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

子账号访问IoT

创建子账号

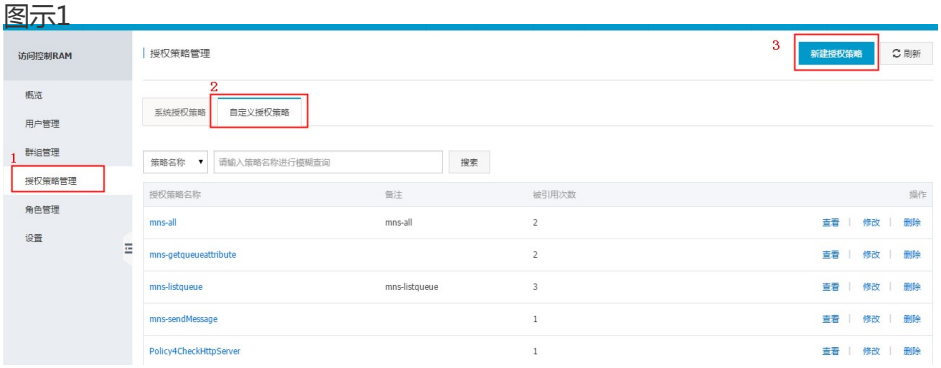
1. 用主账号登录阿里云官网
2. 进入控制台：依次将鼠标移动至 **产品** -> **管理与监控**，单击 **访问控制** 进入访问控制产品页；单击 **获取使用资格** 申请使用权限，然后单击 **管理控制台** 进入控制台；
3. 依次单击 **用户管理** -> **新建用户**（右上角）进入新建用户页面
4. 在创建用户对话框里输入：登陆名（必填）和显示名，备注，邮箱，电话等（可选），同时勾选：**为该用户自动生成AccessKey**
5. 点击 **确定** ，完成子账号创建。创建成功后，会提示是否下载子账号 AK，请选择下载并妥善保管子账号 AK 文件。

图示1：

新建授权策略

- 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

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 。

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。

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: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
DebugRuleSql	iot:DebugRuleSql	*	规则引擎SQL调试

产品管理

本文档主要介绍产品管理控制台的使用说明，帮助用户快速使用阿里云物联网套件。

开通物联网套件

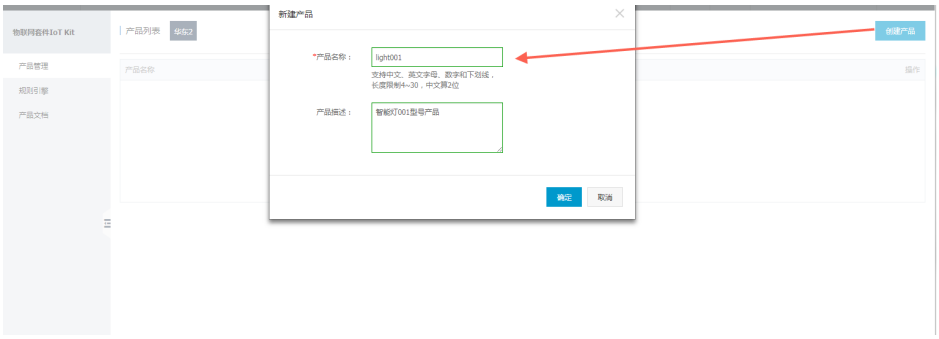
以aliyun账号直接进入IoT控制台，如果还没有开通阿里云物联网套件服务，则需要申请开通。

接入引导

1. 创建产品
2. 添加设备
3. 获取设备的Topic

创建产品

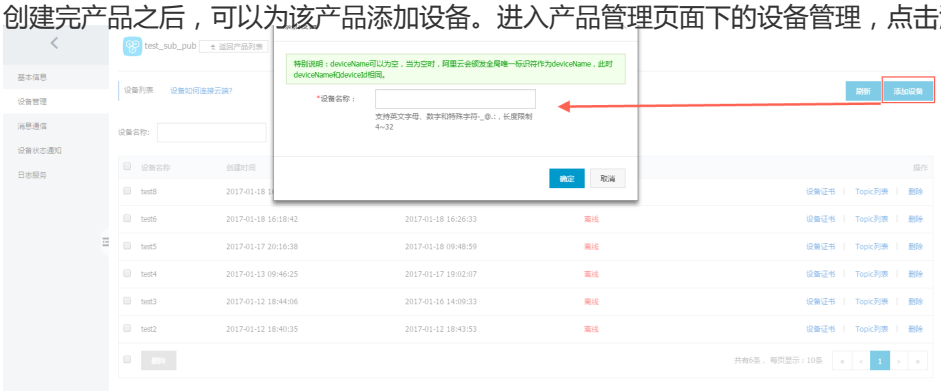
初步进入控制台后，需要创建产品。点击创建产品。产品相当于某一类设备的集合，用户可以根据产品管理其设备等。



- 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。
- productKey：阿里云IoT为产品颁发的全局唯一标识符

添加设备

创建完产品之后，可以为该产品添加设备。进入产品管理页面下的设备管理，点击添加设备。



说明：用户可以自定义设备名称(即deviceName)，这个名称即可作为设备唯一标识符，用户可以基于该设备名称与IoT Hub进行通信，需要指出的是，用户需要保证deviceName产品内唯一。

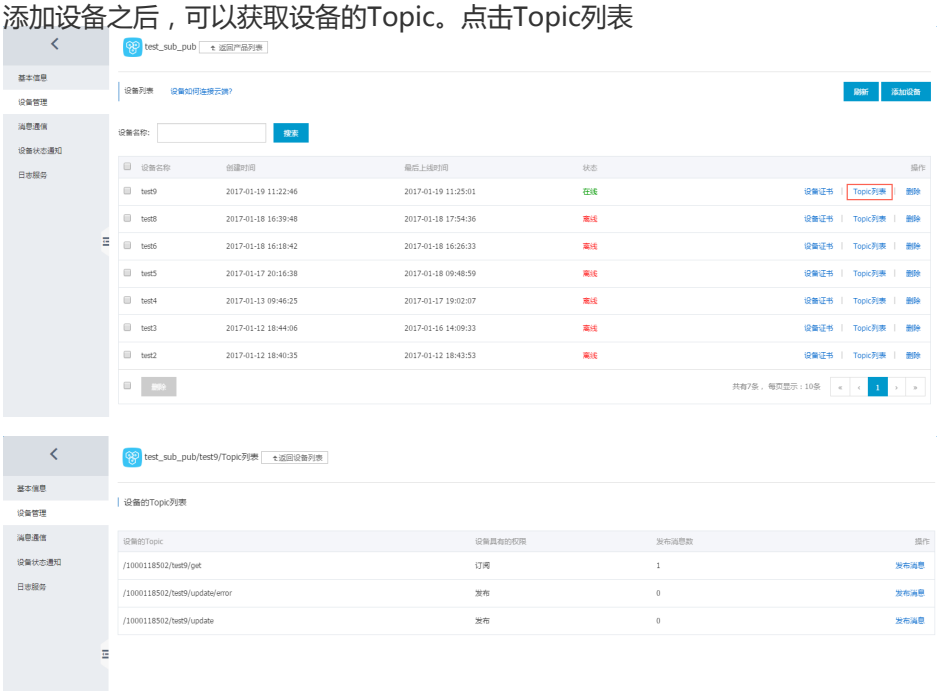
设备证书：添加设备之后，物联网套件为设备颁发的唯一标识符，设备证书用于设备认证以及设备通信，详细的请参考设备接入文档。

- deviceName：用户自定义设备唯一标识符，用于设备认证以及设备通信，用户保证产品维度内唯一。
- deviceSecret：物联网套件为设备颁发的设备秘钥，用于认证加密，与deviceName或者deviceId成对出现。

除了控制台添加设备，也可以通过API接口动态授权，请参考设备注册

获取设备的Topic

添加设备之后，可以获取设备的Topic。点击Topic列表



说明：创建产品之后，物联网套件都会为产品默认定义三个Topic类。那么，在添加设备之后，每个设备都会默认有三个Topic，即图中所示。如果想要增加、修改、删除Topic，请到消息通信重新定义Topic类。

设备可以基于Topic列表中的Topic进行Pub/Sub通信，例如列表中有/1000118502/test9/update，且设备拥有的权限是发布，这就意味着设备可以往这个Topic发布消息；同样，列表中/1000118502/test9/get,权限是

订阅，这就意味着设备可以从这个Topic订阅消息。

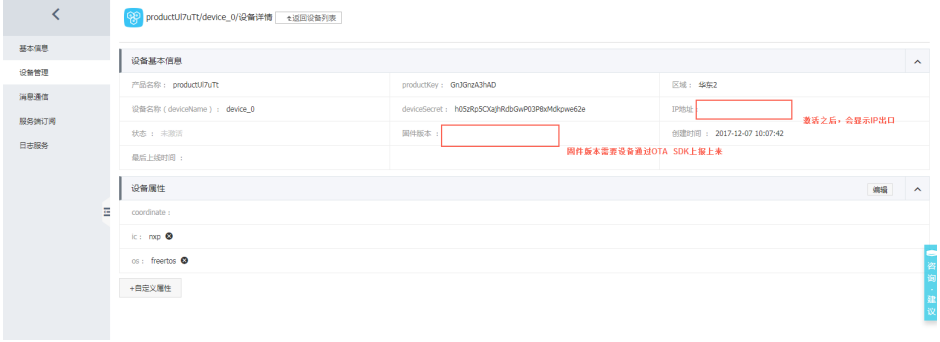
设备接入

获得productKey、设备证书以及设备的Topic这些参数，就可以基于aliyun IoT device SDK for C将设备连接上IoT Hub并进行通信，具体请参考文档设备接入

本文档主要介绍设备管理的一些功能使用。

用户在添加设备之后，可以对设备进行管理。

进入设备详情之后，用户可以查看设备基本信息



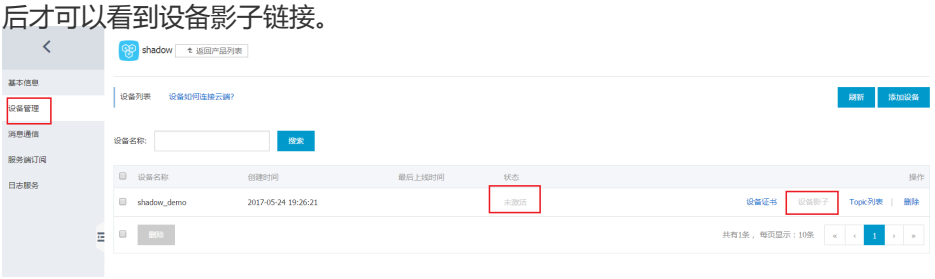
- IP地址，当设备激活连接到套件时，套件会获取到设备入网的出口IP地址
- 固件版本，需要设备连接套件时，通过设备端OTA SDK将版本号上报上来。

设备属性，用户可以自定义，属性定义之后，这些属性值都会在系统中流转，同时规则引擎支持属性函数attribute(key),key输入属性名称，这个函数可以将系统中设备定义的属性名称对应的值提取出来并通过规则引擎进行流转起来，帮助客户实现丰富的应用场景。

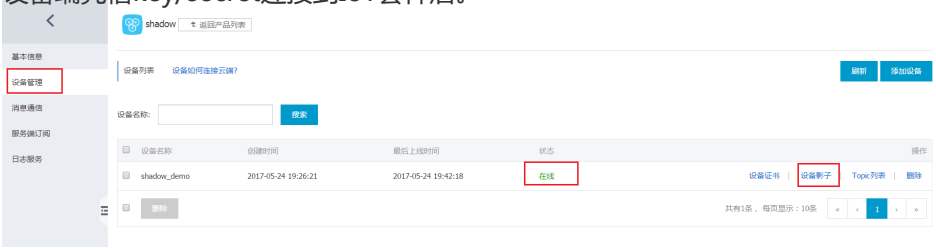
- 提供一个默认属性：坐标，并提供地图服务让用户可以方便为设备确定坐标。
- 其余属性，用户都可以自定义，输入key-value键值对，可以在控制台添加也可以通过API添加，API请参考属性接口

影子初始状态：

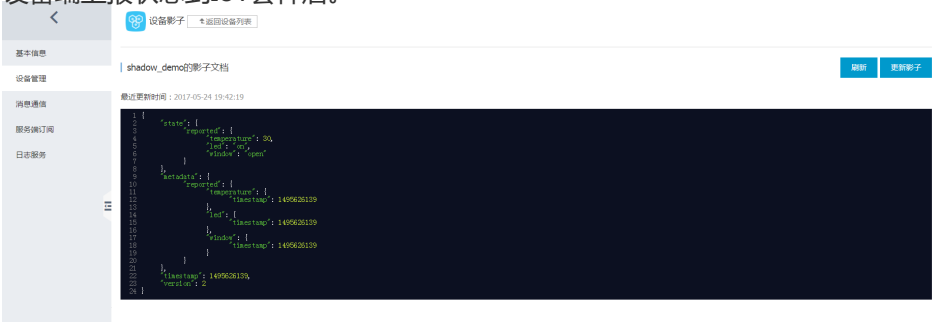
创建产品和设备后，设备处于未激活状态，设备影子链接处于无法点击状态，设备必须连接到IoT套件后才可以看到设备影子链接。



设备端凭借key/secret连接到IoT套件后。

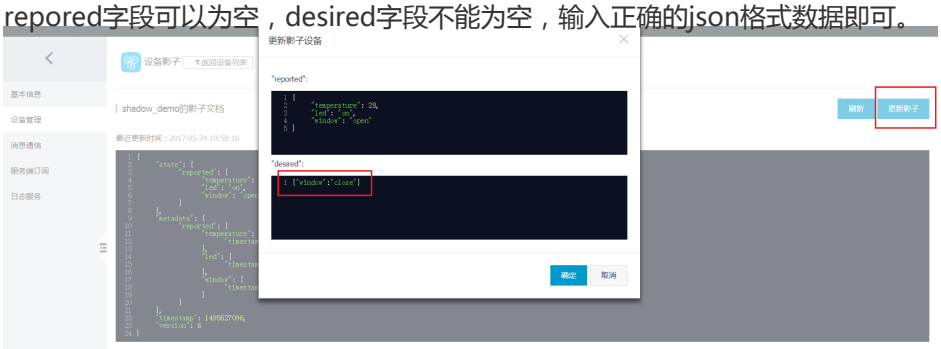


设备端上报状态到IoT套件后。



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}
}
```

用户在创建产品之后，可以在消息通信页面里配置让设备使用PUB/SUB或者使用RRPC这两种通信方式，请根据您的使用场景选择通信方式，详情请参考文档通信方式。

发布订阅（PUB/SUB）

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

云端请求设备端(RRPC)

物联网套件同样提供机制可以让云端请求设备端，并得到设备的返回结果，例如控制设备得到结果。这部分在控制台没有相应功能提供，但是阿里云物联网套件提供了相应的API完成该功能，请参考最佳实践远程控制设备并返回结果。

本文档主要介绍控制台如何配置服务端订阅。

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1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1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，请参考文档服务端快速开始

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。

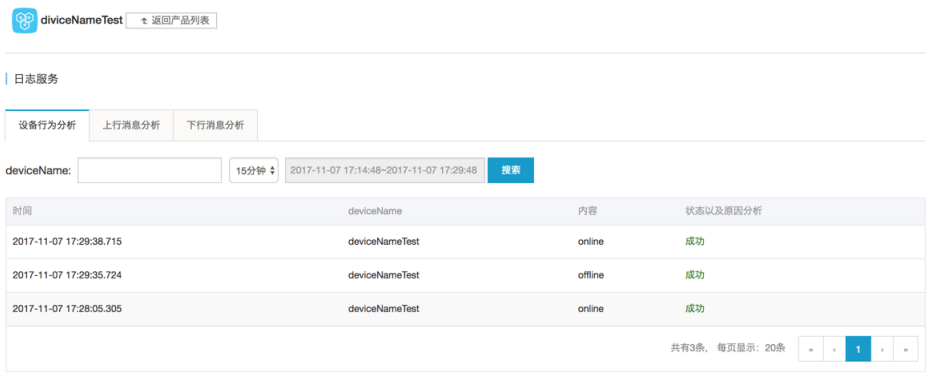
特别注意：

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：

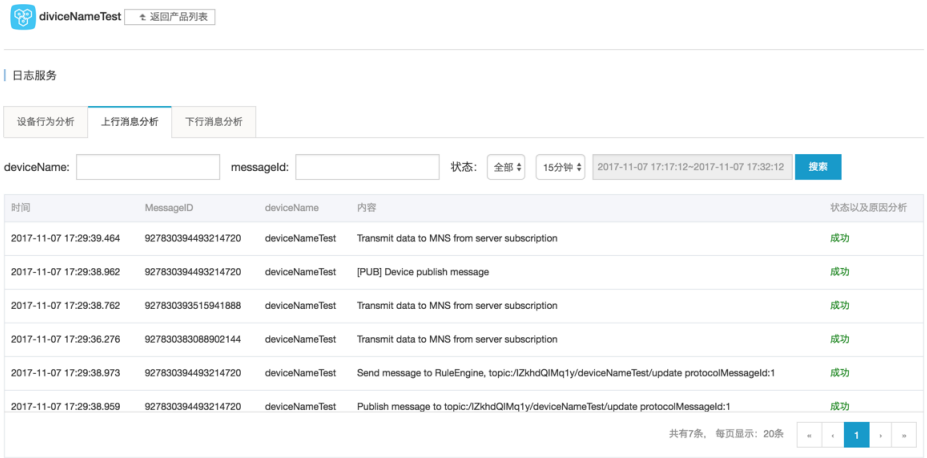


设备离线原因中英文对查表

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）

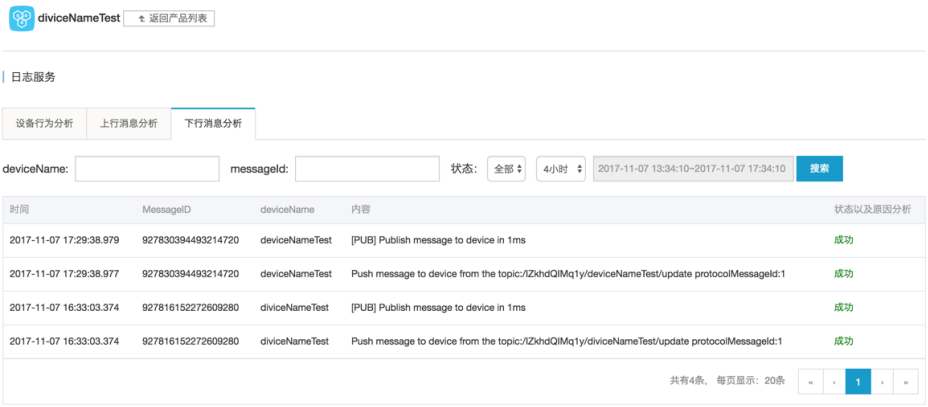
context	error reason	失败原因
Device publish message to topic:{},QoS={},protocolMessageId:{}	Rate limit:{maxQps},current qps:{}	限流{最大流量},当前qps:{}

设备发送消息到opic:{},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）

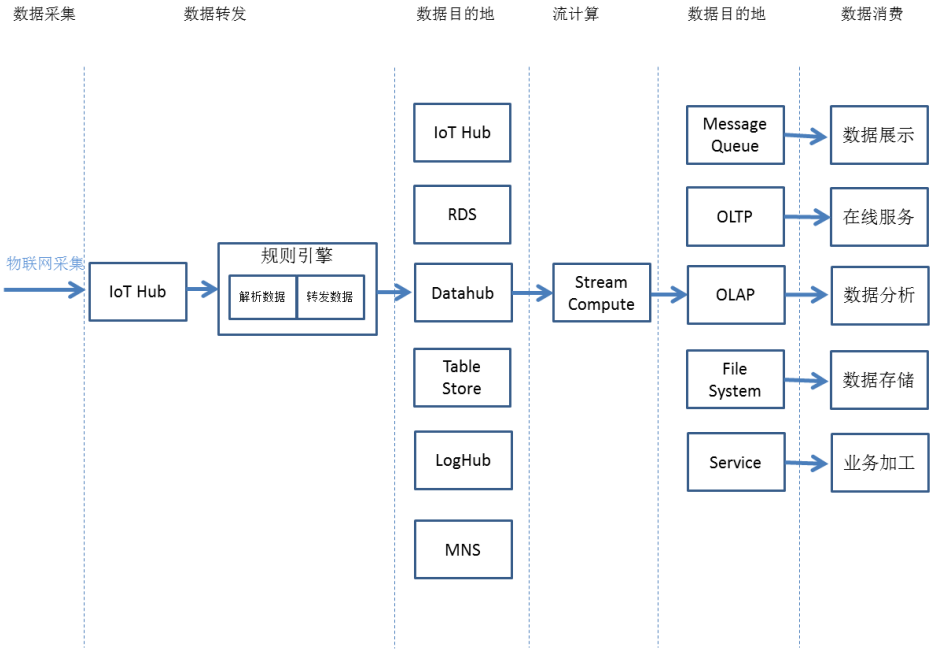
context	error reason	失败原因
Publish message to topic:{},protocolMessageId:{} 推送消息给topic:{},protocolMessageId:{}	No authorization	未授权
Publish message to device,QoS={} IoT Hub发送消息给设备	IoT hub cannot publish messages	因为服务端没有得到设备的puback，会一直发消息，直到超过50条的阈值然后IoT Hub就会无法发送消息
	Device cannot receive messages	设备端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

规则引擎

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。

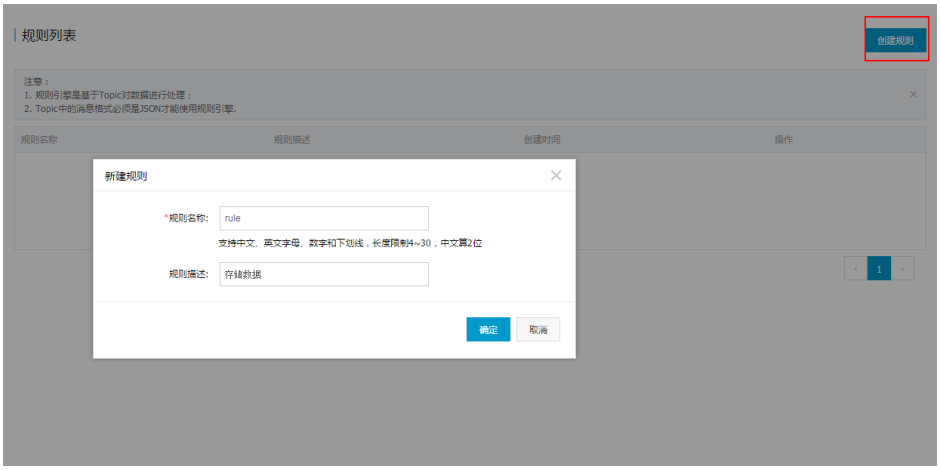


特别注意：

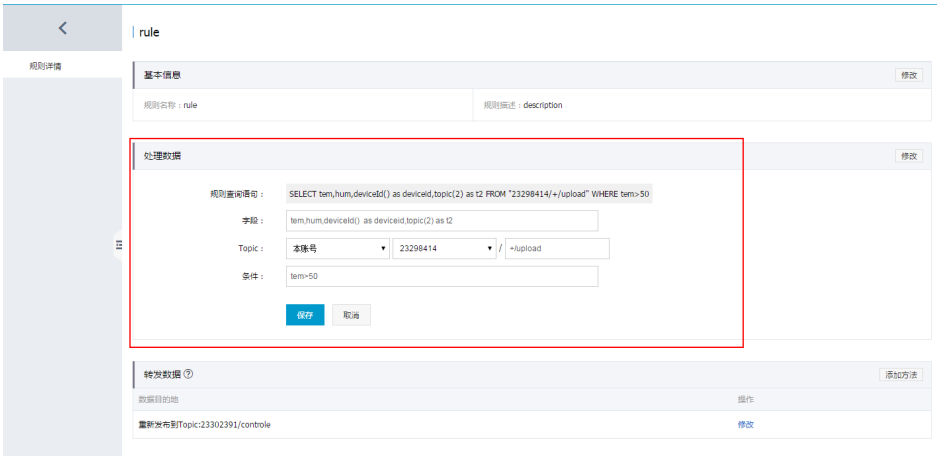
1. 规则引擎是基于Topic对数据进行处理

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



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。

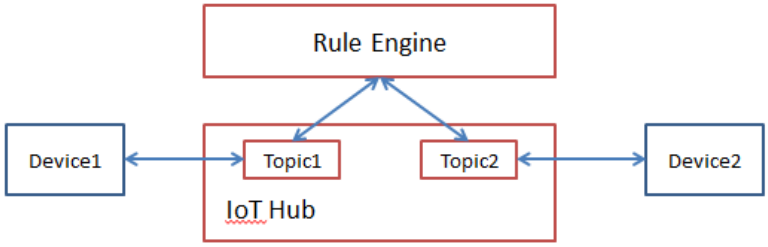
特别注意：

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。

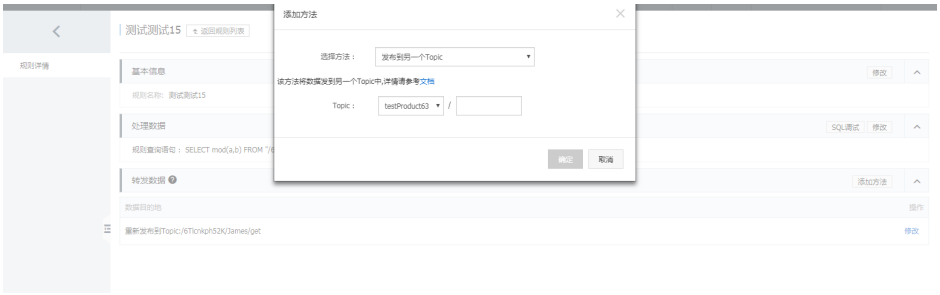
转发数据

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



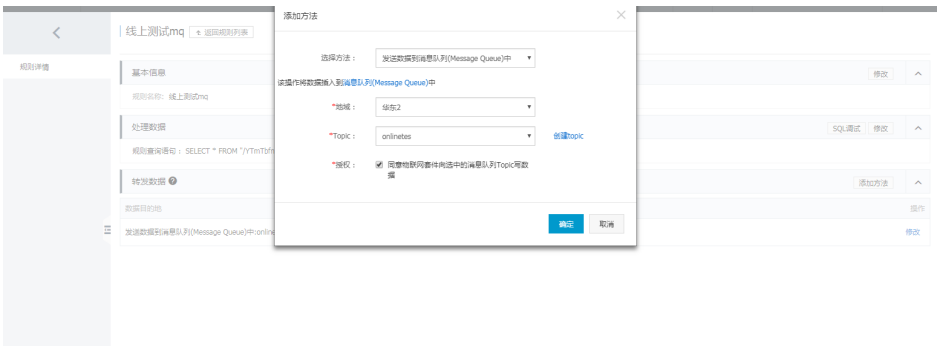
将Topic中的消息处理过后，发送到另一个Topic，可以实现M2M场景，但是又不局限于M2M，帮助节省服务端的开发。



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。

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。



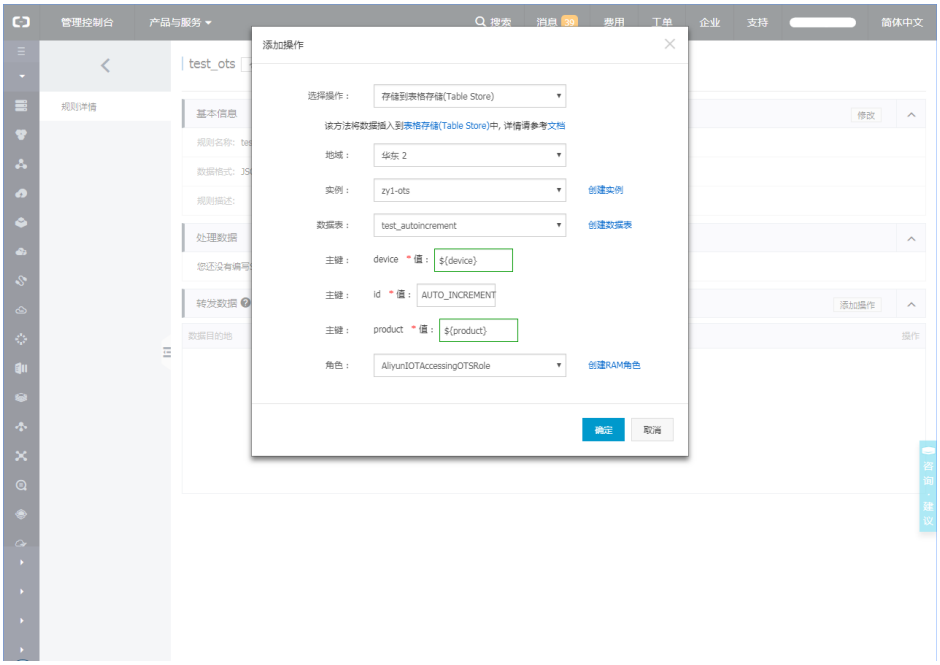
- 用户选择消息队列Topic所在地域。
- 用户根据业务选择需要写入数据的Topic
- 用户授权套件具有写入消息队列Topic的权限

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

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

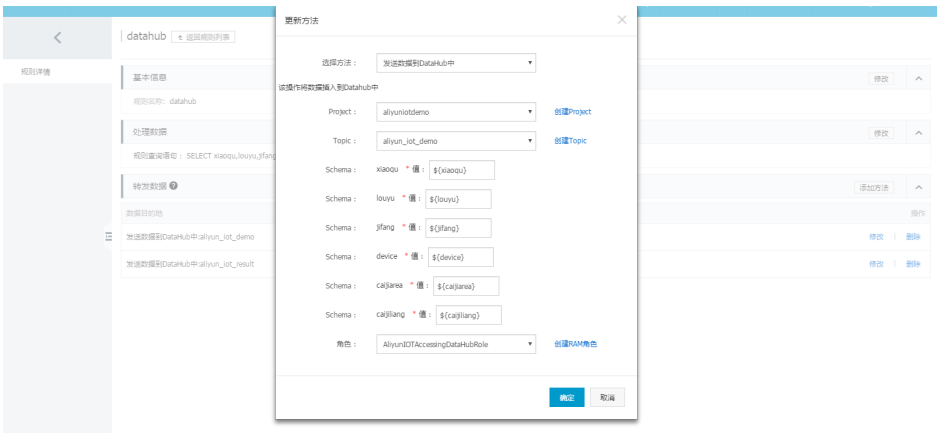
物联网套件的定位在于设备接入和设备管理，对于数据存储和计算，物联网套件会将这部分工作交给阿里云其他云产品。

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。

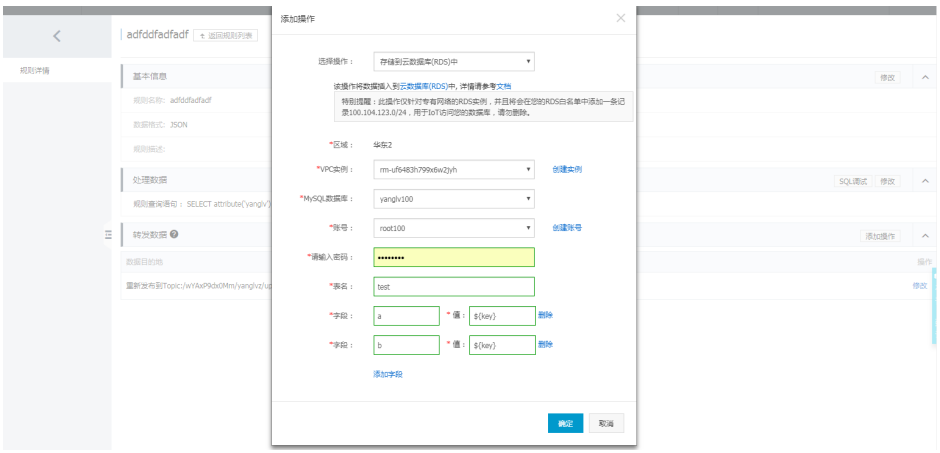
特别注意：将规则引擎筛选出来的数据映射到对应的Schema，需要使用${}，如果不使用的话，存到表中的将会是一个常量；Schema与规则引擎的的数据类型必须保持一致，不然无法存储。

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

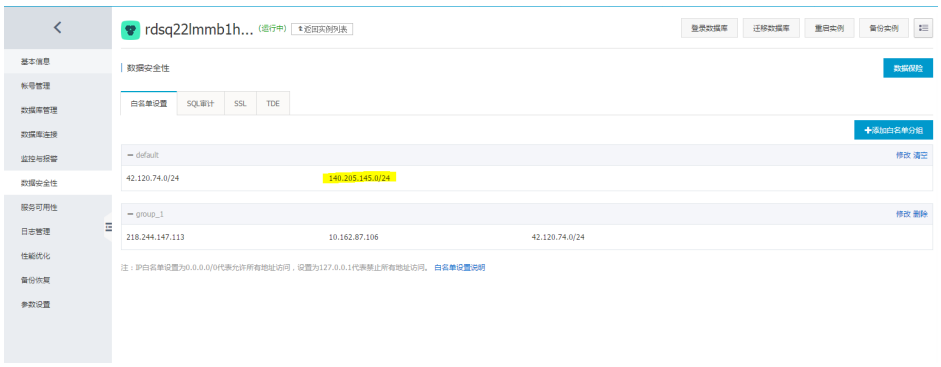
> 假如规则引擎的SQL：SELECT tem FROM mytopic.
>
> 假如RDS数据库有一张表，表中有tem字段，类型是String。
>
> **存储可以在控制台上配置，字段填入的是RDS数据表中的字段，例如tem，值填入的是规则引擎筛选出来的JSON字段，例如${tem},这里要强调两点，需要使用${}，如果不使用的话，存到表中的将会是一个常量，例如填入tem，那数据表存入就是tem这个常量；字段与值的数据类型必须保持一致，不然无法存储。**

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。

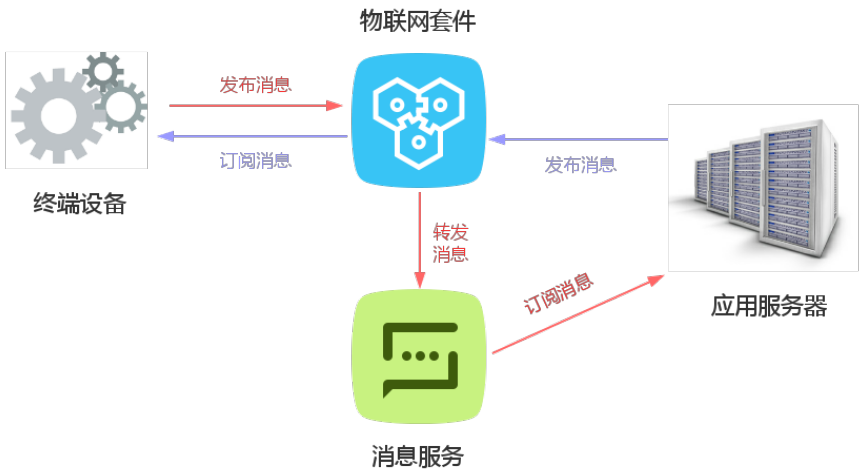
- 规则引擎为了连接RDS数据库，需要使用用户的账号去连接，这就需要用户提供账号和密码给规则引擎。**这里面需要强调的是规则引擎取得用户的账号后，只是负责将规则匹配的数据写进数据库中，不会做其他操作。**
- 规则引擎只是负责将数据写进RDS数据表中，不会去帮用户去创建数据表，所以用户需要在RDS中自行创建数据表。
- **存储需要用户自行保证字段与值的数据类型一致，不然会导致写入不成功；值应该使用函数${}，这样才能存入JSON数据中的value，不然的话将会存入一个常量。**

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

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

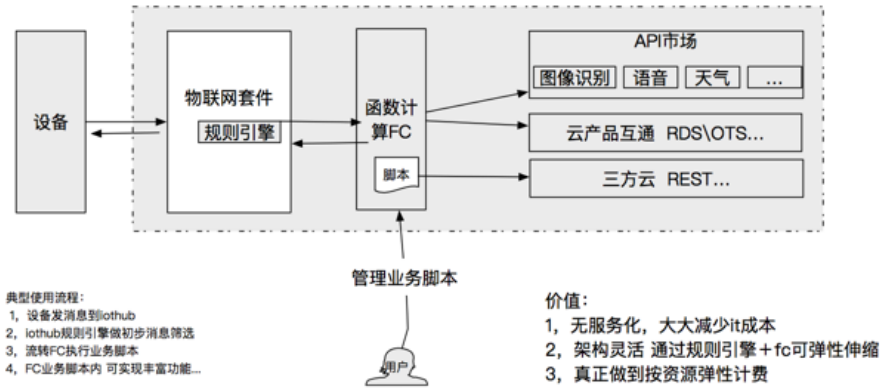
这些IP段不能删除，不然物联网套件就无法连接HiTSDB，进而也就无法将数据写进HiTSDB数据库中。如果没有此记录请手工添加。

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。

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：

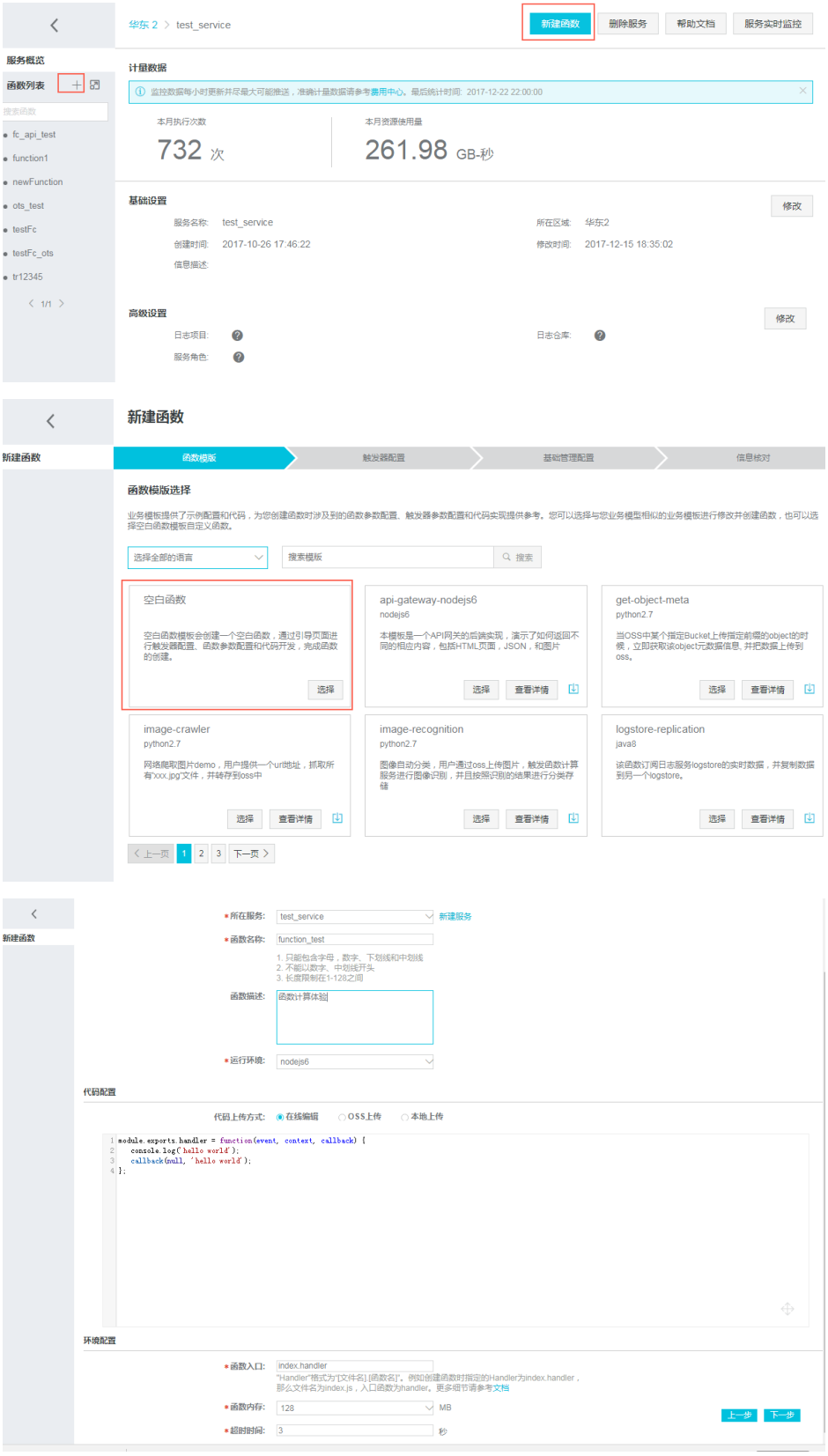
FC控制台操作

FC控制台

1、创建服务

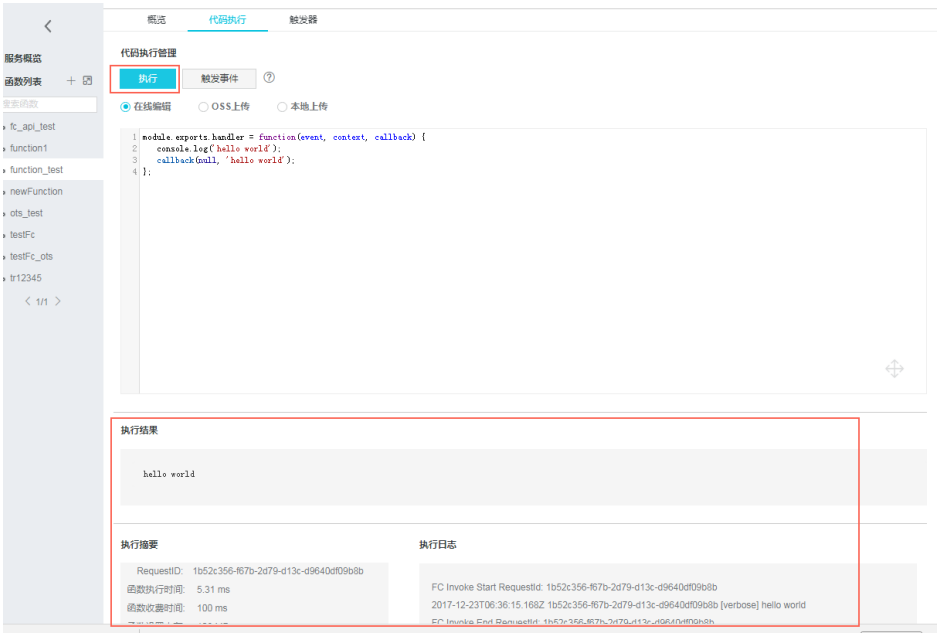
2、创建函数

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

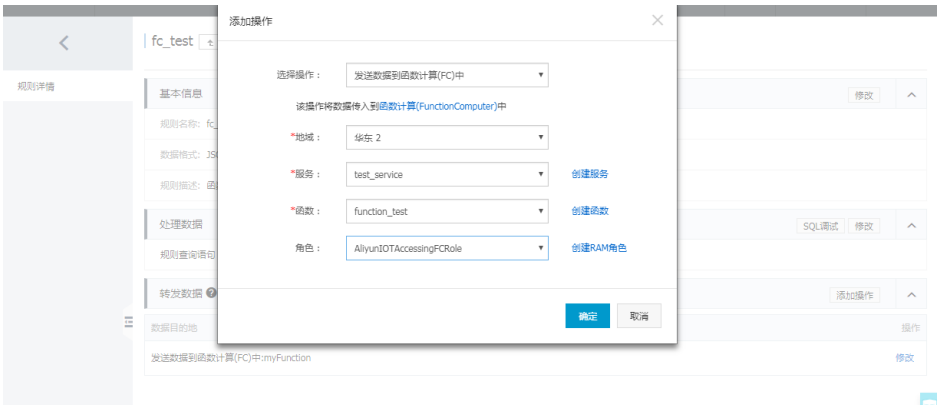
函数成功创建之后，可以直接在fc的控制台执行，以验证函数执行情况。函数计算会直接将函数的输出打印在控制台上，同时还有请求的相关信息。



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



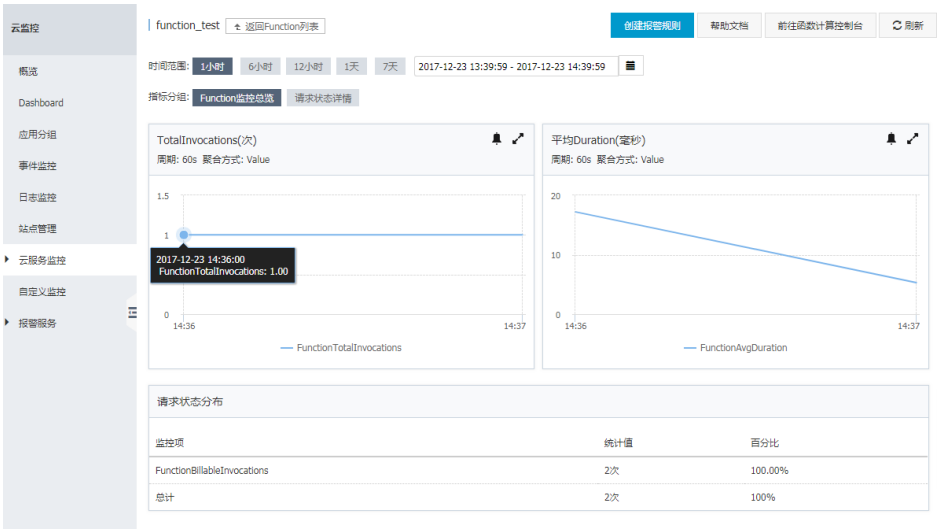
操作说明：

- 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

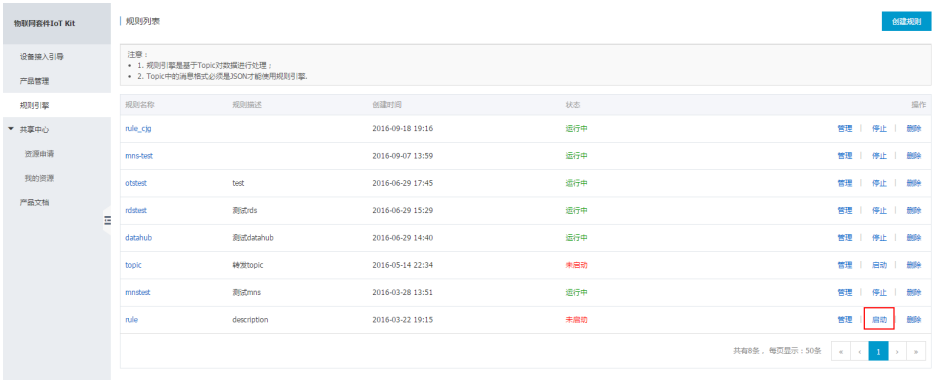
配置好方法之后，运行该规则，就可以将经过SQL语法处理过后的数据转发到函数计算中。

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

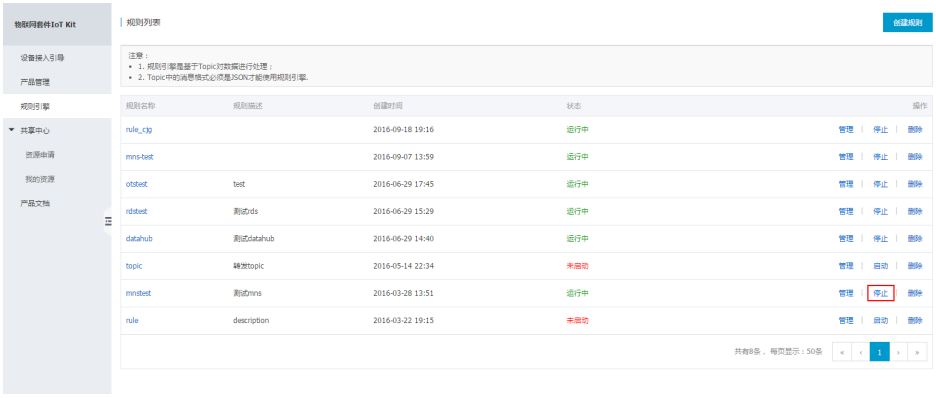


当编写完SQL，添加好方法之后，用户就可以启动规则。这样一旦有消息Pub到SQL语法中的Topic里时，规则就触发，执行规则。



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

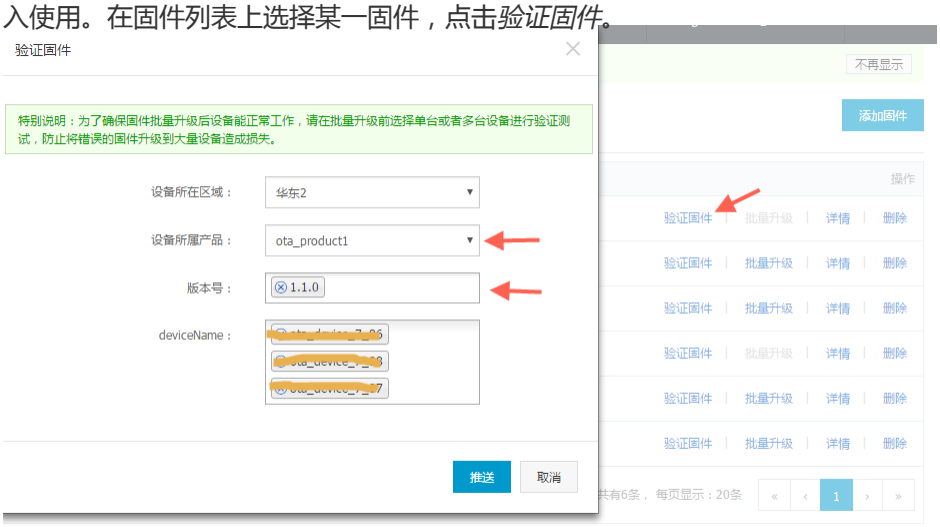
目前只有华东2节点支持固件服务。进入控制台，点击产品管理，选择华东2节点，可以在左侧扩展服务菜单栏看到固件升级菜单。点击固件升级，在右上角有个添加固件按钮，点击*添加固件*。

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。