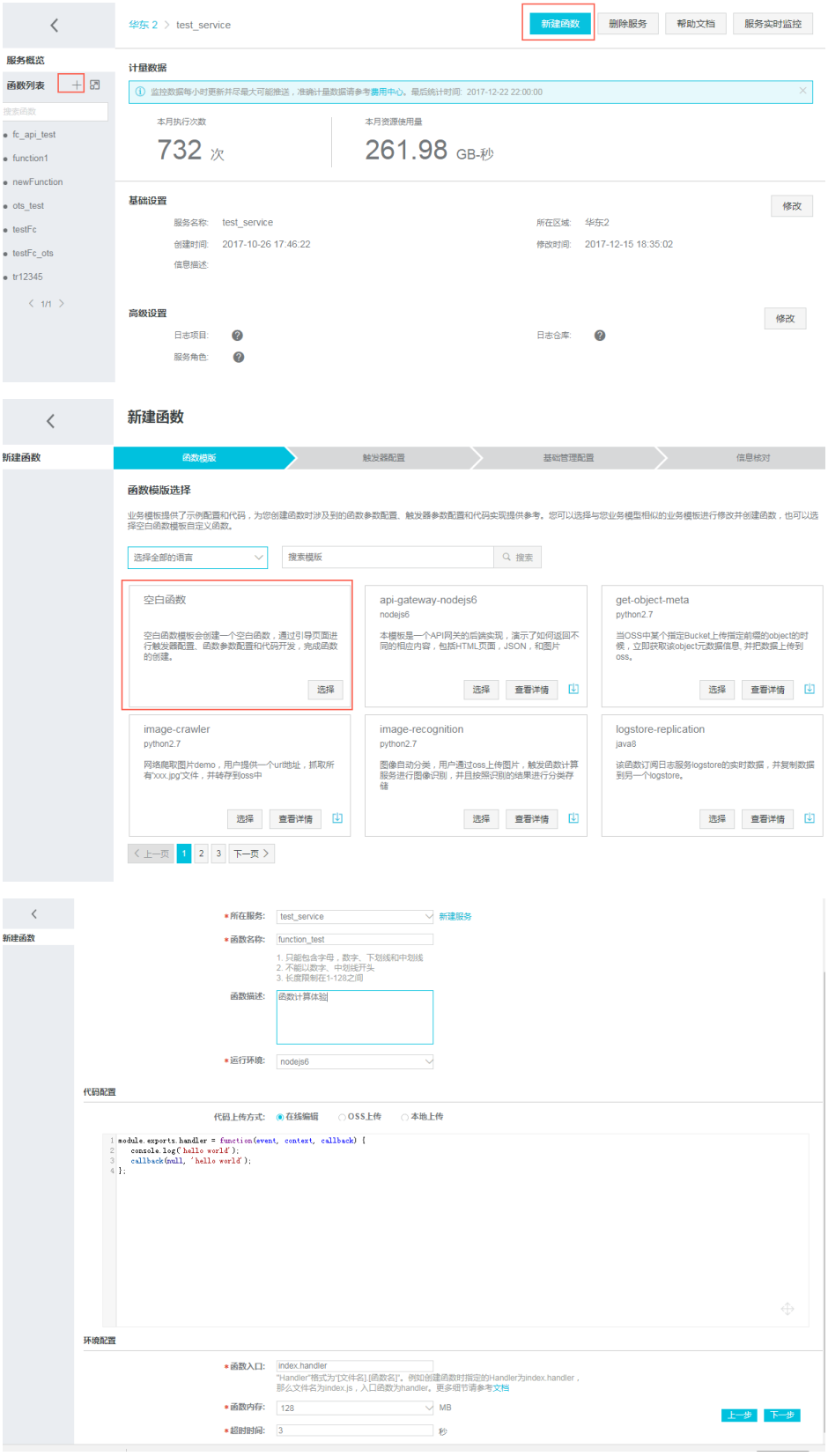
FC控制台操作

FC控制台

1、创建服务

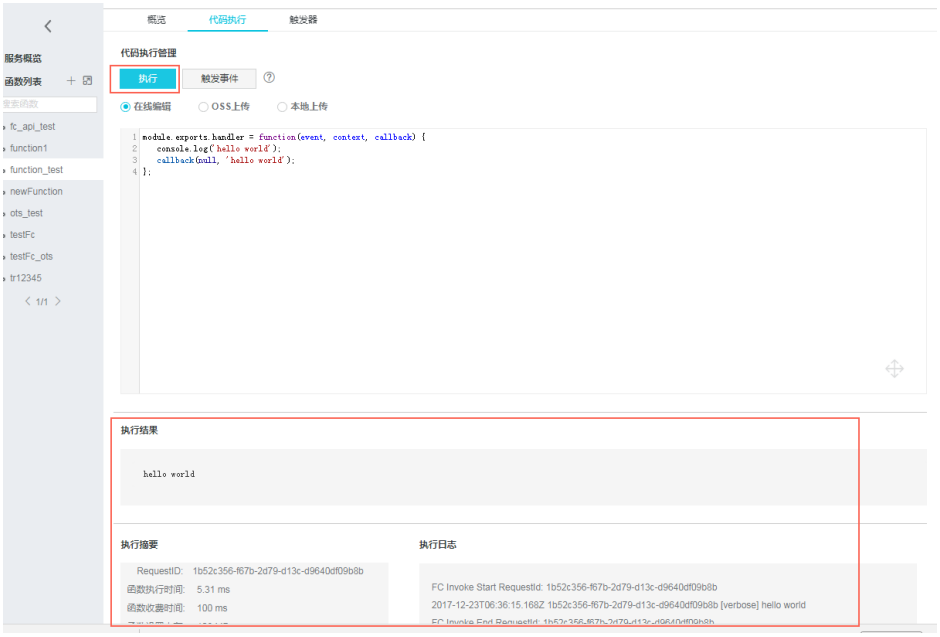
2、创建函数

函数计算有给定很多的函数模板，也有多种运行环境可选择，这个根据实际业务的需要来配置。，这里以最简单的一个空白函数作为示例。



3、函数执行验证

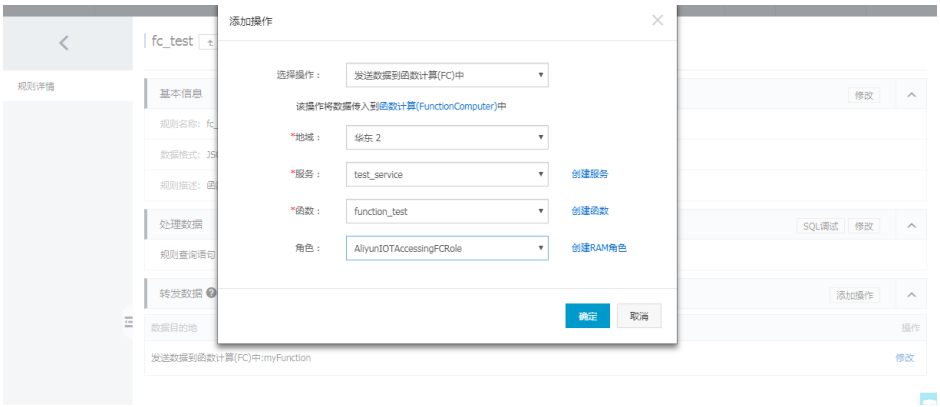
函数成功创建之后，可以直接在fc的控制台执行，以验证函数执行情况。函数计算会直接将函数的输出打印在控制台上，同时还有请求的相关信息。



函数能正常执行。接下来配置iot规则引擎

规则引擎转发

如下图，添加方法，将数据转发到创建函数中



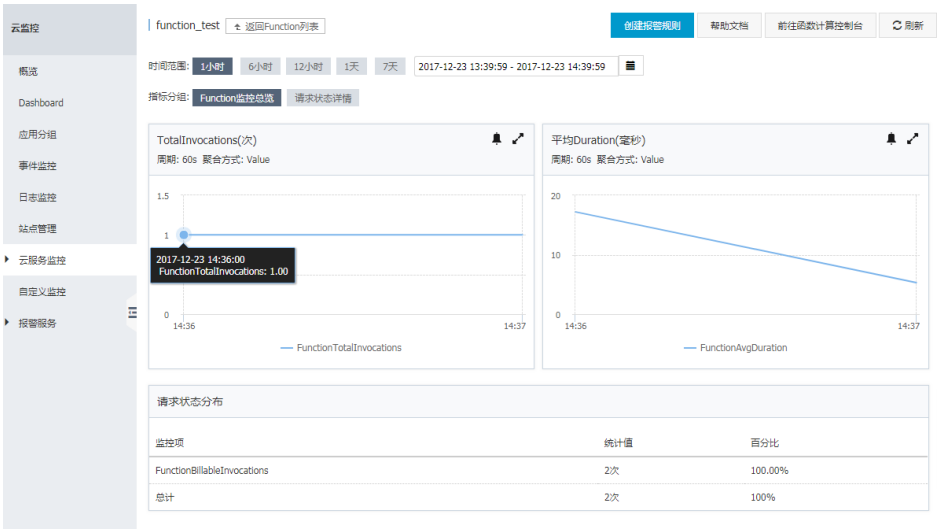
操作说明：

- 在方法中选择发送数据到函数计算（FC）中
- 首先用户需要根据自己的业务选择函数计算的函数作为数据转发目的地。用户需要先选择地域，然后根据地域选择服务。如果没有资源，那就需要去函数计算控制台创建相应的资源。
- 规则引擎不能操作用户的函数计算中的函数，必须经过用户的授权才能执行函数。所以，用户需要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，这样规则引擎才能将处理过后的数据转到函数计算，并执行对应的函数。如果存在该角色，选择该角色，如果不存在，创建该角色。角色具体信息请到RAM控制台查看。
- 目前只支持华东2节点
- json格式和二进制格式都支持转发到函数计算中

配置好方法之后，运行该规则，就可以将经过SQL语法处理过后的数据转发到函数计算中。

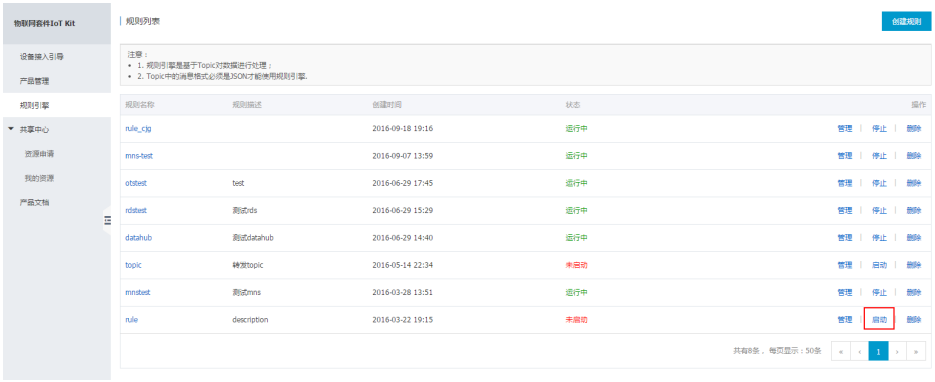
结果验证

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



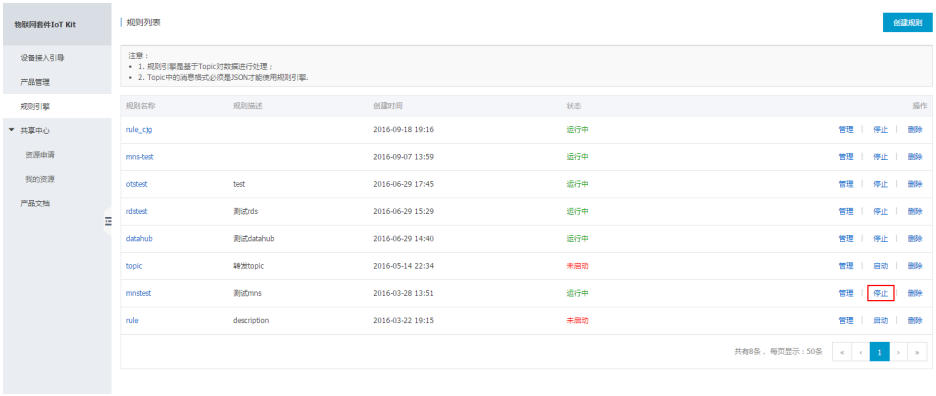
设置规则状态

当编写完SQL，添加好方法之后，用户就可以启动规则。这样一旦有消息Pub到SQL语法中的Topic里时，规则就触发，执行规则。



停止规则

用户也可以停止规则，这样规则就不会被触发。



扩展服务

固件升级

固件升级服务使用指南

本文档主要介绍固件服务的控制台使用说明，帮助用户快速使用固件服务。

开通固件升级服务

以阿里云帐号直接进入IOT控制台，就可以找到*固件升级*。如果还没有开通固件升级服务，则需要点击开通。

使用引导

1. 添加固件
2. 验证固件
3. 批量升级
4. 再次升级

添加固件

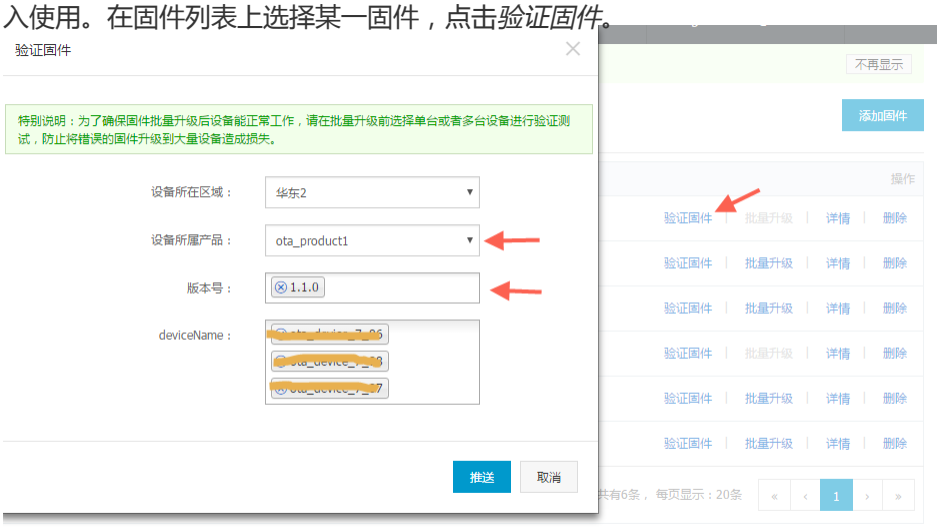
目前只有华东2节点支持固件服务。进入控制台，点击产品管理，选择华东2节点，可以在左侧扩展服务菜单栏看到固件升级菜单。点击固件升级，在右上角有个添加固件按钮，点击*添加固件*。

说明

1. 上传的固件文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1~32。
2. 上传的固件文件必须是bin文件，一般是linux下编译生成的二进制文件。
3. 上传的固件文件大小不能超过10M。
4. 用户最多能上传100个固件文件(**已删除的也包括在内**)。

验证固件

添加完固件后，必须先使用少量设备来验证固件是否可用。若验证固件可用，则此固件才可以在大量设备上投入使用。在固件列表上选择某一固件，点击*验证固件*。



说明：

1. 验证固件会向mqtt接入的设备推送升级通知，在线设备会立即接收到升级通知（不在线的设备下次接入时系统会再重新推送一次）；其他接入方式（如coap或者https）的设备都是短连接的，所以无法收到。
2. 可以反复发起验证固件，验证固件的本质就是指定少量设备进行升级。

3. 对于某一固件来说，上一次的验证动作没有结束，是无法再次发起验证的。
4. 只要用户进行过验证固件操作，未验证的固件的状态就会被置为已验证，不会取决于设备的实际升级结果。
5. 验证固件操作结束，系统会记录下设备相应的升级操作记录（初始状态是**待升级**），只有设备收到升级通知后上传升级进度后，系统才会认为升级正式开始（升级操作记录状态更新为**升级中**）。升级开始后，可以在固件详情页，查看设备的升级进度信息。

批量升级

当进行过验证固件后，确认固件对设备是可用的，则此固件就可以在大批设备上投入使用。批量升级的本质也是对大批设备定向推送升级通知。

说明：

1. 禁止使用未验证的固件进行批量升级操作。
2. 设备从收到升级通知开始直至升级完成是一个渐进的过程，OTA系统在收到设备主动上报的升级进度后更新设备升级进度百分比信息。
3. 批量升级所覆盖的设备可能会因为设备上一次的升级动作没有结束（设备处于待升级或者升级中），而导致本次升级中该部分设备升级失败。
4. 设备在实际升级过程中出现错误（比如说下载失败、校验失败、解压失败等），并且通知给OTA系统后，系统会将本次升级动作置为完成(升级失败)。
5. 可以在固件详情页，看到批量升级对应设备的升级情况，升级失败列表选项卡会显示简要的升级失败原因。

再次升级

若某些设备上一次升级失败（不管是通过验证固件还是批量升级发起升级），则用户可以使用失败的升级操作

记录尝试再次发起升级。

删除固件

添加后的固件不能编辑，但是可以删除，而且必须保证该固件要升级的设备都已完成才能进行删除操作。

子账号访问IoT

创建子账号

创建子账号

1. 用主账号登录阿里云官网
2. 进入控制台：依次将鼠标移动至 **产品** -> **管理与监控**，单击 **访问控制** 进入访问控制产品页；单击 **获取使用资格** 申请使用权限，然后单击 **管理控制台** 进入控制台；
3. 依次单击 **用户管理** -> **新建用户**（右上角）进入新建用户页面
4. 在创建用户对话框里输入：登陆名（必填）和显示名，备注，邮箱，电话等（可选），同时勾选：**为该用户自动生成AccessKey**
5. 点击 **确定** ，完成子账号创建。创建成功后，会提示是否下载子账号 AK，请选择下载并妥善保管子账号 AK 文件。

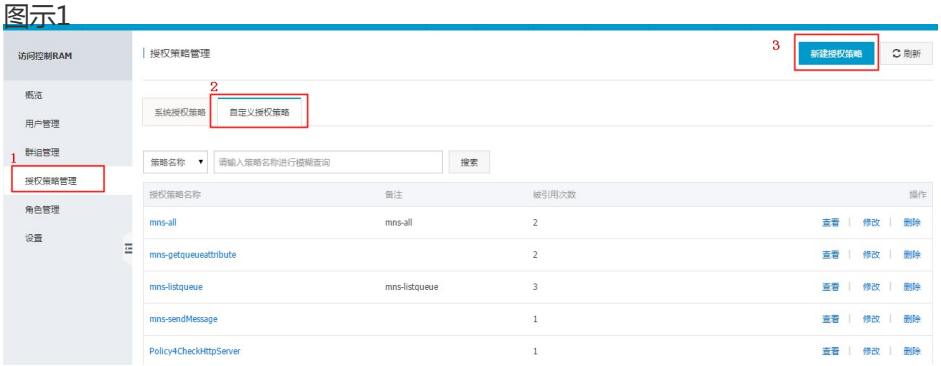
图示1：

新建授权策略

新建授权策略

- 1 在 RAM 控制台上依次点击：**授权策略管理** ->**自定义授权策略**->**新建授权策略**

图示1



- 2 在弹出对话框中：选择授权策略模板（使用空模板）

- 3 编辑授权策略并提交：修改 **授权策略名称**（自定义名称），**备注**，**策略内容**，并提交。
示例：一个 IoT 授权策略内容模板：

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

授权策略是json格式的字符串，其中，
Action ： 表示要授权的操作，IoT 操作都以”iot:”开头，
例如: “iot:CreateProduct” 表示 IoT 服务的API：CreateProduct
其他详见附录：IoT API和授权操作映射表；

Effect ： 表示授权类型， 例如:Allow, Deny

Resrouce： “ *“默认获取所有**详细授权策略的规则说明请参考RAM帮助文档*

图示2

授权子账号访问IoT

1.添加子账号授权策略

- 返回 **用户管理**， 找到第一步创建的子账号，点击右侧 **授权**。
- 在弹出的对话框中，选择授权策略名称，并添加到右侧已选授权策略列表，点击 **确定** 提交。

2.使用子账号进行访问 IoT 服务

- 返回 **概览**， 找到RAM用户登录链接。
- 输入子用户名称和子用户密码登录。

如果没有提示没有权限，说明子账号有权限访问 。

授权策略示例

如何使用带IP限制的访问控制

示例1：在Allow授权中增加IP限制允许通过42.120.88.0/24, 42.120.66.0/24两个IP段访问IoT套件

```
{
"Statement": [
{
"Action": "iot:*",
"Effect": "Allow",
"Resource": "acs:iot:*:*:*"
}
],
"Version": "1",
"Condition":{
"IpAddress": {
"acs:SourceIp": ["42.120.88.0/24", "42.120.66.0/24"]
}
}
}
```

示例2：在Deny授权中增加IP限制如果源IP不在42.120.88.0/24中则禁止对IoT执行任何操作；

```
{
"Statement": [
{
"Action": "iot:*",
"Effect": "Deny",
"Resource": "acs:iot:*:*:*"
}
],
"Version": "1",
"Condition":{
"NotIpAddress": {
"acs:SourceIp": ["42.120.88.0/24"]
}
}
}
```

注意：因为Policy的鉴权规则是Deny优先(即如果用户的访问操作命中任意一条Deny规则，则禁止访问)，所以访问者从42.120.88.0/24以外的IP地址访问时，IoT套件服务会报没有权限。

IoT API 授权映射表

注意事项

- 为保证安全，在添加子账号，添加授权策略和增加授权时，需要手机短信或邮件验证。

IoT API授权映射表

IoT API	RAM 授权操作	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
GetCats	iot:GetCats	*	获取产品类型信息
QueryProductByUserId	iot:QueryProductByUserId	*	根据用户id查询产品
RegistDevice	iot:RegistDevice	*	创建设备
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
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
QueryTopic	iot:QueryTopic	*	查询Topic
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
ServerOnline	iot:ServerOnline	*	服务端订阅者上线
DeviceGrant	iot:DeviceGrant	*	设备授权
DeviceRevokeByTopic	iot:DeviceRevokeByTopic	*	通过Topic名撤销设备权限
DeviceRevokeById	iot:DeviceRevokeByI	*	通过ID撤销设备权限

	d		
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
DebugRuleSql	iot:DebugRuleSql	*	规则引擎SQL调试