Alibaba Cloud IoT Platform

Quick Start

Issue: 20190718

MORE THAN JUST CLOUD | Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades , adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults " and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity , applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified , reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates . The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
⛔	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	⛔ Danger: Resetting will result in the loss of user configuration data.
⚠	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	⚠ Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
📋	This indicates warning informatio n, supplementary instructions, and other content that the user must understand.	❗ Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	📋 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus , page names, and other UI elements.	Click OK.
`Courier font`	It is used for commands.	Run the `cd / d C :/ windows` command to enter the Windows system folder.
Italics	It is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	It indicates that it is a optional value, and only one item can be selected.	`ipconfig` *[-all|-t]*

Style	Description	Example
{} or {a|b}	**It indicates that it is a required value, and only one item can be selected.**	`swich` *{stand | slave}*

Contents

1 Create products and devices

The first step in using IoT Platform is to create products and devices. A product is a collection of devices that typically have the same features. You can manage devices in batch by managing the corresponding product.

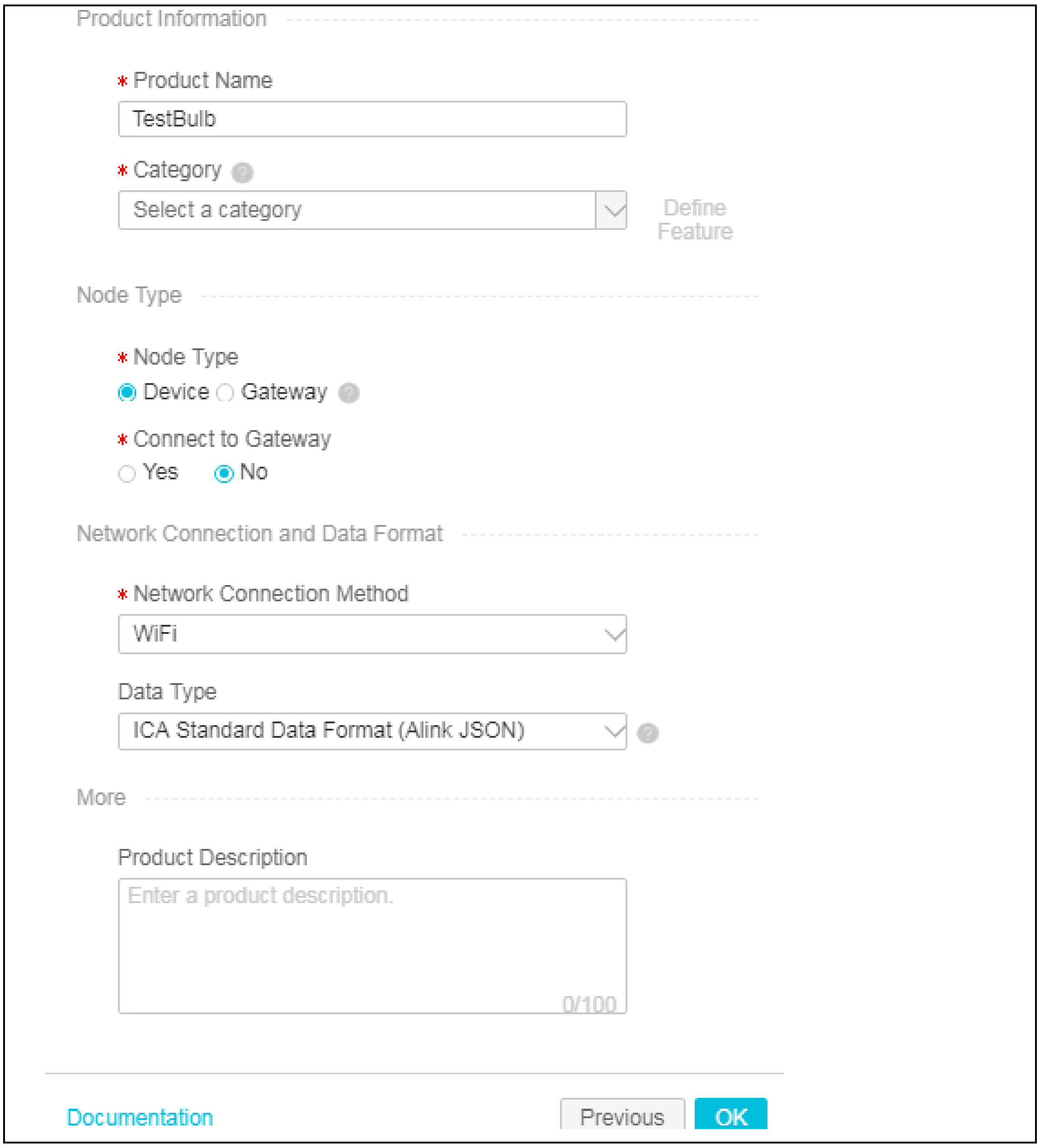
Procedure

1. Log on to the IoT Platform console.

2. Create a product.

   a) In the left-side navigation pane, click Devices > Product. On the Products page, click Create Product.

   b) Enter all the required information and then click OK.



Product Information

* Product Name

TestBulb

* Category

Select a category

Node Type

* Node Type

[✓ Device] Gateway

* Connect to Gateway

Yes [✓ No]

Network Connection and Data Format

* Network Connection Method

WiFi

Data Type

ICA Standard Data Format (Alink JSON)

More

Product Description

Enter a product description.

0/100

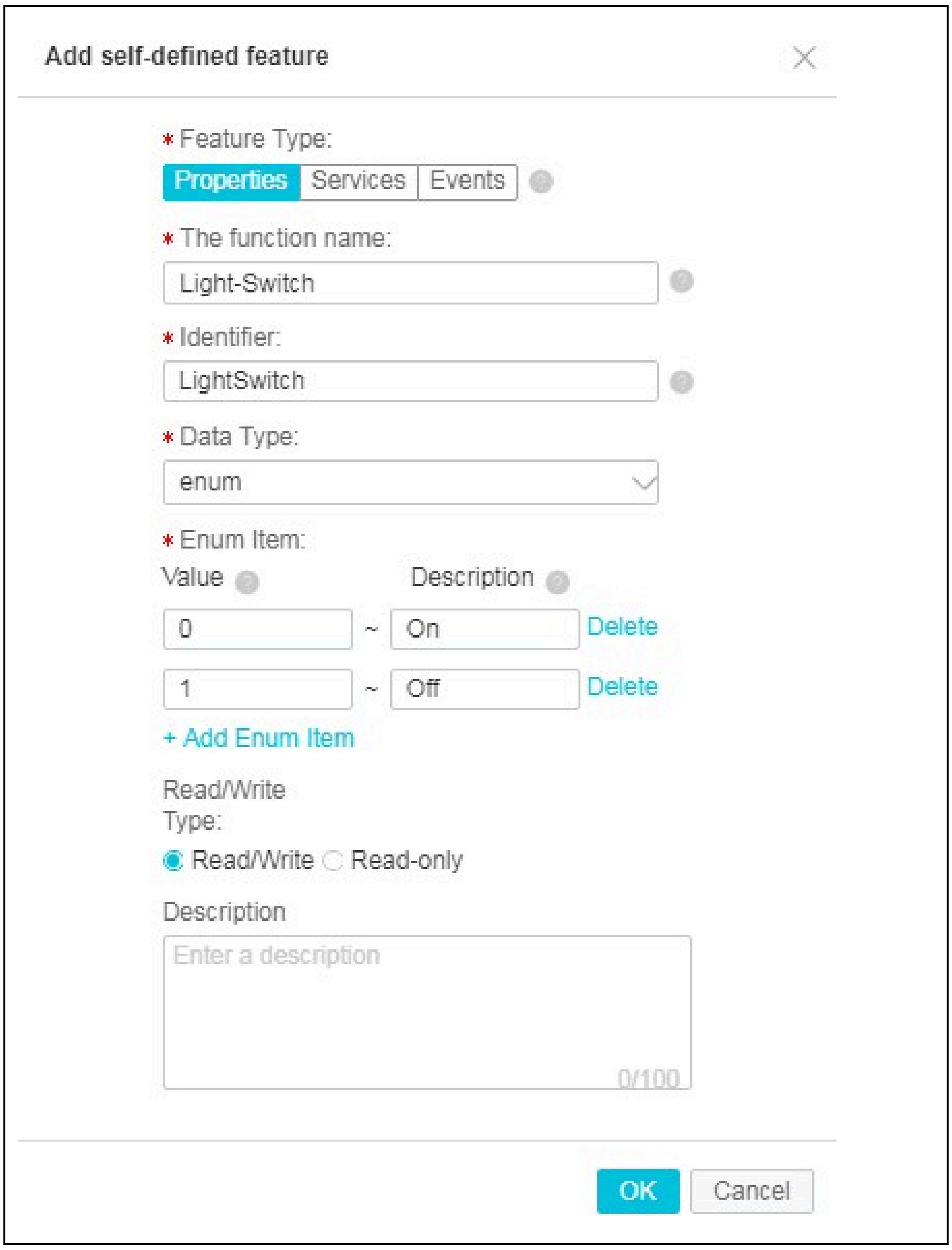
The parameters are described as follows:

Parameter	Description
Product Name	In this example, the product is named as TestBulb. The product name must be unique within the account. A Product name is 4 to 30 characters in length, and can contain Chinese characters, English letters, digits and underscores. A Chinese character counts as two characters .
Category	In this example, the product category is Custom category indicating that features of the product is self-defined.
Node Type	In this example, the node type is Device. · Device: Indicates that devices of this product cannot be mounted with sub-devices. This kind of devices can connect to IoT Platform directly or as sub-devices of gateway devices. · Gateway: Indicates that devices of this product connect to IoT Platform directly and can be mounted with sub-devices. A gateway can manage sub-devices, maintain topological relationships with sub-devices, and synchronize topological relationships to IoT Platform.
Connect to Gateway Note: This parameter appears if the node type is Device.	Indicates whether or not devices of this product can be connected to gateways as sub-devices. · Yes: Devices of this product can be connected to a gateway. · No: Devices of this product cannot be connected to a gateway.
Network Connection Method Note: This parameter appears if you select No for Connect to Gateway.	Select a network connection method for the devices. In this example, WiFi is selected.

Parameter	Description
Data Type	Select a format in which devices exchange data with IoT Platform. In this example, ICA Standard Data Format (Alink JSON) is selected. ICA Standard Data Format (Alink JSON): The standard data format defined by IoT Platform for device and IoT Platform communication.
Product Description	Describe the product information. You can enter up to 100 characters.

Once the product is created successfully, it appears in the product list.

3. Define features for the product.
    a) In the product list, find the product and click View.
    b) On the product details page, click Define Feature.
    c) Click Add Feature corresponding to Self-Defined Feature.
    d) Define a property. In this example, a light switch property is defined. 0 indicates turning the light on and 1 indicates turning the light off.


Add self-defined feature

* Feature Type: [✓ Properties] Services Events

* The function name: Light-Switch

* Identifier: LightSwitch

* Data Type: enum

* Enum Item:

Value		Description	
0	~	On	Delete
1	~	Off	Delete

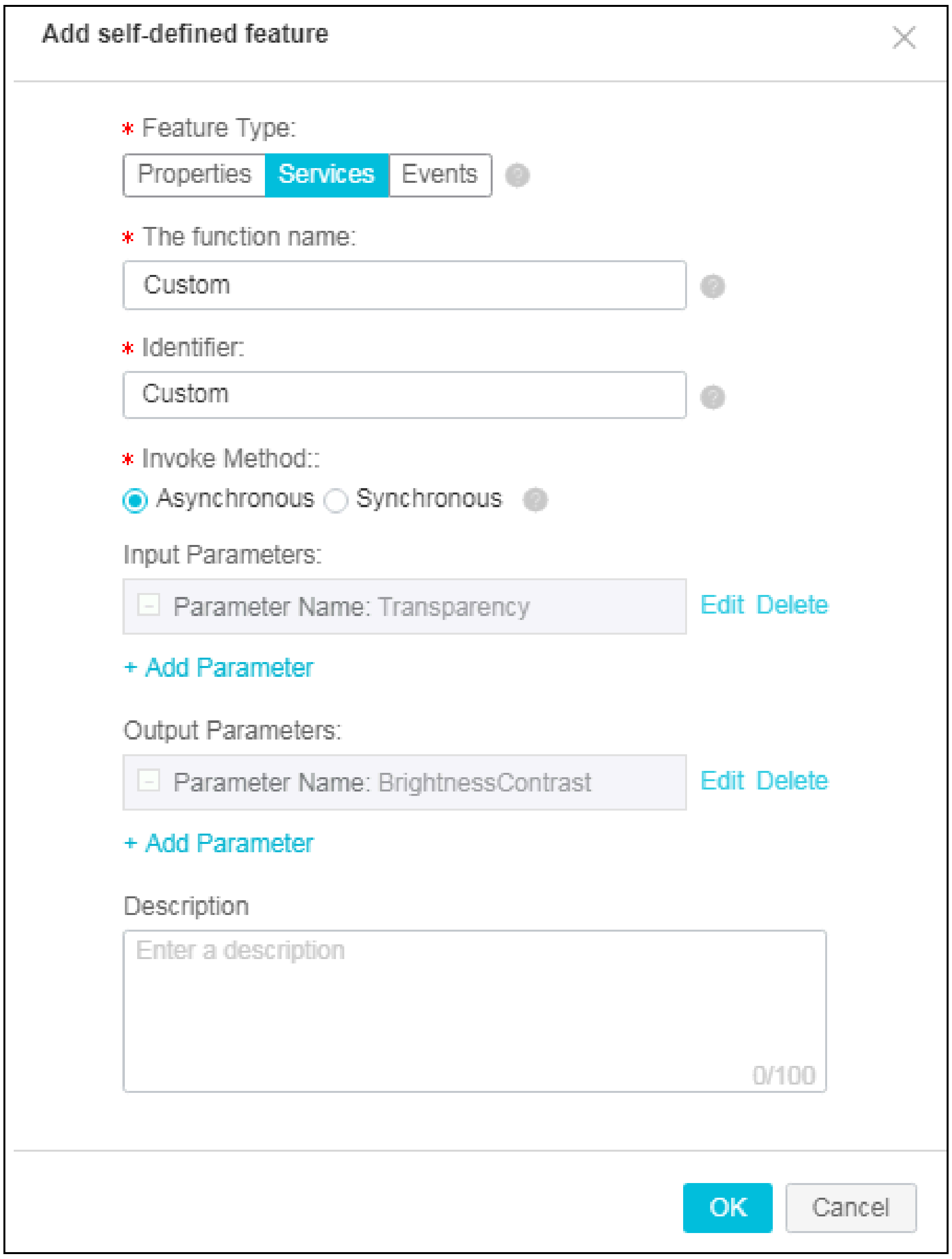
+ Add Enum Item

Read/Write Type: [✓ Read/Write] Read-only

Description: Enter a description 0/100

OK Cancel

    e) Define a service. For example, you can add an input parameter for adjusting the brightness of the bulb, and add an output parameter for the bulb to report the brightness contrast between the bulb and the room environment.


Add self-defined feature

* Feature Type:

Properties [✓ Services] Events

* The function name:

Custom

* Identifier:

Custom

* Invoke Method::

[✓ Asynchronous] Synchronous

Input Parameters:

Parameter Name: Transparency Edit Delete

+ Add Parameter

Output Parameters:

Parameter Name: BrightnessContrast Edit Delete

+ Add Parameter

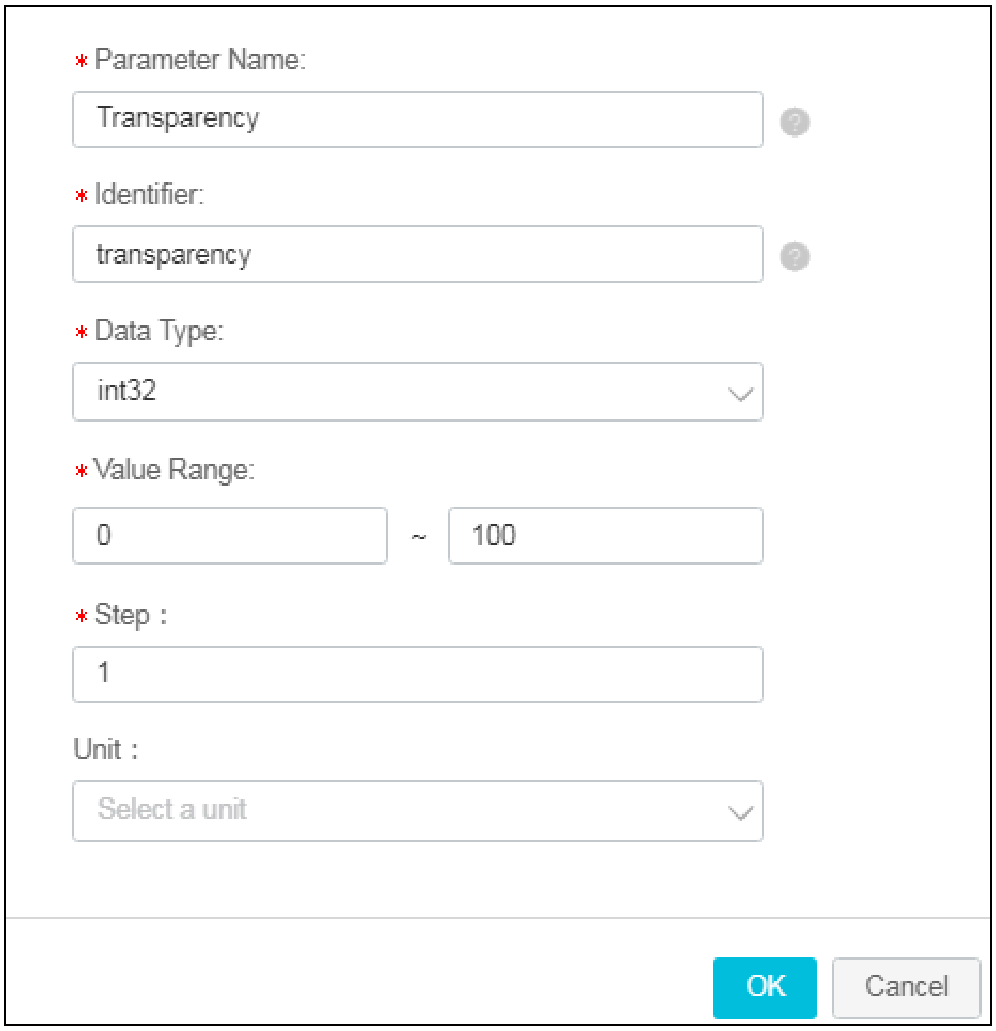
Description

Enter a description

0/100

OK Cancel

The following figure shows an example of input parameter.



* Parameter Name: Transparency

* Identifier: transparency

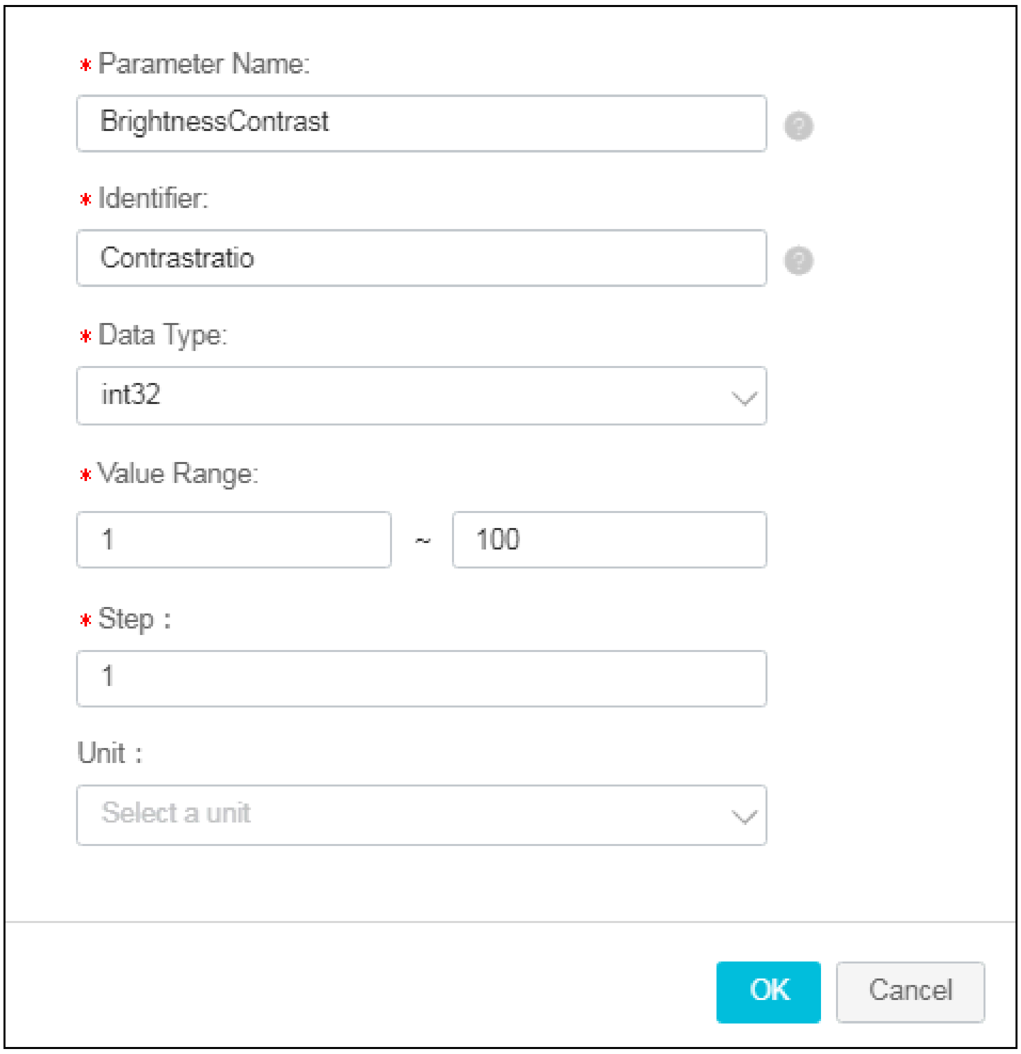
* Data Type: int32

* Value Range: 0 ~ 100

* Step : 1

Unit : Select a unit

The following figure shows an example of output parameter.


* Parameter Name:

BrightnessContrast

* Identifier:

Contrastratio

* Data Type:

int32

* Value Range:

1 ~ 100

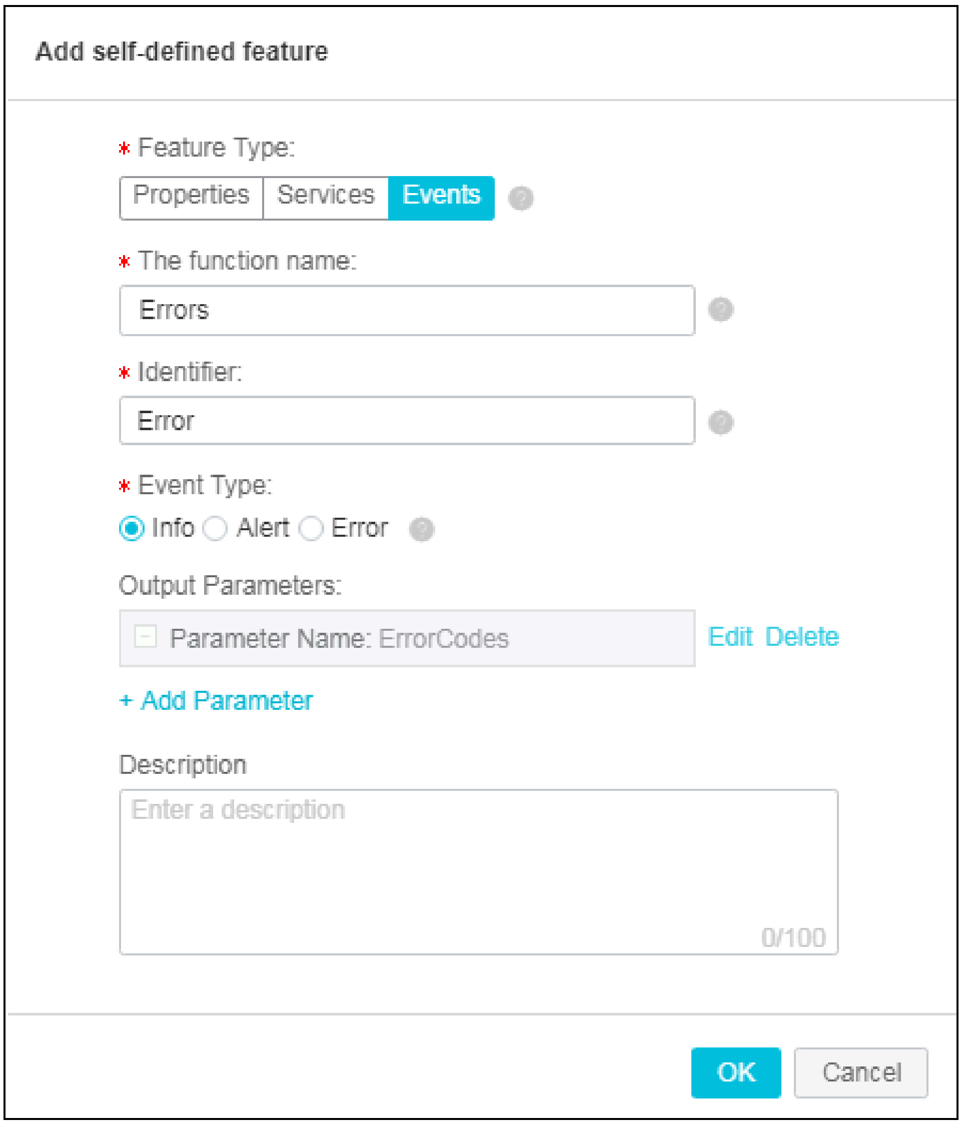
* Step :

1

Unit :

Select a unit

f) Define an event. You can define an event for devices to report errors.



Add self-defined feature

* Feature Type: Properties Services [✓ Events]

* The function name: Errors

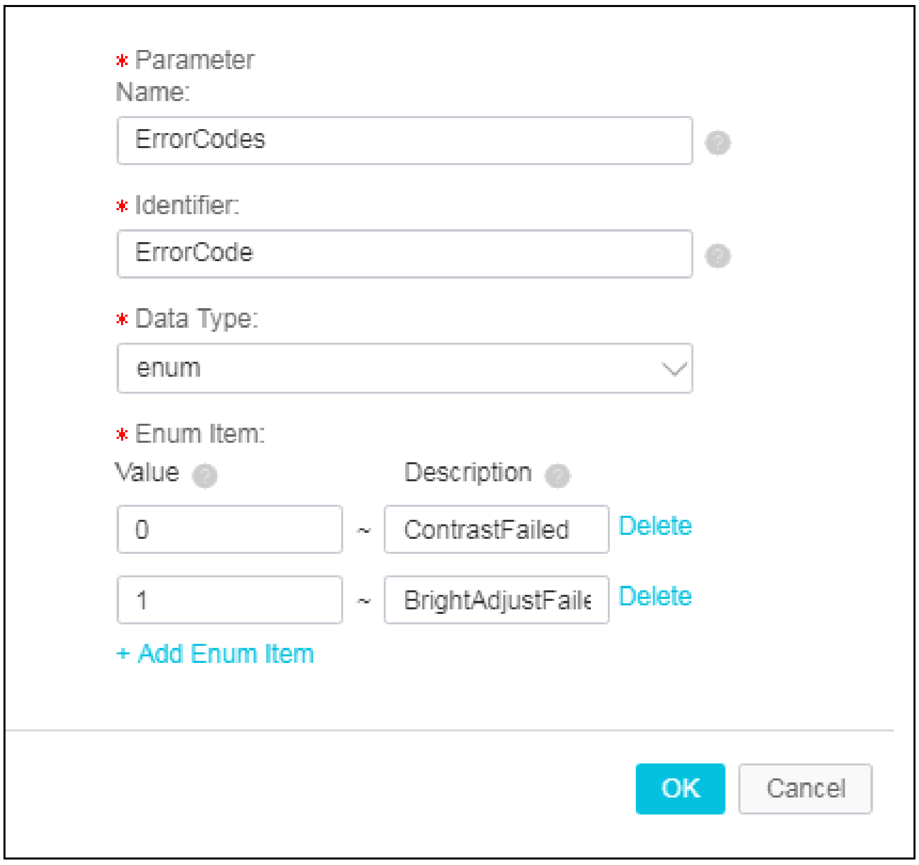
* Identifier: Error

* Event Type: [✓ Info] Alert Error

Output Parameters: Parameter Name: ErrorCodes

Description: Enter a description 0/100

The following figure shows an example of output parameter.



* Parameter Name: ErrorCodes

* Identifier: ErrorCode

* Data Type: enum

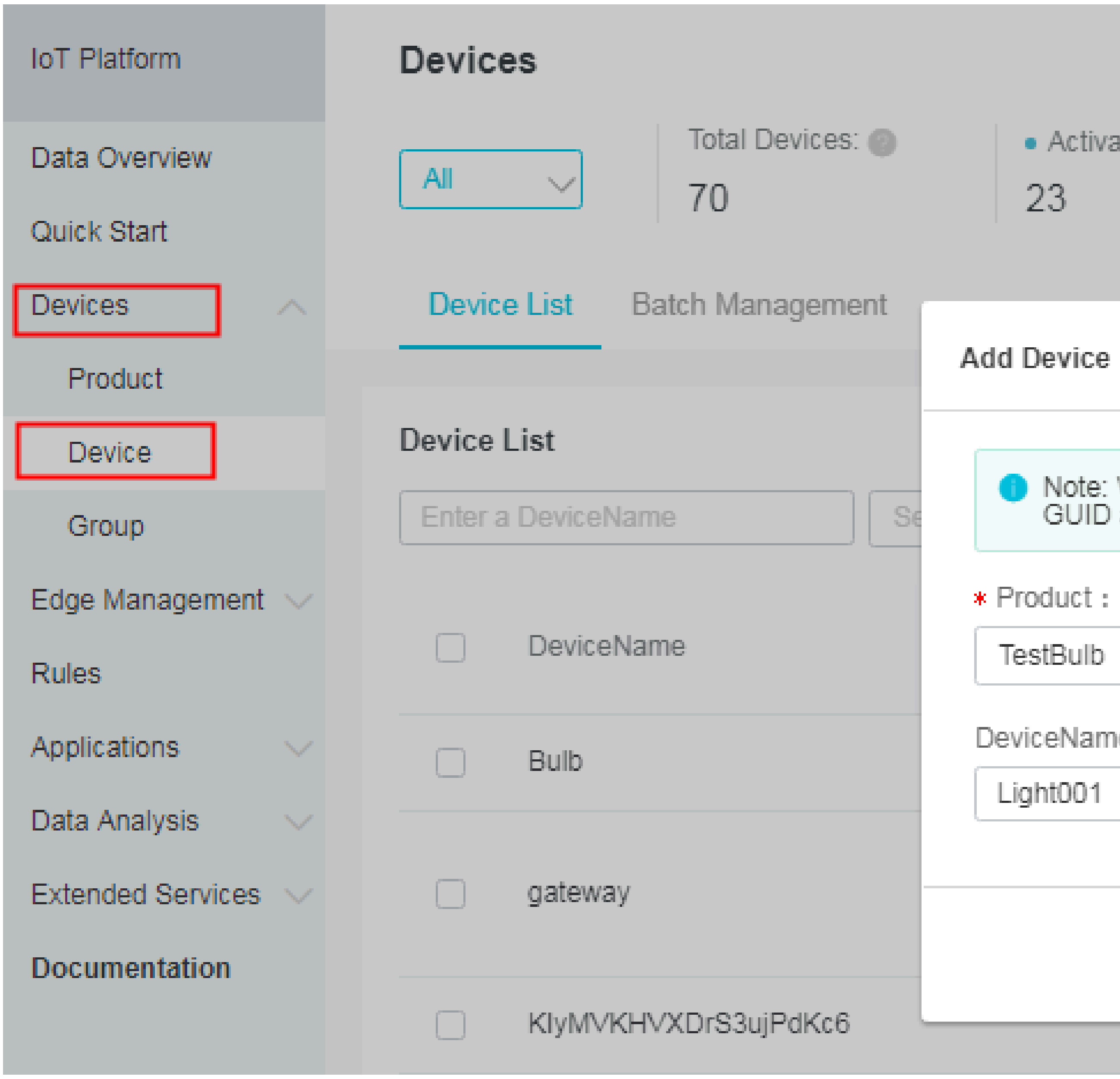
* Enum Item:

Value		Description	
0	~	ContrastFailed	Delete
1	~	BrightAdjustFaile	Delete

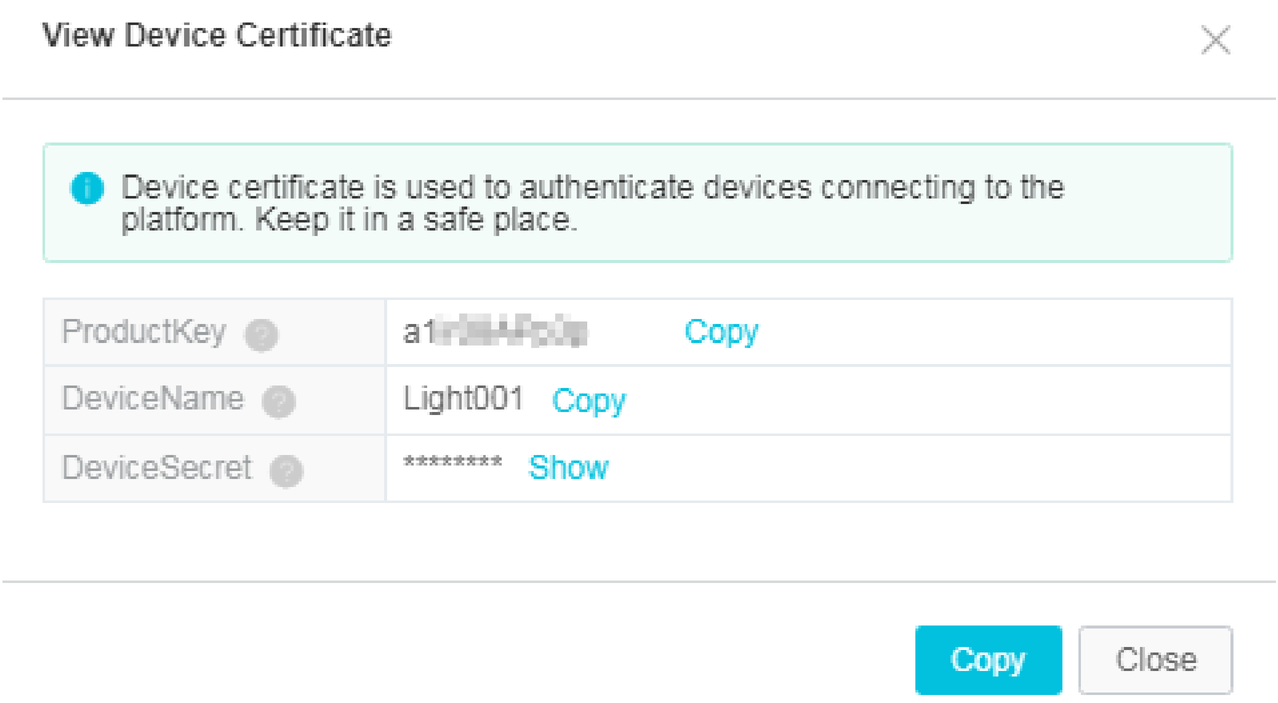
4. Create a device.

   a) In the left-side navigation pane, click Devices > Device.

   b) On the device management page, click Add Device. Select a product to which the device to be created belongs, and then enter a name for the device (DeviceName). Click OK.



   c) Save the device certificate information. The certificate information includes ProductKey, DeviceName, and DeviceSecret. Keep this information confidential, because it is the certificate that will be used for device authentication when the device is connecting to IoT Platform.


View Device Certificate

Device certificate is used to authenticate devices connecting to the platform. Keep it in a safe place.

ProductKey	a1[illegible] Copy
DeviceName	Light001 Copy
DeviceSecret	******** Show

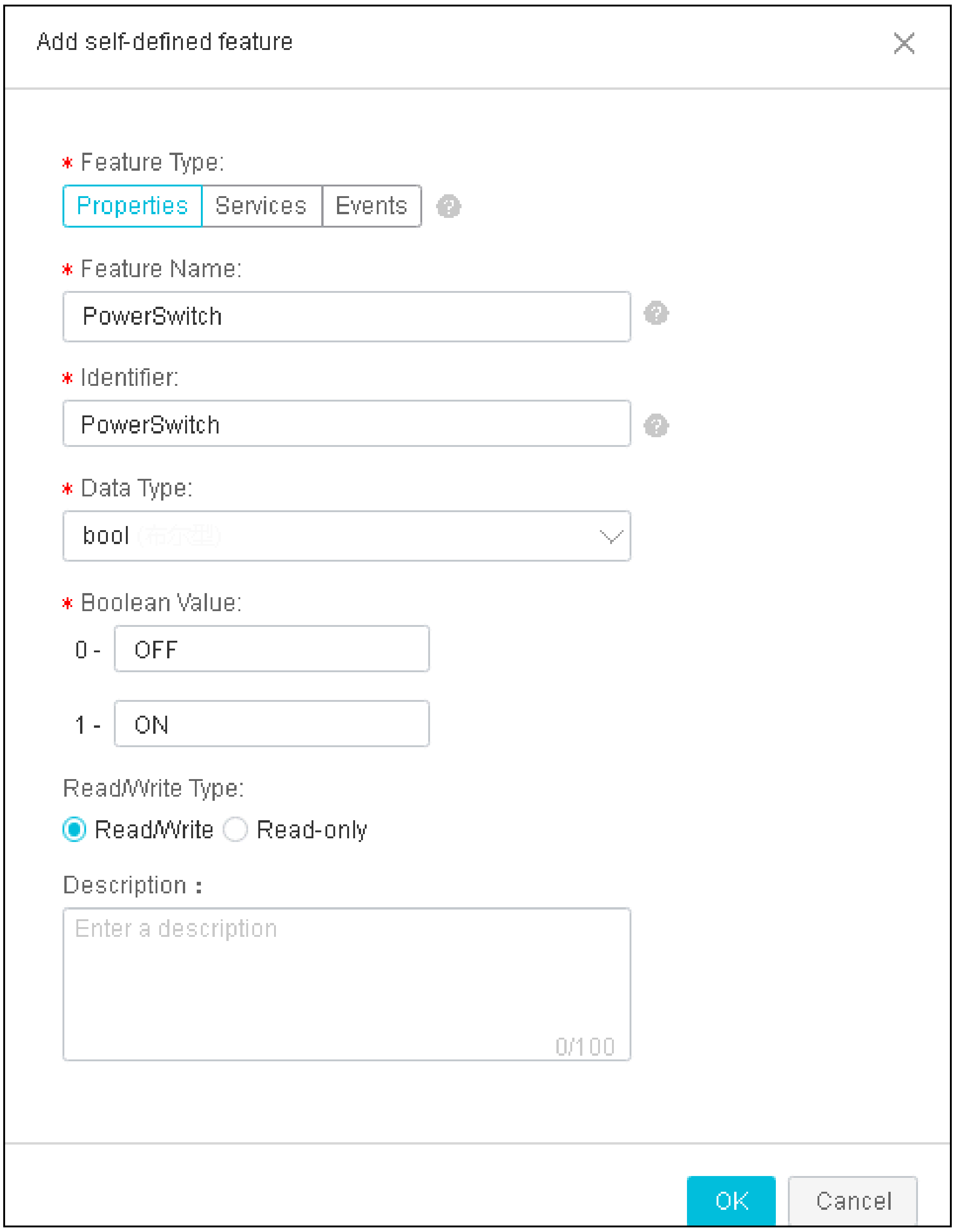
2 Define product features

IoT Platform allows you to define features for products. You can use a TSL model to describe product features, including properties, services, and events. The TSL model makes it easy to manage products and data transmission. After you create a product, you can define a TSL model to describe product features. Devices under this product automatically inherit its features.

Procedure

1. In the product list, select the product and click View.
2. On the Product Details page, click Define Feature.
3. In the Self-Defined Feature section, click Add Feature.

4. As shown below, add a property to define a switch. Click OK.



Add self-defined feature

* Feature Type:

[✓ Properties] Services Events

* Feature Name:

PowerSwitch

* Identifier:

PowerSwitch

* Data Type:

bool (布尔型)

* Boolean Value:

0 - OFF

1 - ON

Read/Write Type:

[✓ Read/Write] Read-only

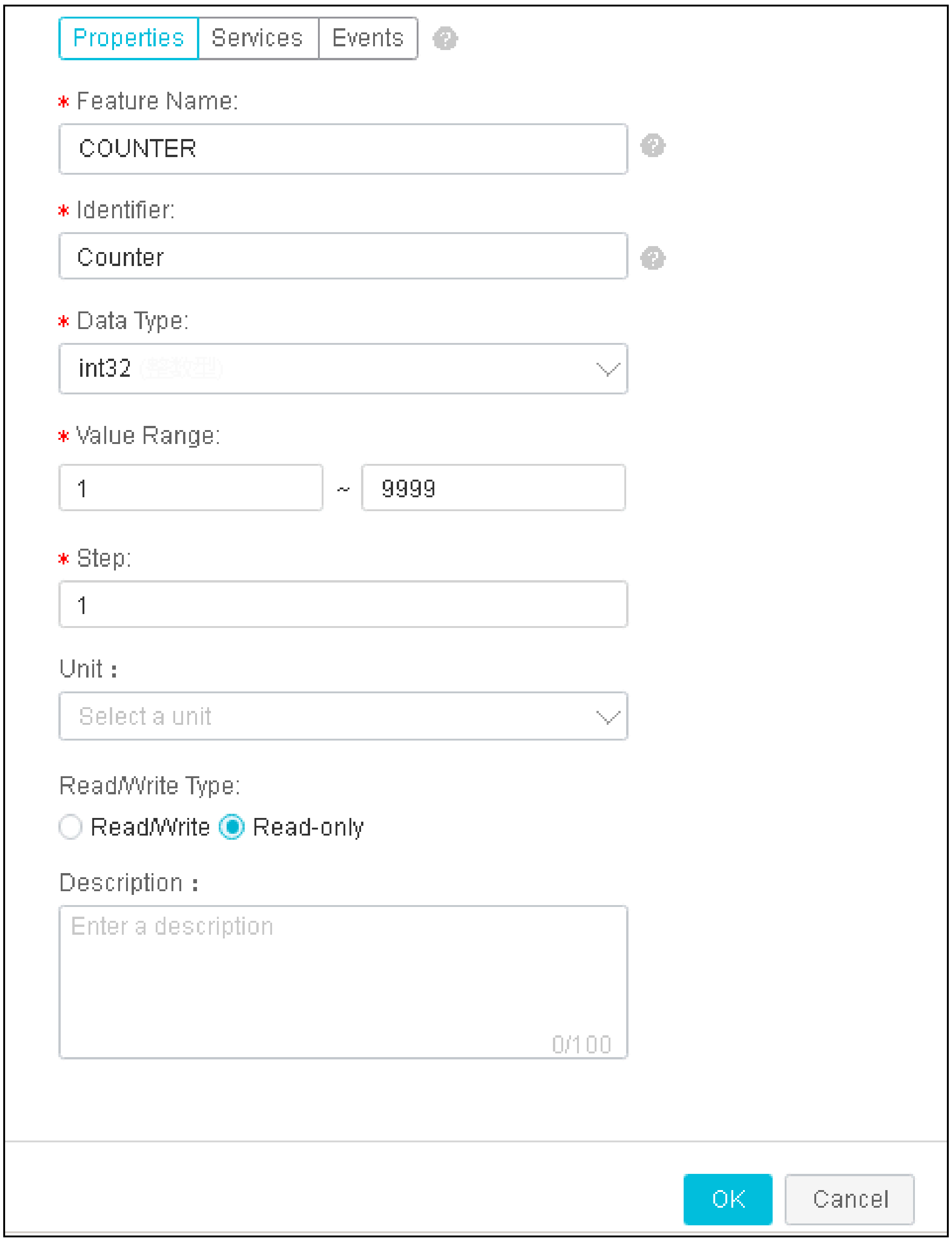
Description :

Enter a description

0/100

OK Cancel

5. As shown below, add a property to define a counter. Click OK.



* Feature Name: COUNTER

* Identifier: Counter

* Data Type: int32 (整数型)

* Value Range: 1 ~ 9999

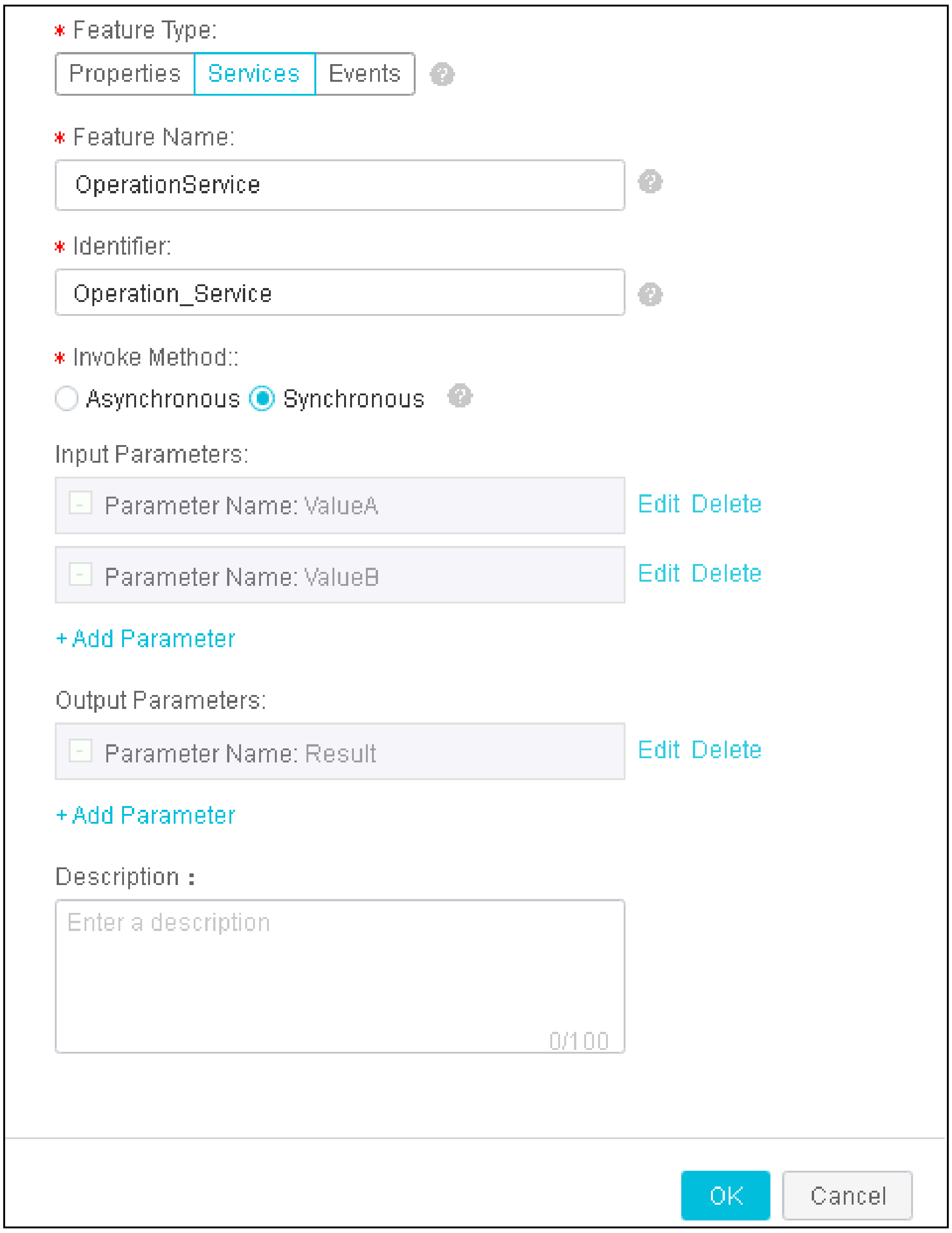
* Step: 1

Unit : Select a unit

Read/Write Type: Read/Write [✓ Read-only]

Description : Enter a description 0/100

6. As shown below, add a service to support numerical calculations. Click OK.



* Feature Type:

Properties [✓ Services] Events

* Feature Name:

OperationService

* Identifier:

Operation_Service

* Invoke Method::

Asynchronous [✓ Synchronous]

Input Parameters:

Parameter Name: ValueA Edit Delete

Parameter Name: ValueB Edit Delete

+ Add Parameter

Output Parameters:

Parameter Name: Result Edit Delete

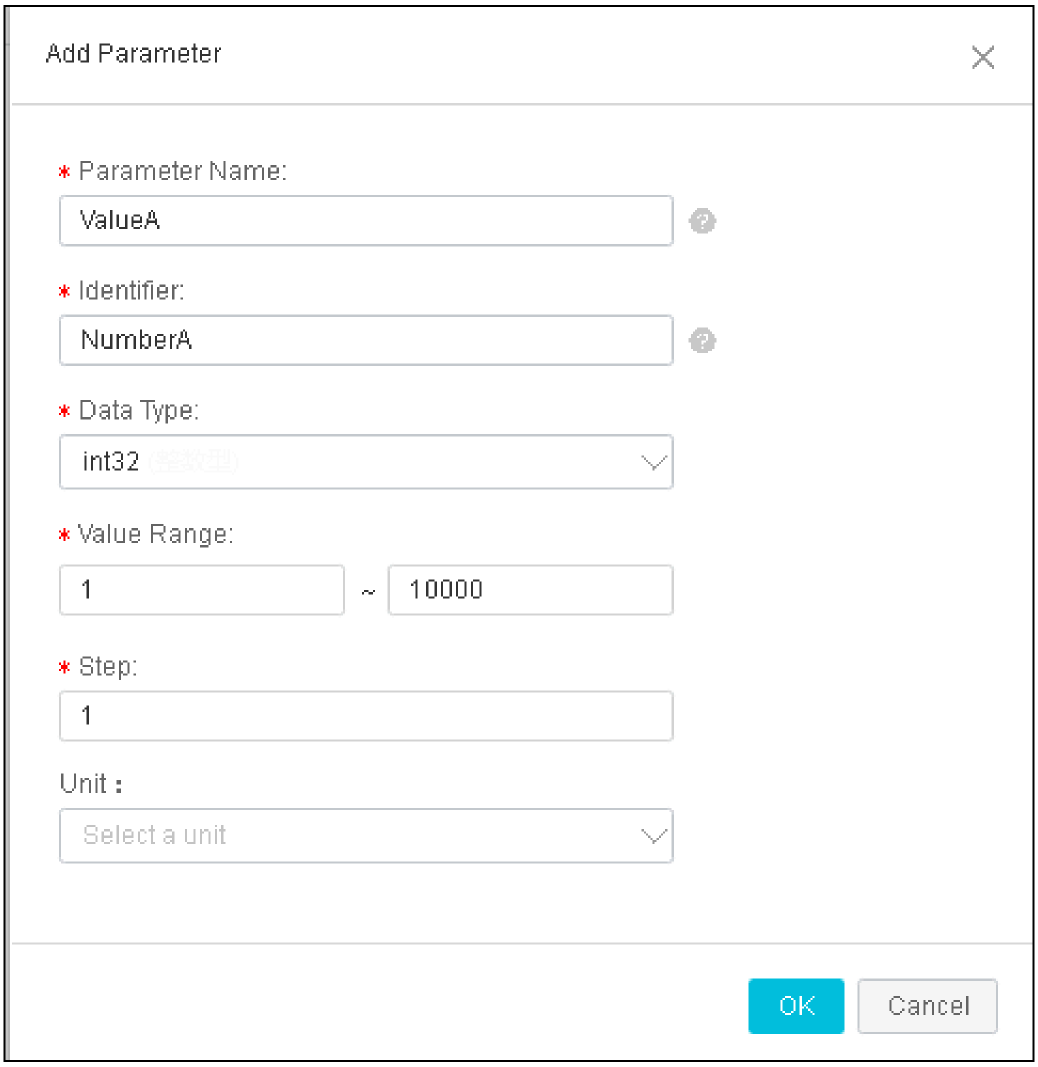
+ Add Parameter

Description :

Enter a description

0/100

- Value A is defined as follows:

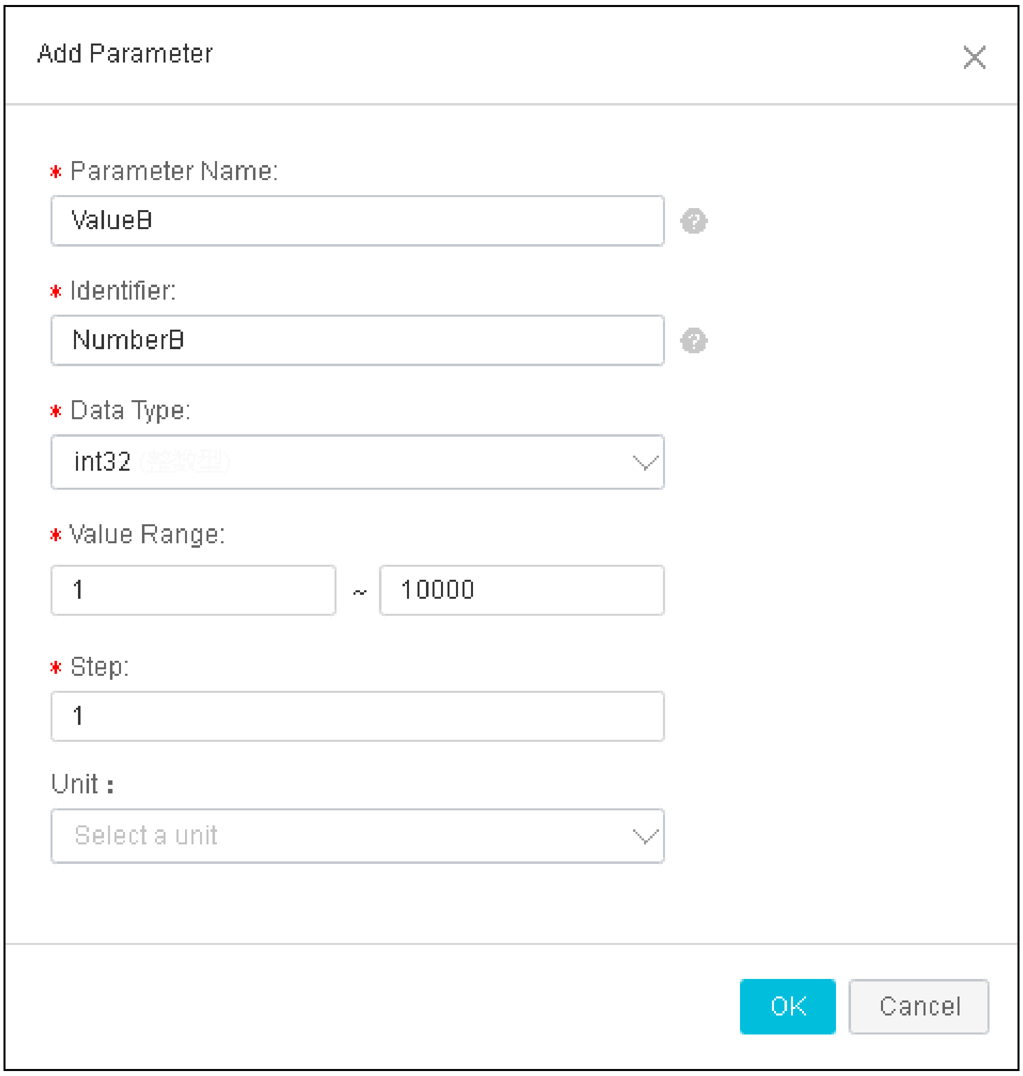


Add Parameter

* Parameter Name: ValueA
* Identifier: NumberA
* Data Type: int32 (整数型)
* Value Range: 1 ~ 10000
* Step: 1

Unit : Select a unit

- **Value B is defined as follows:**


Add Parameter

* Parameter Name: ValueB

* Identifier: NumberB

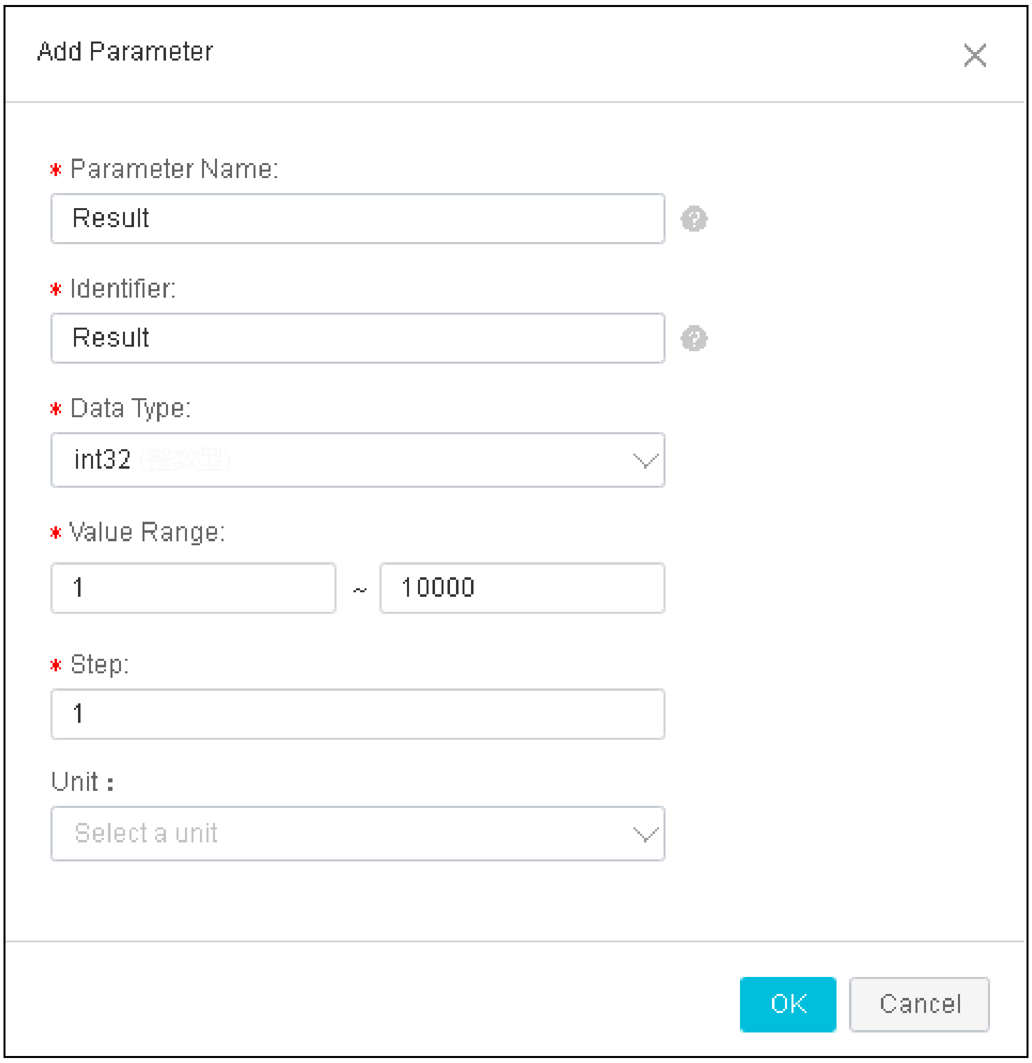
* Data Type: int32

* Value Range: 1 ~ 10000

* Step: 1

Unit : Select a unit

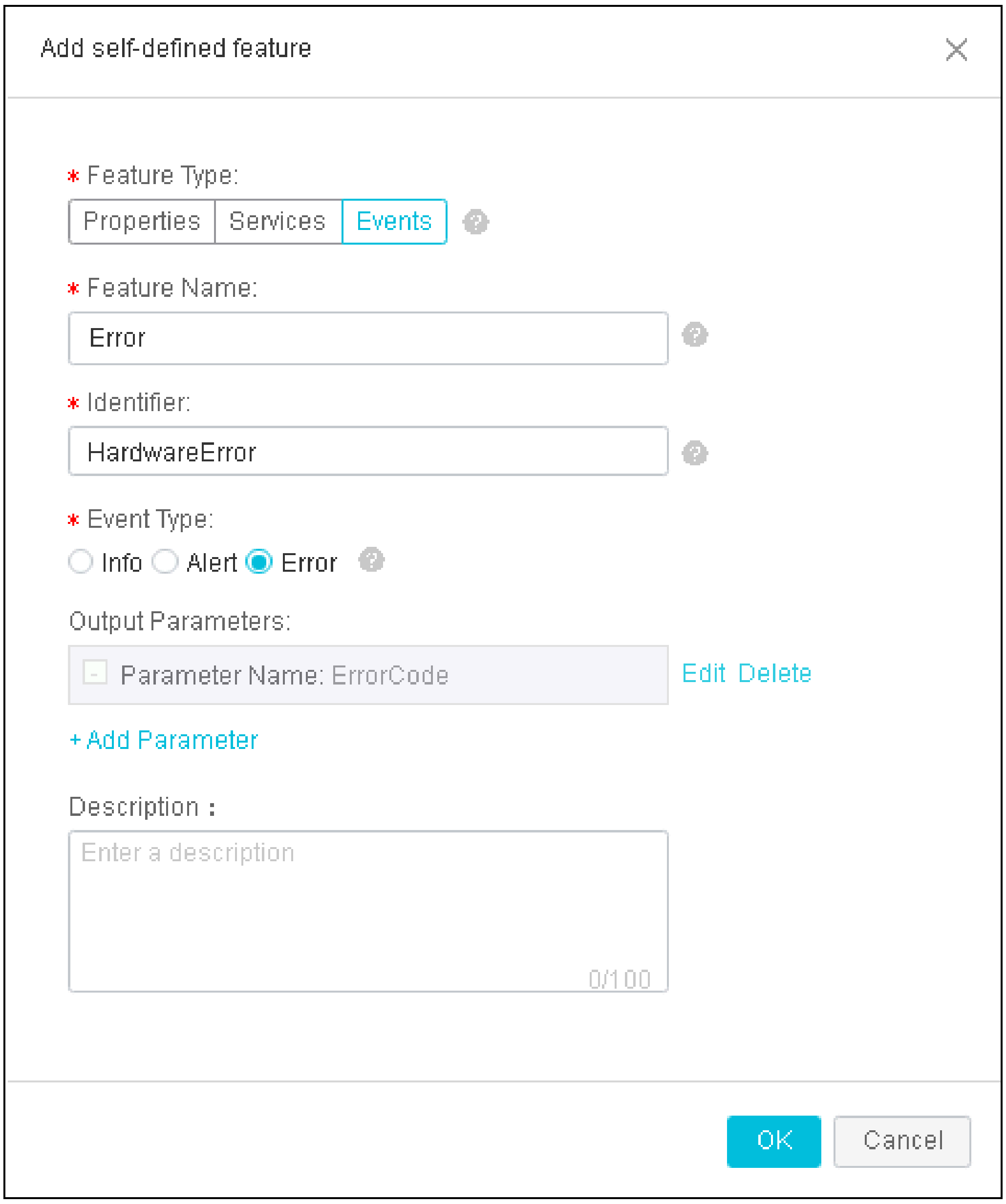
- The output parameter indicates the calculation result.


Add Parameter

* Parameter Name: Result
* Identifier: Result
* Data Type: int32
* Value Range: 1 ~ 10000
* Step: 1

Unit : Select a unit

7. As shown below, add an event to define a hardware error. Click OK.



Add self-defined feature

* Feature Type: Properties | Services | [✓ Events]

* Feature Name: Error

* Identifier: HardwareError

* Event Type: Info Alert [✓ Error]

Output Parameters:

Parameter Name: ErrorCode Edit Delete

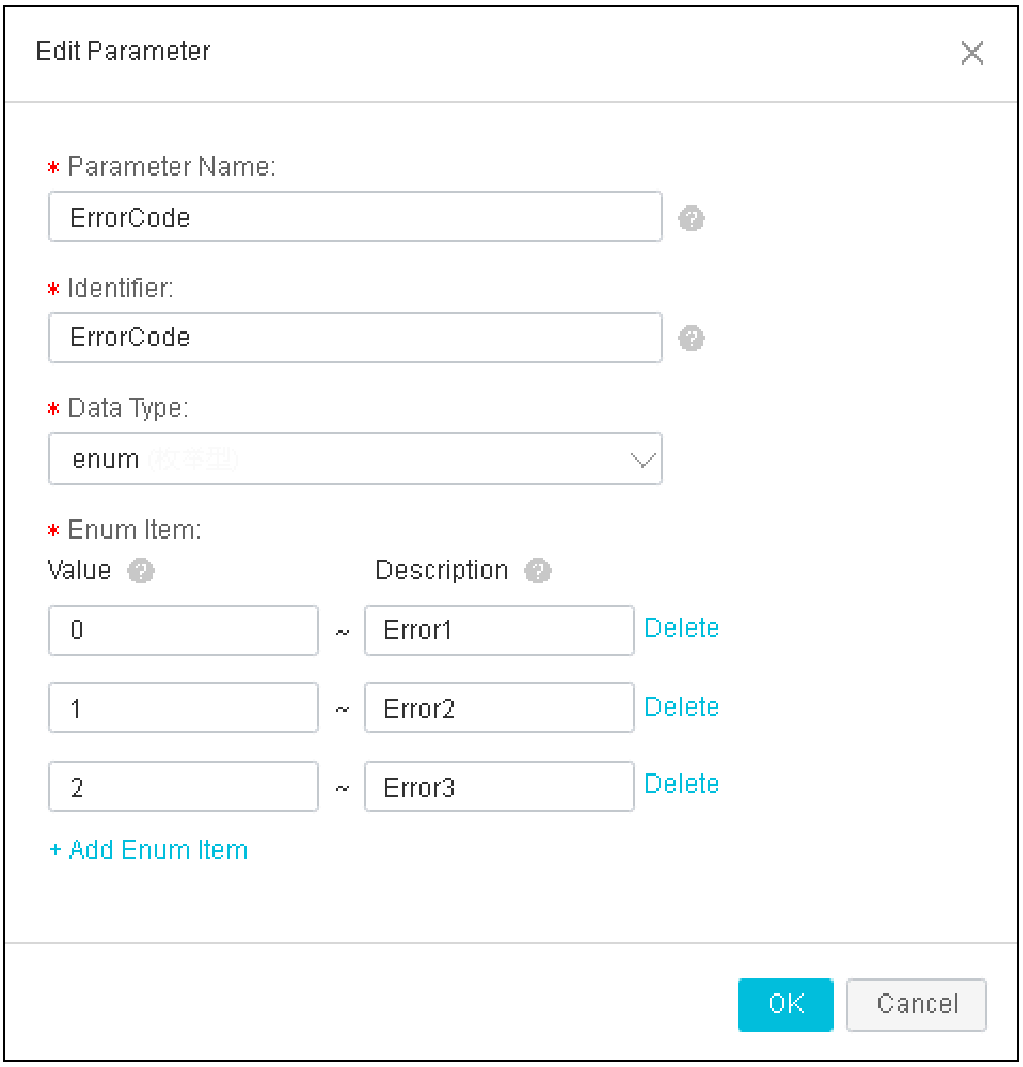
+ Add Parameter

Description：

Enter a description 0/100

OK Cancel

- The output parameter indicates the error code.



* Parameter Name: ErrorCode

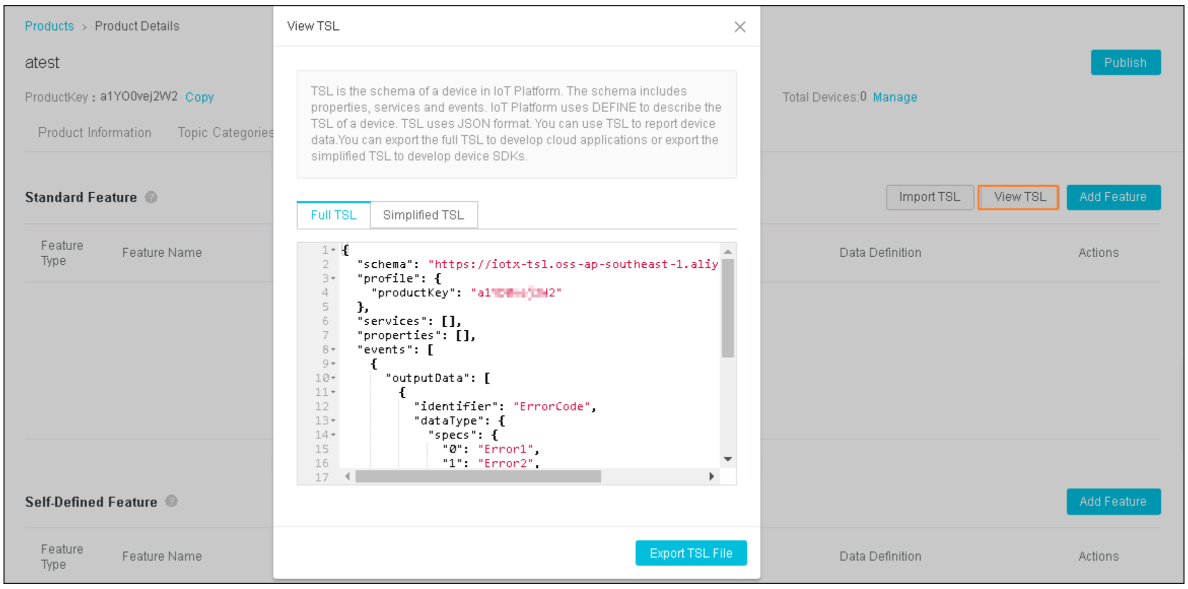
* Identifier: ErrorCode

* Data Type: enum (枚举型)

* Enum Item:

Value		Description
0	~	Error1
1	~	Error2
2	~	Error3

8. Click View TSL and choose Full TSL to view the TSL definitions in JSON format.



TSL is the schema of a device in IoT Platform. The schema includes properties, services and events. IoT Platform uses DEFINE to describe the TSL of a device. TSL uses JSON format. You can use TSL to report device data.You can export the full TSL to develop cloud applications or export the simplified TSL to develop device SDKs.

```
{
  "schema": "https://iotx-tsl.oss-ap-southeast-1.aliy
  "profile": {
    "productKey": "a1[redacted]2"
  },
  "services": [],
  "properties": [],
  "events": [
    {
      "outputData": [
        {
          "identifier": "ErrorCode",
          "dataType": {
            "specs": {
              "0": "Error1",
              "1": "Error2",
```

What's next

Establish a connection between a device and IoT Platform

3 Establish a connection between a device and IoT Platform

Alibaba Cloud IoT Platform provides device SDKs that allow devices to connect to IoT Platform. This article uses a sample program provided by IoT Platform to introduce how to connect the device to IoT Platform using the provided SDK.

Prerequisites

- The SDK used in this example is a C SDK for Linux system. We recommend that you develop this SDK on Ubuntu16.04 (64-bit)
- Software used in the development of the SDK: `make - 4 . 1` , `git - 2 . 7 . 4` , `gcc - 5 . 4 . 0` , `gcov - 5 . 4 . 0` , `lcov - 1 . 12` , `bash - 4 . 3 . 48` , `tar - 1 . 28` , and `mingw - 5 . 3 . 1` Using the following command to install the software:

```
apt - get   install  - y   build - essential   make   git   gcc
```

Procedure

1. Log on to your Linux VM instance.
2. Download the C SDK 2.3.0.

```
wget   https :// github . com / aliyun / iotkit - embedded / archive
/ v2 . 3 . 0 . zip ? spm = a2c4g .  11186623 . 2 . 13 . 1f41492b5W
HpzV & file = v2 . 3 . 0 . zip
```

3. Use the unzip command to extract files from the package.
4. Open the demo program

```
vi   iotkit - embedded - 2 . 3 . 0 / examples / linkkit / linkkit_ex
ample_solo . c
```

5. Change the values of ProductKey, DeviceName, and DeviceSecret in the demo to be your device certificate information, and then save the file.

   See the following example:

```
//   for   demo   only
# define   PRODUCT_KE  Y       " a1I1nn8vPf  4 "
# define   DEVICE_NAM  E       " Light00 "
# define   DEVICE_SEC  RET     " n27gKXTxrU  x ********* QZEmoUX8Tc
 eM "
```

6. In the top level directory, use make command to compile the sample program.

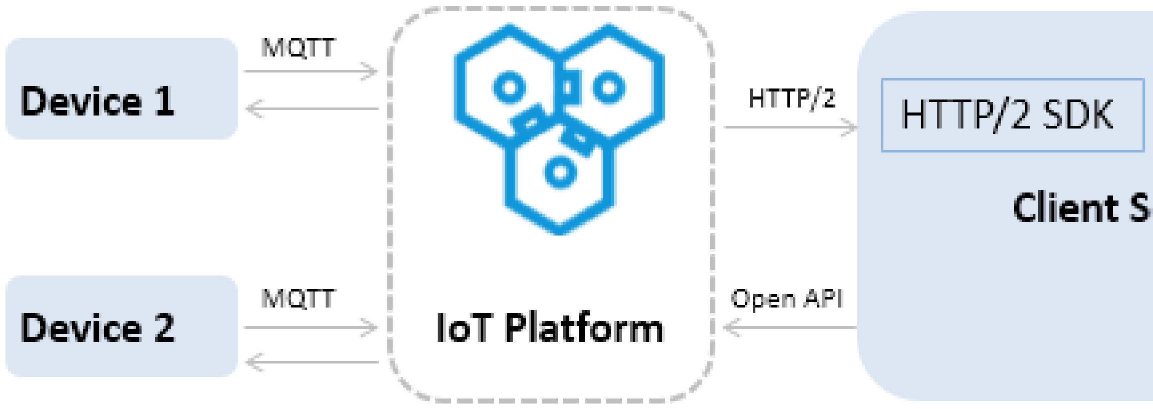
```
$  make   distclean
$  make
```

7. Run the sample program to connect the device to IoT Platform. In the IoT Platform console, you see that the device status is online, indicating that the device has been connected to IoT Platform successfully.

   Once the device has been connected to IoT Platform, it automatically report messages to IoT Platform. You see the device logs for message contents.

4 Subscribe to device messages from IoT Platform

After a device is connected to IoT Platform, the device directly reports data to IoT Platform. Then, the data is forwarded to your server over an HTTP/2 connection. This topic describes how to configure the service subscription function. You can connect your server to an HTTP/2 SDK to receive device data.



Procedure

1. Configure the service subscription function for your product in the IoT Platform console .

   a) On the Products page, click View next to the target product.

   b) On the Product Details page, click Service Subscription > Set.

   c) Select the types of messages to which you want to subscribe, and click Save.

Message type	Description
Device Upstream Notification	Indicates the custom data and TSL model data that are reported by the device. The data includes property data , event data, property setting responses, and service call responses.
Device Status Change Notifications	Indicates the notifications that are sent by the system when the status of a device changes. For example, the connection and disconnection notifications.
Device Changes Throughout Lifecycle	Indicates the notifications about device creation, deletion, disabling, and enabling.

Message type	Description
Sub-Device Data Report Detected by Gateway	A gateway reports the information about the discovered sub-devices to IoT Platform. Make sure that the gateway has an application that can discover and report sub-device information.
Device Topological Relation Changes	Indicates notifications about the creation and removal of the topological relationships between a gateway and its sub-devices.

After you configure service subscription in the console, it takes approximately 1 minute for the settings to take effect.

2. Add dependencies.

   If you use Apache Maven to manage Java projects, you must add the following dependencies to the pom.xml file.

   Note:
   Currently, only Java 8 and .NET SDKs are supported. For more information about SDK configuration, see Development guide for Java HTTP/2 SDK or Development guide for .NET HTTP/2 SDK.

```
< dependency >
    < groupId > com . aliyun . openservic  es </ 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 >
```

3. Use the AccessKey information of your Alibaba Cloud account for identity authentication and connect the HTTP/2 SDK to IoT Platform.

```
// AccessKey   ID   of   your   Alibaba   Cloud   account
         String   accessKey  = " xxxxxxxxxx  xxxxx ";
        // AccessKey   Secret   of   your   Alibaba   Cloud
 account
         String   accessSecr  et  = " xxxxxxxxxx  xxxxx ";
        //  Region   ID   of   your   IoT   Platform   service
         String   regionId  = " cn - shanghai ";
        // User   ID   of   your   Alibaba   Cloud   account
         String   uid  = " xxxxxxxxxx  xx ";
        // Endpoint :  https ://${ uid }. iot - as - http2 .${
 region }. aliyuncs . com
         String   endPoint  = " https ://" +  uid  + ". iot - as -
 http2 ." +  regionId  + ". aliyuncs . com ";
```

```
       // Connection   configurat  ion
        Profile   profile  =  Profile . getAccessK  eyProfile (
endPoint ,  regionId ,  accessKey ,  accessSecr  et );

       // Construct   the   client
        MessageCli  ent   client  =  MessageCli  entFactory .
messageCli  ent ( profile );

       // Receive   data
        client . connect ( messageTok  en  -> {
            Message   m  =  messageTok  en . getMessage ();
            System . out . println (" receive   message   from  "
+  m );
            return   MessageCal  lback . Action . CommitSucc  ess ;
       });
```

Parameter description

Parameter	Description
accessKey	The AccessKey ID of your Alibaba Cloud account. To obtain the AccessKey ID, log on to the Alibaba Cloud console, hover over your account avatar, and click AccessKey. You are redirected to the Security Management page of the User Management console.
accessSecret	The AccessKey Secret of your Alibaba Cloud account. Obtain the AccessKey Secret in the same way you obtain the AccessKey ID.
uid	The account ID. To obtain the account ID, log on to the Alibaba Cloud console by using your Alibaba Cloud account, and click the account avatar. You are redirected to the Security Settings page of the Account Management console.
regionId	The region ID of your IoT Platform service. In the IoT Platform console, you can view the region in the left corner of the top navigation bar. For more information about regions, see Regions and zones .

4. Verify that the HTTP/2 SDK can receive messages from the device.

   If messages can be received, you can obtain the following data from the message callback of the SDK.

Parameter	Description
`messageId`	The message ID generated by IoT Platform .
`topic`	The source topic of the message.
`payload`	The payload of the message. For more information, see Data format.
`generateTi me`	The timestamp when the message was generated, in milliseconds.
`qos`	· 0: The message will be delivered only once. · 1: The message will be delivered at least once.

5 Devices receive commands from IoT Platform

You can use applications in the cloud to call the `SetDeviceProperty` interface to send property setting commands to devices. This article introduces how to configure the device SDK to receive commands from IoT Platform.

Procedure

1. Import the SDK dependency into the maven project.

   The following examples show how to import the IoT Platform Java SDK dependency into the maven project.

   ```
   <! --  https :// mvnreposit  ory . com / artifact / com . aliyun /
    aliyun - java - sdk - iot  -->
   < dependency >
       < groupId > com . aliyun </ groupId >
       < artifactId > aliyun - java - sdk - iot </ artifactId >
       < version > 6 . 4 . 0 </ version >
   </ dependency >
   ```

   Import the core module of the SDK.

   ```
   < dependency >
       < groupId > com . aliyun </ groupId >
       < artifactId > aliyun - java - sdk - core </ artifactId >
       < version > 3 . 5 . 1 </ version >
   </ dependency >
   ```

2. Initialize the SDK.

   The region ID in the endpoint must be the same as the region ID of the device. In the following example, the region ID is cn-shanghai.

   ```
    String   accessKey  = "< your   accessKey >";
    String   accessSecr  et  = "< your   accessSecr  et >";
    DefaultPro  file . addEndpoin  t (" cn - shanghai ", " cn -
    shanghai ", " Iot ", " iot . cn - shanghai . aliyuncs . com ");
    IClientPro  file   profile  =  DefaultPro  file . getProfile (" cn
    - shanghai ",  accessKey ,  accessSecr  et );
    DefaultAcs  Client   client  =  new   DefaultAcs  Client ( profile
    );
   ```

3. Call the SetDeviceProperty operation to send a property setting request to a device. In the following example, the value of the property LightSwitch is set to 1.

   Example:

   ```
    SetDeviceP  ropertyReq  uest   request  =  new   SetDeviceP
    ropertyReq  uest ();
    request . setProduct  Key (" a1I1xxxxPf  4 ");
    request . setDeviceN  ame (" Light001 ");
   ```

```
 JSONObject   itemJson  =  new   JSONObject ();
 itemJson . put (" LightSwitc  h ",  1 );
 request . setItems ( itemJson . toString ());

 try  {
     SetDeviceP ropertyRes ponse   response  =  client .
 getAcsResp  onse ( request );
     System . out . println ( response . getRequest  Id () + ",
 success : " +  response . getSuccess ());
}  catch  ( ClientExce ption   e ) {
     e . printStack  Trace ();
}
```

Note:

For more information about how to call the SetDeviceProperty operation, see SetDeviceProperty.

4. If the device has received the request, the log output is as follows:

```
[ inf ]  iotx_mc_ha  ndle_recv_  PUBLISH ( 1617 ):  Downstream
   Topic : '/ sys / a1I1nn8vPf  4 / Light001 / thing / service /
 property / set '
[ inf ]  iotx_mc_ha  ndle_recv_  PUBLISH ( 1618 ):  Downstream
 Payload :

< {
<     " method ": " thing . service . property . set ",
<     " id ": " 200864995 ",
<     " params ": {
<         " LightSwitc  h ":  1
<     },
<     " version ": " 1 . 0 . 0 "
< }

[ dbg ]  iotx_mc_ha  ndle_recv_  PUBLISH ( 1623 ):          Packet
 Ident  :  00000000
[ dbg ]  iotx_mc_ha  ndle_recv_  PUBLISH ( 1624 ):          Topic
 Length  :  52
[ dbg ]  iotx_mc_ha  ndle_recv_  PUBLISH ( 1628 ):            Topic
   Name  : / sys / a1I1nn8vPf  4 / Light001 / thing / service /
 property / set
[ dbg ]  iotx_mc_ha  ndle_recv_  PUBLISH ( 1631 ):      Payload
 Len / Room  :  101  /  109
[ dbg ]  iotx_mc_ha  ndle_recv_  PUBLISH ( 1632 ):        Receive
 Buflen  :  166
[ dbg ]  iotx_mc_ha  ndle_recv_  PUBLISH ( 1643 ): delivering   msg
  ...
[ dbg ]  iotx_mc_de  liver_mess  age ( 1344 ):  topic   be
 matched
[ inf ]  dm_msg_pro  c_thing_se  rvice_prop  erty_set ( 134 ):
 thing / service / property / set
[ dbg ]  dm_msg_req  uest_parse ( 130 ):  Current   Request
 Message   ID :  200864995
[ dbg ]  dm_msg_req  uest_parse ( 131 ):  Current   Request
 Message   Version :  1 . 0 . 0
[ dbg ]  dm_msg_req  uest_parse ( 132 ):  Current   Request
 Message   Method :  thing . service . property . set
[ dbg ]  dm_msg_req  uest_parse ( 133 ):  Current   Request
 Message   Params : {" LightSwitc  h ": 1 }
```

```
[ dbg ]  dm_ipc_msg  _insert ( 87 ):  dm   msg   list   size :  0 ,
 max   size :  50
[ inf ]  dm_msg_res  ponse ( 262 ):  Send   URI : / sys /
 a1I1nn8vPf  4 / Light001 / thing / service / property / set_reply
 ,  Payload : {" id ":" 200864995 "," code ": 200 ," data ":{}}
[ inf ]  MQTTPublis  h ( 515 ):  Upstream   Topic : '/ sys /
 a1I1nn8vPf  4 / Light001 / thing / service / property / set_reply '
[ inf ]  MQTTPublis  h ( 516 ):  Upstream   Payload :

> {
>     " id ": " 200864995 ",
>     " code ":  200 ,
>     " data ": {
>     }
> }

[ inf ]  dm_client_  publish ( 121 ):  Publish   Result :  0
[ inf ]  _iotx_link  kit_event_  callback ( 223 ):  Receive
 Message   Type :  15
[ inf ]  _iotx_link  kit_event_  callback ( 225 ):  Receive
 Message : {" devid ": 0 ," payload ":{" LightSwitc  h ": 1 }}
[ dbg ]  _iotx_link  kit_event_  callback ( 403 ):  Current   Devid
 :  0
[ dbg ]  _iotx_link  kit_event_  callback ( 404 ):  Current
 Payload : {" LightSwitc  h ": 1 }
 user_prope  rty_set_ev  ent_handle  r .  160 :  Property   Set
 Received ,  Devid :  0 ,  Request : {" LightSwitc  h ": 1 }
```