Intelligent EdgeFabric

User Guide

Issue	02
Date	2021-07-23



HUAWEI TECHNOLOGIES CO., LTD.

Copyright © Huawei Technologies Co., Ltd. 2021. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

Contents

1 Edge Computing with IEF....1
2 Service Instances....3
3 Node Management....5
3.1 Edge Node Overview....5
3.2 Configuring the Edge Node Environment....6
3.3 Registering a Custom Edge Node....9
3.4 Managing an Edge Node....13
3.5 Registering an IEC Node....17
3.6 Edge Node Groups....19
3.7 Upgrading an Edge Node....22
3.8 Logs, Monitoring, and Alarms....24
3.9 Copying GPU Driver Files....31
4 End Device Management....32
4.1 End Devices and Device Twins....32
4.2 Device Templates....34
4.3 End Devices....36
4.4 Binding an End Device to an Edge Node....39
4.5 Device Twin Working Principles....41
4.6 Migrating Device Data to the Cloud....44
4.7 Performing Security Authentication Using Certificate....51
4.8 MQTT Topics....60
4.8.1 Device Twin Update....60
4.8.2 Device Twin Delta....62
4.8.3 Device Member Update....63
4.8.4 Device Property Update....64
4.8.5 Device Member Acquisition....65
4.8.6 Device Member Acquisition Result....65
4.8.7 Device Twin Acquisition....66
4.8.8 Device Twin Acquisition Result....67
4.8.9 Device Twin Modification....68
4.8.10 Device Twin Modification Result....69
4.8.11 Encryption Data Request....70

 Copyright © Huawei Technologies Co., Ltd.

4.8.12 Encryption Data Acquisition ... 71
4.8.13 Alarm Addition ... 72
4.8.14 Alarm Clearance ... 76
4.8.15 Custom Topics ... 77
5 Containerized Application Management ... 78
5.1 Containerized Applications ... 78
5.2 Application Templates ... 86
5.3 Offline Self-Healing ... 91
5.4 Affinity and Anti-Affinity Scheduling ... 95
5.5 ConfigMaps ... 98
5.6 Secrets ... 100
5.7 Encryption Data ... 101
5.8 Health Check Configuration ... 103
6 Function Management ... 105
7 Application Mesh ... 109
8 Edge-Cloud Messages ... 114
8.1 Edge-Cloud Message Overview ... 114
8.2 Cloud Delivering Messages to Edge Nodes ... 117
8.3 Edge Nodes Reporting Messages to the Cloud ... 120
8.4 System Subscriptions ... 124
9 Batch Management ... 129
10 Edge Marketplace ... 130
11 Auditing ... 131
11.1 IEF Operations Supported by CTS ... 131
11.2 Viewing Tracing Logs ... 134
12 Permissions Management ... 135
12.1 Creating a User and Granting Permissions ... 135
12.2 IEF Custom Policies ... 136
12.3 IEF Resources ... 136
12.4 IEF Request Conditions ... 137

 Copyright © Huawei Technologies Co., Ltd.

1 Edge Computing with IEF

Intelligent EdgeFabric (IEF) provides you a complete edge computing solution where cloud applications are extended to the edge. By leveraging edge-cloud synergy, you can manage edge nodes and applications remotely while still processing data nearby. In addition, you can perform O&M in the cloud, including edge node monitoring, edge application monitoring, and log collection.

Edge Computing with IEF

Figure 1-1 shows how IEF implements edge computing.

1. Enable IEF to manage edge nodes, and bind end devices to the nodes.

   Install edge node software on edge nodes, and bind end devices to the nodes. Then you can use IEF to deploy applications on and deliver functions to those edge nodes.

   For details on how to manage edge nodes and bind end devices, see **Node Management** and **End Device Management**.

2. Develop applications, create images, and upload the images to **SoftWare Repository for Container (SWR)**.

   This step ensures that edge nodes can pull images from SWR when IEF delivers applications.

   There is no sequential order between this step and **step 1**. Either of them can be performed first as needed.

3. Deploy applications and deliver functions.

   After the edge nodes are managed and applications are developed, you can use IEF to deploy applications on edge nodes and start running your services.

   IEF can also deliver **FunctionGraph** functions to provide higher flexibility for applications.

   Running applications can be monitored and alarms can be reported through the Application Operations Management (AOM) service.

   For details on how to deploy applications and deliver functions, see **Containerized Application Management** and **Function Management**.

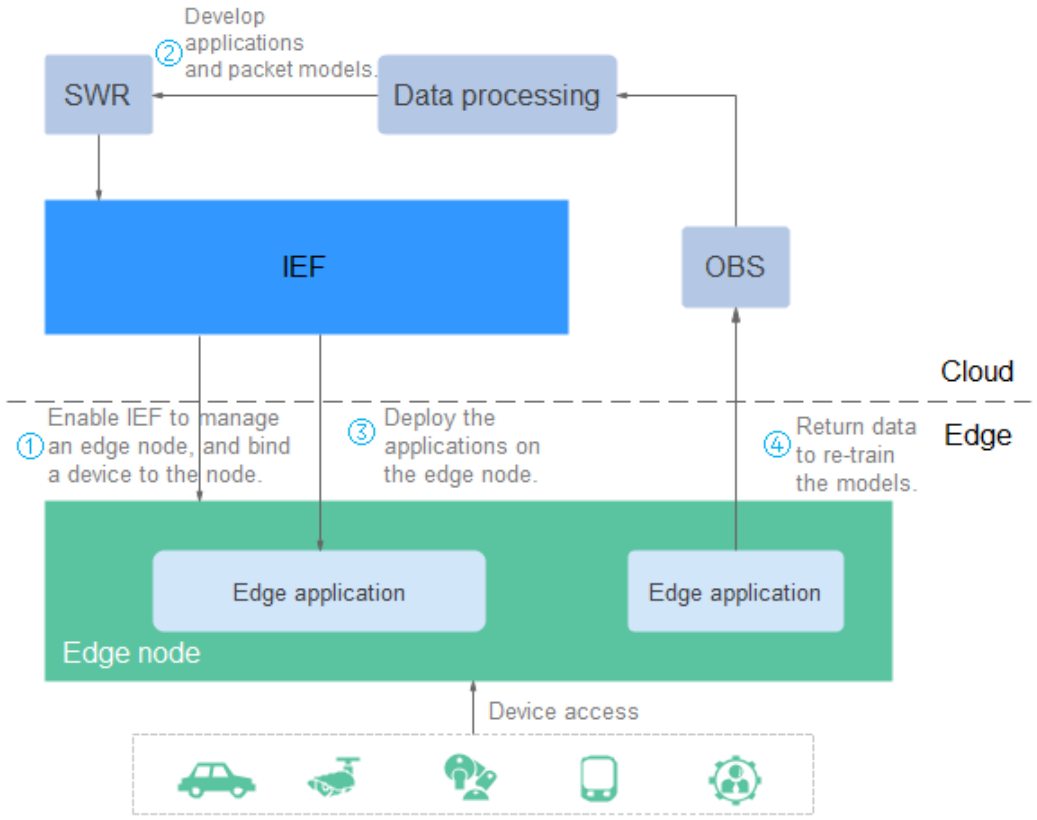
4. (Optional) Send the data back to the cloud for further processing, and update the applications based on the results.

 Copyright © Huawei Technologies Co., Ltd.

This step is not closely related to the use of IEF, but it is a common way to optimize applications based on processing data. You can determine whether to perform this step based on your requirements.

For example, if you deploy a facial recognition application on an edge node, you can periodically upload collected data to OBS and add the data to a training set to generate a new and better model. Then, you can package the new model into a new image and update the image to the application deployed on the edge node.

Figure 1-1 Implementing edge computing



 Copyright © Huawei Technologies Co., Ltd.

2 Service Instances

A service instance is a management cluster that IEF uses to manage edge nodes and deliver applications. IEF supports two types of service instances: Professional and Platinum.

- Professional (default): All users share the same management cluster.
- Platinum: Users need to create their own platinum service instances and then exclusively use those management clusters. For details on how to create and switch to a platinum service instance, see **Creating a Platinum Service Instance** and **Switching an Instance**.

Creating a Platinum Service Instance

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Dashboard**. On the page displayed on the right, click **Create Platinum Service Instance** in the upper right corner of the page.

Step 3 Configure parameters.

- **Region**: Select the region where the service instance is deployed. Instances can only be used in the region specified when they are created. Select the region based on service requirements.
- **Instance Name**: Enter the instance name.
- **Access Type**: **Internet** and **Direct Connect** are available.
- **Edge Nodes**: The number of edge nodes that can be managed by the service instance. Currently, 50, 200, and 1,000 nodes can be selected.
- **Access Bandwidth**: If **Access Type** is set to **Internet**, select 5 Mbit/s, 10 Mbit/s, or 30 Mbit/s depending on the number of edge nodes. If **Access Type** is set to **Direct Connect**, the bandwidth is determined by the Direct Connect service.
- **Advanced Settings**: Multi-AZ deployment is supported. If this option is enabled, platinum service instances can be deployed in multiple AZs to facilitate disaster recovery. However, the cluster performance is decreased.

Step 4 Click **Next**.

Step 5 Confirm the order details, and click **Create**.

 Copyright © Huawei Technologies Co., Ltd.

It takes 20-30 minutes to create a service instance. You can view the service instance status in the service instance list.

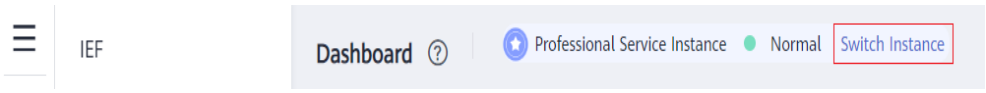
----End

Switching an Instance

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Dashboard**. On the page displayed on the right, click **Switch Instance** in the upper middle of the page.

Figure 2-1 Switching an instance



Step 3 Under the target instance, click **Select Instance**.

----End

Copyright © Huawei Technologies Co., Ltd.

3 Node Management

3.1 Edge Node Overview

Edge nodes are your own edge computing devices. They are used to run edge applications, process your data, and collaborate with cloud applications securely and conveniently. You can deploy system applications using IEF to extend cloud service capabilities to edge nodes, or deploy your own applications to implement edge computing capabilities.

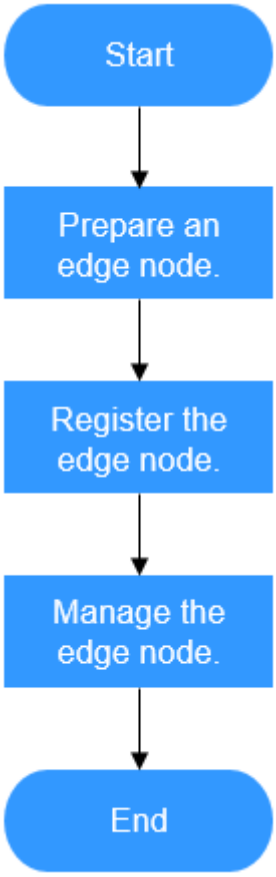
To enable IEF to manage your edge node, perform the following steps:

1. Prepare the node. The edge node must meet requirements listed in **Configuring the Edge Node Environment**.
2. Register the node on IEF, and download the EdgeCore installer and configuration file. For details, see **Registering a Custom Edge Node**.
3. Manage the edge node using the EdgeCore installer and configuration file. For details, see **Managing an Edge Node**.

IEF allows you to register an IEC edge instance as an edge node. A registered instance is automatically managed by IEF. For details, see **Registering an IEC Node**.

 Copyright © Huawei Technologies Co., Ltd.

Figure 3-1 Procedure



3.2 Configuring the Edge Node Environment

Specifications Requirements

An edge node can be a physical machine or a virtual machine (VM). Edge nodes must meet the specifications listed in **Table 3-1**.

Table 3-1 Edge node requirements

Item	Specifications
OS	The language of the operating system must be English. • x86_64 architecture Ubuntu 16.04 LTS (Xenial Xerus), Ubuntu 18.04 LTS (Bionic Beaver), RHEL 7.x, Kylin 4.0.2, NewStart CGSL v5.5, NeoKylin v7.0, and CentOS 7.x • Armv7i (Arm32) architecture Raspbian GNU/Linux 9 (stretch) • AArch64 (Arm64) architecture Ubuntu 18.04.2 LTS (Bionic Beaver)

Copyright © Huawei Technologies Co., Ltd.

Item	Specifications
Memory	More than 256 MB of memory is recommended as 128 MB of memory is required to run the edge software.
CPU	≥ 1 core
Hard disk	≥ 1 GB
GPU (optional)	The GPU models on the same edge node must be the same. **NOTE** Currently, NVIDIA Tesla GPUs such as P4, P40, and T4 are supported. If an edge node is equipped with GPUs, you can choose not to enable its GPUs when registering it on IEF. If you choose to enable GPUs of an edge node, the GPU driver has to be installed on the edge node before you can manage it on IEF.
NPU (optional)	Huawei Ascend AI processors **NOTE** Currently, only edge nodes integrated with Huawei Ascend 310 or 910 Processors are supported, such as Atlas 300 inference cards, Atlas 500 AI edge stations, and Atlas 800 inference servers. If you choose to enable NPUs of an edge node, ensure that the NPU driver has been installed on it. Currently, Ascend 310 supports only firmware versions 1.3.x.x and 1.32.x.x, for example, 1.3.2.B893. You can run the **npu-smi info** command to view your firmware version. If the driver is not installed, contact the device manufacturer for assistance.
Container engine (mandatory when containerized applications are deployed)	The Docker version must be later than 17.06. Docker 18.06.3 is recommended. (However, Docker 18.09.0 is not recommended as it has a serious bug. For details, see **https://github.com/docker/for-linux/issues/543**. If this version has been installed, upgrade it at the earliest possible opportunity. This issue has been resolved in Docker 18.09.0.60 embedded in Atlas 500 AI edge stations.) **NOTICE** After Docker is installed, configure the Docker process to start at host startup. This configuration prevents system exceptions caused by Docker startup failures after the host is restarted. The Docker Cgroup driver must be set to **cgroupfs**.
Glibc	The Glibc version must be later than 2.17.
Port	Edge nodes require port 8883, which is the listening port of the built-in MQTT broker on edge nodes. Ensure that this port works properly.

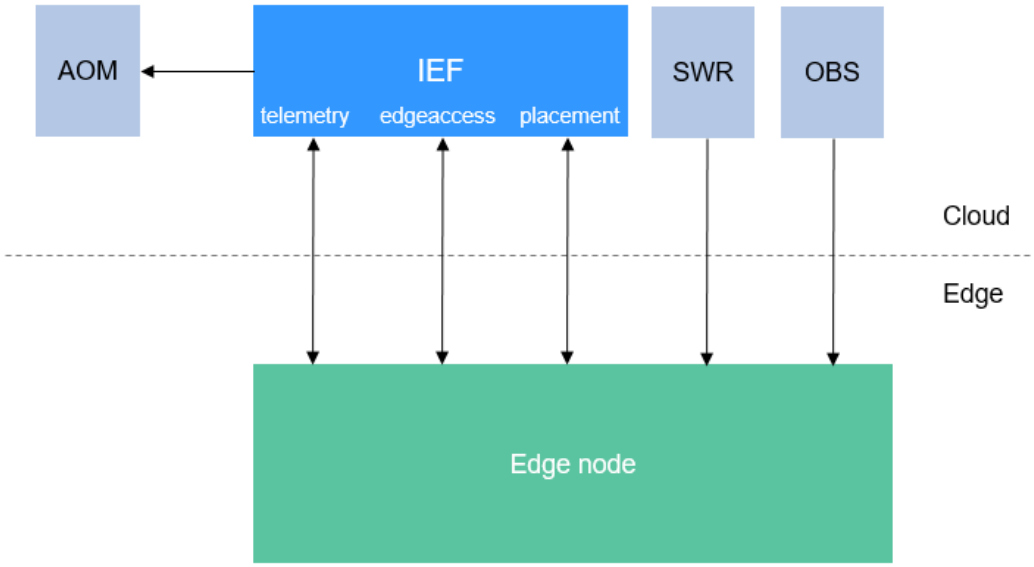
Copyright © Huawei Technologies Co., Ltd.

Item	Specifications
Time synchronization	The time on the edge node must be consistent with the UTC time. Otherwise, the monitoring data and logs of the edge node may be inaccurate. You can select an NTP server for time synchronization. For details, see **How Do I Synchronize Time with the NTP Server**.

Connection Requirements

If you need to manage your edge nodes on IEF, ensure that the edge nodes can communicate with IEF, SWR, and OBS. If any air wall or network limit exists in the environment where your edge nodes are located, obtain the IP addresses of these cloud services by referring to **How Do I Obtain the IP Addresses of IEF Cloud Services** and configure the firewall.

Figure 3-2 Connections between an edge node and cloud services



Edge nodes must be able to connect to IEF through the Internet. Alternatively, they can connect to IEF through Direct Connect or VPN. For details, see **Connecting to IEF Through Direct Connect or VPN**.

Configuring the Edge Node Environment

Step 1 Log in to an edge node as a user with sudo permissions.

Step 2 Configure the GPU driver.

If your edge node has been equipped with a GPU, the GPU driver has to be installed on the edge node before you can manage it on IEF. For details, see **Copying GPU Driver Files**.

Step 3 Configure the NPU driver.

 Copyright © Huawei Technologies Co., Ltd.

If your edge node uses Huawei Ascend AI processors, ensure that the corresponding driver has been installed.

Step 4 Install Docker on the edge node and check the Docker status.

This step is mandatory when containerized applications need to be deployed. If only functions are to be deployed, then you can skip this step.

The Docker version must be later than 17.06. Docker 18.06.3 is recommended. However, Docker 18.09.0 is not recommended as it has a serious bug. If you have used this version, upgrade it at the earliest possible opportunity.

After Docker installation is complete, run **docker -v** to confirm that Docker was installed properly. If the following information is displayed, Docker was installed properly.

```
# docker -v
Docker version 19.03.12, build 48a66213fee
```

Step 5 Configure firewall rules for the edge node.

Check the firewall status on the edge node.

```
systemctl status firewalld
firewall-cmd --state
```

In the command output, **notrunning** indicates that the firewall is disabled and **running** indicates that the firewall is enabled.

If the firewall is enabled, enable port 8883 or disable the firewall.

- To enable port 8883, run the following commands:
  ```
  firewall-cmd --add-port=8883/tcp --permanent
  systemctl restart firewalld
  ```
- To disable the firewall, run the following commands:
  ```
  systemctl disable firewalld
  systemctl stop firewalld
  ```

----End

3.3 Registering a Custom Edge Node

Registering a custom edge node is the process of creating an edge node on IEF and obtaining the node configuration file and EdgeCore installer.

Registering an Edge Node

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Managed Resources** > **Edge Nodes**. Then, click **Register Edge Node** in the upper right corner.

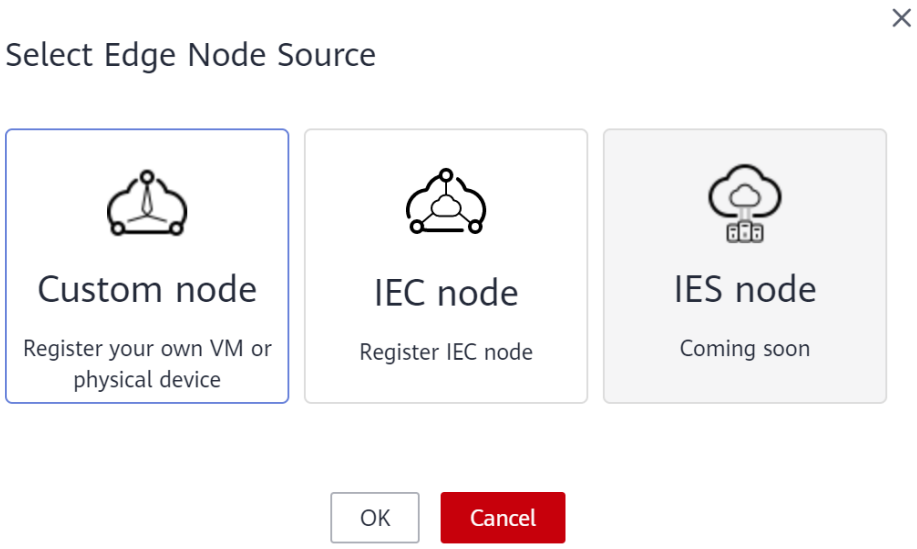
Step 3 Choose **Custom node** and click **OK**.

NOTE

This option is available only for IEF of the platinum edition.

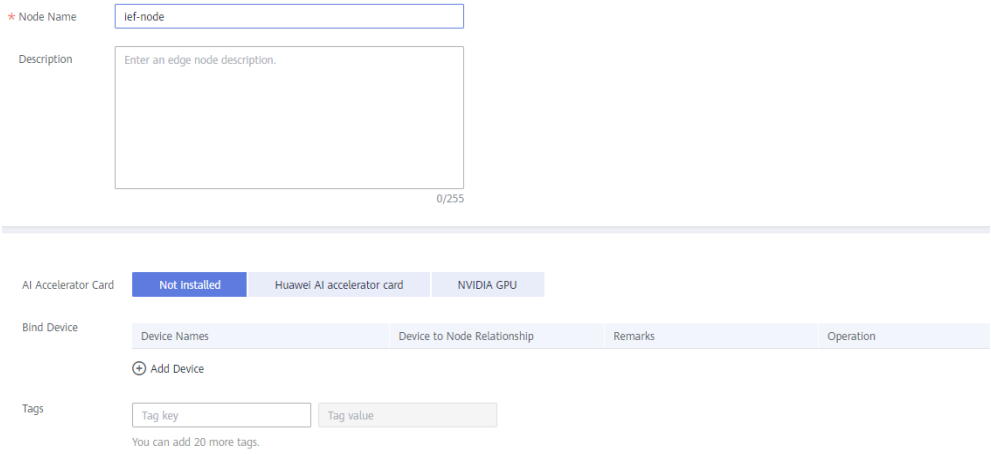
 Copyright © Huawei Technologies Co., Ltd.

Figure 3-3 Selecting an edge node source



Step 4 Configure basic edge node information.

Figure 3-4 Basic edge node information



- **Node Name**: name of an edge node. The name can contain 1 to 64 characters, including letters, digits, hyphens (-), and underscores (_).
- **AI Accelerator Card**

  Huawei AI accelerator card: Select this option if you register an edge node equipped with Huawei Ascend 310 or 910 Processors (NPUs), for example, an Atlas 500 AI edge station. After you select **Huawei AI accelerator card**, select **Ascend 310** or **Ascend 910** accordingly.

  NVIDIA GPU: Select this option if your edge node is equipped with an NVIDIA GPU.

  Not installed: Select this option if your edge node does not use any AI accelerator card.

 Copyright © Huawei Technologies Co., Ltd.

NOTE

If you select **NVIDIA GPU** but your edge node is not equipped with any NVIDIA GPU, the edge node cannot be managed by IEF.

If your edge node has been equipped with an NVIDIA GPU, the GPU driver has to be installed on the edge node before you can manage it on IEF. For details, see **Copying GPU Driver Files**.

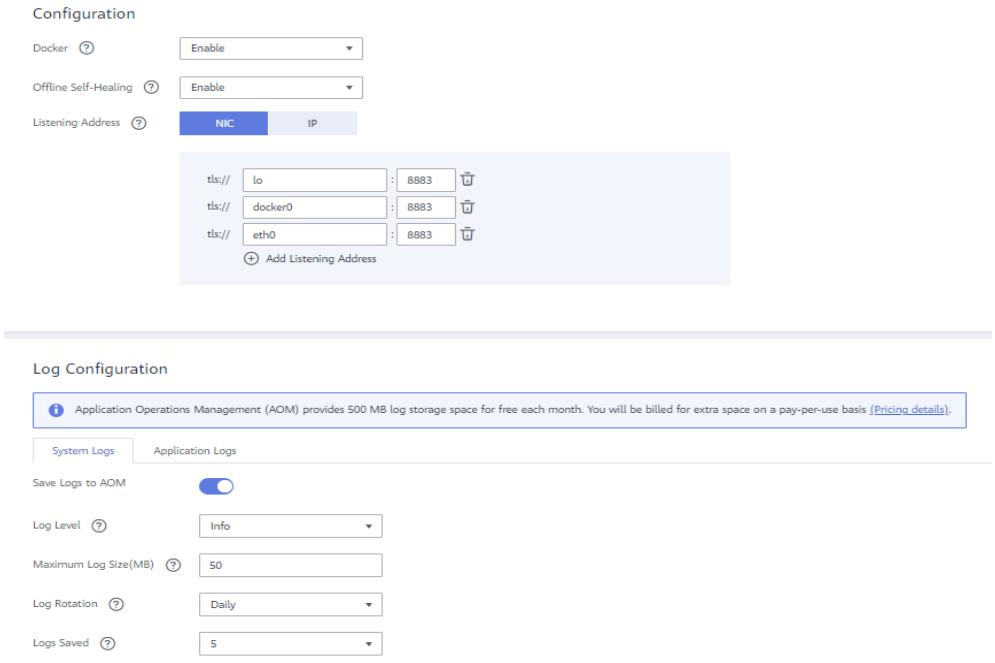
- **Bind Device**: Bind an end device to the edge node. If you have not registered an end device, register one by referring to **End Device Management**. If you do not bind an end device in this step, you can bind one after the edge node is registered.
- **Tags**

  Tags can be used to classify resources, facilitating resource management.

  If you need to use the same tag to identify multiple cloud resources, you can select the same tag from the **Tags** drop-down list box. It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources.

Step 5 Click **Next** to configure advanced settings for the edge node.

Figure 3-5 Advanced settings



- **Docker**: If you enable Docker, containerized applications can be deployed. If you do not enable Docker, only functions can be deployed.
- **Offline Self-Healing** (available only for platinum service instances): If you enable this option, then if the instance becomes abnormal or the edge node where the instance had originally been deployed becomes faulty, the instance will be automatically rescheduled to other available edge nodes. For details, see **Offline Self-Healing**.
- **Listening Address**

 Copyright © Huawei Technologies Co., Ltd.

The address on which the built-in MQTT broker of the edge node listens. It is used for edge-cloud messaging. For details about edge-cloud messaging, see **Device Twin Working Principles** and **Edge-Cloud Message Overview**.

By default, local NICs lo (localhost) and docker0 are listened to. You can specify an NIC name or IP address to set the NIC to be listened to. Multiple NICs or IP addresses can be added.

Currently, system and application logs can be configured for edge nodes.

- **System Logs**: logs generated by the IEF software (such as edge-core, edge-logger, and edge-monitor) installed on edge nodes.
- **Application Logs**: logs generated by applications deployed on edge nodes.

Set the following parameters for both system and application logs:

- **Save Logs to AOM**

  You can enable or disable log uploading to AOM. If it is enabled, you can view the logs in AOM. For details, see **Viewing Logs on AOM**.

- **Log Level**

  Options for system log are **Error**, **Warning**, **Info**, and **Debug**. The logs to be uploaded depend on the specified log level.

- **Maximum Log Size (MB)**: maximum size of a log file, in MBs. The default value is **50**. The value range is 10–1000. If a log file reaches the size specified here, the system compresses the log file and dumps it to a specified directory on the host.
  - System logs are stored in the **/var/IEF/sys/log/** directory on edge nodes and will be dumped to AOM when the maximum log size is reached.
  - Application logs include the standard output of containers and the logs mounted to **/var/IEF/app/log** on the edge node. These logs will be dumped to AOM when the maximum log size is reached.
- **Log Rotation**: rate at which logs are compressed and dumped into a specified directory. The value can be **Daily**, **Weekly**, **Monthly**, or **Yearly**. Log files will be dumped when either **Maximum Log Size (MB)** or **Log Rotation** is reached.
- **Logs Saved**: maximum number of logs being retained at a specified period. The default value is **5**. The value range is 1–10. Once this number is reached, when new log files are dumped in, old log files will be deleted on a first-in first-out (FIFO) basis.

NOTE

If you register an Atlas 500 equipped with AI accelerator cards, you are advised to set log parameters as follows:

Log Level: **Info**

Maximum Log Size (MB): **10**

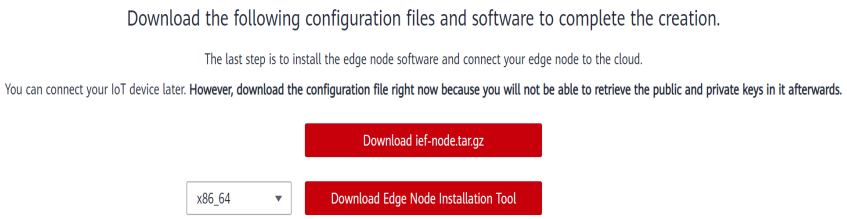
Log Rotation: **Daily**

Logs Saved: **5**

Step 6 Click **Register** in the lower right corner. Download the configuration file and EdgeCore installer, which will be used in **Managing an Edge Node**.

 Copyright © Huawei Technologies Co., Ltd.

Figure 3-6 Downloading the configuration file and EdgeCore installer

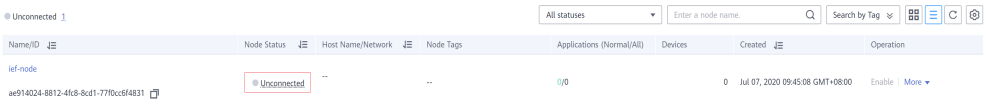


1. Click **Download *Edge node name*.tar.gz**.
2. Select the CPU architecture of your edge node, and click **Download Edge Node Installation Tool**.

Step 7 In the lower right corner, select **I have confirmed that the configuration file has been downloaded** and click **Finish**.

The edge node is in the **Unconnected** state because the EdgeCore installer downloaded in **Step 6** has not been installed yet. For details, see **Managing an Edge Node**.

Figure 3-7 Unconnected edge node



----End

Follow-up Operation

After the registration is complete, you need to manage the edge node. For details, see **Managing an Edge Node**.

3.4 Managing an Edge Node

Managing an edge node is the process of using the EdgeCore installer and configuration file downloaded in **Registering a Custom Edge Node** to install EdgeCore on the edge node. In this way, the edge node can be connected to and managed by IEF.

When you manage your edge node on IEF for the first time, IEF automatically pushes the latest EdgeCore. For example, if three EdgeCore versions 2.22.0, 2.23.0, and 2.24.0 are available, IEF will push the latest version 2.24.0.

NOTE

The edge nodes registered on IEF have one-to-one relationships with the actual devices. The EdgeCore installer and configuration file of an edge node can be installed on only one actual device.

 Copyright © Huawei Technologies Co., Ltd.

Prerequisites

- You have prepared a node that meets the specified environment requirements. For details, see **Configuring the Edge Node Environment**.
- You have registered a node on IEF and obtained the node configuration file and EdgeCore installer. For details, see **Registering a Custom Edge Node**.

Managing an Edge Node

If you want to register an Atlas 500 AI edge station as an edge node, follow the procedure described in **Managing an Atlas 500 AI Edge Station**.

If you want to register an ITS800 as an edge node, follow the procedure described in **Managing an ITS800 Node**.

Step 1 Log in to an edge node as a user with sudo permissions.

Step 2 Upload the EdgeCore installer and configuration file downloaded in **Registering a Custom Edge Node** to the edge node, and run the following command to decompress the software package to the **/opt** folder:

sudo tar -zxvf *edge-installer_1.0.0_x86_64.tar.gz* **-C /opt**

Replace *edge-installer_1.0.0_x86_64.tar.gz* with the name of the EdgeCore installer package downloaded in **Registering a Custom Edge Node**.

Step 3 Run the following command to decompress the configuration file to the **opt/IEF/Cert** directory:

sudo mkdir -p /opt/IEF/Cert; sudo tar -zxvf *Edge node name.tar.gz* **-C /opt/IEF/Cert**

Replace *Edge node name.tar.gz* with the name of the configuration file downloaded in **Registering a Custom Edge Node**.

Step 4 Run the following commands to check whether the edge node meets the management requirements:

chmod +x /opt/edge-installer/edgectl

./opt/edge-installer/edgectl diagnose installation

- If the edge node meets the requirements, the following information is displayed:

```
+---------------------------------+
| Installation diagnose succeed. |
+---------------------------------+
```

- If the edge node does not meet the requirements, an error message similar to the following is displayed. You can locate and rectify the fault based on the error code in the message. For details, see **Error Code Overview**.

```
Failed to get Docker version.
ERROR6102: Make sure that Docker is running.
For more information, see IEF documentation.

+-------------------------------+
| Installation diagnose failed. |
+-------------------------------+
```

Step 5 Run the following command to manage the edge node:

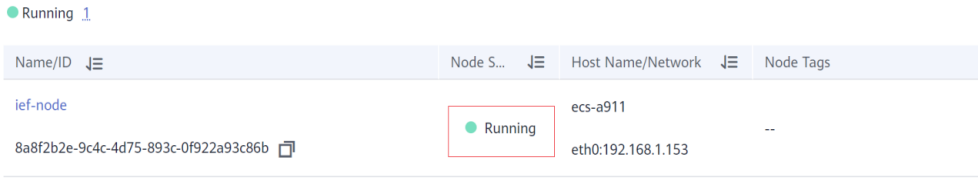
cd /opt/edge-installer; sudo ./installer -op=install

 Copyright © Huawei Technologies Co., Ltd.

Step 6 Verify whether the edge node is managed successfully.

1. Log in to the IEF console.
2. In the navigation pane, choose **Managed Resources** > **Edge Nodes**.
3. Check the status of the edge node. If the status is **Running**, the edge node has been managed by IEF.

Figure 3-8 Checking the edge node status



----End

NOTICE

After the edge node is managed by IEF, do not delete the **/opt** directory on the edge node, or you will need to register the edge node and enable IEF to manage it again.

Managing an Atlas 500 AI Edge Station

If your edge node is not an Atlas 500 AI edge station, see **Managing an Edge Node**.

Step 1 Log in to an Atlas 500 AI edge station as user **admin**, and check whether the domain name resolution of the DNS server is normal.

1. Switch to the develop mode, and enter the password of user **root**.

```
IES:/->develop
Tips:The root password is an initial password. Change it for security purposes.
input root passwd
Password:
Last login: Wed Aug 14 07:31:06 UTC 2019 on pts/0
Euler:~ #
```

2. Ping the service address of HUAWEI CLOUD IEF, and check whether the domain name can be resolved.

   ping ief-edgeaccess.cn-south-1.myhuaweicloud.com

   - If the following information is displayed, the domain name cannot be resolved:

```
Euler:~ # ping ief-edgeaccess.cn-north-1.myhuaweicloud.com
ping: ief-edgeaccess.cn-north-1.myhuaweicloud.com: System error
Euler:~ #
```

     In this case, use **vi** to edit **resolv.conf** and add the IP address (for example, 114.114.114.114) of the DNS server on the public network.

     vi /etc/resolv.conf

   - If the following information is displayed, the domain name can be resolved, and you can continue to **Step 2**.

Copyright © Huawei Technologies Co., Ltd.

```
Euler:~ # ping ief-edgeaccess.cn-north-1.myhuaweicloud.com
PING ief-edgeaccess.cn-north-1.myhuaweicloud.com (      ) 56(84) bytes of data.
^C
--- ief-edgeaccess.cn-north-1.myhuaweicloud.com ping statistics ---
7 packets transmitted, 0 received, 100% packet loss, time 6134ms
```

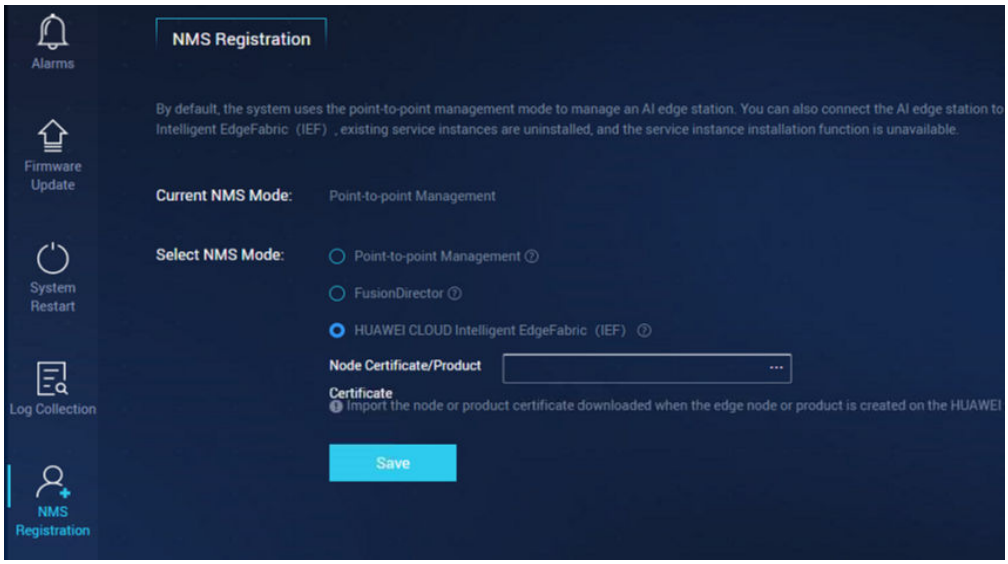
Step 2 Log in to the WebUI of the Atlas 500 AI edge station (**https://***{hostIP}*).

{hostIP} is the IP address of the Atlas 500 AI edge station.

Step 3 Choose **Maintenance** in the navigation pane on the top of the page, and click **NMS Registration** on the left of the page.

Step 4 Select **HUAWEI CLOUD Intelligent EdgeFabric (IEF)** for **Select NMS Mode** and upload the configuration file (node certificate) downloaded in **Registering a Custom Edge Node**.

Figure 3-9 Uploading the node certificate



Step 5 Click **Save**. Wait for the Atlas 500 AI edge station to be managed by IEF.

----End

Managing an ITS800 Node

Step 1 Log in to the ITS800 OMU portal as the **admin** user.

Step 2 Choose **System Management** > **IEF Edge**.

Step 3 Upload the installation package and certificate package.

- The certificate package refers to the configuration file downloaded in **Registering a Custom Edge Node**, for example, **test.tar.gz**.
- The installation package refers to the EdgeCore installer package downloaded in **Registering a Custom Edge Node**, for example, **edge-installer_1.0.8_arm64.tar.gz**.

Step 4 Click **Install** to install the algorithm.

After the algorithm is installed, the value of **Status** changes to **Installed**.

----End

Copyright © Huawei Technologies Co., Ltd.

3.5 Registering an IEC Node

NOTE

IEC nodes are available only for IEF of the platinum edition.

Context

IEF allows you to register an IEC edge instance as an edge node. When you register an IEC node, an edge instance is automatically created in IEC and automatically managed by IEF.

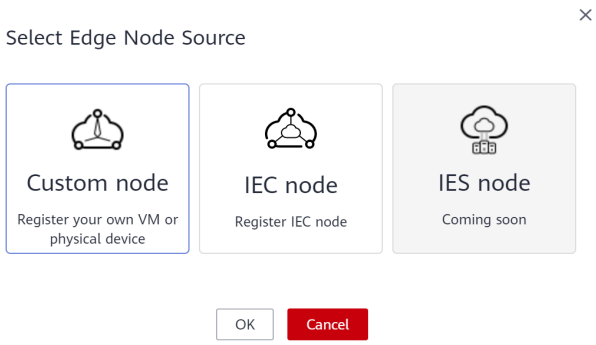
Procedure

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Managed Resources** > **Edge Nodes**. Then, click **Register Edge Node** in the upper right corner.

Step 3 Choose **IEC node** and click **OK**.

Figure 3-10 Selecting an edge node source



For details about how to register a custom node, see **Registering a Custom Edge Node**.

Step 4 Configure the edge node.

- **Node Name**: name of an edge node. The name can contain 1 to 48 characters, including letters, digits, hyphens (-), and underscores (_).
- **Tags**

  Tags can be used to classify resources, facilitating resource management.

  If you need to use the same tag to identify multiple cloud resources, you can select the same tag from the **Tags** drop-down list box. It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources.
- **Region**: Select the **edge site** where the edge node is located.
- **CPU Architecture**: Select the CPU architecture.

Copyright © Huawei Technologies Co., Ltd.

- **Specifications**: Select the specifications of the edge node.
- **Operating System**: OS used by the edge node. Currently, only CentOS 7.6 is supported.
- **System Disk**: Select the type and size of the system disk.
- **Edge VPC**: Select the edge VPC where the edge node is located. If no VPC is available, **create one**.
- **EIP**: Currently, only **On** is available.
- **Address Pool**: Select a carrier line for the EIP.
- **Security Group**: Select a security group for the instance. If no options are available, **create a security group** and **configure security rules**.

  NOTE

  The outbound rules of the security group must enable all ports.

- **Login Mode**
  - **Password**: A username and its initial password will be used for edge instance authentication.

    The initial password of user **root** will be used for authenticating Linux login, while that of user **Administrator** will be used for authenticating Windows login.

    This mode can be used for both public and private images.

    A password must meet the following requirements:

    - Contains 8 to 26 characters.
    - Contains at least three of the following character types: uppercase letters, lowercase letters, digits, and special characters !@$%^-_=+[{}]:,./?
    - Cannot contain the username or the username spelled backwards.
  - **Key pair**: A key pair will be used for edge instance authentication. You can select an existing key pair, or click **Create Key Pair** to **create one**.

    Key pair is recommended because it is more secure than **Password**. If you select **Password**, ensure that the password meets complexity requirements to prevent malicious attacks.

    If you use an existing key pair, make sure that you have saved the key file locally. Otherwise, you will not be able to log in to the edge instance.

Step 5 Click **Next** to configure advanced settings for the edge node.

- **Bind Device**: Bind an end device to the edge node. If you have not registered an end device, register one by referring to **End Device Management**. If you do not bind an end device in this step, you can bind one after the edge node is registered.
- **Docker**: If you enable Docker, containerized applications can be deployed. If you do not enable Docker, only functions can be deployed.
- **Listening Address**

  The address on which the built-in MQTT broker of the edge node listens. It is used for edge-cloud messaging. For details about edge-cloud messaging, see **Device Twin Working Principles** and **Edge-Cloud Message Overview**.

 Copyright © Huawei Technologies Co., Ltd.

By default, local NICs lo (localhost) and docker0 are listened to. You can specify an NIC name or IP address to set the NIC to be listened to. Multiple NICs or IP addresses can be added.

Currently, system and application logs can be collected from edge nodes.

- **System Logs**: logs generated by the IEF software (such as edge-core, edge-logger, and edge-monitor) installed on edge nodes.
- **Application Logs**: logs generated by applications deployed on edge nodes.

Step 6 Click **Next** and review the edge node specifications. If the specifications are correct, click **Buy Now**.

After the purchase is complete, the edge node is displayed in the edge node list, and **IEC node** is displayed as its node source.

----End

3.6 Edge Node Groups

NOTE

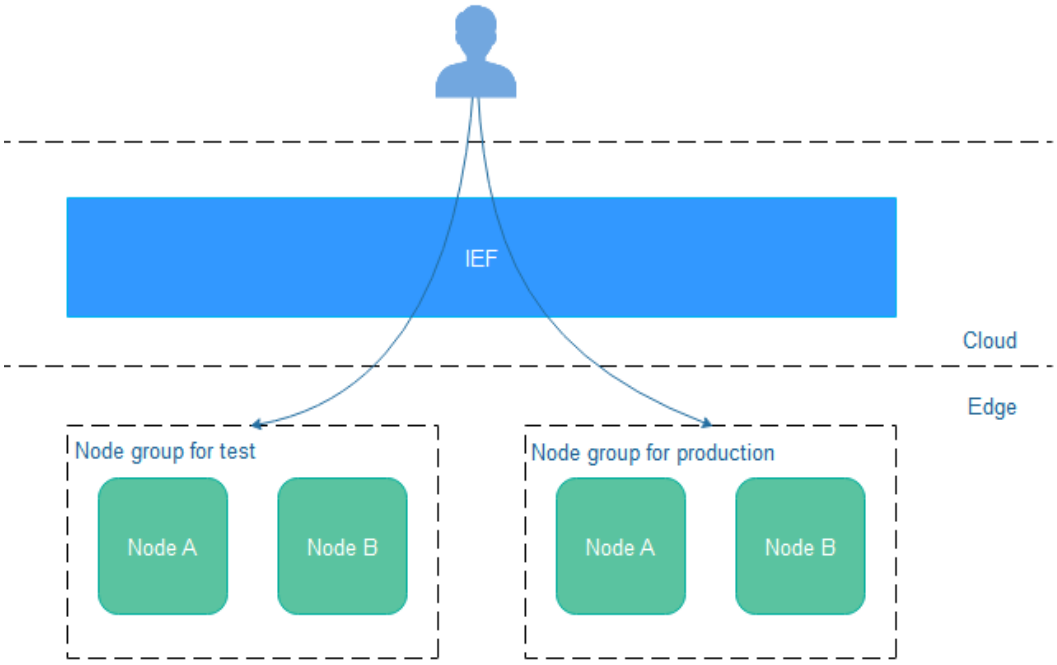
Edge node groups are available only for platinum service instances.

An edge node group can divide nodes and end devices and facilitate deployment of applications on nodes in the group. With node groups, you can:

- Add nodes and end devices to node groups, and remove nodes and end devices from node groups.
- Allocate security certificates for accessing nodes in a node group and applications deployed in the node group.
- Automatically schedule applications onto nodes in a node group based on the resource usage of the nodes.
- Reschedule an application onto other normal nodes when a node in a node group is faulty.
- Specify affinity and anti-affinity scheduling rules for applications and nodes.

 Copyright © Huawei Technologies Co., Ltd.

Figure 3-11 Node groups



Creating an Edge Node Group

Step 1 Switch to the platinum service instance.

If no platinum service instances have been created, create one by following the procedure described in **Creating a Platinum Service Instance**.

Step 2 Log in to the IEF console.

Step 3 In the navigation pane, choose **Managed Resources** > **Edge Node Groups**. Then, click **Create Edge Node Group** in the upper right corner.

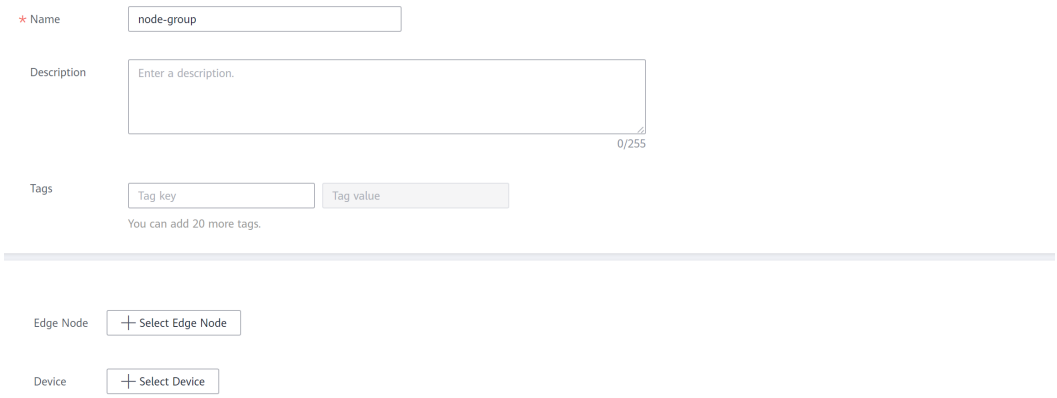
Step 4 Set the name and description of the node group, and select the edge nodes and end devices to be added to the node group.

Nodes added to the same node group must use the same architecture, for example, Arm or x86.

After a node group is created, you can still add or remove nodes and end devices.

 Copyright © Huawei Technologies Co., Ltd.

Figure 3-12 Creating an edge node group

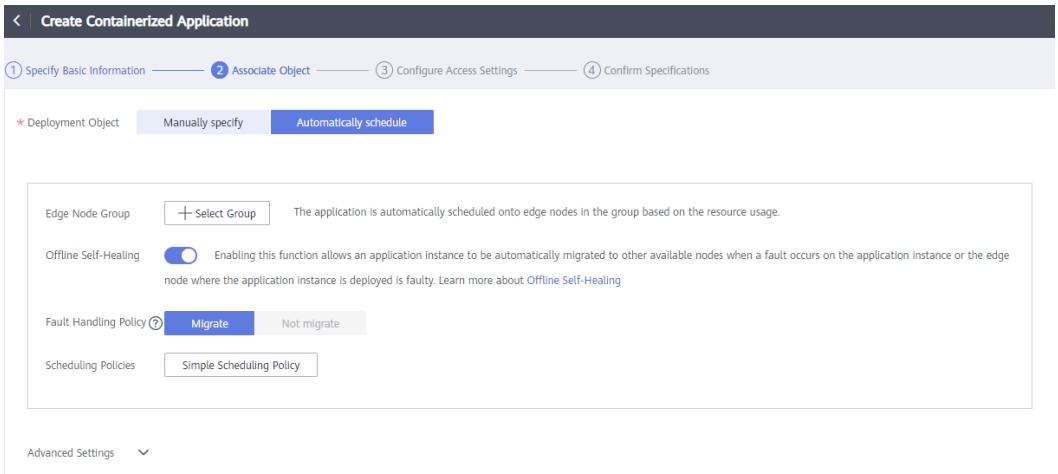


----End

Deploying an Application in a Specified Node Group

You can specify a node group where an application is to be deployed. The application will be deployed to the most suitable node in the group based on the resources of the nodes in the group. For details on how to deploy applications, see **Containerized Applications**.

Figure 3-13 Creating a containerized application



- **Fault Handling Policy**: Indicates whether the application instance is rescheduled onto another available node in the edge node group when the edge node that runs the application instance is unavailable.
- **Time Window (s)**: Indicates the duration of a fault before rescheduling is triggered.

Adding a Node Group Certificate

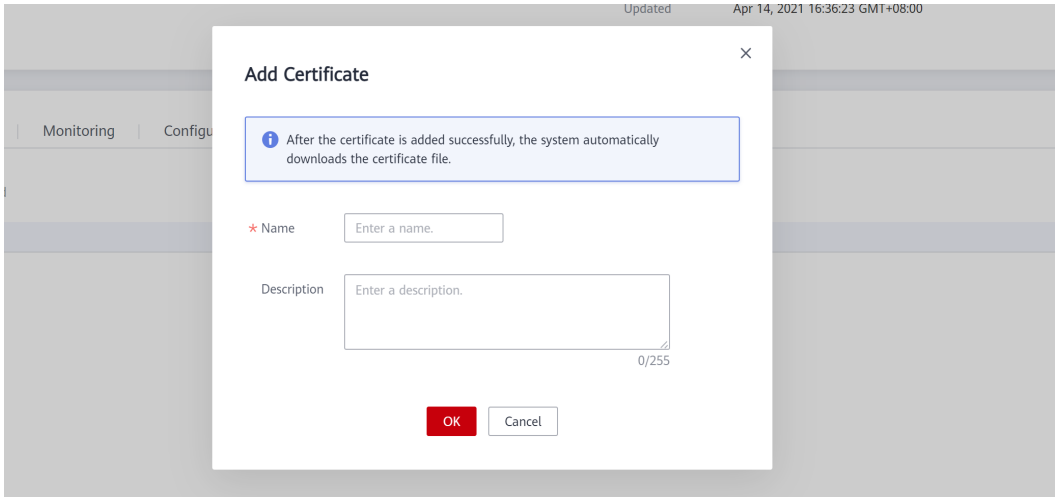
Applications deployed in a node group may be scheduled onto any node in the group, and end devices may also access any node in the node group. The certificates added for nodes cannot implement access to the MQTT broker on nodes from applications. A node group certificate is required.

 Copyright © Huawei Technologies Co., Ltd.

NOTE

The validity period of the certificate is 5 years.

Figure 3-14 Adding a node group certificate



3.7 Upgrading an Edge Node

Context

IEF releases new versions of EdgeCore irregularly. You can upgrade the EdgeCore software installed on your edge node as required.

Version Support Policy

IEF only maintains the edge software versions released within one year. Therefore, you are advised to upgrade your edge nodes at least once a year.

Version Upgrade Rule

During edge node upgrade, IEF automatically selects the EdgeCore of the latest version.

For example, if EdgeCore 2.22.0, 2.23.0, and 2.24.0 are available and the EdgeCore version on your edge node is 2.12.0, IEF will push the EdgeCore 2.24.0 to your edge node.

Precautions

- IEF does not proactively upgrade EdgeCore on your edge nodes. You are advised to upgrade the node manually in the time window with the minimum impact on your services.
- Upgrading a version within the maintenance period will not interrupt your applications running on the edge node. However, if message routing is used, services may be temporarily affected.
- Upgrading a version beyond the maintenance period may temporarily interrupt services due to container restart.

 Copyright © Huawei Technologies Co., Ltd.

- Do not change node configurations during the node upgrade, such as restarting Docker, installing or uninstalling the GPU/NPU driver, upgrading the OS kernel, or modifying network configurations, which may result in node upgrade failures.

Procedure

Step 1 Log in to an edge node, and configure firewall rules.

Check the firewall status on the edge node.

```
systemctl status firewalld
firewall-cmd --state
```

In the command output, **notrunning** indicates that the firewall is disabled and **running** indicates that the firewall is enabled.

If the firewall is enabled, enable port 8883 or disable the firewall.

- To enable port 8883, run the following commands:

```
firewall-cmd --add-port=8883/tcp --permanent
systemctl restart firewalld
```

- To disable the firewall, run the following commands:

```
systemctl disable firewalld
systemctl stop firewalld
```

Step 2 Log in to the IEF console.

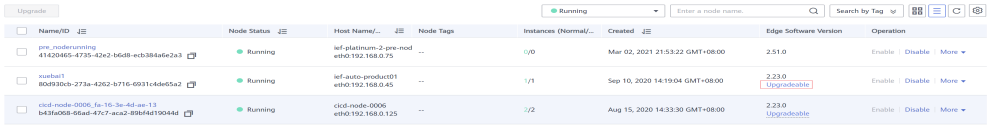
Step 3 In the navigation pane, choose **Managed Resources** > **Edge Nodes**.

Step 4 View the **Edge Software Version** column to check whether an upgrade can be performed.

Only edge nodes in the **Running** state can be upgraded.

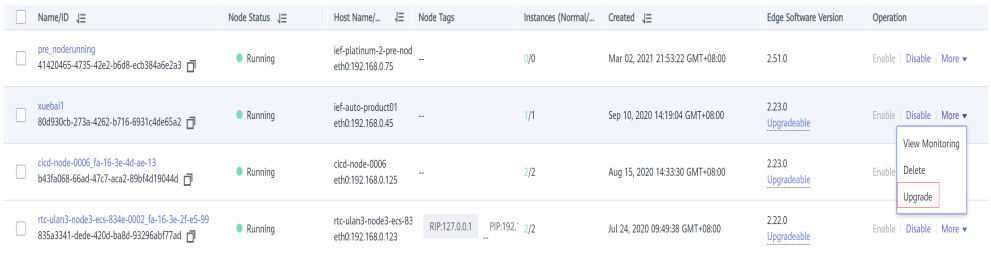
- If **Upgradeable** is displayed in the **Edge Software Version** column, you can perform an upgrade.
- If an edge node is in the **Running** state and **Upgradeable** is not displayed, the edge node already runs the latest EdgeCore.

Figure 3-15 Checking whether an edge node can be upgraded



Step 5 Choose **More** > **Upgrade**.

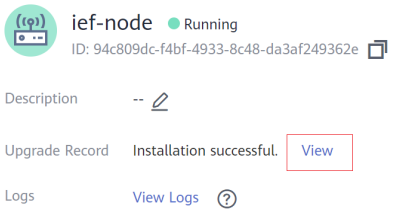
Figure 3-16 Upgrading an edge node



 Copyright © Huawei Technologies Co., Ltd.

Step 6 Click the node name to go to the node details page, and view the upgrade record.

Figure 3-17 Upgrade record



----End

3.8 Logs, Monitoring, and Alarms

Log Description

If the log function is enabled for an edge node on the IEF console, the system and application logs generated for the edge node will be uploaded to AOM.

- **System Logs**: logs generated by the IEF software (such as edge-core, edge-logger, and edge-monitor) installed on edge nodes.
- **Application Logs**: logs generated by applications deployed on edge nodes.
  - Edge nodes will upload logs in the **/var/IEF/app/log** directory to AOM through IEF. You can mount the **/var/IEF/app/log/*{appName}*** directory into the container when creating an application. For details, see **hostPath: used for mounting a directory of the host into the container**. You can view logs on AOM by *{appName}*.
  - Edge nodes will also upload container logs in ***{{DOCKER_ROOT_DIR}}*/containers/*{containerID}*/*{containerID}*-json.log** to AOM. You can run the **docker info** command to query the value of *DOCKER_ROOT_DIR*. *containerID* indicates the container ID.

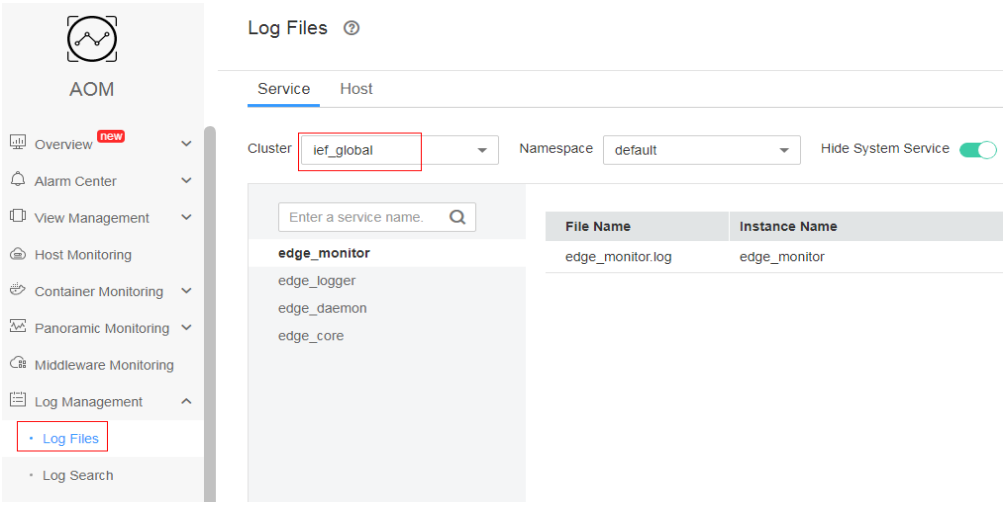
Viewing Logs on AOM

Step 1 Log in to the AOM console.

Step 2 In the navigation pane, choose **Log Management** > **Log Files**. On the displayed page, select cluster **ief_global** and namespace **default**.

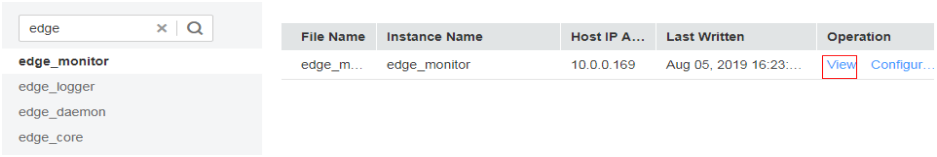
 Copyright © Huawei Technologies Co., Ltd.

Figure 3-18 Selecting a cluster and a namespace



Step 3 Search for logs by application name, and click **View** in the row where the log file resides to view detailed logs.

Figure 3-19 Viewing logs



----End

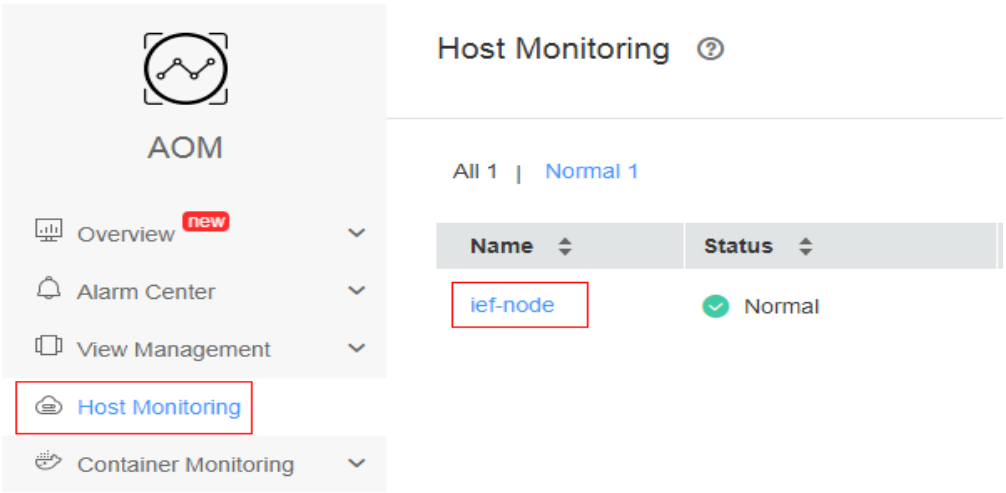
Viewing Node Monitoring Information on AOM

You can view the node monitoring information on AOM.

Step 1 Log in to the AOM console.

Step 2 Click the name of the node whose monitoring information is to be viewed.

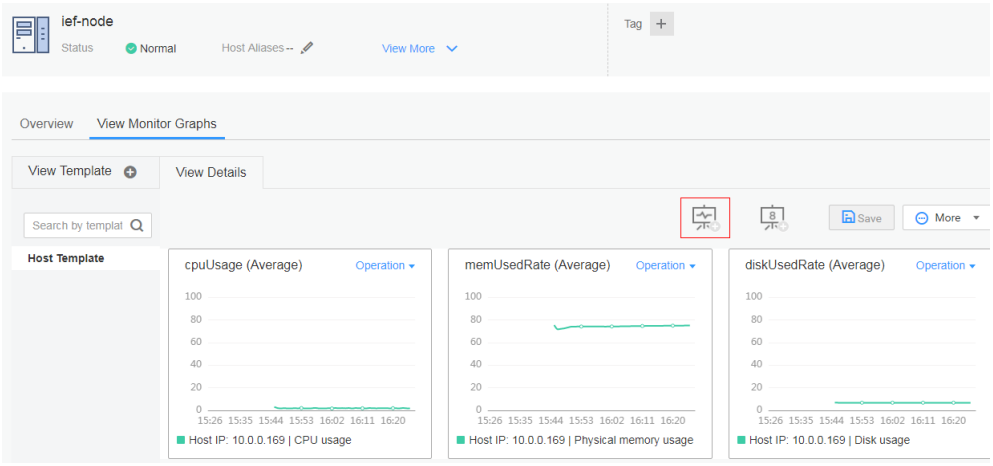
Figure 3-20 Selecting a monitored node



 Copyright © Huawei Technologies Co., Ltd.

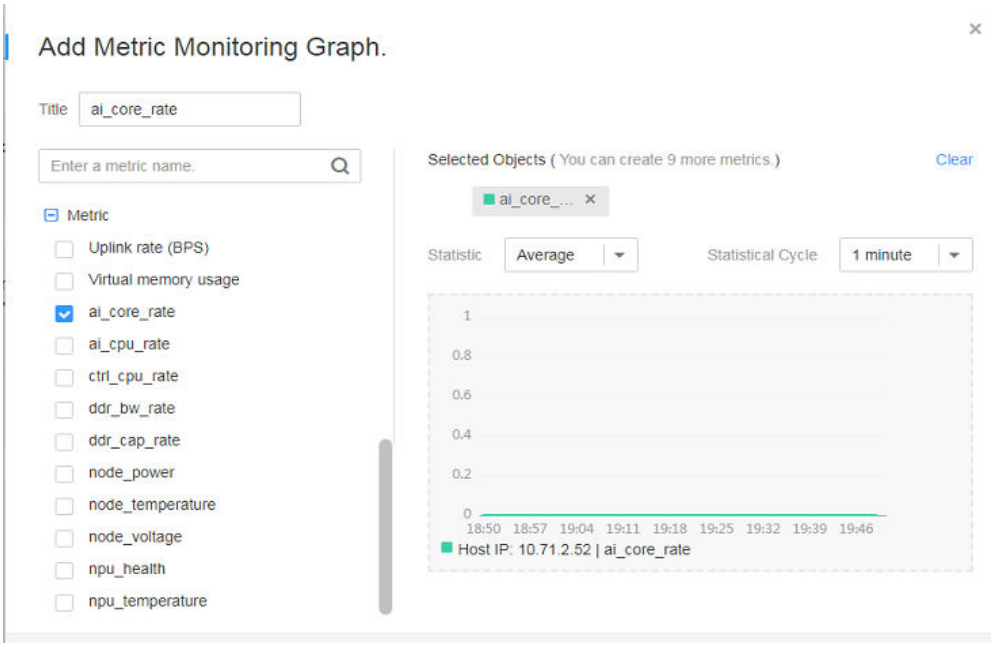
Step 3 On the **View Monitor Graphs** tab page, view the resource usage of the node, such as the CPU usage and memory usage.

Figure 3-21 Viewing monitoring information



If the edge node is an Atlas 500 AI edge station, you can also view NPU information.

Figure 3-22 Viewing monitoring information about an Atlas 500 AI edge station



The following table describes the metrics about an Atlas 500 AI edge station.

Metric	Description
ai_core_rate	AI core usage
ai_cpu_rate	AI CPU usage
ctrl_cpu_rate	Control CPU usage

 Copyright © Huawei Technologies Co., Ltd.

Metric	Description
ddr_cap_rate	DDR memory usage
ddr_bw_rate	DDR bandwidth usage
node_power	Node power
node_temperature	Node temperature
node_voltage	Node voltage
npu_health	Chip health status
npu_temperature	Chip temperature

----End

Viewing Container Monitoring Information on AOM

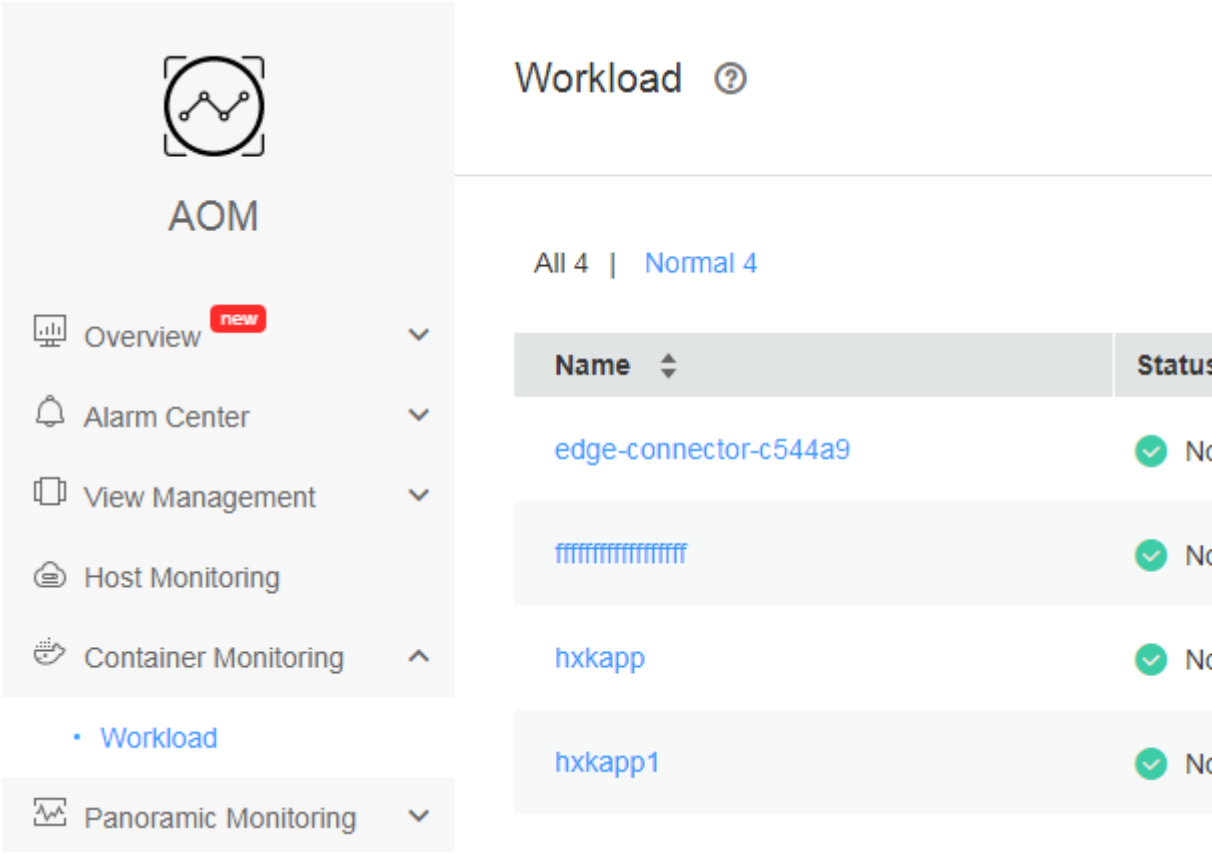
You can view the monitoring information about the application containers deployed on edge nodes on AOM.

Step 1 Log in to the AOM console.

Step 2 Select the workload whose monitoring information is to be viewed.

 Copyright © Huawei Technologies Co., Ltd.

Figure 3-23 Selecting a workload



Step 3 On the **View Monitor Graphs** tab page, view the resource usage of the container, such as the CPU usage and memory usage.

----End

Alarms Preset on IEF

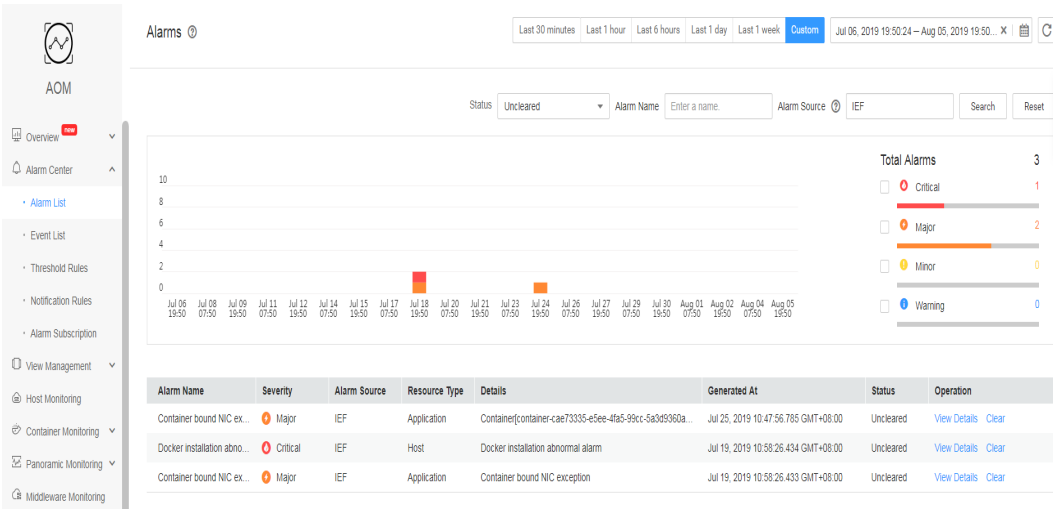
IEF comes with seven types of alarms preconfigured for each edge node. These alarms will be automatically reported to AOM.

Alarm Name	Trigger	Clearance Condition
Container Engine Is Abnormal	Docker has been enabled on an edge node, and the Docker information fails to be queried.	Docker is running properly, and edge-core can obtain the Docker information.
Application Liveness Probe Is Abnormal	A liveness probe has been configured for the application, and the probe detects an exception.	The probe detects that the container is normal.
Failed to Obtain GPU Resources	GPU resources could not be obtained during GPU application deployment.	GPU resources are obtained.

Copyright © Huawei Technologies Co., Ltd.

Alarm Name	Trigger	Clearance Condition
Failed to Obtain GPU Information	GPU has been enabled on an edge node, and the GPU information fails to be queried.	The GPU information is successfully queried.
Invalid AK/SK	The edge-hub has distributed 10 consecutive temporary AK/SK pairs and detects that the AK/SK has expired or is in abnormal state.	The edge-hub successfully distributes the temporary AK/SK.
Application Restarted	The application container restarts unexpectedly.	This alarm does not need to be cleared.
NIC Bound to the Container Is Faulty	The NIC bound to the container is faulty.	The NIC bound to the container becomes normal.

Figure 3-24 Viewing alarms



Setting Alarms on AOM

You can create threshold rules on AOM to monitor the metrics of edge nodes.

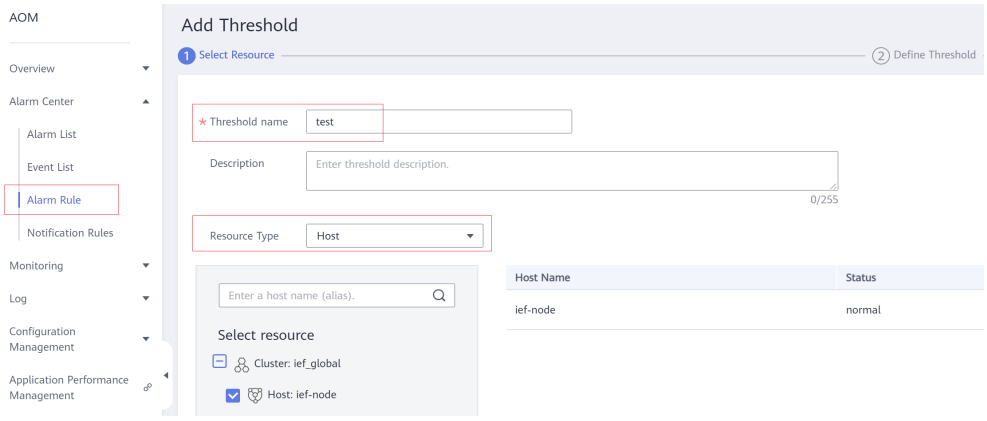
Step 1 Log in to the AOM console.

Step 2 In the navigation pane, choose **Alarm Center** > **Alarm Rule**. Then, click **Add Threshold**.

Step 3 Enter a threshold name, and select a resource.

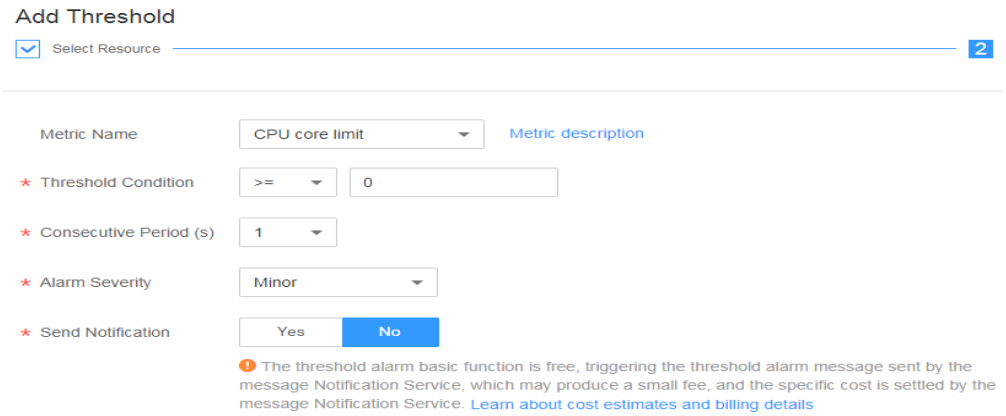
 Copyright © Huawei Technologies Co., Ltd.

Figure 3-25 Selecting a resource



Step 4 Click **Next**, and set a rule.

Figure 3-26 Setting a rule



Step 5 Click **Submit**.

After the threshold rule is created, you can view it in the threshold rule list.

When a metric meets the rule condition, an alarm is generated. You can view the alarm in the alarm list.

----End

Reporting User-Defined Alarms to AOM

IEF can report customized alarms from edge nodes to AOM. To be specific, after the MQTT client publishes alarm information to the MQTT broker, IEF will automatically report the alarms to AOM.

For details, see **Alarm Addition** and **Alarm Clearance**.

 Copyright © Huawei Technologies Co., Ltd.

3.9 Copying GPU Driver Files

Context

For a GPU-enabled edge node, you need to copy GPU driver files to specific directories on the edge node before managing it.

Currently, IEF supports NVIDIA Tesla GPUs such as P4, P40 and T4, and the GPU drivers that match CUDA Toolkit 8.0 to 10.0.

Prerequisites

The GPU driver has been installed on the edge node following the official NVIDIA guide.

Procedure

Step 1 Log in to the edge node as user **root**.

Step 2 Run the following command:

nvidia-modprobe -c0 -u

Step 3 Create directories.

mkdir -p /var/IEF/nvidia/drivers /var/IEF/nvidia/bin /var/IEF/nvidia/lib64

Step 4 Copy GPU driver files to the directories.

cp /lib/modules/*{Kernel version of the current environment}***/kernel/drivers/video/nvi* /var/IEF/nvidia/drivers/**

cp /usr/bin/nvidia-* /var/IEF/nvidia/bin/

cp -rd /usr/lib64/libcuda* /var/IEF/nvidia/lib64/

cp -rd /usr/lib64/libEG* /var/IEF/nvidia/lib64/

cp -rd /usr/lib64/libGL* /var/IEF/nvidia/lib64/

cp -rd /usr/lib64/libnv* /var/IEF/nvidia/lib64/

cp -rd /usr/lib64/libOpen* /var/IEF/nvidia/lib64/

cp -rd /usr/lib64/libvdpau_nvidia* /var/IEF/nvidia/lib64/

cp -rd /usr/lib64/vdpau /var/IEF/nvidia/lib64/

You can run the **uname -r** command to view the kernel version of the current environment. The following is an example. Replace the kernel version with the actual value.

```
# uname -r
3.10.0-514.e17.x86_64
```

----End

 Copyright © Huawei Technologies Co., Ltd.

4 End Device Management

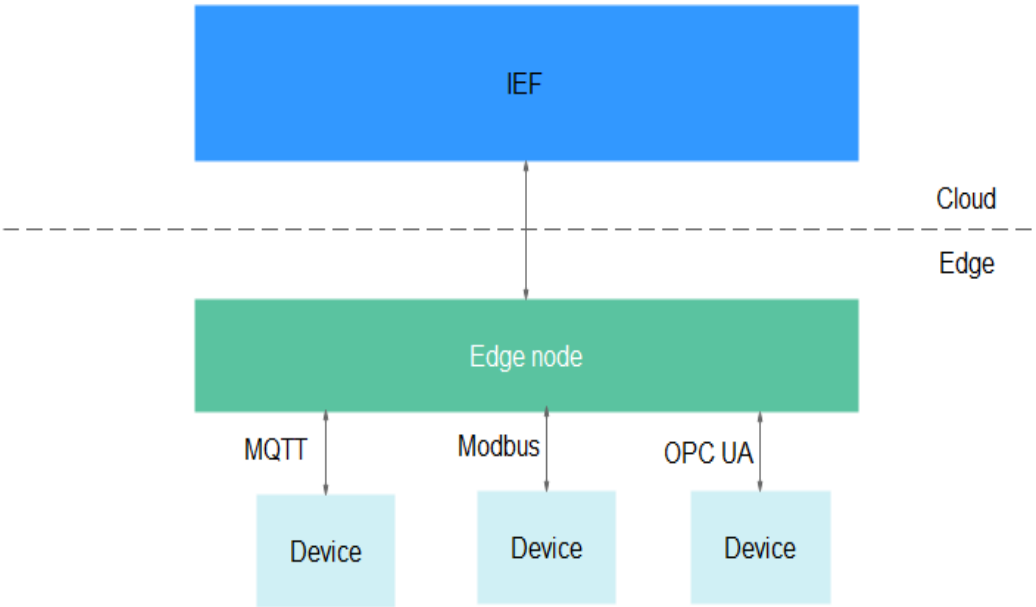
4.1 End Devices and Device Twins

End Device

End devices can be as small as a sensor or controller or as large as a smart camera or computer numerical control (CNC) machine tool.

These devices can be connected to IEF through edge nodes by using MQTT, Modbus, or OPC UA. After end devices are connected to IEF, you can manage them on IEF in a unified manner.

Figure 4-1 End device management



Device Twins

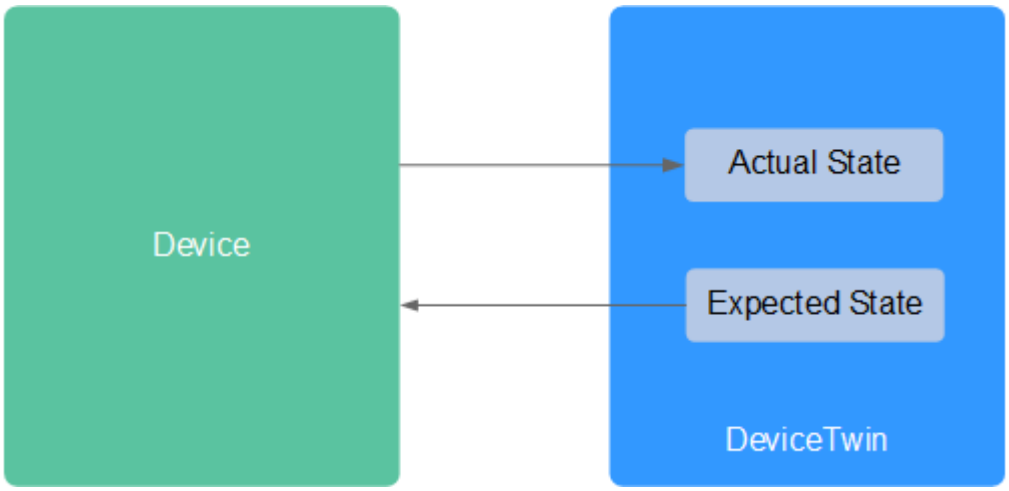
An end device usually has two types of data:

 Copyright © Huawei Technologies Co., Ltd.

- Metadata that does not change, which includes the serial number, asset identifier, MAC address, and other detailed information about the device. This type of data can be called static properties or **device properties**.
- Dynamic device data, which includes dedicated real-time device data for a specific scenario, for example, the on and off states of a light. This type of data can be called **twin properties**.

DeviceTwin has the same features as physical devices and facilitates better communication between end devices and applications. The commands sent by an application reach DeviceTwin (an internal component on an edge node). Then, DeviceTwin updates the device status according to **Expected State** set by the application. In addition, the end device reports its **Actual State** in real time. DeviceTwin records both **Actual State** and **Expected State** of the end device. In this way, the status of an offline device can be synchronized when it comes back online.

Figure 4-2 DeviceTwin



On the IEF console, you can register an end device and bind it to an edge node. After the binding, the device and twin properties will be stored on the edge node. Applications deployed on the edge node can obtain device and twin properties, and modify **Expected State** and **Actual State** recorded in DeviceTwin. In addition, IEF synchronizes twin properties between the cloud and edge. If a conflict occurs, the properties at the edge are used.

For details about the edge-cloud synergy for the end device status, see **Device Twin Working Principles**.

Procedure

To enable IEF to manage and control an end device, perform the following steps:

1. Define a device template (including device and twin properties).
2. Use the template to register a device.

   You can also register a device without using a template.
3. Bind the device to an edge node.
4. Perform related operations, such as monitoring the device status.

 Copyright © Huawei Technologies Co., Ltd.

4.2 Device Templates

In an edge computing scenario, there are a large number of end devices. You can define a template for end devices of the same type and use it to register the devices on IEF. For example, you can create a device template named **Camera**. After you use this template to register end devices, these devices will use all properties defined in the template. In this way, you do not need to keep repeating the same configuration.

IEF allows you to define templates based on access protocols, including commonly used protocols, **Modbus** and **OPC UA**, and custom protocols.

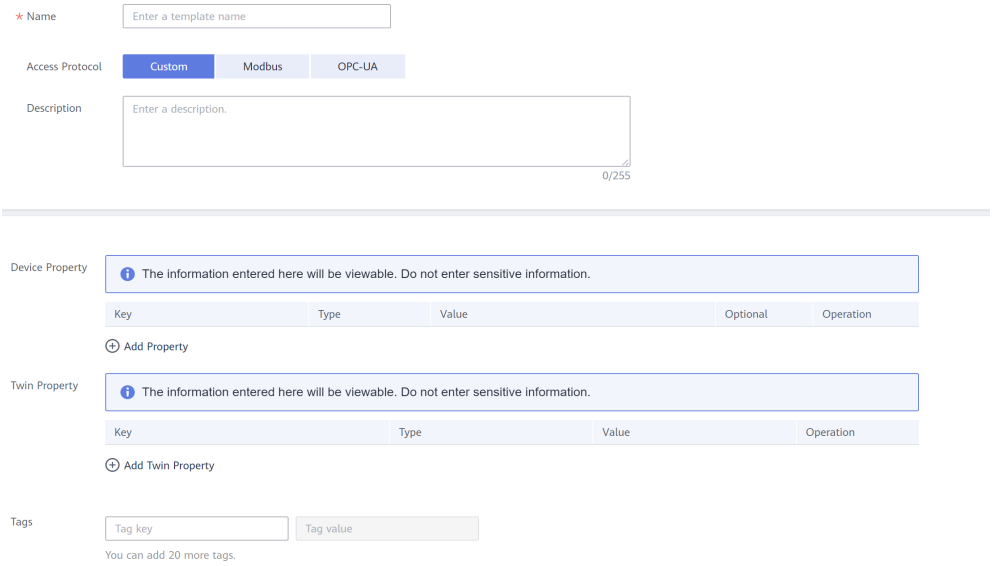
Creating a Device Template

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Managed Resources** > **End Devices**. Then, click **Create Device Template** in the upper right corner.

Step 3 Enter a template name, select an access protocol, and set template description, device properties, twin properties, and device tags.

Figure 4-3 Creating a device template



- **Name**: name of a device template.
- **Access Protocol**: IEF supports **Custom**, **Modbus**, and **OPC-UA**.
- **Device Property**:

  Properties are defined as key-value pairs. Enter a property name and value, and select a type.

  Metadata that does not change, such as serial numbers, asset identifiers, and MAC addresses, is defined in templates as device properties.
- **Twin Property**:

 Copyright © Huawei Technologies Co., Ltd.

The way twin properties are configured depends on the protocol type.

Dynamic device data, such as dedicated real-time device data for a specific scenario, is set as twin properties. For example, the on and off states of a light are real-time data.

- **Custom**: Twin properties are defined as key-value pairs. Enter a property name and value, and select a type.
- **Modbus**: For details, see **Twin Properties of Devices Using Modbus**.
- **OPC-UA**: For details, see **Twin Properties of Devices Using OPC UA**.

NOTICE

IEF does not provide any encryption or decryption tools, and does not sense your device attribute values. If the device attribute values are encrypted, you need to decrypt them.

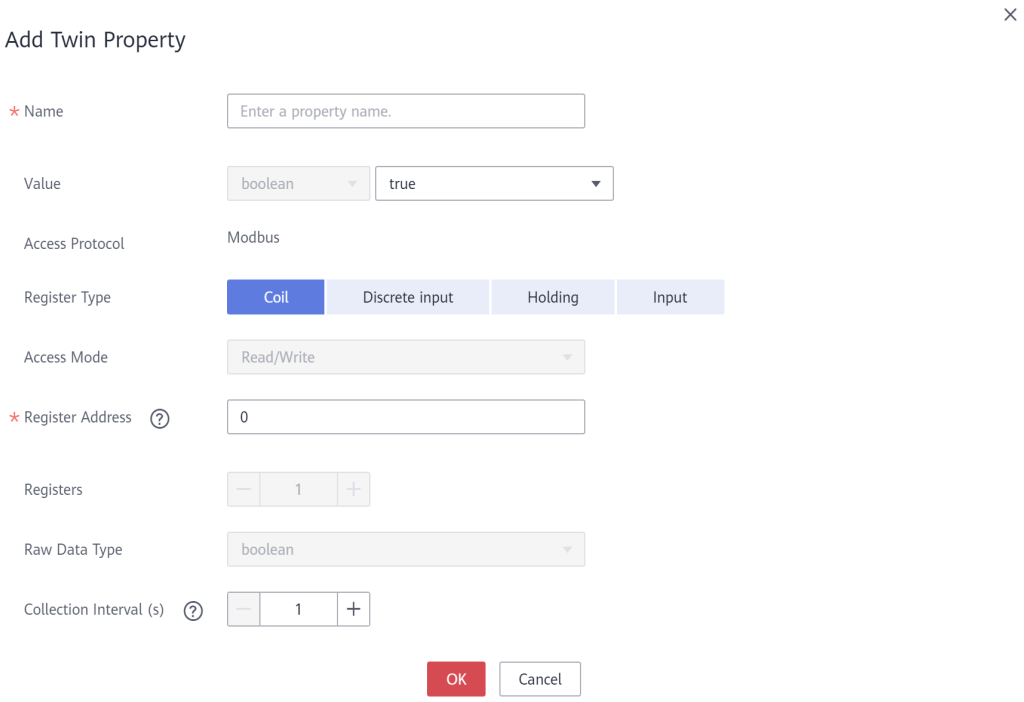
- **Tags**: Tags are used to classify end devices. You can quickly search for desired devices by tag. Tags also facilitate end device management by category.

Step 4 Click **Create**.

----End

Twin Properties of Devices Using Modbus

Figure 4-4 Twin properties of devices using Modbus



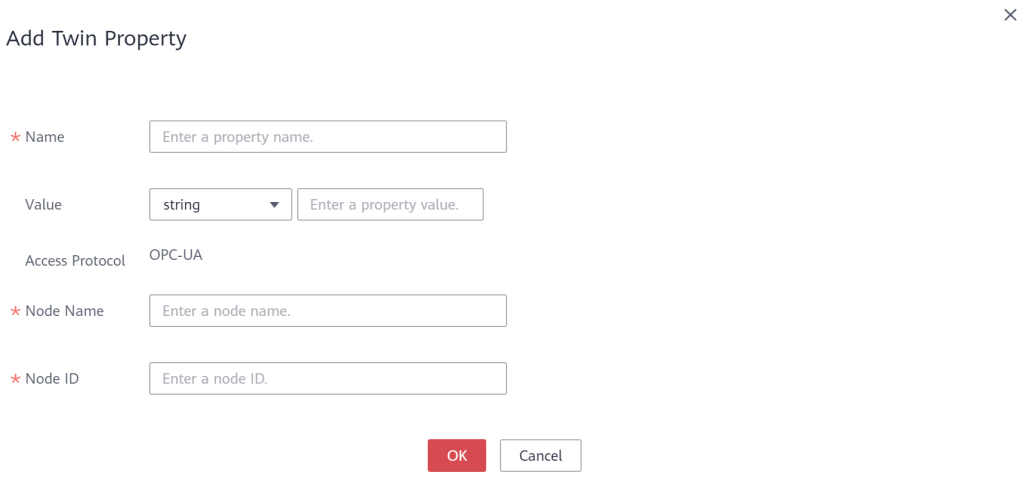
- **Register Type**: supports **Coil**, **Discrete input**, **Holding**, and **Input**.

 Copyright © Huawei Technologies Co., Ltd.

- **Access Mode**: The access mode depends on the register type. Coil and holding registers support the read/write mode. Discrete input registers and input registers support the read-only mode.
- **Register Address**: register offset address.
- **Registers**: number of registers.
- **Raw Data Type**: initial type of data stored in devices.
- **Collection Interval**: interval at which device data is collected and reported.
- **Bit Transposition**: transposes the high and low bits of a register.
- **Register Transposition**: transposes the high and low bits among registers.

Twin Properties of Devices Using OPC UA

Figure 4-5 Twin properties of devices using OPC UA



Twin properties are defined as key-value pairs. Enter a property name and value, and select a type.

In addition to the property name and value, you also need to specify the node name and ID.

In the OPC UA protocol, devices are defined and stored by node. The node name and ID mentioned here are the name and ID in device properties stored in the OPC UA server.

4.3 End Devices

End devices can be connected to IEF through edge nodes by using the custom, Modbus, or OPC UA protocol. After end devices are connected to IEF, you can manage them on IEF in a unified manner.

Registering a Device

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Managed Resources** > **End Devices**. Then, click **Register End Device** in the upper right corner.

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Set device parameters.

- **Name**: name of an end device.
- **ID**: You can customize an ID or enable IEF to automatically generate an ID. If you want to customize a device ID, select **Custom device ID** and enter an ID.
- **Access Protocol**: **Custom**, **Modbus**, and **OPC-UA** are available.
- **Device Configuration**: Select an existing device template to add properties for the device automatically. The template defines device properties, twin properties, and device tags. You can also manually add device properties and tags without using a template. The meanings of the properties and tags manually added are the same as those defined in the template. For details, see **Device Templates**.
- **Access Configuration**: This configuration is required when **Access Protocol** is set to **Modbus** or **OPC-UA**. For details, see **Modbus-based Access Configuration** and **OPC UA-based Access Configuration**.

Step 4 Click **Register**.

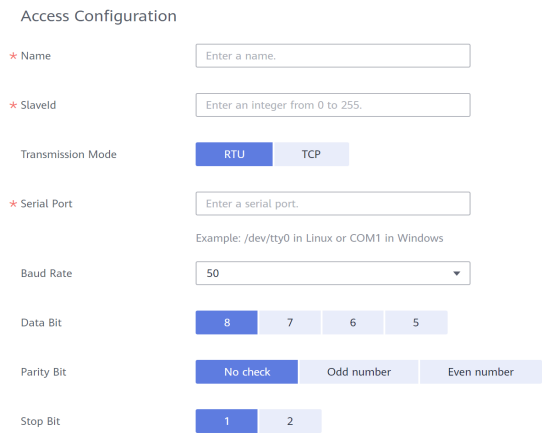
----End

Modbus-based Access Configuration

Modbus supports two transmission modes: RTU and TCP.

- The access configuration parameters in RTU mode are as follows:

Figure 4-6 Access configuration parameters in RTU mode

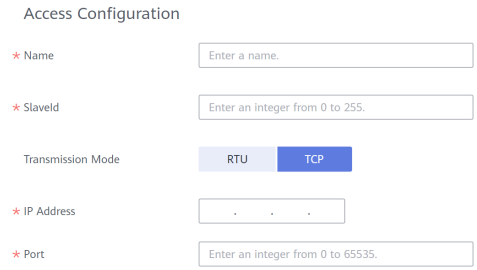


  - **SlaveId**: identifier for accessing register values.
  - **Serial Port**: serial port used to connect to an end device. The value varies according to the operating system of an edge node.
  - **Baud Rate**: number of symbols transmitted per second, which is used to measure the data transmission rate.
  - **Data Bit**: measures the actual data bit in communications.
  - **Parity Bit**: a simple error correction method for determining whether noise is interfering with communications or whether transmitted data is not synchronizing properly with received data.

 Copyright © Huawei Technologies Co., Ltd.

  - **Stop Bit**: the last bit of a data packet.
- The access configuration parameters in TCP mode are as follows:

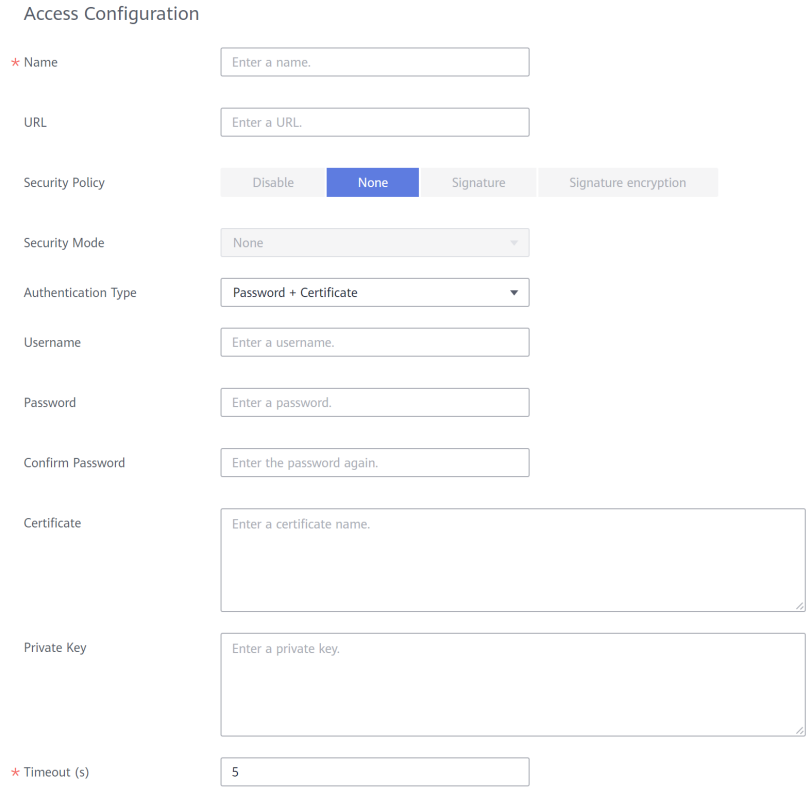
Figure 4-7 Access configuration parameters in TCP mode



  - **IP Address**: IP address of an end device.
  - **Port**: port number of an end device.

OPC UA-based Access Configuration

Figure 4-8 OPC UA-based access configuration



- **Security Policy**: supports **Disable**, **None**, **Signature**, and **Signature encryption**.
- **Security Mode**: authentication mode for communication between end devices and edge nodes.

 Copyright © Huawei Technologies Co., Ltd.

- **Authentication Type**: supports **Password**, **Certificate**, and **Password + Certificate**.
- **Certificate** and **Private Key**: The values are encoded in Base64. For details, see **Base64 Encoding**.

Base64 Encoding

To perform Base64 encoding on a character string, run **echo -n** *{Content to be encoded}* **| base64**. For example:

```
root@ubuntu:~# echo -n "3306" | base64
MzMwNg==
```

Follow-up Procedure

You can bind end devices to edge nodes. For details, see **Binding an End Device to an Edge Node**.

4.4 Binding an End Device to an Edge Node

An edge node can be bound to multiple end devices, but each end device can be bound to only one edge node. By binding an end device to an edge node, you can deploy applications on the edge node so that you can manage the device and monitor its status on IEF.

Binding an Edge Node

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Managed Resources** > **End Devices**.

Step 3 Click **Bind Node** in the row where the end device is located.

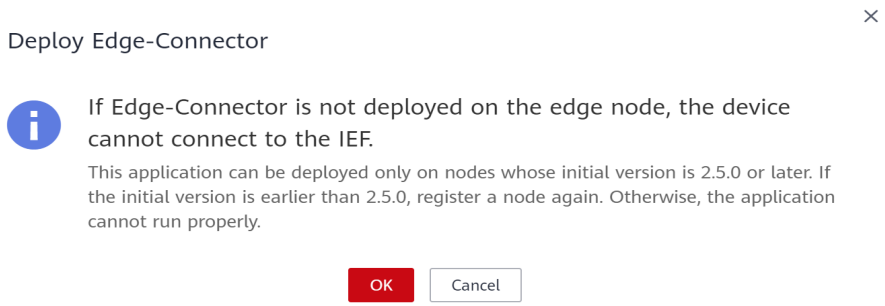
Step 4 Enter the relationship between the device and the node, select the edge node to be bound, and click **OK**.

If an end device using Modbus or OPC UA is to be bound to a node, a message that prompts you to deploy an Edge-Connector application will be displayed. In this case, click **OK**.

The Edge-Connector application converts Modbus and OPC UA messages into MQTT topics so that the end device can communicate with IEF. For details about the communication among end devices, edge nodes, and IEF, see **Device Twin Working Principles**.

 Copyright © Huawei Technologies Co., Ltd.

Figure 4-9 Prompt for Edge-Connector deployment



NOTE

After an end device is bound to a node, a ConfigMap will be automatically created. For details, see **Additional Description About Binding of an End Device Using Modbus or OPC UA**.

You can deploy an Edge-Connector application during binding or after binding. For details, see **Manually Deploying an Edge-Connector Application**.

----End

Binding an End Device on the Edge Node Details Page

You can also bind an end device on the details page of an edge node.

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Managed Resources** > **Edge Nodes**. Then, click an edge node name.

Step 3 On the edge node details page, click the **End Devices** tab and then click **Bind**.

Step 4 In the dialog box that is displayed, select the device to be bound, enter the device description, and click **OK**.

----End

Additional Description About Binding of an End Device Using Modbus or OPC UA

After an end device using Modbus or OPC UA is bound to an edge node, a DPL ConfigMap and an Edge-Connector application will be automatically created on IEF. You can view the Edge-Connector application in the containerized application list. The application is named in the format of **edge-connector-*{First six digits of the node ID}***,

The DPL ConfigMap is named using the format **dpl-cm-*{First six digits of the node ID}***. This ConfigMap stores the configuration details for the Edge-Connector application. You can view the DPL ConfigMap in the ConfigMap list.

 Copyright © Huawei Technologies Co., Ltd.

Manually Deploying an Edge-Connector Application

If an Edge-Connector application is not deployed when you bind an end device using Modbus or OPC UA, you can deploy this application in the edge marketplace.

Step 1 Log in to the IEF console. In the navigation pane, choose **Edge Marketplace**.

Step 2 On the **Edge Application Center** page, select the CPU architecture of the edge node, and click **Deploy** on the **Modbus&OPC UA Edge-Connector** card.

Step 3 Enter an application name, and click **Next**.

Step 4 Select an edge node, and select the DPL ConfigMap in the **Volumes** area. The DPL ConfigMap is named using the format **dpl-cm-*{First six digits of the node ID}***.

Step 5 Click **Create**.

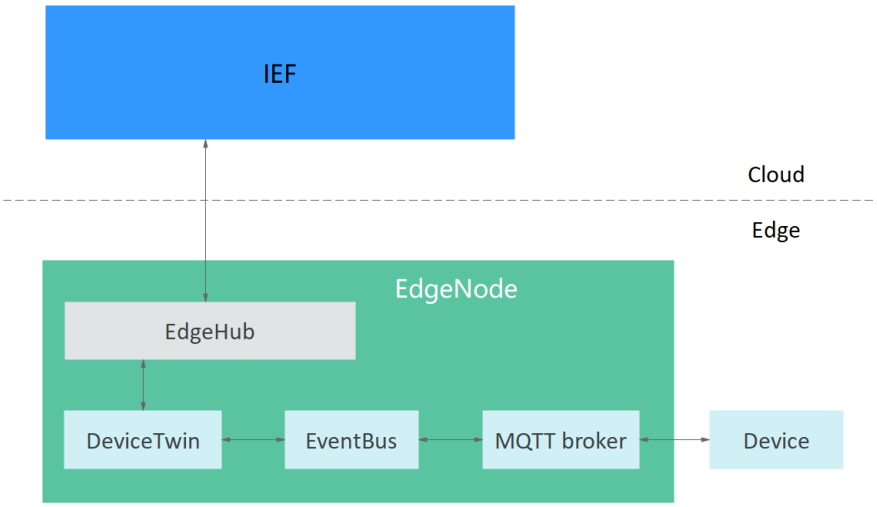
----End

4.5 Device Twin Working Principles

After an edge node is managed, Edge Agent is installed on the edge node. Edge Agent provides the following components for end device management:

- EdgeHub: a WebSocket client, which provides functions such as synchronizing cloud resource updates and reporting edge node and end device data to the cloud.
- DeviceTwin: stores end device status and synchronizes the end device status to the cloud.
- EventBus: a client that interacts with the MQTT server and provides the functions of subscribing to and publishing messages for other components.
- MQTT broker: an MQTT server.

Figure 4-10 End device management



DeviceTwin plays an important role during the communication among end devices, edge nodes, and IEF. It maintains the dynamic device data, including the

 Copyright © Huawei Technologies Co., Ltd.

dedicated real-time data of devices in a specific scenario, such as the on/off status of lights.

DeviceTwin has the same features as physical devices and facilitates better communication between end devices and applications. The commands sent by an application reach DeviceTwin. Then, DeviceTwin updates the device status according to **Expected State** set by the application. In addition, the end device reports its **Actual State** in real time. DeviceTwin records both **Actual State** and **Expected State** of the end device. In this way, the status of an offline device can be synchronized when it comes back online.

The following is an example of a device twin:

```
{
   "device": {
      "id": "989e4fc8-9f24-44d7-9f9c-a0bd3bfb1949",
      "description": "my home light",
      "name": "light",
      "state": "online",
      "twin": {
         "powerstatus": {
            "expected": {
               "value": "ON"
            },
            "actual": {
               "value": "OFF"
            },
            "metadata": {
               "type": "string"
            }
         }
      }
   }
}
```

In this example, a twin property **powerstatus** is defined for the device named **light**, in which the expected value is **ON** and the actual value is **OFF**.

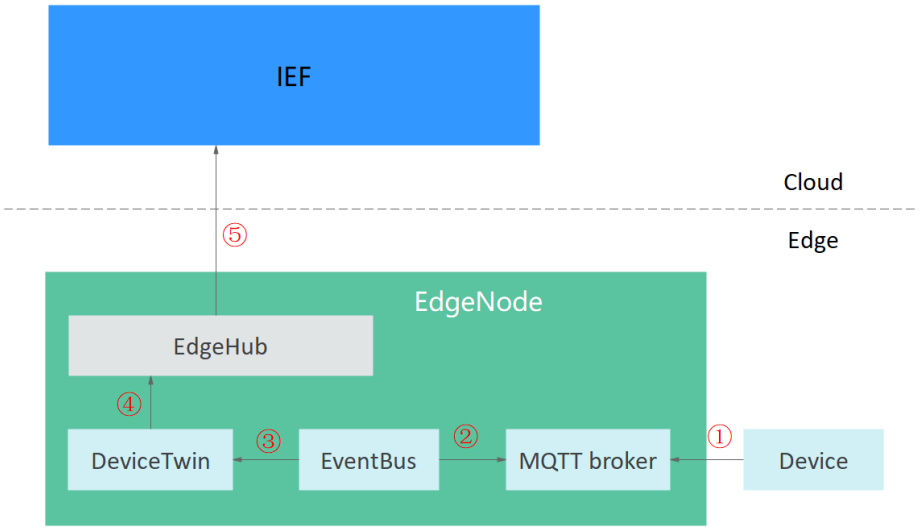
The following describes how this end device communicates with the edge node and IEF.

Reporting the Actual Device Status to the Cloud

Figure 4-11 shows the process for an end device to report its actual status to the cloud.

 Copyright © Huawei Technologies Co., Ltd.

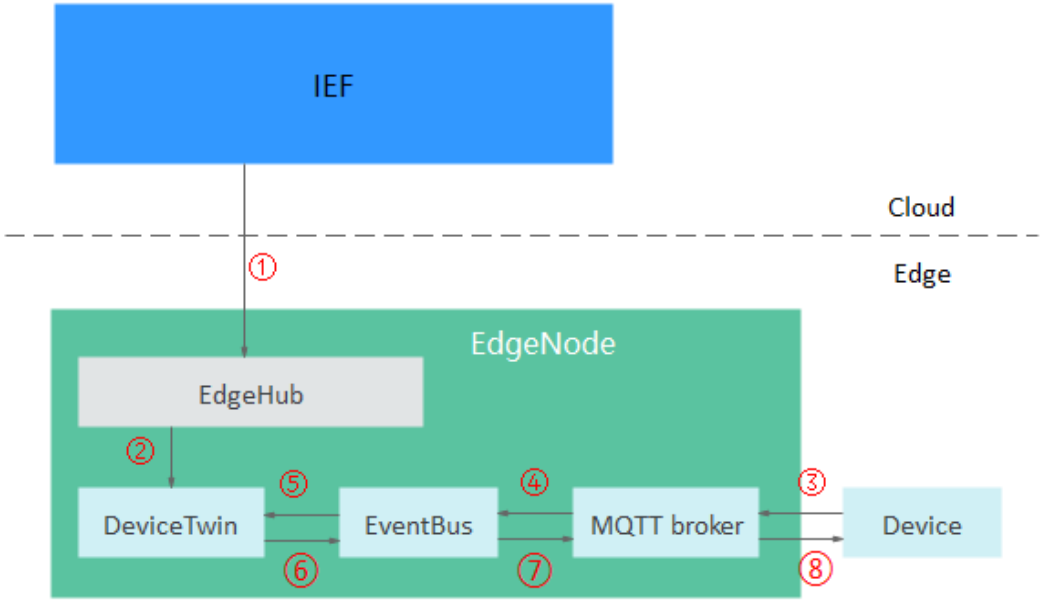
Figure 4-11 Reporting the device status



1. The end device reports its actual status to the MQTT broker in real time.
2. EventBus receives a subscription message from the MQTT broker. The message contains the actual status of the end device.
3. EventBus sends the actual device status to DeviceTwin. Then, DeviceTwin stores the actual device status on the edge node.
4. DeviceTwin synchronizes the actual status to EdgeHub (a WebSocket client).
5. EdgeHub sends a message to IEF.

Modifying Twin Properties in the Cloud to Control Device Status

Figure 4-12 Modifying the device status



1. Modify the twin properties of the end device on IEF. Then, IEF sends the expected status of the end device to EdgeHub on the edge node.

 Copyright © Huawei Technologies Co., Ltd.

2. EdgeHub sends a message containing the expected device status to DeviceTwin. Then, DeviceTwin stores the expected device status on the edge node.
3. The end device sends a message to the MQTT broker in real time to query the expected device status.
4. EventBus receives the message from the MQTT broker.
5. EventBus queries the expected device status based on the message.
6. DeviceTwin sends the expected device status to EventBus.
7. EventBus sends the expected device status to the MQTT broker.
8. The end device receives a subscription message from the MQTT broker and adjusts its status to the expected status.

4.6 Migrating Device Data to the Cloud

MQTT Broker

End devices can communicate with IEF using the MQTT protocol. You can also control end devices through pub/sub messaging.

An edge node has a **built-in MQTT broker**. The built-in MQTT broker uses port 8883 to communicate with end devices. End devices can communicate with the built-in MQTT broker only after they pass security authentication. For details, see **Performing Security Authentication Using Certificate**.

Edge nodes also support **external MQTT brokers**. You can install an MQTT broker (for example, **Mosquitto**, an open-source MQTT broker) on an edge node. Then devices can communicate with the edge node over port 1883.

NOTE

If an external MQTT broker is used, ensure that the communication port of the external MQTT broker is available.

MQTT Topics

The MQTT broker forwards messages between end devices and edge nodes to enable the end devices to communicate with the nodes and IEF. By default, the MQTT broker provides message topics described in **Table 4-1**. Device status can be reported and controlled through pub/sub messaging.

After an application is compiled, you can deploy the application to an edge node through IEF. For details, see **Containerized Application Management**.

Copyright © Huawei Technologies Co., Ltd.

Table 4-1 Default topics provided by IEF

Name	Type	Topic	Description
Device Twin Update	Subscribe	$hw/events/device/{device_id}/twin/update/document	This topic is used to subscribe to device twin updates. It reflects the differences before and after a device twin update.
Device Twin Delta	Subscribe	$hw/events/device/{device_id}/twin/update/delta	This topic is used to subscribe to device twin delta events. When a device twin changes, the twin properties whose actual values are different from expected values are returned.
Device Member Update	Subscribe	$hw/events/node/{node_id}/membership/updated	This topic is used to subscribe to end device binding updates.
Device Property Update	Subscribe	$hw/events/device/{device_id}/updated	This topic is used to subscribe to end device property updates.
Device Member Acquisition	Publish	$hw/events/node/{node_id}/membership/get	This topic is used to subscribe to end device bindings.
Device Member Acquisition Result	Subscribe	$hw/events/node/{node_id}/membership/get/result	This topic is used to subscribe to the result of obtaining information about end device members.
Device Twin Acquisition	Publish	$hw/events/device/{device_id}/twin/get	This topic is used to publish the request for obtaining device twins.
Device Twin Acquisition Result	Subscribe	$hw/events/device/{device_id}/twin/get/result	This topic is used to subscribe to the result of obtaining device twins.
Device Twin Modification	Publish	$hw/events/device/{device_id}/twin/update	This topic is used to publish device twin modifications.
Device Twin Modification Result	Subscribe	$hw/events/device/{device_id}/twin/update/result	This topic is used to subscribe to the result of modifying device twins.

Copyright © Huawei Technologies Co., Ltd.

Name	Type	Topic	Description
Encryption Data Request	Publish	$hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/decrypt	This topic is used to publish the request for obtaining encryption data.
Encryption Data Acquisition	Subscribe	$hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/plaintext	This topic is used to subscribe to encryption data.
Alarm Reporting	Publish	$hw/alarm/{appname}/add	This topic is used to report alarms to AOM.
Alarm Clearance	Publish	$hw/alarm/{appname}/clear	This topic is used to clear alarms in AOM.
Custom Topics	Publish	{project_id}/nodes/{node_id}/user/{custom_topic}	Topics customized based on actual requirements. You can publish end device data to a custom topic in the MQTT broker of an edge node. IEF then forwards the device data to a DIS stream or an API Gateway address. Then, you can extract the data for processing and analysis.

The following describes how to obtain end device data at the edge, receive control messages from the cloud, and report end device data to the cloud. For details about the sample code for MQTT to send and receive messages, see **Go-Language Code Sample** and **Java-Language Code Sample**.

Obtaining End Devices Associated with an Edge Node

Step 1 Send a request to the topic **Device Member Acquisition**.

Topic: $hw/events/node/{node_id}/membership/get

Payload: {"event_id": "Custom ID"}

For example:

```
$hw/events/node/{node_id}/membership/get

{"event_id":""}
```

 Copyright © Huawei Technologies Co., Ltd.

Step 2 Subscribe to the topic **Device Member Acquisition Result**.

Topic: $hw/events/node/{node_id}/membership/get/result

The following is an example of the returned result:

```
{
   "event_id": "",
   "timestamp": 1554986455386,
   "devices": [
      {
         "id": "2144773f-13f1-43f5-af07-51991d4fd064",
         "name": "equipmentA",
         "state": "unknown",
         "attributes": {
            "name": {
               "value": "a",
               "optional": true,
               "metadata": {
                  "type": "string"
               }
            }
         }
      }
   ]
}
```

----End

Obtaining Device Twins

Step 1 Send a request to the topic used to **Device Twin Acquisition**.

Topic: $hw/events/device/{device_id}/twin/get

Payload: {"event_id": "Custom ID"}

Step 2 Subscribe to the topic **Device Twin Acquisition Result**.

Topic: $hw/events/device/{device_id}/twin/get/result

The following is an example of the returned result:

```
{
   "event_id": "",
   "timestamp": 1554988425592,
   "twin": {
      "humidity": {
         "expected": {
            "value": "0",
            "metadata": {
               "timestamp": 1554988419529
            }
         },
         "optional": true,
         "metadata": {
            "type": "int"
         }
      },
      "temperature": {
         "expected": {
            "value": "0",
            "metadata": {
               "timestamp": 1554988419529
            }
         },
         "optional": true,
         "metadata": {
            "type": "int"
```

 Copyright © Huawei Technologies Co., Ltd.

```
        }
      }
    }
}
```

----End

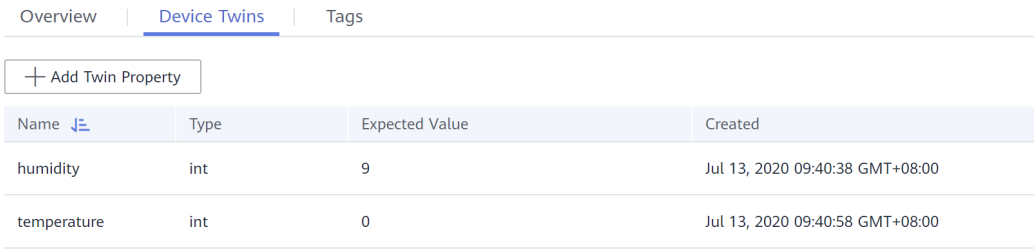
Listening to Device Twin Events

After **Obtaining End Devices Associated with an Edge Node** and **Obtaining Device Twins** are performed, the ID of the end device bound to the node is obtained. Then, the events of the end device can be listened to.

Update the device twin by setting the expected value in the cloud to control the end device.

For example, a device has two twin properties: humidity and temperature.

Figure 4-13 Twin properties



Change the value of twin property **humidity** from **9** to **10**. After the twin property is changed, the device receives the following two events:

- **Device twin change event**: includes the twin information before and after the change.
- **Device twin delta event**: includes the device twin details and the delta information about the inconsistency between the expected and actual values of twin properties.

The device can receive the two events only after it has subscribed to the two topics.

Step 1 Subscribe to the topic **Device Twin Update**.

Topic: $hw/events/device/{device_id}/twin/update/document

The update message received at the edge is as follows:

```
{
  "event_id": "0f921313-4074-46a2-96f6-aac610721059",
  "timestamp": 1555313685831,
  "twin": {
    "humidity": {
      "last": {
        "expected": {
          "value": "9",
          "metadata": {
            "timestamp": 1555313665978
          }
        },
```

 Copyright © Huawei Technologies Co., Ltd.

```
            "optional": true,
            "metadata": {
               "type": "int"
            }
         },
         "current": {
            "expected": {
               "value": "10",
               "metadata": {
                  "timestamp": 1555313685831
               }
            },
            "optional": true,
            "metadata": {
               "type": "int"
            }
         }
      },
      "temperature": {
         "last": {
            "expected": {
               "value": "0",
               "metadata": {
                  "timestamp": 1555313665978
               }
            },
            "actual": {
               "value": "2",
               "metadata": {
                  "timestamp": 1555299457284
               }
            },
            "optional": true,
            "metadata": {
               "type": "int"
            }
         },
         "current": {
            "expected": {
               "value": "0",
               "metadata": {
                  "timestamp": 1555313685831
               }
            },
            "actual": {
               "value": "2",
               "metadata": {
                  "timestamp": 1555299457284
               }
            },
            "optional": true,
            "metadata": {
               "type": "int"
            }
         }
      }
   }
}
```


Step 2 Subscribe to the topic **Device Twin Delta**.

Topic: $hw/events/device/{device_id}/twin/update/delta

The update message received at the edge is as follows:


```
{
   "event_id": "60fb5baf-d4ad-47b0-a21e-8b57b52d0978",
   "timestamp": 1555313685837,
   "twin": {
      "humidity": {
         "expected": {
```


 Copyright © Huawei Technologies Co., Ltd.

```
            "value": "10",
            "metadata": {
               "timestamp": 1555313685831
            }
         },
         "optional": true,
         "metadata": {
            "type": "int"
         }
      },
      "temperature": {
         "expected": {
            "value": "0",
            "metadata": {
               "timestamp": 1555313685831
            }
         },
         "actual": {
            "value": "2",
            "metadata": {
               "timestamp": 1555299457284
            }
         },
         "optional": true,
         "metadata": {
            "type": "int"
         }
      }
   },
   "delta": {
      "humidity": "10",
      "temperature": "0"
   }
}
```


----End

Reporting Actual Values of Device Properties

Step 1 Publish a twin update event for an end device.

Topic: $hw/events/device/{device_id}/twin/update

Payload: {"event_id":"", "timestamp":0, "twin":{"Property":{"actual": {"value":"Actual value"}}}}

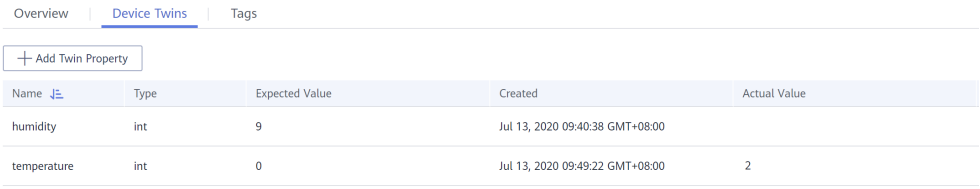
For example:


```
{
   "event_id": "",
   "timestamp": 0,
   "twin": {
      "temperature": {
         "actual": {
            "value": "2"
         }
      }
   }
}
```


After the event is published, you can find that the actual value of the device twin has changed in the cloud.

 Copyright © Huawei Technologies Co., Ltd.

Figure 4-14 Updated twin properties



Step 2 Subscribe to the topic **Device Twin Modification Result** at the edge. The edge node then can receive the result of the device twin update event.

Topic: $hw/events/device/{device_id}/twin/update/result

The update result is as follows:

```
{
    "event_id": "",
    "timestamp": 1554992093859,
    "twin": {
        "temperature": {
            "actual": {
                "value": "2",
                "metadata": {
                    "timestamp": 1554992093859
                }
            },
            "optional": true,
            "metadata": {
                "type": "int"
            }
        }
    }
}
```

----End

4.7 Performing Security Authentication Using Certificate

Scenario

By default, the built-in MQTT broker enables the port for Transport Layer Security (TLS) authentication. A client can access the MQTT broker only when it has a certificate.

End devices and applications can use the certificates added on the node details page for security authentication.

NOTE

Applications deployed in a node group may be scheduled onto any node in the node group. End devices may also access any node in the node group. The certificates added on the node details page cannot implement access to the MQTT broker on nodes from applications. Therefore, a node group certificate is required. For details on how to add a node group certificate, see **Adding a Node Group Certificate**.

 Copyright © Huawei Technologies Co., Ltd.

Constraints

- Edge connection certificates are bound to edge nodes. The certificates applied for on an edge node can be used only to access the MQTT broker of this edge node. If these certificates are used to access the MQTT broker of another edge node, the authentication will fail.
- A maximum of 10 certificates can be applied for an edge node.
- The validity period of a certificate is 5 years.
- There are constraints on using MQTT.

Table 4-2 MQTT constraints

Description	Restriction
Supported MQTT version	3.1.1
Differences from the standard MQTT protocol	• Quality of Service (QoS) 0 is supported. • Topic customization is supported. • QoS 1 and QoS 2 are not supported. • **will** and **retain** messages are not supported.
MQTTS security levels	TCP channel basic + TLS protocol (TLS v1.2)

Applying for a Certificate

NOTE

The validity period of a certificate is 5 years.

Step 1 Log in to the IEF console (**https://console.huaweicloud.com/ief2.0?locale=en-us**).

Step 2 In the navigation pane, choose **Managed Resources** > **Edge Nodes**.

Step 3 Click an edge node name. The edge node details page is displayed.

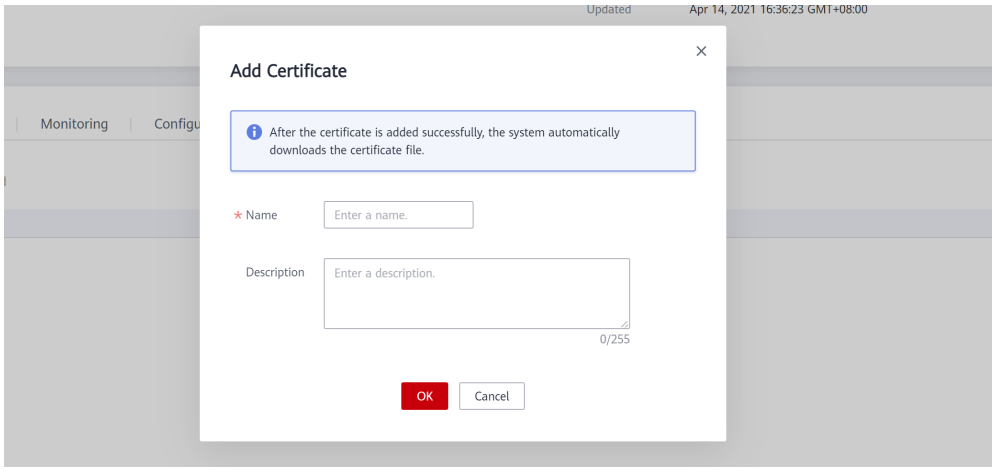
Step 4 Click the **Certificates** tab, and click **Add Certificate**.

Step 5 Enter a certificate name, select the certificate type, and click **OK**.

After the certificate is created, the system will automatically download the certificate file. Keep the certificate file secure.

Copyright © Huawei Technologies Co., Ltd.

Figure 4-15 Adding a certificate



----End

Using a Certificate

A certificate is used for authentication when an end device communicates with the MQTT broker.

Go-Language Code Sample and **Java-Language Code Sample** illustrate how to use certificates for authentication.

NOTE

- The client does not need to verify the server certificate. In other words, one-way authentication is required.
- Port 8883 of the built-in MQTT broker is enabled by default.
- In the Go-language code sample, the MQTT client references github.com/eclipse/paho.mqtt.golang (an open-source library).
- The MQTT client is required to process disconnection events and reestablish connections to improve the connection reliability.

Go-Language Code Sample

```
package main

import (
    "crypto/tls"
    "crypto/x509"
    "fmt"
    "math/rand"
    "sync"
    "time"

    MQTT "github.com/eclipse/paho.mqtt.golang"
)

func main() {
    subClient := InitMqttClient(onSubConnectionLost)
    pubClient := InitMqttClient(onPubConnectionLost)

    wait := sync.WaitGroup{}
    wait.Add(1)

    go func() {
```

 Copyright © Huawei Technologies Co., Ltd.

```
        for {
            time.Sleep(1*time.Second)
            pubClient.Publish("topic", 0, false, "hello world")
        }
    }()

    subClient.Subscribe("topic", 0, onReceived)

    wait.Wait()
}

func InitMqttClient(onConnectionLost MQTT.ConnectionLostHandler) MQTT.Client {
    pool := x509.NewCertPool()
    cert, err := tls.LoadX509KeyPair("/tmp/example_cert.crt", /tmp/example_cert.key")
    if err != nil {
        panic(err)
    }

    tlsConfig := &tls.Config{
        RootCAs: pool,
        Certificates: []tls.Certificate{cert},
         // One-way authentication, that is, the client does not verify the server certificate.
        InsecureSkipVerify: true,
    }
    // Use the TLS or SSL protocol to connect to port 8883.
    opts := MQTT.NewClientOptions().AddBroker("tls://
127.0.0.1:8883").SetClientID(fmt.Sprintf("%f",rand.Float64()))
    opts.SetTLSConfig(tlsConfig)
    opts.OnConnect = onConnect
    opts.AutoReconnect = false
    // Callback function. It is triggered immediately after the client is disconnected from the server.
    opts.OnConnectionLost = onConnectionLost
    client := MQTT.NewClient(opts)
    loopConnect(client)
    return client
}

func onReceived(client MQTT.Client, message MQTT.Message) {
    fmt.Printf("Receive topic: %s,  payload: %s \n", message.Topic(), string(message.Payload()))
}

// The reconnection mechanism is triggered after the subscribe client is disconnected from the server.
func onSubConnectionLost(client MQTT.Client, err error) {
    fmt.Println("on sub connect lost, try to reconnect")
    loopConnect(client)
    client.Subscribe("topic", 0, onReceived)
}

// The reconnection mechanism is triggered after the publish client is disconnected from the server.
func onPubConnectionLost(client MQTT.Client, err error) {
    fmt.Println("on pub connect lost, try to reconnect")
    loopConnect(client)
}

func onConnect(client MQTT.Client) {
    fmt.Println("on connect")
}

func loopConnect(client MQTT.Client) {
    for {
        token := client.Connect()
        if rs, err := CheckClientToken(token); !rs {
            fmt.Printf("connect error: %s\n", err.Error())
        } else {
            break
        }
        time.Sleep(1 * time.Second)
    }
}
```

 Copyright © Huawei Technologies Co., Ltd.

```
func CheckClientToken(token MQTT.Token) (bool, error) {
    if token.Wait() && token.Error() != nil {
        return false, token.Error()
    }
    return true, nil
}
```

Java-Language Code Sample

The format of an MqttClientDemo.java file is as follows:

```
/**********************************************************************************
 Description: A java demo of MQTT message sending and receiving. You need to create an edge node and
download the client certificate.
 **********************************************************************************/

package com.example.demo;

import javax.net.ssl.SSLSocketFactory;

import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

/**********************************************************************
 * MQTT demo shows how the client connects to the MQTT broker of the edge node to send and receive
messages and how SSL security authentication is performed on the connection. The demo illustrates the
following flows:
* 1. The MQTT subscribe client receives MQTT messages.
* 2. The MQTT publish client sends MQTT messages.
 **********************************************************************/
public class MqttClientDemo {
    private static int QOS_TYPE = 2;
    //MQTT server address
    private static final String MQTT_HOST = "ssl://x.x.x.x:8883";
    //MQTT publish client ID
    private static final String MQTT_PUB_CLIENT_ID = "pub_client_1";
    //MQTT subscribe client ID
    private static final String MQTT_SUB_CLIENT_ID = "sub_client_1";
    //MQTT channel subscription topic
    private static final String TOPIC = "/hello";
    //Paths of the SSL certificate used for MQTT client connection
    public static final String CLIENT_CRT_FILE_PATH = "example_cert.crt";
    public static final String CLIENT_KEY_FILE_PATH = "example_cert.key";
    //MQTT client connection timeout interval (s)
    public static final int TIME_OUT_INTERVAL = 10;
    //Interval at which the MQTT client sends a heartbeat message, in seconds
    public static final int HEART_TIME_INTERVAL = 20;
    //Interval at which the MQTT client retries upon disconnection, in milliseconds
    public static final int RECONNECT_INTERVAL = 10000;
    //Interval at which the MQTT client sends a message, in seconds
    public static final int PUBLISH_MSG_INTERVAL = 3000;

    //MQTT client
    private MqttClient mqttClient;
    //MQTT client ID.
    private String clientId;
    //MQTT client connection options
    private MqttConnectOptions connOpts;
    //Initialized MQTT client has not subscribed to any topic.
    private boolean isSubscribe = false;

    public MqttClientDemo(String id) throws MqttException {
        setClientId(id);
```

 Copyright © Huawei Technologies Co., Ltd.

```
        initMqttClient();
        initCallback();
        initConnectOptions();
        connectMqtt();
    }

    /**********************************************************************
     * Sending messages
     * @param message Message to be sent
     * @throws MqttException
     **********************************************************************/
    public void publishMessage(String message) throws MqttException {
        MqttMessage mqttMessage = new MqttMessage(message.getBytes());
        mqttMessage.setQos(QOS_TYPE);
        mqttMessage.setRetained(false);
        mqttClient.publish(TOPIC, mqttMessage);
        System.out.println(String.format("MQTT Client[%s] publish message[%s]", clientId, message));
    }

    /**********************************************************************
     * Subscribing to topics
     * @throws MqttException
     **********************************************************************/
    public void subscribeTopic() throws MqttException {
        int[] Qos = {QOS_TYPE};
        String[] topics = {TOPIC};
        mqttClient.subscribe(topics, Qos);
        isSubscribe = true;
    }

    /**********************************************************************
     * Starting the thread to periodically send MQTT messages
     * @throws MqttException
     **********************************************************************/
    public void startPublishMessage() {
        new Thread() {
            @Override
            public void run() {
                while (true) {
                    try {
                        Thread.sleep(PUBLISH_MSG_INTERVAL);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    try {
                        publishMessage("hello world!");
                    } catch (MqttException e) {
                        System.out.println(String.format("MQTT client[%s] publish message error,errorMsg[%s]",
clientId, e.getMessage()));
                    }
                }
            }
        }.start();
    }

    /**********************************************************************
     * Initializing the MQTT client
     * @throws MqttException Abnormal connection
     **********************************************************************/
    private void initMqttClient() throws MqttException {
        MemoryPersistence persistence = new MemoryPersistence();
        mqttClient = new MqttClient(MQTT_HOST, clientId, persistence);
    }

    /**********************************************************************
     * Initializing connection options
     * @throws MqttException Abnormal connection
     **********************************************************************/
    private void initConnectOptions() {
```

 Copyright © Huawei Technologies Co., Ltd.

```
        connOpts = new MqttConnectOptions();
        // Specify whether to clear the session. If the value is false, the server retains the client connection records. If the value is true, the client connects to the server as a new client.
        connOpts.setCleanSession(true);
        connOpts.setHttpsHostnameVerificationEnabled(false);
        // Set the timeout interval, in seconds.
        connOpts.setConnectionTimeout(TIME_OUT_INTERVAL);
        // Set the interval at which a session heartbeat message is sent, in seconds. The server sends a message to the client every 1.5x20 seconds to check whether the client is online. However, this method does not provide the reconnection mechanism.
        connOpts.setKeepAliveInterval(HEART_TIME_INTERVAL);
        SSLSocketFactory factory = null;
        try {
            factory = SslUtil.getSocketFactory(CLIENT_CRT_FILE_PATH, CLIENT_KEY_FILE_PATH);
        } catch (Exception e) {
            e.printStackTrace();
        }
        // TLS connection configuration
        connOpts.setSocketFactory(factory);
    }

    /**********************************************************************
     * Initiate an MQTT connect request.
     * @throws MqttException Abnormal connection
     **********************************************************************/
    private void connectMqtt() throws MqttException {
        mqttClient.connect(connOpts);
        System.out.println(String.format("MQTT client[%s] is connected,the connctOptions: \n%s", clientId, connOpts.toString()));
    }

    /**********************************************************************
     * Set the callback API.
     * @throws MqttException Abnormal connection
     **********************************************************************/
    private void initCallback() {
        mqttClient.setCallback(new MqttMessageCallback());
    }

    private void setClientId(String id) {
        clientId = id;
    }

    /**********************************************************************
     * MQTT client reconnection function. This function is called to check whether a topic has been subscribed to. If yes, the topic will be re-subscribed to.
     * @throws MqttException
     **********************************************************************/
    private void rconnectMqtt() throws MqttException {
        connectMqtt();
        if (isSubscribe) {
            subscribeTopic();
        }
    }

    /**********************************************************************
     * After the MQTT client subscribes to topics, the MQTT client receives messages through the callback API if the MQTT channel has data.
     * @version V1.0
     **********************************************************************/
    private class MqttMessageCallback implements MqttCallback {

        @Override
        public void connectionLost(Throwable cause) {
            System.out.println(String.format("MQTT Client[%s] connect lost,Retry in 10 seconds,info[%s]", clientId, cause.getMessage()));
            while (!mqttClient.isConnected()) {
                try {
                    Thread.sleep(RECONNECT_INTERVAL);
```

 Copyright © Huawei Technologies Co., Ltd.

```
                System.out.println(String.format("MQTT Client[%s] reconnect ....", clientId));
                rconnectMqtt();
            } catch (Exception e) {
                continue;
            }
        }

    }

    @Override
    public void messageArrived(String topic, MqttMessage mqttMessage) {
        String message = new String(mqttMessage.getPayload());
        System.out.println(String.format("MQTT Client[%s] receive message[%s] from topic[%s]", clientId,
message, topic));
    }

    @Override
    public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {

    }
  }


  public static void main(String[] args) throws MqttException {
    try {
      //Subscribe to the MQTT channel.
       MqttClientDemo mqttsubClientDemo = new
MqttClientDemo(MqttClientDemo.MQTT_SUB_CLIENT_ID);
       mqttsubClientDemo.subscribeTopic();
       //Send hello world to the MQTT channel.
       MqttClientDemo mqttpubClientDemo = new
MqttClientDemo(MqttClientDemo.MQTT_PUB_CLIENT_ID);
       mqttpubClientDemo.startPublishMessage();
    } catch (MqttException e) {
       System.out.println(String.format("program start error,errorMessage[%s]", e.getMessage()));
    }
  }
}
```

The format of an SSLUtil.java file is as follows:

```
/**********************************************************************
 Description: SSL utility class. Load the client SSL certificate configuration and ignore server certificate
verification.
 **********************************************************************/

package com.example.demo;

import java.io.ByteArrayInputStream;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyPair;
import java.security.KeyStore;
import java.security.Security;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.X509Certificate;

import javax.net.ssl.KeyManagerFactory;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSocketFactory;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.openssl.PEMReader;
import org.bouncycastle.openssl.PasswordFinder;

public class SslUtil {
```

```
    /**********************************************************************
     * Verify and obtain the SSLSocketFactory.
     **********************************************************************/
    public static SSLSocketFactory getSocketFactory(final String crtFile, final String keyFile) throws Exception
{
        Security.addProvider(new BouncyCastleProvider());

        // 1. Load the client certificate.
        PEMReader reader_client =
                new PEMReader(new InputStreamReader(new
ByteArrayInputStream(Files.readAllBytes(Paths.get(crtFile)))));
        X509Certificate cert = (X509Certificate) reader_client.readObject();
        reader_client.close();

        // 2. Load the client key.
        reader_client = new PEMReader(
                new InputStreamReader(new ByteArrayInputStream(Files.readAllBytes(Paths.get(keyFile)))),
                new PasswordFinder() {
                    @Override
                    public char[] getPassword() {
                        return null;
                    }
                }
        );

        // 3. Send the client key and certificate to the server for identity authentication.
        KeyStore ks = KeyStore.getInstance(KeyStore.getDefaultType());
        ks.load(null, null);
        ks.setCertificateEntry("certificate", cert);
        ks.setKeyEntry("private-key", ((KeyPair) reader_client.readObject()).getPrivate(), "".toCharArray(), new
Certificate[]{cert});
        KeyManagerFactory kmf =
KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
        kmf.init(ks, "".toCharArray());

        // 4. Create a socket factory.
        SSLContext context = SSLContext.getInstance("TLSv1.2");
        TrustManager[] tms = new TrustManager[1];
        TrustManager miTM = new TrustAllManager();
        tms[0] = miTM;
        context.init(kmf.getKeyManagers(), tms, null);

        reader_client.close();

        return context.getSocketFactory();
    }


    /**********************************************************************
     * Ignore server certificate verification.
     **********************************************************************/
    static class TrustAllManager implements TrustManager, X509TrustManager {
        @Override
        public X509Certificate[] getAcceptedIssuers() {
            return null;
        }

        @Override
        public void checkServerTrusted(X509Certificate[] certs, String authType)
                throws CertificateException {
        }

        public boolean isServerTrusted(X509Certificate[] certs) {
            return true;
        }

        public boolean isClientTrusted(X509Certificate[] certs) {
```

 Copyright © Huawei Technologies Co., Ltd.

```
            return true;
        }

        @Override
        public void checkClientTrusted(X509Certificate[] certs, String authType)
                throws CertificateException {
        }
    }
}
```

The format of a pom.xml file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>mqtt.example</artifactId>
    <version>1.0-SNAPSHOT</version>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>7</source>
                    <target>7</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
    <!-- https://mvnrepository.com/artifact/org.eclipse.paho/org.eclipse.paho.client.mqttv3 -->
    <dependency>
        <groupId>org.eclipse.paho</groupId>
        <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
        <version>1.2.1</version>
    </dependency>
     <!-- https://mvnrepository.com/artifact/org.bouncycastle/bcprov-jdk16 -->
    <dependency>
        <groupId>org.bouncycastle</groupId>
        <artifactId>bcprov-jdk16</artifactId>
        <version>1.45</version>
    </dependency>
    </dependencies>

</project>
```

4.8 MQTT Topics

4.8.1 Device Twin Update

This topic is used to subscribe to device twin updates. It reflects the differences before and after a device twin update.

Topic

$hw/events/device/{device_id}/twin/update/document

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Type	Description
device_id	String	End device ID.

Usage

Use an MQTT client to subscribe to this topic.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID.
timestamp	Int64	Timestamp when the event occurred.
twin	Object	A set of updated device twins. Each twin is in the key-value pair format. The value contains **last** (twin information before the update) and **last** (twin information after the update). The twin information contains the option flag, the twin metadata contains the value type, the expected status contains the expected value and update time, and the actual status contains the actual value and update time.

Example

When an end device is bound to an edge node, the following message is received:

```
$hw/events/device/{device_id}/twin/update/document
{
    "event_id":"",
    "timestamp":1557314742122,
    "twin":{
        "state":{
            "last":null,
            "current":{
                "expected":{
                    "value":"running",
                    "metadata":{
                        "timestamp":1557314742122
                    }
                },
                "optional":true,
                "metadata":{
                    "type":"string"
                }
            }
        }
    }
}
```

 Copyright © Huawei Technologies Co., Ltd.

4.8.2 Device Twin Delta

This topic is used to subscribe to device twin delta events. When a device twin changes, the twin properties whose actual values are different from expected values are returned.

Topic

$hw/events/device/{device_id}/twin/update/delta

Parameter	Type	Description
device_id	String	End device ID.

Usage

Use an MQTT client to subscribe to this topic.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID.
timestamp	Int64	Timestamp when the event occurred.
twin	Object	A set of updated device twins. Each twin is in the key-value pair format. The value contains the option flag, the twin metadata contains the value type, the expected status contains the expected value and update time, and the actual status contains the actual value and update time.
delta	Map	Names of twin properties whose actual values are different from expected values, and expected values of these properties.

Example

When an end device is bound to an edge node, the following message is received:

```
$hw/events/device/{device_id}/twin/update/delta

{
    "event_id":"b9625811-f34f-4252-bee9-98185e7e1ec7",
    "timestamp":1557314742131,
    "twin":{
        "state":{
            "expected":{
                "value":"running",
                "metadata":{
                    "timestamp":1557314742122
                }
```

 Copyright © Huawei Technologies Co., Ltd.

```
            },
            "optional":true,
            "metadata":{
                "type":"string"
            }
        }
    },
    "delta":{
        "state":"running"
    }
}
```

4.8.3 Device Member Update

This topic is used to subscribe to updates on bindings between end devices and edge nodes.

Topic

$hw/events/node/{node_id}/membership/updated

Parameter	Type	Description
node_id	String	Node ID.

Usage

Use an MQTT client to subscribe to this topic.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID.
timestamp	Int64	Timestamp when the event occurred.
added_devices	Array	Information about end devices, including the device ID, name, properties, and twin properties.
removed_devices	Array	Information about end devices, including the device ID, name, properties, and twin properties.

Example

When an end device is bound to an edge node, the following message is received:

```
$hw/events/node/{node_id}/membership/updated

{
    "event_id":"04a975ab-fd51-49be-85f5-5967e994f640",
    "timestamp":1557314742136,
    "added_devices":[
        {
            "id":"ab39361a-6fc0-4c94-b919-72b1e08ca690",
            "name":"IEF-device",
```

 Copyright © Huawei Technologies Co., Ltd.

```
            "state":"unknown",
            "attributes":{
               "address":{
                  "value":"shenzhen",
                  "optional":true,
                  "metadata":{
                     "type":"string"
                  }
               }
            },
            "twin":{
               "state":{
                  "expected":{
                     "value":"running",
                     "metadata":{
                        "timestamp":1557314434570
                     }
                  },
                  "optional":true,
                  "metadata":{
                     "type":"string"
                  }
               }
            }
         }
      ],
      "removed_devices":null
}
```

4.8.4 Device Property Update

This topic is used to subscribe to updates on end device properties.

Topic

$hw/events/device/{device_id}/updated

Parameter	Type	Description
device_id	String	End device ID.

Usage

Use an MQTT client to subscribe to this topic.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID.
timestamp	Int64	Timestamp when the event occurred.
attributes	Object	A set of updated end device properties, which are in the key-value pair format. The key indicates the property name, the value contains the property value and the optional flag, and metadata contains the value type.

 Copyright © Huawei Technologies Co., Ltd.

Example

When an end device is bound to an edge node, the following message is received:

```
$hw/events/device/{device_id}/updated
{
    "event_id":"",
    "timestamp":1557314742136,
    "attributes":{
        "address":{
            "value":"shenzhen",
            "optional":true,
            "metadata":{
                "type":"string"
            }
        }
    }
}
```

4.8.5 Device Member Acquisition

This topic is used to publish the request for obtaining end device member information.

Topic

$hw/events/node/{node_id}/membership/get

Parameter	Type	Description
node_id	String	Node ID.

Usage

Use an MQTT client to publish this topic. This topic must be used together with **Device Member Acquisition Result**.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID, which is user-defined.

Example

```
$hw/events/node/3fbb5b8d-32db-4271-a34f-a013e021b6ce/membership/get

{
    "event_id":"bc876bc-345d-4050-86a8-319a5b13cc10"
}
```

4.8.6 Device Member Acquisition Result

This topic is used to subscribe to the result of obtaining end device member information.

 Copyright © Huawei Technologies Co., Ltd.

Topic

$hw/events/node/{node_id}/membership/get/result

Parameter	Type	Description
node_id	String	Node ID.

Usage

Use an MQTT client to subscribe to this topic. This topic must be used together with **Device Member Acquisition**.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID.
timestamp	Int64	Timestamp when the event occurred.
devices	Array	Information about end devices, including the device ID, name, and properties. The properties are in the key-value pair format. The key indicates the property name, the value contains the property value and the optional flag, and metadata contains the value type.

Example

```
$hw/events/node/3fbb5b8d-32db-4271-a34f-a013e021b6ce/membership/get/result

{
    "event_id":"bc876bc-345d-4050-86a8-319a5b13cc10",
    "timestamp":1557317193524,
    "devices":[
        {
            "id":"ab39361a-6fc0-4c94-b919-72b1e08ca690",
            "name":"IEF-device",
            "state":"unknown",
            "attributes":{
                "address":{
                    "value":"longgang",
                    "optional":true,
                    "metadata":{
                        "type":"string"
                    }
                }
            }
        }
    ]
}
```

4.8.7 Device Twin Acquisition

This topic is used to publish the request for obtaining device twins.

 Copyright © Huawei Technologies Co., Ltd.

Topic

$hw/events/device/{device_id}/twin/get

Parameter	Type	Description
device_id	String	End device ID.

Usage

Use an MQTT client to publish this topic. This topic must be used together with **Device Twin Acquisition Result**.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID.

Example

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/get

{
    "event_id":"123456"
}
```

4.8.8 Device Twin Acquisition Result

This topic is used to subscribe to the result of obtaining device twins.

Topic

$hw/events/device/{device_id}/twin/get/result

Parameter	Type	Description
device_id	String	Device ID.

Usage

Use an MQTT client to subscribe to this topic. This topic must be used together with **Device Twin Acquisition**.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID.

Copyright © Huawei Technologies Co., Ltd.

Parameter	Type	Description
timestamp	Int64	Timestamp when the event occurred.
twin	Object	A set of end device twin information. Each twin is in the key-value pair format. The value contains the option flag, the twin metadata contains the value type, the expected status contains the expected value and update time, and the actual status contains the actual value and update time.

Example

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/get/result

{
    "event_id":"123456",
    "timestamp":1557317510926,
    "twin":{
        "state":{
            "expected":{
                "value":"stop",
                "metadata":{
                    "timestamp":1557316778931
                }
            },
            "optional":true,
            "metadata":{
                "type":"string"
            }
        }
    }
}
```

4.8.9 Device Twin Modification

This topic is used to publish device twin modifications.

Topic

$hw/events/device/{device_id}/twin/update

Parameter	Type	Description
device_id	String	End device ID.

Usage

Use an MQTT client to publish this topic. This topic must be used together with **Device Twin Modification Result**.

 Copyright © Huawei Technologies Co., Ltd.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID.
timestamp	Int64	Timestamp when the event occurred.
twin	Object	A set of device twin information to be modified. The twin property is in the key-value pair format. The key is the name of the twin property to be modified, and the value contains the expected value of the expected status to be modified or the actual value of the actual status to be modified.

Example

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/update

{
    "event_id":"123457",
    "twin":{
        "state":{
            "actual":{
                "value":"stop"
            }
        }
    }
}
```

4.8.10 Device Twin Modification Result

This topic is used to subscribe to the result of modifying device twins.

Topic

$hw/events/device/{device_id}/twin/update/result

Parameter	Type	Description
device_id	String	End device ID.

Usage

Use an MQTT client to subscribe to this topic. This topic must be used together with **Device Twin Modification**.

Parameter Description

Parameter	Type	Description
event_id	String	Event ID.

Copyright © Huawei Technologies Co., Ltd.

Parameter	Type	Description
timestamp	Int64	Timestamp when the event occurred.
twin	Object	A set of modified device twins. Each twin is in the key-value pair format. The value contains the option flag, the twin metadata contains the value type, the expected status contains the expected value and update time, and the actual status contains the actual value and update time.

Example

```
$hw/events/device/ab39361a-6fc0-4c94-b919-72b1e08ca690/twin/update/result

{
    "event_id":"123457",
    "timestamp":1557317614026,
    "twin":{
        "state":{
            "actual":{
                "value":"stop",
                "metadata":{
                    "timestamp":1557317614026
                }
            },
            "optional":true,
            "metadata":{
                "type":"string"
            }
        }
    }
}
```

4.8.11 Encryption Data Request

This topic is used to publish the request for obtaining encryption data.

Topic

$hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/decrypt

Parameter	Type	Description
project_id	String	Project ID. For details on how to obtain a project ID, see **Obtaining a Project ID**.
encryptdata_name	String	Name of an encryption data record.
properties_name	String	Key of an encryption item.

 Copyright © Huawei Technologies Co., Ltd.

Usage

Use an MQTT client to publish this topic. This topic must be used together with **Encryption Data Acquisition**.

A certificate must be used for security authentication when encryption data is requested. For details about the authentication method, see **Performing Security Authentication Using Certificate**.

Parameter Description

None.

Example

Publish an empty message.

4.8.12 Encryption Data Acquisition

This topic is used to subscribe to encryption data.

Topic

$hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/plaintext

Parameter	Type	Description
project_id	String	Project ID. For details on how to obtain a project ID, see **Obtaining a Project ID**.
encryptdata_name	String	Name of an encryption data record.
properties_name	String	Key of an encryption item.

Usage

Use an MQTT client to subscribe to this topic. This topic must be used together with **Encryption Data Request**.

A certificate must be used for security authentication when encryption data is requested. For details about the authentication method, see **Performing Security Authentication Using Certificate**.

Parameter Description

Parameter	Type	Description
plain_text	String	Plaintext value of an encryption item.

Copyright © Huawei Technologies Co., Ltd.

Parameter	Type	Description
ret_message	String	Error message.

Example

If the request is correct, the following message is received:

```
{
   "ret_code": 200,
   "plain_text": "xxxxxxxxxxx"
}
```

If the request is incorrect, the following message is received:

```
{
   "ret_code": 400,
   "ret_message": "xxxxxxxxxxx"
}
```

4.8.13 Alarm Addition

This topic is used to report alarms to AOM.

Topic

$hw/alarm/{appname}/add

Parameter	Type	Description
appname	String	Application name, which is user-defined.

Usage

Use an MQTT client to publish this topic.

Parameter Description

Parameter	Type	Description
alarmName	String	Alarm name.
alarmId	String	Alarm ID, which must be unique. For details, see **Generating an alarmId**.
detailedInformation	String	Alarm description.
url	String	URL for root cause analysis. If the URL is not available, leave this field blank.

Copyright © Huawei Technologies Co., Ltd.

Parameter	Type	Description
source	String	Alarm source. The value is a character string consisting of uppercase and lowercase letters.
cleared	Boolean	Whether the alarm is cleared. • **true**: The alarm has been cleared. • **false**: The alarm needs to be cleared manually.
policyID	String	ID of an alarm rule. For a threshold rule, set this parameter to **ruleId**. If no threshold rule is configured, leave this field blank.
objectInstance	String	Location information. If this field is not specified, the value of this parameter is the same as that of **alarmId** by default.
perceivedSeverity	Integer	Alarm severity. • 1: Critical • 2: Major • 3: Minor • 4: Warning
resourceId	Object	Alarm information. For details, see **Table 4-3**.
neType	String	Type of the resource for which the alarm is generated. • Application • DB • Host
eventType	Integer	Alarm type. • 21: dynamic threshold alarm • 22: batch threshold alarm • 23: threshold alarm • 24: system alarm • 25: probe added or deleted • 26: Agent installation alarm • 27: Quota threshold-crossing alarm

Table 4-3 resourceId

Parameter	Type	Description
namespace	String	Resource type. The options are as follows: • **PAAS.CONTAINER**: container metrics • **PAAS.NODE**: node metrics

 Copyright © Huawei Technologies Co., Ltd.

Parameter	Type	Description
dimension	Object	Dimension information, which is associated with the node application information reported by the monitoring module. For details, see **Table 4-4**. Currently, alarms are about nodes and applications. Therefore, you only need to pay attention to these two dimensions. The application dimension covers service, instance, container, and process information. You can select one or more types of information to be reported. • Service information: clusterId, nameSpace, and serviceID • Instance information: podID and podName • Container information: containerID and containerName • Process information: processID and processName

Table 4-4 Dimension

Parameter	Type	Description
clusterId	String	Project ID. For details on how to obtain a project ID, see **Obtaining a Project ID**.
nameSpace	String	The default value is **default**.
nodeIP	String	Node IP address.
serviceID	String	Service ID. • For applications, the value is the MD5 value of {projectId}_{hostid}_{appName}. • For processes, the value is the MD5 value of {projectId}_{hostid}_{Process name}_{Process pid}. • **processID** can be calculated as follows: md5({projectId})_{hostid}_md5({Process name})_{process pid}. **md5(projectId)** indicates the MD5 value of **projectId**, and **md5({Process name})** indicates the MD5 value of the process name. The dimension settings must be consistent with those reported by the monitoring module. Otherwise, the corresponding resources cannot be associated.
podID	String	Instance ID.
podName	String	Instance name.

Copyright © Huawei Technologies Co., Ltd.

Parameter	Type	Description
containerID	String	Container ID.
containerName	String	Container name.
processID	String	Process ID.
processName	String	Process name.

Generating an alarmId

NOTE

You do not need to use this method to generate the alarmId. However, you need to ensure that the alarmId is unique.

Generate the MD5 value of **{projectId}_{Service name for which the alarm is generated}_{Dimension information}_{Metric name}_{Alarm type}_{Rule information}**.

Where:

- **projectId**: Project ID. For details on how to obtain a project ID, see **Obtaining a Project ID**.
- **Dimension information**:
  - Node information: {clusterId}_{namespace}_{ip}
  - Container information: {clusterId}_{namespace}_{appName}_{podName}_{containerId}
  - Application information: {clusterId}_{namespace}_{appName}
- **Alarm type**:
  - 21: dynamic threshold alarm
  - 22: batch threshold alarm
  - 23: threshold alarm
  - 24: system alarm
  - 25: probe added or deleted
  - 26: Agent installation alarm
- **Rule information**: Set this field to a rule name for threshold alarms and to **NA** for alarms generated by the service itself. For dynamic threshold alarms, set this field to **policyId**.

Example

- Node alarm

```
{
  "alarmName": "test",
   "alarmId": "73ccbccce05de74f9d3dda42f6ecfe20",
   "detailedInformation": "test",
   "url": "",
   "source": "IEF",
```

```
    "cleared": false,
    "policyID": "",
    "perceivedSeverity": 4,
    "resourceId": {
        "namespace": "PAAS.NODE",
        "dimension": {
            "clusterId": "e277befa37a64ed1aa25b522e686bc28",
            "nameSpace": "default",
            "nodeIP": "192.168.0.164"
        }
    },
    "neType": "Host",
    "eventType": 23
}
```

- Application alarm:

```
{
    "alarmName": "Application restart",
    "alarmId": "b09076ff565c59d4da0db0c9223781",
    "detailedInformation": "Application restart test",
    "url": "",
    "source": "IEF",
    "cleared": false,
    "policyID": "",
    "perceivedSeverity": 3,
    "resourceId": {
        "namespace": "PAAS.CONTAINER",
        "dimension": {
            "containerName": "container-e991acd3-864c-4038-8a90-e042eebab496",
            "containerID": "70b385315c8ac507b3de7dfe1258932cea0b53a850b7d030ce7ed0a55c47877c",
            "podID": "0e9ce4fd-b732-11e9-8a30-fa163e9b3546",
            "podName": "hxkapp1-7898f5bd4b-2lj8z"
        }
    },
    "neType": "Application",
    "eventType": 23
}
```

4.8.14 Alarm Clearance

This topic is used to clear alarms in AOM. The message body is the same as that described in **Alarm Addition**. To be specific, the only difference between the requests for reporting and clearing alarms is the topic name.

Topic

$hw/alarm/{appname}/clear

Parameter	Type	Description
appname	String	Application name, which is user-defined.

Usage

Use an MQTT client to publish this topic.

Parameter Description

See **Alarm Addition**.

 Copyright © Huawei Technologies Co., Ltd.

4.8.15 Custom Topics

IEF allows you to customize topics based on actual requirements.

You can publish end device data to a custom topic in the MQTT broker of an edge node. IEF then forwards the device data to a DIS stream or an API Gateway address. Then, you can extract the data for processing and analysis.

To use a custom topic, create a message route in which a topic is defined on the IEF console. IEF will forward the data in the topic based on the message route. For details on how to create a message route, see **Edge-Cloud Message Overview**.

Topic

{project_id}/nodes/{node_id}/user/{custom_topic}

Usage

Use an MQTT client to publish this topic.

Parameter Description

IEF can transparently forward any content in a custom topic.

{custom_topic} supports the wildcards **#** and **+**. Therefore, multiple messages that meet a wildcard rule can be forwarded in a unified manner.

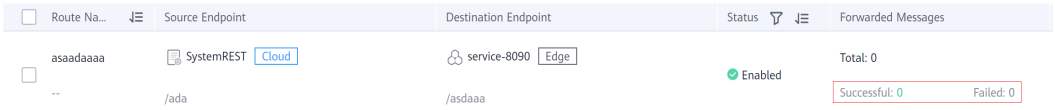
is a wildcard that matches any number (0 or greater) of levels in a topic. **+** is a wildcard that matches only one level in a topic.

IEF performs the minimum match on the messages that match the wildcard topic and then forwards the messages. For example, you can configure 123/+/567 as a wildcard rule for message topics 123/aaa/567 and 123/bbb/567.

Example

After a route is created, you can publish messages (data) to the custom topic defined in the route. IEF then forwards the messages to the specified endpoint. IEF also records the number of forwarded messages, including successful and failed messages.

Figure 4-16 Number of forwarded messages



 Copyright © Huawei Technologies Co., Ltd.

5 Containerized Application Management

5.1 Containerized Applications

IEF delivers containerized applications to edge nodes. The applications can be first released in the edge marketplace or they can be custom edge applications:

- Edge marketplace applications. For details about the edge marketplace, see **Edge Marketplace**.
- Custom edge applications. This section describes how to create a custom edge application.

Constraints

- When the disk usage of an edge node exceeds 70%, the image reclamation mechanism is started to reclaim the disk space occupied by the container image. In this case, if you deploy a containerized application, the application container will take a long time to start. Therefore, properly plan the disk space of the edge node before deploying the containerized application.
- When you create a containerized application, the edge node needs to pull an image from SWR. If the image size is too large and the download bandwidth of the edge node is small, the container image fails to be pulled. In this case, a message will be displayed on the IEF console, indicating that the containerized application fails to be created. However, the image pull will not be interrupted. After the container image is pulled successfully, the containerized application will be automatically created. To prevent such a problem, you can pull the container image to the edge node and then create a containerized application.
- The architecture of the container image to be used must be the same as that of the edge node on which a containerized application is to be deployed. For example, if the edge node uses x86, the container image architecture must also be x86.
- In a platinum service instance, the number of application instances can be reduced to zero.

 Copyright © Huawei Technologies Co., Ltd.

Creating an Edge Application

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Containerized Applications**. Then, click **Create Containerized Application** in the upper right corner.

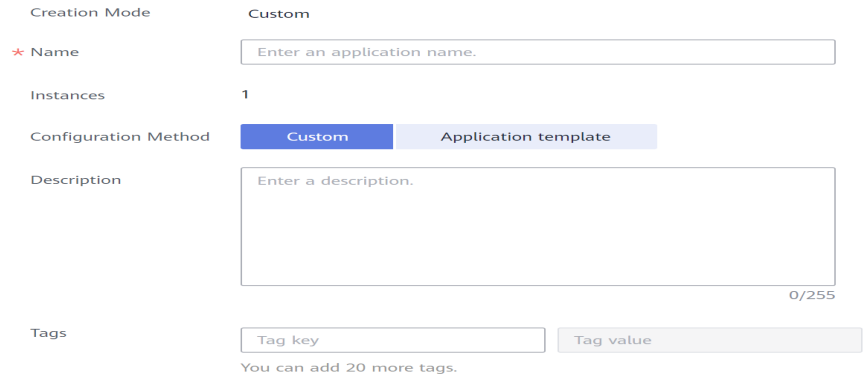
Step 3 Specify basic information.

- **Name**: name of a containerized application.
- **Instances**: number of application instances.
- **Configuration Method**
  - **Custom**: Configure the application step by step. For details, see **Step 4** to **Step 6**.
  - **Application template**: Select a predefined application template and modify it based on your requirements. This method reduces repeated operations. The configurations in a template are the same as those set in **Step 4** to **Step 6**. For details on how to create a template, see **Application Templates**.
- **Tags**

  Tags can be used to classify resources, facilitating resource management.

  If you need to use the same tag to identify multiple cloud resources, you can select the same tag from the **Tags** drop-down list box. It is recommended that you use TMS's predefined tag function to add the same tag to different cloud resources.

Figure 5-1 Basic information



Step 4 Configure containers.

Click **Use Image** under the image to be used to deploy the application.

- **My Images**: shows all the images you have created in **SWR**.
- **Shared Images**: displays images shared by other users. Shared images are managed and maintained in SWR. For details, see **Sharing Private Images**.

After selecting an image, specify container configurations.

- **Image Version**: Select the version of the image to be used to deploy the application.

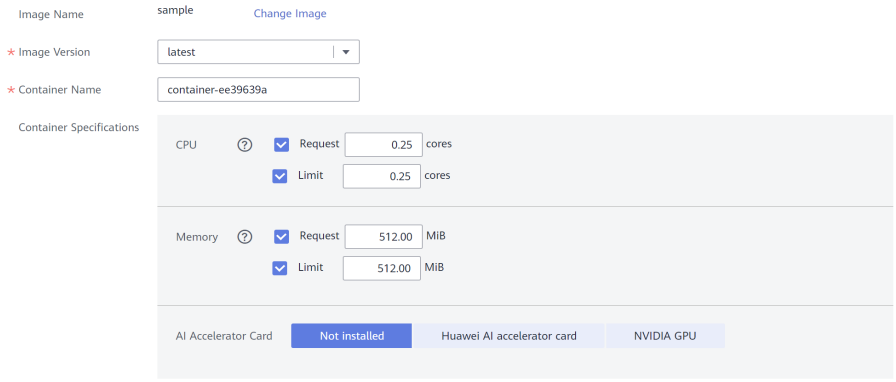
 Copyright © Huawei Technologies Co., Ltd.

NOTICE

Do not use version **latest** when deploying containers in the production environment. Because this will make it difficult to determine the version of the running image and to roll back the application properly.

- **Container Specifications**: Specify CPU, memory, and AI accelerator card quotas. Note that one Ascend processor can be allocated to only one containerized application.

Figure 5-2 Container configuration



You can also configure the following advanced settings for the container:

- **Command**

  A container image has metadata that stores image details. If lifecycle commands and arguments are not set, IEF runs the default commands and arguments provided during image creation, that is, **ENTRYPOINT** and **CMD** in the Dockerfile.

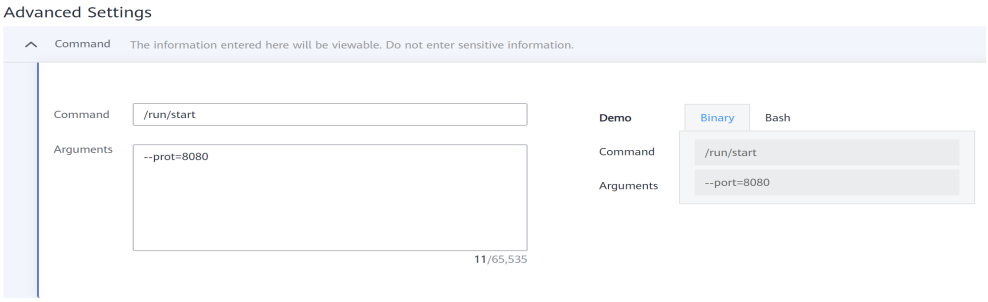
  If the commands and parameters used to run a container are set during application creation, the default commands **ENTRYPOINT** and **CMD** are overwritten during image building. **Table 5-1** lists the specific rules.

Table 5-1 Commands and parameters used to run a container

Image ENTRYPOINT	Image CMD	Command to Run a Container	Parameters to Run a Container	Command Executed
[touch]	[/root/test]	Not set	Not set	[touch /root/test]
[touch]	[/root/test]	[mkdir]	Not set	[mkdir]
[touch]	[/root/test]	Not set	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

 Copyright © Huawei Technologies Co., Ltd.

Figure 5-3 Startup command



  - **Command**

    Enter an executable command, for example, **/run/start**.

    If there are multiple commands, separate them with spaces. If the command contains a space, enclose the command in quotation marks ("").

    NOTE

    In the case of multiple commands, you are advised to run **/bin/sh** or other **shell** commands. Other commands are used as parameters.

  - **Arguments**

    Enter the argument that controls the container running command, for example, **--port=8080**.

    If there are multiple arguments, separate them with line breaks.

- **Additional Options**

  You can enable **Privileged Mode** to grant root permission to the container for accessing host devices (such as GPUs and FPGAs).

- **Environment Variables**

  An environment variable is a variable whose value can affect the way a running container will behave. Currently, environment variables can be manually entered, imported from secrets or ConfigMaps, or referenced from **hostIP**. For **Type**, you can select:

  - **Added manually**: Customize a variable name and value.
  - **Added from Secret**: You can customize a variable name. The variable value is referenced from secret configuration data. For details on how to create a secret, see **Secrets**.
  - **Added from ConfigMap**: You can customize a variable name. The variable value is referenced from ConfigMap configuration data. For details on how to create a ConfigMap, see **ConfigMaps**.
  - **Variable reference**: The variable value is referenced from **hostIP**, that is, the IP address of an edge node.

  NOTE

  IEF does not encrypt the environment variables you entered.

  If the environment variables you attempt to configure contain sensitive information, you need to encrypt them before entering them and also need to decrypt them through applications.

  IEF does not provide any encryption and decryption tools. If you need to configure cypher text, use other encryption and decryption tools.

 Copyright © Huawei Technologies Co., Ltd.

- **Data Storage**

  You can define a local volume and mount the local storage directory of the edge node to the container for persistent data storage.

  Currently, the following four types of local volumes are supported:

  - **hostPath**: used for mounting a host directory to the container. hostPath is a persistent volume. After an application is deleted, the data in hostPath still exists in the local disk directory of the edge node. If the application is re-created later, previously written data can still be read after the directory is mounted.

    You can mount the application log directory to the **var/IEF/app/log/{appName}** directory of the host. In the directory name, **{appName}** indicates the application name. The edge node will upload the .log and .trace files in the **/var/IEF/app/log/{appName}** directory to AOM.

    Figure 5-4 Log volume mounting

  - **emptyDir**: a simple empty directory used for storing transient data. It can be created in hard disks or memory. The application can read files from and write files into the directory. emptyDir has the same lifecycle as the application. If the application is deleted, the data in emptyDir is deleted along with it.
  - **configMap**: a type of resources that store configuration details required by workloads. For details on how to create a ConfigMap, see **ConfigMaps**.
  - **secret**: a type of resources that store sensitive data, such as authentication and key details. For details on how to create a secret, see **Secrets**.

  NOTE

  - The container path cannot be a system directory, such as **/** or **/var/run**. Otherwise, an exception occurs. You are advised to mount an empty directory to the container. If the directory is not empty, ensure that the directory does not contain any files that affect container startup. Otherwise, the files will be replaced, making it impossible for the container to be properly started. As a result, the application creation will fail.
  - When an NPU-equipped device such as Atlas 500 is used, do not mount the **/var/dlog** directory to the container because this directory has been mounted to the NPU-Plugin container of Atlas.
  - If the container is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.

- **Health Check**

  Health check regularly checks the status of containers or workloads.

  - **Liveness Probe**: The system executes the probe to check if a container is still alive, and restarts the instance if the probe fails. Currently, the system probes a container by HTTP request or command and determines whether the container is alive based on the response from the container.

 Copyright © Huawei Technologies Co., Ltd.

- **Readiness Probe**: The system invokes the probe to determine whether the instance is ready. If the instance is not ready, the system does not forward requests to the instance.

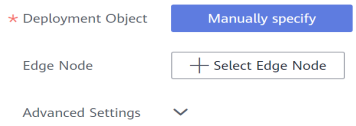
For details, see **Health Check Configuration**.

Step 5 Click **Next**.

Select a deployment object. Currently, two deployment objects are supported.

- **Manually specify**

  Figure 5-5 Specifying an edge node

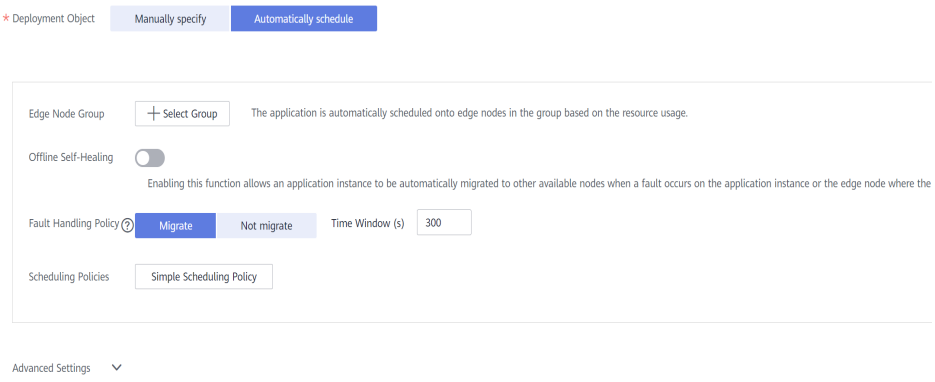
  

- **Automatically schedule** (supported only by platinum service instances)

  The containerized application will be automatically scheduled in the edge node group based on the resource usage of the nodes. You can also set a restart policy to determine whether the application instance is re-scheduled onto other available nodes in the edge node group when the edge node where the application instance originally runs is unavailable.

  You can also configure self-healing and scheduling policies. For details, see **Offline Self-Healing** and **Affinity and Anti-Affinity Scheduling**.

  Figure 5-6 Automatic scheduling

- **Restart Policy**
  - **Always restart**: The system restarts the container regardless of whether it had quit normally or unexpectedly.
  - **Restart upon failure**: The system restarts the container only if it had previously quit unexpectedly.
  - **Do not restart**: The system does not restart the container regardless of whether it had quit normally or unexpectedly.

 Copyright © Huawei Technologies Co., Ltd.

NOTICE

You are allowed to upgrade a containerized application and modify its instance quantity and access configuration only after you select **Always restart**.

- **Host PID**

  Enabling this function allows containers to share the PID namespace with the host where the edge node resides. In this way, you can perform container operations on the edge node, for example, starting and stopping a container. Similarly, you can perform operations related to the edge node in containers, for example, starting and stopping a process of the edge node.

  This function can only be enabled on edge nodes running software v2.8.0 or later.

Step 6 Click **Next**.

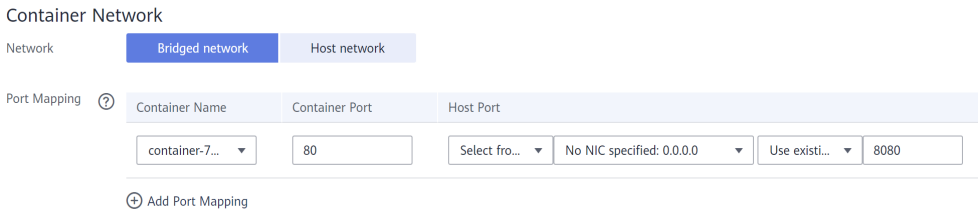
Containers can be accessed through a bridged network or a host network.

- **Bridged network**

  The container uses an independent virtual network. Configure relationships between container and host ports to enable external communications. After the ports are configured, traffic destined for a host port is directed to a specified container port. For example, if container port 80 is mapped to host port 8080, the traffic destined for host port 8080 will be directed to container port 80.

  You can select a host NIC to enable the port bound to this NIC to communicate with the container port. Note that port mapping does not support NICs with IPv6 addresses.

Figure 5-7 Bridged network



- **Host network**

  The network of the host (edge node) is used. To be specific, the container and the host use the same IP address, and network isolation is not required between them.

- **Service**

  A Service defines more refined access mode for applications. You can create a Service after an application is created. For details, see **Creating a Service**.

Step 7 Click **Next**. Confirm the specifications of the containerized application and click **Create**.

----End

 Copyright © Huawei Technologies Co., Ltd.

Querying Application Information

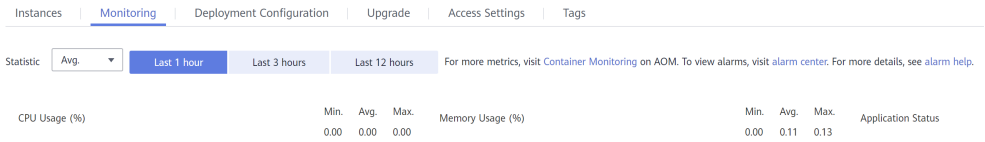
After an application is deployed, you can view the CPU and memory information of the application on the IEF console. You can also go to the **Container Monitoring** page on AOM to view additional metrics and add or view alarms.

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Containerized Applications**. Then, click an application name.

Step 3 Click the **Monitoring** tab to view the monitoring details for the application.

Figure 5-8 Application monitoring details



----End

Upgrading an Application

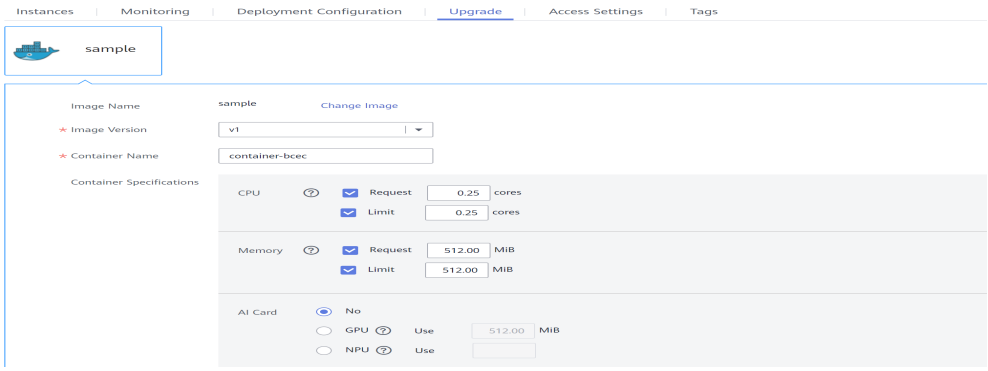
After an application is deployed, you can upgrade it. Rolling upgrades are used. That is, a new application instance is created first, and then the old application instance is deleted.

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Containerized Applications**. Then, click an application name.

Step 3 Click the **Upgrade** tab, and modify the container configuration. The parameter settings are the same as those in **Step 4**.

Figure 5-9 Upgrading an application



Step 4 Click **Submit**.

----End

 Copyright © Huawei Technologies Co., Ltd.

Modifying Access Settings

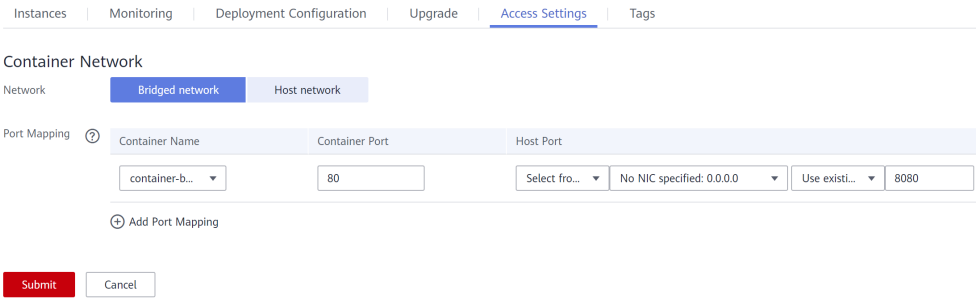
After an application is deployed, you can modify the access settings of the application.

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Containerized Applications**. Then, click an application name.

Step 3 Click the **Access Settings** tab, and modify the configuration. The parameter settings are the same as those in **Step 6**.

Figure 5-10 Modifying access settings



Step 4 Click **Submit**.

----End

5.2 Application Templates

An application template defines the details required by containerized applications, such as the container image, configuration, disk mounting information, and resource usage.

Applications are created from an image. You need to create and upload an image to the image repository in advance.

Creating an Application Template

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Application Templates**. Then, click **Create Application Template** in the upper right corner.

Step 3 Specify basic information.

- **Name** (mandatory): name of an application template.
- **Version** (mandatory): version number of the application template.
- **Alias**: alias of the application template.
- **Architecture**: architecture supported by the application.
- **Description**: description of the application template.
- **Tags**: Tags can be used to classify application templates, facilitating template management. The tags specified in this step are only used to identify

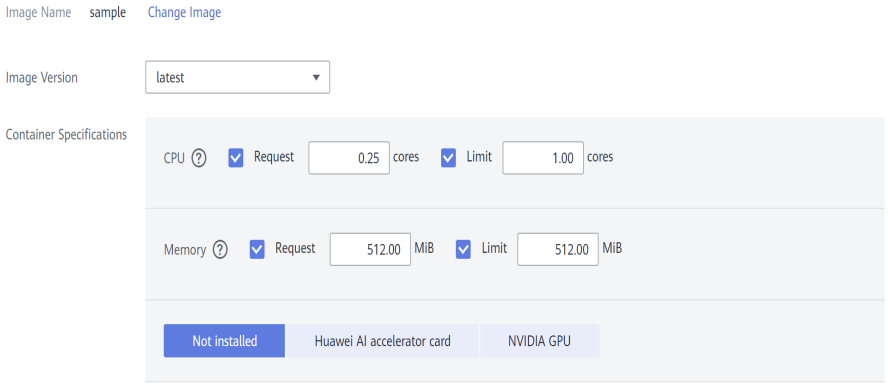
 Copyright © Huawei Technologies Co., Ltd.

application templates. You can use these tags to filter the application templates during search.

Step 4 Click **Next**. Then, add a container.

1. Click **Use Image** under the target image.
   - **My Images**: shows all the images you have uploaded to the image repository.
   - **Shared Images**: shows the images shared by other users.
2. Click **Next**. Then, specify container specifications.

Figure 5-11 Image information



   - **Image Version**: Select the version of the image to be used to deploy the application.
   - **Container Specifications**: Specify CPU, memory, Huawei AI accelerator card, and GPU quotas.

Step 5 Click **Next**. Then, configure the application.

- **Startup Commands**

  A container image has metadata that stores image details. If lifecycle commands and arguments are not set, IEF runs the default commands and arguments provided during image creation, that is, **ENTRYPOINT** and **CMD** in the Dockerfile.

  If the commands and parameters used to run a container are set during application creation, the default commands **ENTRYPOINT** and **CMD** are overwritten during image building. **Table 5-2** lists the specific rules.

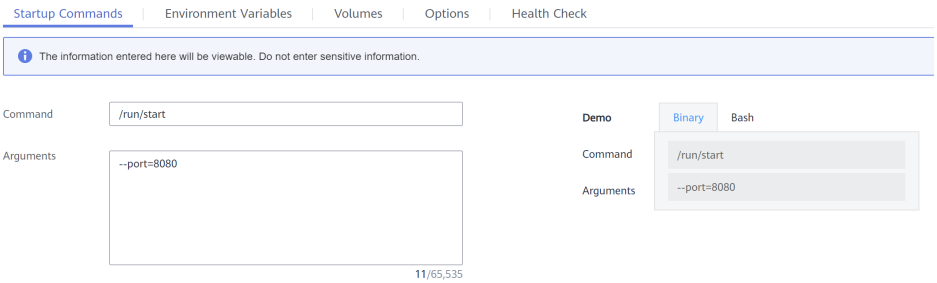
Table 5-2 Commands and parameters used to run a container

Image ENTRYPOINT	Image CMD	Command to Run a Container	Parameters to Run a Container	Command Executed
[touch]	[/root/test]	Not set	Not set	[touch /root/test]
[touch]	[/root/test]	[mkdir]	Not set	[mkdir]

 Copyright © Huawei Technologies Co., Ltd.

Image ENTRYPOINT	Image CMD	Command to Run a Container	Parameters to Run a Container	Command Executed
[touch]	[/root/test]	Not set	[/opt/test]	[touch /opt/test]
[touch]	[/root/test]	[mkdir]	[/opt/test]	[mkdir /opt/test]

Figure 5-12 Startup command



- **Command**

  Enter an executable command, for example, **/run/start**.

  If there are multiple commands, separate them with spaces. If the command contains a space, enclose the command in quotation marks ("").

  In the case of multiple commands, you are advised to run **/bin/sh** or other **shell** commands. Other commands are used as parameters.

- **Arguments**

  Enter the argument that controls the container running command, for example, **--port=8080**.

  If there are multiple arguments, separate them with line breaks.

- **Environment Variables**

  An environment variable is a variable whose value can affect the way a running container will behave. You can modify environment variables even after applications are deployed.

  Click **Add Environment Variable** to add a variable. Currently, environment variables can be manually entered, imported from secrets or ConfigMaps, or referenced from **hostIP**. For **Type**, you can select:

  - **Added manually**: Customize a variable name and value.
  - **Added from Secret**: You can customize a variable name. The variable value is referenced from secret configuration data. For details on how to create a secret, see **Secrets**.
  - **Added from ConfigMap**: You can customize a variable name. The variable value is referenced from ConfigMap configuration data. For details on how to create a ConfigMap, see **ConfigMaps**.

 Copyright © Huawei Technologies Co., Ltd.

- **Variable reference**: The variable value is referenced from **hostIP**, that is, the IP address of an edge node.

  NOTE

  IEF does not encrypt the environment variables you entered.

  If the environment variables you attempt to configure contain sensitive information, you need to encrypt them before entering them and also need to decrypt them through applications.

  IEF does not provide any encryption and decryption tools. If you need to configure cypher text, use other encryption and decryption tools.

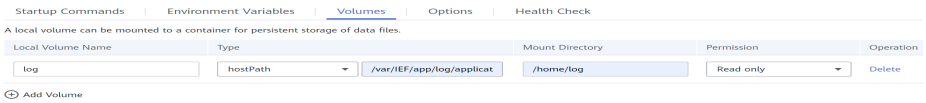
- **Volumes**

  A volume is used for storage when the container is running. Currently, the following four volume types are supported:

  - **hostPath**: used for mounting a host directory to the container. hostPath is a persistent volume. After an application is deleted, the data in hostPath still exists in the local disk directory of the edge node. If the application is re-created later, previously written data can still be read after the directory is mounted.

    You can mount the application log directory to the **var/IEF/app/log/{appName}** directory of the host. In the directory name, **{appName}** indicates the application name. The edge node will upload the .log and .trace files in the **/var/IEF/app/log/{appName}** directory to AOM.

    Figure 5-13 Log volume mounting

  - **emptyDir**: a simple empty directory used for storing transient data. It can be created in hard disks or memory. emptyDir is an empty directory after being mounted. The application can read files from and write files into the directory. emptyDir has the same lifecycle as the application. If the application is deleted, the data in emptyDir is deleted along with it.
  - **configMap**: a type of resources that store configuration details required by workloads. For details on how to create a ConfigMap, see **ConfigMaps**.
  - **secret**: a type of resources that store sensitive data, such as authentication and key details. For details on how to create a secret, see **Secrets**.

 Copyright © Huawei Technologies Co., Ltd.

NOTE

- The container path cannot be a system directory, such as **/** or **/var/run**. Otherwise, an exception occurs. You are advised to mount the container to an empty directory. If the directory is not empty, ensure that the directory does not contain any files that affect container startup. Otherwise, the files will be replaced, making it impossible for the container to be properly started. As a result, the application creation will fail.
- When an NPU-equipped device such as Atlas 500 is used, do not mount the **/var/dlog** directory to the container because this directory has been mounted to the NPU-Plugin container of Atlas.
- If the container is mounted to a high-risk directory, you are advised to use an account with minimum permissions to start the container. Otherwise, high-risk files on the host may be damaged.

- **Options**
  - **Privileged Mode**

    You can enable **Privileged Mode** to grant root permission to the container for accessing host devices (such as GPUs and FPGAs).
  - **Restart Policy**

    Always restart: The system restarts the container regardless of whether it had quit normally or unexpectedly.

    Restart upon failure: The system restarts the container only if it had previously quit unexpectedly.

    Do not restart: The system does not restart the container regardless of whether it had quit normally or unexpectedly.
  - **Container Network**

    Host network: The network of the host (edge node) is used. To be specific, the container and the host use the same IP address, and network isolation is not required between them.

    Bridged network: The container uses an independent virtual network. Configure relationships between container and host ports to enable external communications. After the ports are configured, traffic destined for a host port is directed to a specified container port. For example, if container port 80 is mapped to host port 8080, the traffic destined for host port 8080 will be directed to container port 80.
- **Health Check**
  - **Liveness Probe**: The system executes the probe to check if a container is still alive, and restarts the instance if the probe fails. Currently, the system probes a container by HTTP request or command and determines whether the container is alive based on the response from the container.
  - **Readiness Probe**: The system invokes the probe to determine whether the instance is ready. If the instance is not ready, the system does not forward requests to the instance.

  For details, see **Health Check Configuration**.

Step 6 After configuration, click **Create**.

----End

 Copyright © Huawei Technologies Co., Ltd.

Creating an Application Template Version

You can create multiple application versions for an application template to facilitate application management.

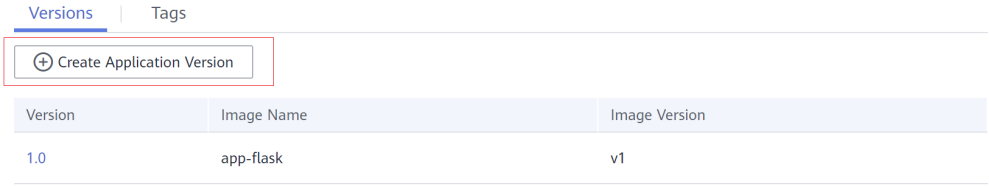
Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Application Templates**.

Step 3 Click the application template whose version is to be added. The application template details page is displayed.

Step 4 Click **Create Application Version** on the **Versions** page.

Figure 5-14 Creating an application version



Step 5 Enter a version, and click **Next**. Then, configure other parameters by following the description in **Creating an Application Template**.

----End

Sharing Templates

Application templates in platinum service instances can be shared with other platinum service instances.

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Application Templates**.

Step 3 On the **Private Templates** page, click the application template to be shared. The application template details page is displayed.

Step 4 Click the **Service Instances** tab, click **Add Service Instance**, and select service instances.

Step 5 Click **OK**.

----End

5.3 Offline Self-Healing

NOTE

Only platinum service instances support offline self-healing.

Offline self-healing is only possible for applications that are scheduled to node groups.

Introduction

Offline self-healing means that if a fault occurs on an application instance, or if the edge node where the application instance is originally deployed becomes

 Copyright © Huawei Technologies Co., Ltd.

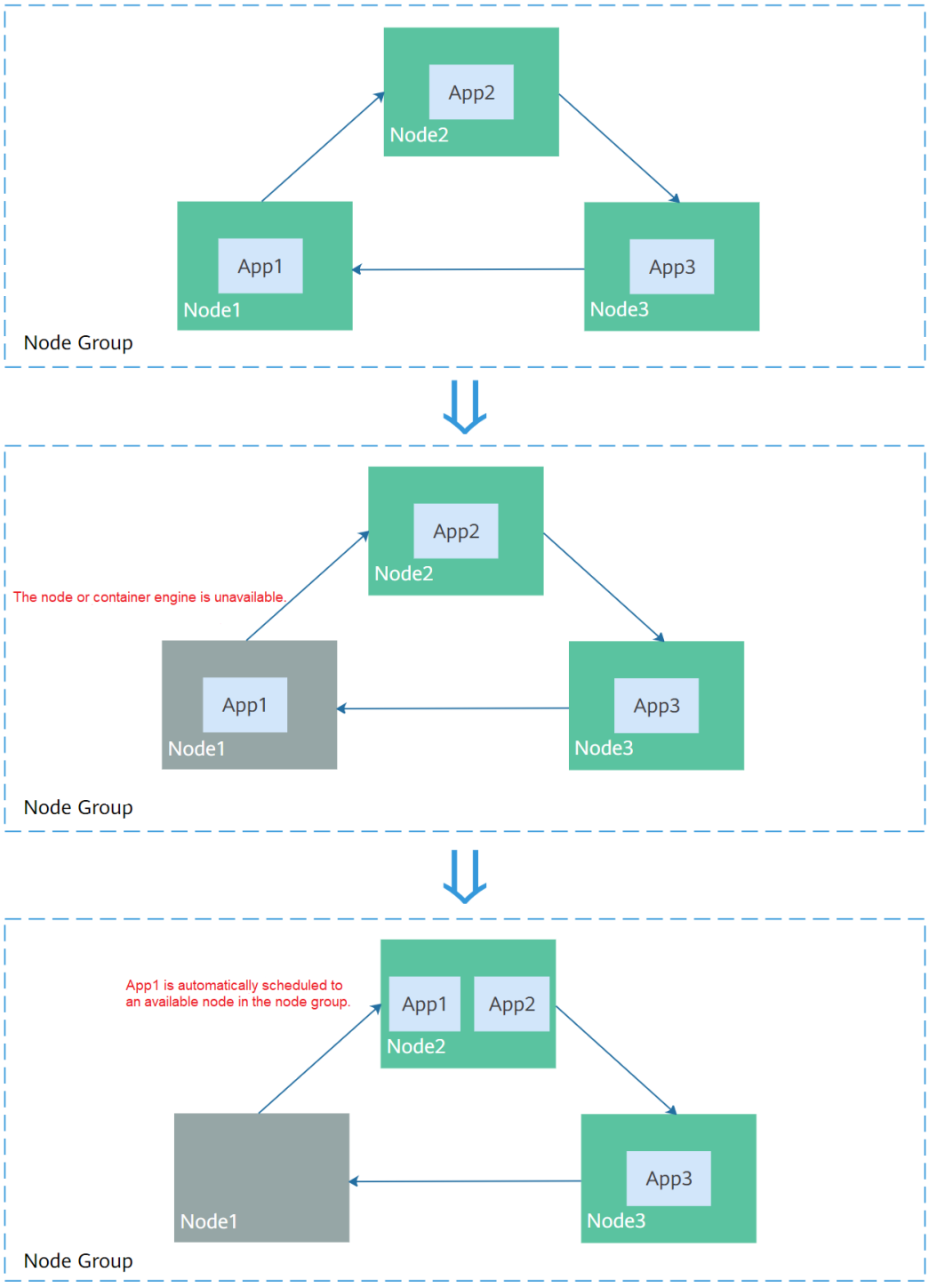
faulty, the application instance can be scheduled to other available nodes in the node group. The application instance can still be automatically scheduled even when the edge node is disconnected from IEF.

If offline self-healing is enabled, the nodes in the node group start loop monitoring, that is, the next-hop node monitors the status of its previous node and applications deployed on it. If a node an application instance is running on becomes faulty, the next-hop node monitoring the node in the loop detects the fault and checks whether it meets requirements to run the instance. If it does, it automatically starts a same application instance. If it does, a scheduling request is sent to the next-hop node in the loop. If no nodes meet requirements, this application instance is considered faulty.

In **Figure 5-15**, if Node1 running App1 fails, Node2 (which monitors Node1) checks whether it meets requirements for running App1. If it does, it starts App1.

 Copyright © Huawei Technologies Co., Ltd.

Figure 5-15 Offline self-healing



Differences Between Offline Self-Healing and Application Migration Upon Node Group Failure

With node groups, if a node fails, applications can be migrated to different nodes. If one node in a node group becomes faulty, the applications on it can be automatically scheduled to other normal nodes in the group. Application migration is an online scheduling function. However, if all of the nodes in a node

 Copyright © Huawei Technologies Co., Ltd.

group go offline, application migration is not possible because there are no nodes available. Offline self-healing, however, can still work properly.

Application migration is triggered immediately anytime any node in a node group is disconnected from IEF. In this case, the node may be running properly, which would make it unnecessary to schedule the instances on the node to other nodes. Offline self-healing can avoid such unnecessary scheduling as it can identify the running status of nodes even when they go offline.

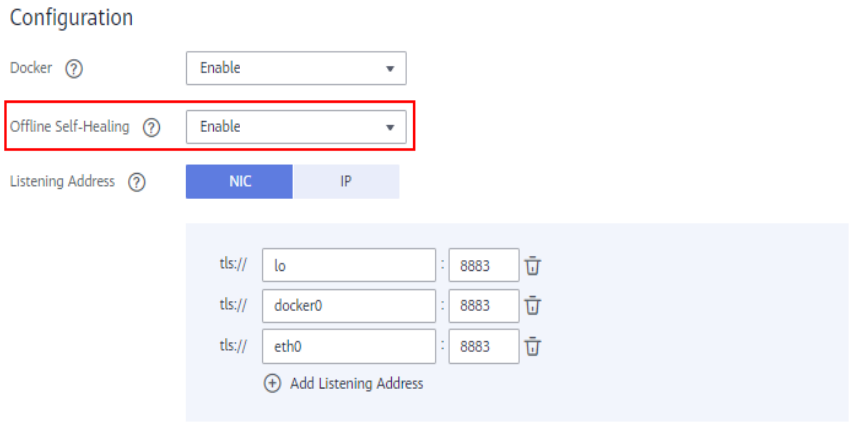
NOTE

If offline self-healing is enabled, application migration upon node group failure is automatically disabled.

Constraints

- A node group must contain at least two edge nodes.
- The offline self-healing function must be enabled for all nodes in a node group, and port 8883 of each node must be accessible.
- An application with offline self-healing enabled cannot detect the addition or deletion of nodes in the node group where it is deployed.
- If you need to deploy applications with offline self-healing enabled on the node, enable the offline self-healing option for a node during registration, as shown in **Figure 5-16**. For details, see **Registering an Edge Node**.

Figure 5-16 Offline self-healing option

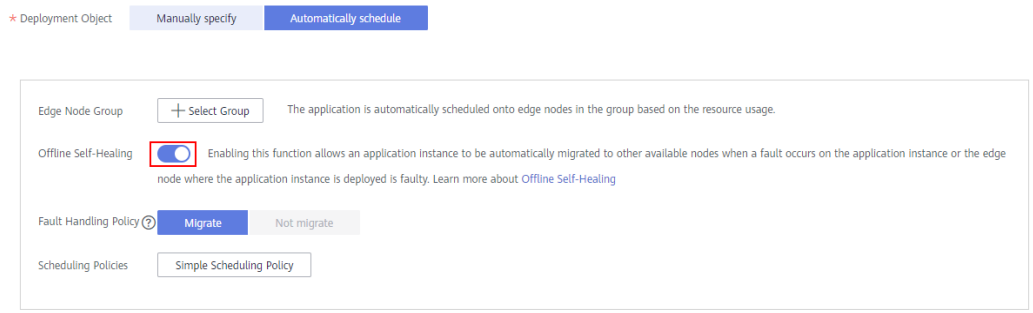


Configuration

Enable the offline self-healing function when **creating an edge application**.

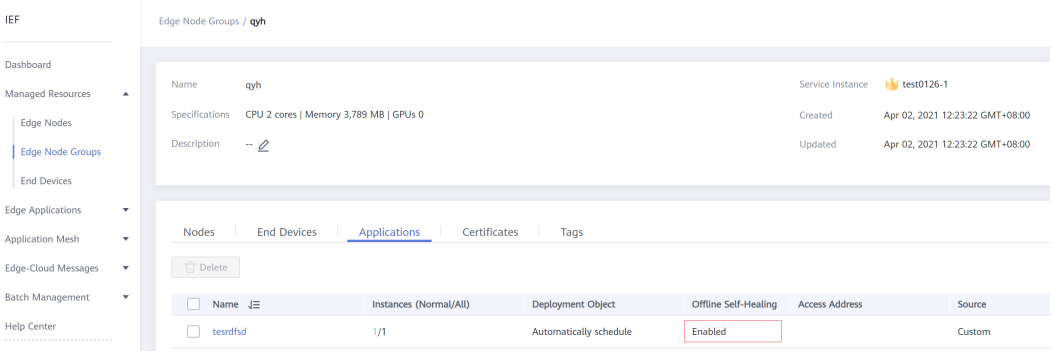
Copyright © Huawei Technologies Co., Ltd.

Figure 5-17 Enabling offline self-healing



After a containerized application is deployed, you can check whether offline self-healing is enabled for the application on the **Applications** page on the details page of the edge node group, as shown in **Figure 5-18**.

Figure 5-18 Checking whether offline self-healing is enabled



5.4 Affinity and Anti-Affinity Scheduling

NOTE

Only platinum service instances support affinity and anti-affinity scheduling.

Affinity and Anti-Affinity Scheduling Policies

When creating a containerized application, you can set affinity and anti-affinity scheduling policies. For example, you can deploy applications of a certain type on specific nodes or deploy different applications on different nodes.

IEF supports the following simple scheduling policies:

- Affinity
  - Application-node affinity: If the affinity is defined between a containerized application and certain nodes, the application will be scheduled only to these nodes. If you have set the affinity between an application and certain nodes, you do not need to set the anti-affinity between the application and other nodes because the affinity determines that the application can be scheduled only to the specified nodes.

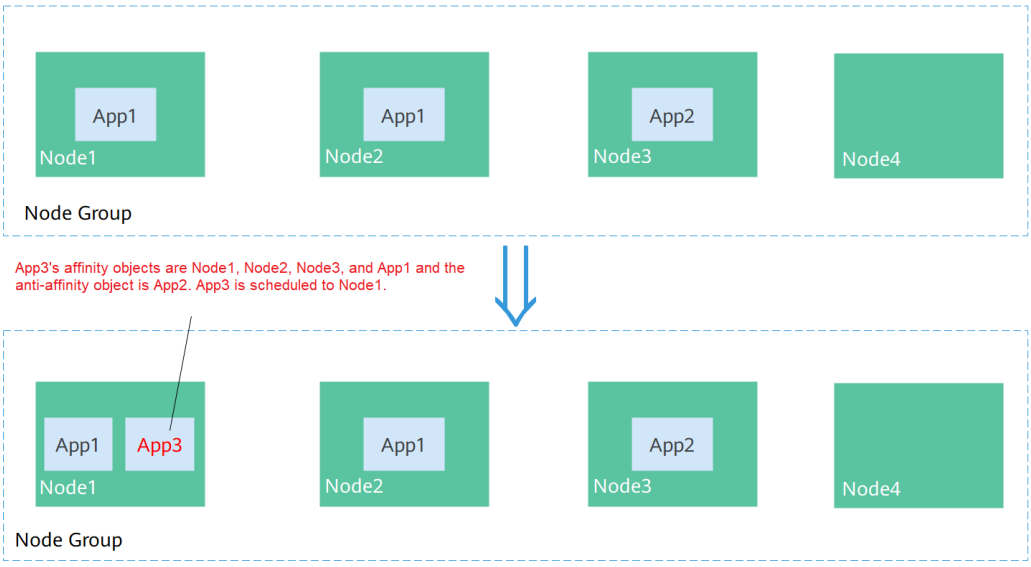
 Copyright © Huawei Technologies Co., Ltd.

  - Affinity between containerized applications: If the affinity is defined between containerized applications A and B, application A will be scheduled only to the nodes where application B is located.
- Anti-Affinity
  - Application-node anti-affinity: If the anti-affinity is defined between a containerized application and certain nodes, the application will not be scheduled to these nodes.
  - Anti-affinity between containerized applications: If the anti-affinity is defined between containerized applications A and B, application A will not be scheduled to the nodes where application B is located.

Affinity and Anti-Affinity Policy Example

You can configure multiple affinity and anti-affinity objects for an application. For example, a node group contains four nodes: Node1, Node2, Node3, and Node4. Instances of containerized application App1 run on Node1 and Node2, and instances of containerized application App2 run on Node3. Assume that you create containerized application App3, for which the affinity objects are set to Node1, Node2, Node3, and App1 and the anti-affinity object is set to App2. In this case, App3 will be scheduled only to Node1 and Node2. Since App3 contains only one instance, it is scheduled to Node1, as shown in **Figure 5-19**.

Figure 5-19 App3 scheduled to Node1



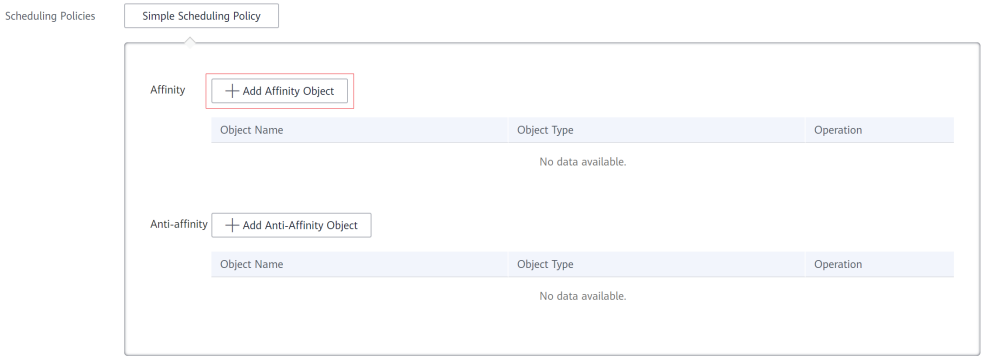
Adding an Affinity Object

Once you have selected an edge node group in the **Associate Object** step during creation of a containerized application, you can configure an affinity policy.

Step 1 Click **Add Affinity Object**.

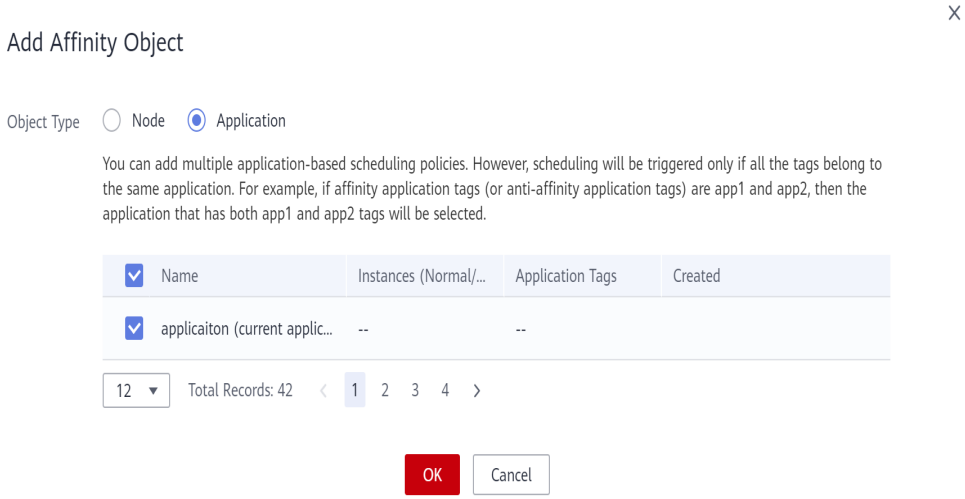
 Copyright © Huawei Technologies Co., Ltd.

Figure 5-20 Adding an affinity object



Step 2 In the dialog box that is displayed, select one or more affinity nodes or applications and click **OK**.

Figure 5-21 Selecting an affinity object



----End

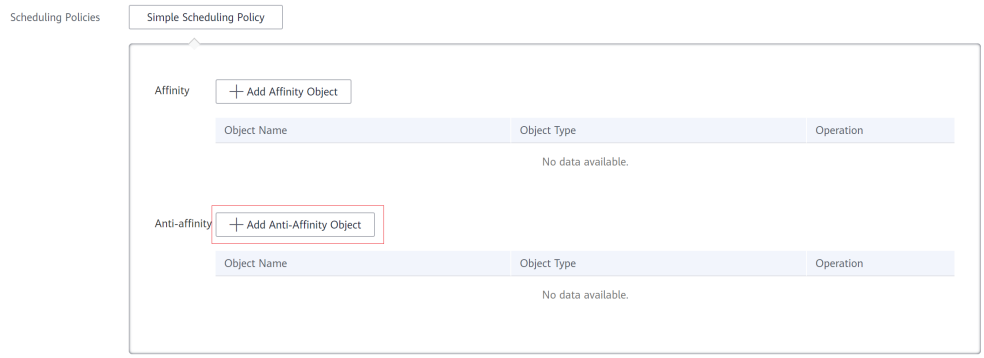
Adding an Anti-affinity Object

You can configure an anti-affinity policy after you select an edge node group in the **Associate Object** step during creation of a containerized application.

Step 1 Click **Add Anti-Affinity Object**.

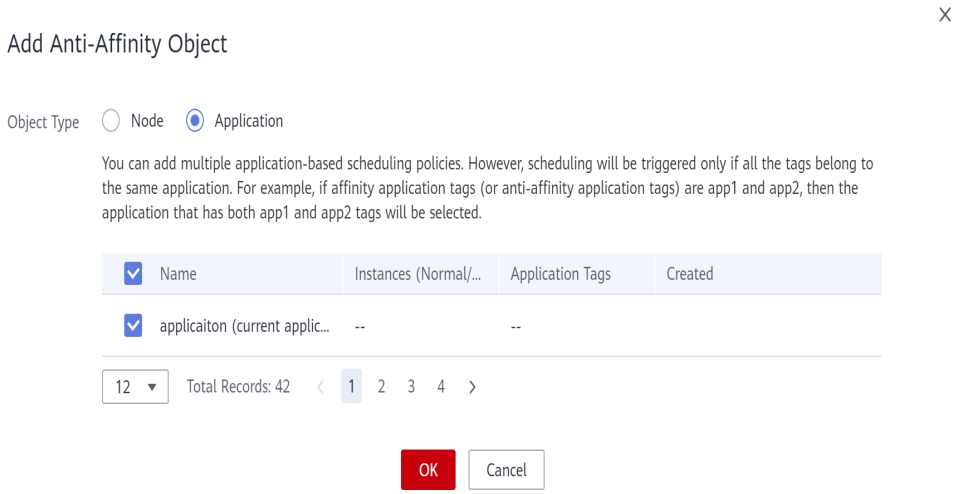
 Copyright © Huawei Technologies Co., Ltd.

Figure 5-22 Adding an anti-affinity object



Step 2 In the dialog box that is displayed, select one or more anti-affinity nodes or applications and click **OK**.

Figure 5-23 Selecting an anti-affinity object



----End

5.5 ConfigMaps

ConfigMaps store configuration details required for workloads. ConfigMaps decouple configuration files from container images to enhance workload portability.

ConfigMaps allow you to:

- Manage configurations for different environments and services.
- Deploy workloads in different environments. Multiple versions are supported for configuration files so that you can update and roll back workloads easily.
- Quickly import configurations in the form of files to containers.

 Copyright © Huawei Technologies Co., Ltd.

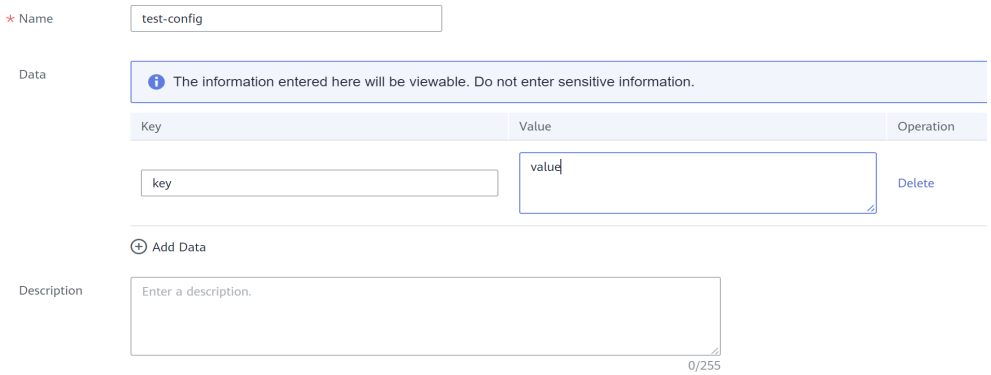
Creating a ConfigMap

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Application Configuration**. Then, click **Create ConfigMap** in the upper right corner.

Step 3 Specify the ConfigMap details.

Figure 5-24 Creating a ConfigMap



- **Name**: name of the ConfigMap.
- **Data**: The configuration data is key-value pairs. Enter a property name and value.

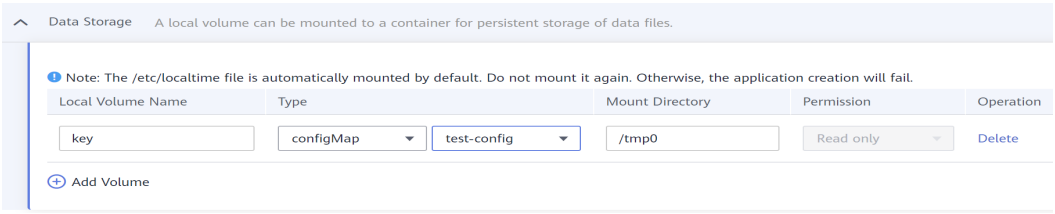
Step 4 Click **Create**. After the ConfigMap is created, the ConfigMap list page is displayed.

----End

Using a ConfigMap

You can use ConfigMaps when configuring volumes during application creation.

Figure 5-25 Using a ConfigMap



After the ConfigMap is mounted to the container, files are created in the mount directory based on the ConfigMap content. Each property name and value pair is generated as a file. The property name is the file name, and the value is the content of the file. For instance, a ConfigMap could have the property name set to **key** and the property value set to **value**. After the ConfigMap is mounted to the **/tmp0** directory, a file named **key** is generated in the **/tmp0** directory and the file content is **value**.

 Copyright © Huawei Technologies Co., Ltd.

5.6 Secrets

Secrets store sensitive, user-defined information such as authentication details or keys, and can then be loaded to containerized applications. For example, a secret can be loaded into a container as a file during volume configuration.

NOTICE

Secrets may involve sensitive user information. If the secrets you attempt to configure contain sensitive information, you need to encrypt them before entering them and also need to decrypt them through applications.

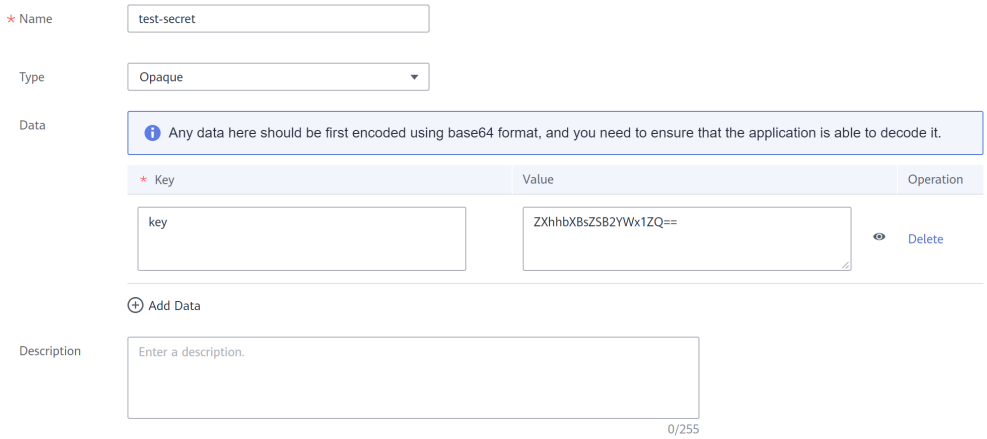
Creating a Secret

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Application Configuration**. Then, click **Create Secret** in the upper right corner.

Step 3 Specify the secret details.

Figure 5-26 Creating a secret



- **Name**: name of the secret.
- **Type**: **Opaque** secrets are supported. The data of the **Opaque** type is key-value pairs. The values are encoded in Base64. For details, see **Base64 Encoding**.
- **Data**: They are key-value pairs. Enter a property name and value. The value must be a Base64-encoded string.

Step 4 Click **Create**. After the secret is created, the secret list page is displayed.

----End

 Copyright © Huawei Technologies Co., Ltd.

Base64 Encoding

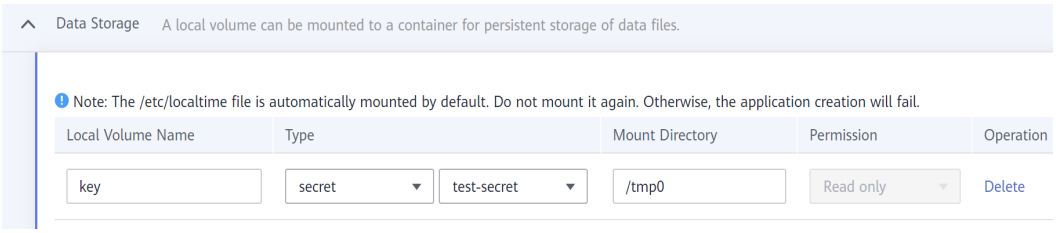
To perform Base64 encoding on a character string, run the **echo -n** *{Content to be encoded}* **| base64** command. For example:

```
root@ubuntu:~# echo -n "example value" | base64
ZXhhbXBsZSB2YWx1ZQ==
```

Using a Secret

You can use secrets when configuring volumes during application creation.

Figure 5-27 Using a secret



After the secret is mounted to the container, files are created in the mount directory based on the secret content. Each property name and value pair is generated as a file. The property name is the file name, and the value is the content of the file. For instance, a secret could have the property name set to **key** and the property value set to **ZXhhbXBsZSB2YWx1ZQ==**. After the secret is mounted to the **/tmp2** directory, a file named **key** is generated in the **/tmp2** directory and the file content is **example value**.

5.7 Encryption Data

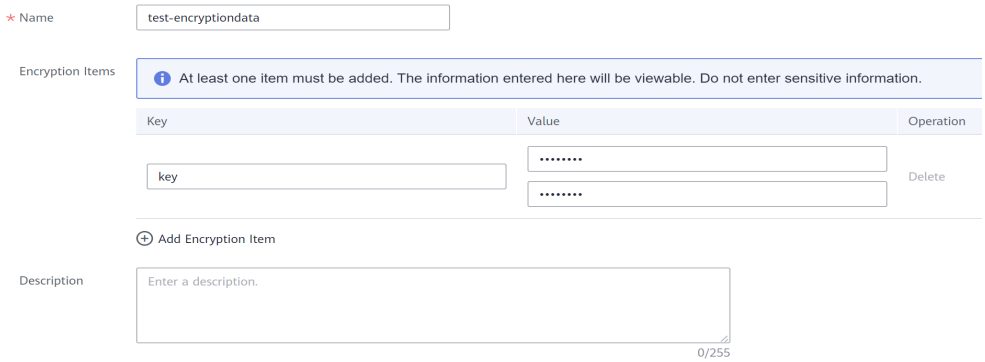
Encryption data is used to store and encrypt sensitive information. Edge applications can gain access to plaintext data through MQTT Broker.

Creating Encryption Data

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Application Configuration**. Then, click **Create Encryption Data** in the upper right corner. On the page that is displayed, specify the parameters.

Figure 5-28 Creating encryption data



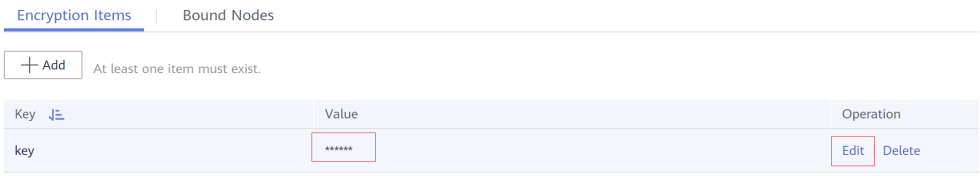
 Copyright © Huawei Technologies Co., Ltd.

The encryption data is stored in key-value pairs. The value needs to be entered twice. Multiple encryption items can be added to each encryption data record.

Step 3 Click **Create**.

You can view encryption data after creating it. However, for data privacy protection, data cannot be viewed in plaintext. You can also edit or delete encryption data.

Figure 5-29 Encryption data



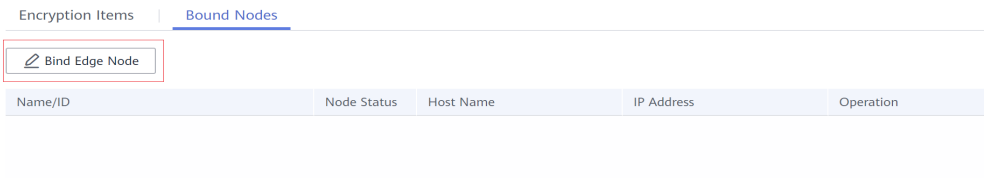
----End

Binding Encryption Data to Edge Nodes

Encryption data can be bound to edge nodes that are not in the running state. When an edge node is restored to **Running**, the bound data will be automatically synchronized to the node. You can bind encryption data to an edge node in either of the following ways:

- On the encryption data details page, click **Bind Node** and then **Bind Edge Node**.

  Figure 5-30 Binding an edge node

- On the edge node details page, click the **Configuration** tab. In the **Encryption Data** area, click **Bind**.

  Figure 5-31 Binding encryption data to an edge node

Using Encryption Data

After encryption data is bound to an edge node, you can obtain the data by using the MQTT client on the edge node.

 Copyright © Huawei Technologies Co., Ltd.

A certificate must be used for security authentication when encryption data is requested. For details about the authentication method, see **Performing Security Authentication Using Certificate**.

Step 1 Subscribe to the topic described in **Encryption Data Acquisition**.

Topic: $hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/plaintext

Step 2 Publish the topic described in **Encryption Data Request**.

Topic: $hw/{project_id}/encryptdatas/{encryptdata_name}/properties/{properties_name}/decrypt

After the request is published, the decrypted data is sent the topic subscribed to in **Step 1**.

----End

5.8 Health Check Configuration

Health check regularly checks the status of containers or workloads. There is a liveness probe and a readiness probe:

- **Liveness Probe**: The system checks if a container is still alive, and restarts the instance if the probe fails. Currently, the system probes a container by HTTP request or command and determines whether the container is alive based on the response from the container.
- **Readiness Probe**: The system determines whether an instance is ready. If the instance is not ready, the system does not forward requests to the instance.

IEF can perform a health check by HTTP request or through a CLI.

HTTP Request-based Check

IEF periodically initiates an HTTP GET request to a container. If HTTP code 2xx or 3xx is received, the container is healthy.

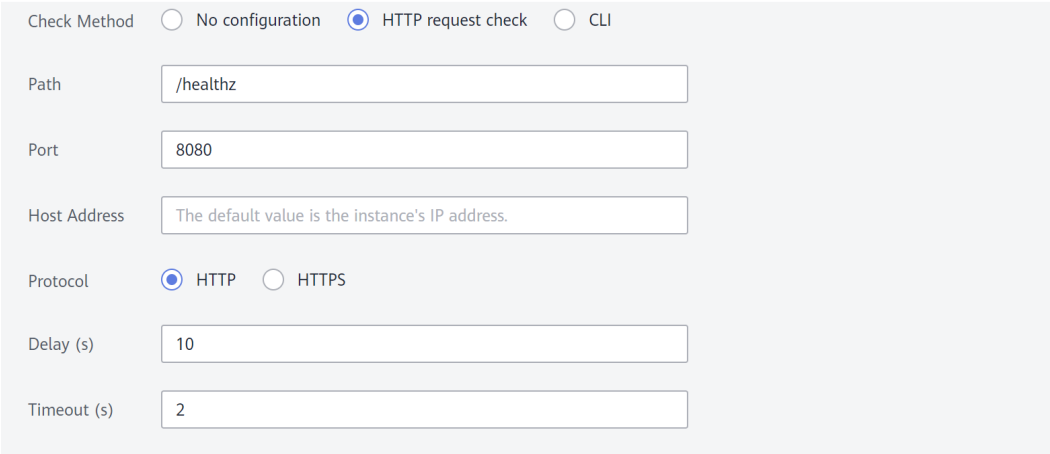
For example, if health check parameters are set as shown in the following figure, IEF will send an HTTP GET request to **http://{Instance's IP address}/healthz:8080** 10 seconds after the container starts. If no response is received within 2 seconds, the health check fails. If status code 2xx or 3xx is received, the container is healthy.

NOTE

You do not need to specify the host address. By default, the instance's IP address is used (that is, requests are sent to the container) unless you have special requirements.

 Copyright © Huawei Technologies Co., Ltd.

Figure 5-32 HTTP request-based check

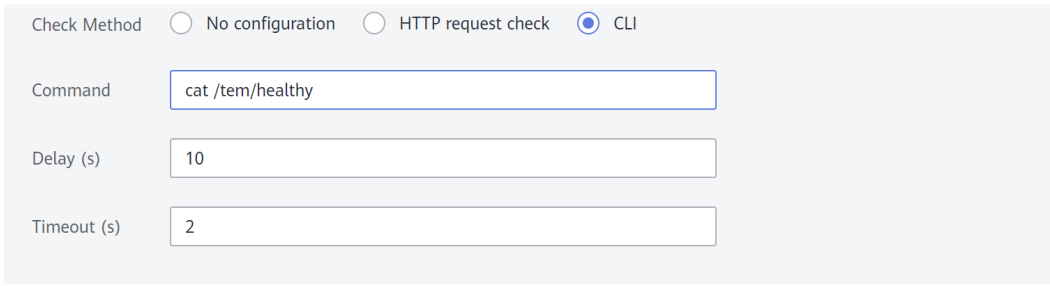


CLI-based Check

The probe runs commands in the container and checks the command output. If the command output is 0, the container is healthy.

For example, if health check parameters are set as shown in the following figure, IEF will run **cat /tmp/healthy** 10 seconds after the container starts. If no response is received within 2 seconds, the health check fails. If the command output is 0, the container is healthy.

Figure 5-33 CLI-based check



 Copyright © Huawei Technologies Co., Ltd.

6 Function Management

Scenario

FunctionGraph is a serverless function computing service on HUAWEI CLOUD. With IEF, you can quickly deploy functions created in FunctionGraph to edge nodes.

NOTE

Currently, IEF supports only Python functions.

Creating a Function

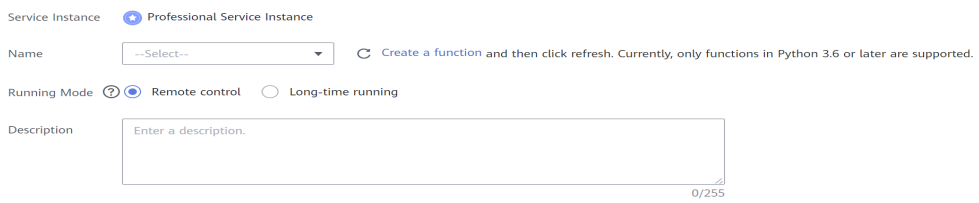
Before creating a function on IEF, ensure that the function has been created and a new version has been published on FunctionGraph. For details on how to create a function on FunctionGraph, see **Getting Started**. For details on how to publish a function version, see **Version Management**.

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Functions**. Then, click **Create Function** in the upper right corner.

Step 3 Specify the function details.

Figure 6-1 Creating a function



- **Name**: Select a function created in FunctionGraph. If there are no functions available, click **Create Function** on the right to create one.
- **Running Mode**: Select **Remote control** or **Long-term running**.
  - **Remote control**: The function can be enabled or disabled in the cloud and can be managed throughout its lifecycle.

 Copyright © Huawei Technologies Co., Ltd.

- **Long-time running**: The function will keep running on edge nodes.

Step 4 Click **Create**. The function is successfully created, and the function list is displayed.

----End

Creating a Dependency

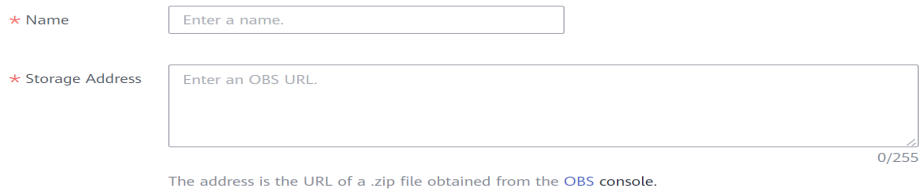
A dependency is a file that a function depends on to run. It can be bound to the function for dynamic updating. IEF supports function dependencies imported from OBS in .zip format. For details on how to use OBS, see **Basic Operation Procedure**.

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Functions**. Then, click **Create Dependency** in the upper right corner.

Step 3 Enter a dependency name and an OBS URL.

Figure 6-2 Creating a dependency



- **Name**: Enter a dependency name.
- **Storage Address**: Enter the URL of a ZIP dependency file.

Step 4 Click **Create**. After the dependency is created, the dependency list page is displayed.

----End

Configuring a Dependency

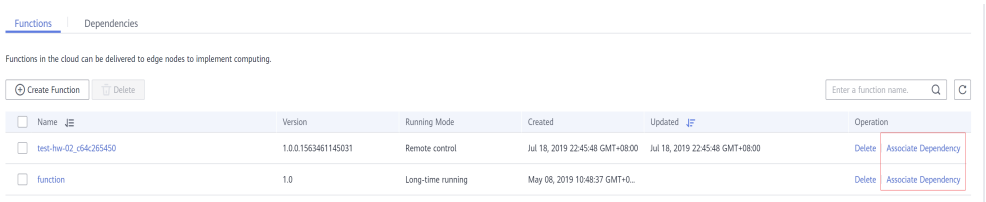
Dependencies need to be associated with edge functions so that they can be delivered together to edge nodes during application deployment.

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge Applications** > **Functions**.

Step 3 In the **Operation** column of the function for which you want to configure a dependency, click **Associate Dependency**.

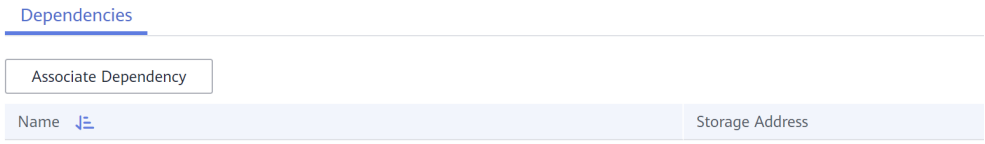
Figure 6-3 Associating with a dependency



 Copyright © Huawei Technologies Co., Ltd.

Step 4 On the **Associate Dependency** page, click **Add Dependency**.

Figure 6-4 Adding a dependency



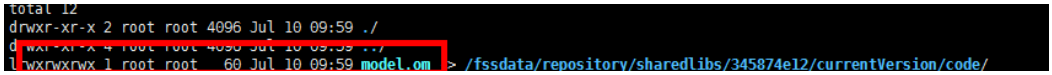
Step 5 On the **Select Dependency** page, select the dependency to be associated with the function, and click **OK**.

Step 6 Enter the alias of the dependency in **Alias**, and click **OK**.

All the associated dependencies are stored in the **depends** sub-directory of the function work directory on an edge node.



The specified alias is used as the sub-directory name in the **depends** sub-directory on the edge node. The dependency file is stored in the alias sub-directory.



----End

Configuring Functions for an Edge Node

Functions need to be associated with an edge node before they can be delivered to an edge node.

Step 1 Log in to the IEF console.

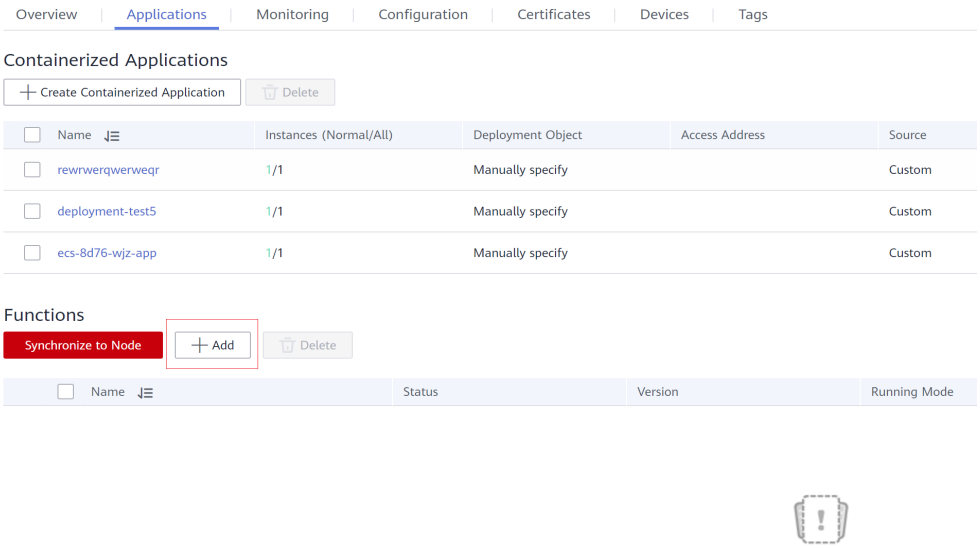
Step 2 In the navigation pane, choose **Edge Resources** > **Edge Nodes**.

Step 3 Click the edge node to be associated with functions.

Step 4 On the node details page, click the **Applications** tab. Then, click **Add** in the **Functions** area.

 Copyright © Huawei Technologies Co., Ltd.

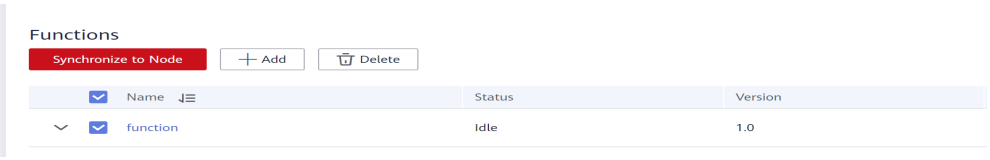
Figure 6-5 Function configuration



Step 5 Select a function name and version and click **OK**.

Step 6 Select the function to be deployed, and click **Synchronize to Node**. The function will be deployed to the edge node.

Figure 6-6 Deploying a function



After the deployment is successful, the functions change to the **Running** state regardless of their running mode. A function can be in any of the following states:

- **Idle**: The function is being deployed or failed to be deployed on the edge node.
- **Running**: The function has started and is running properly.
- **Disabled**: The function was successfully deployed but it has been stopped.
- **Startup error**: The function was successfully deployed but failed to start.

----End

Remotely Controlling a Function

IEF allows you to enable and disable functions whose **Running Mode** is **Remote control** in the cloud.

On the **Applications** tab page of the edge node details page, you can use **Stop** or **Enable** in the **Operation** column of a function to disable or enable the function.

 Copyright © Huawei Technologies Co., Ltd.

7 Application Mesh

NOTE

The application mesh function is available only for platinum service instances.

Scenario

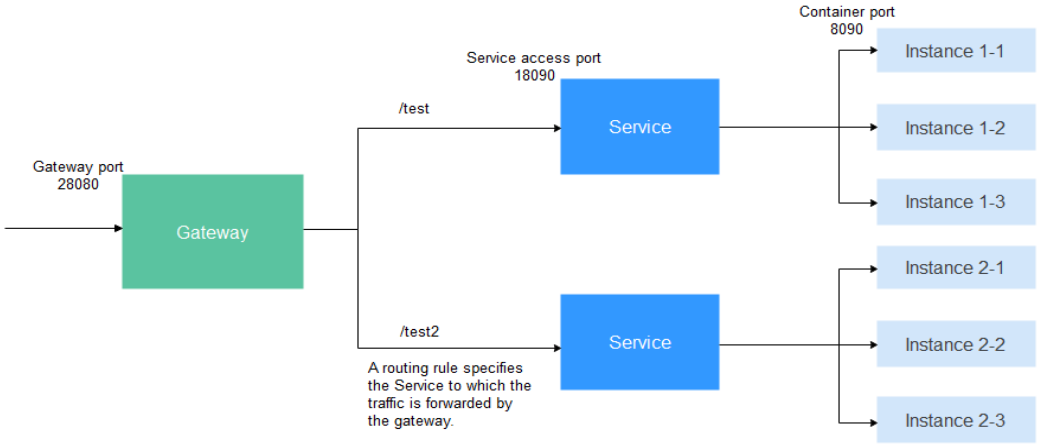
The application mesh is a non-intrusive microservice governance solution that supports full-lifecycle management and traffic management such as load balancing.

The application mesh involves Services and gateways:

- Services: A Service defines a set of instances and a means of accessing them. The Service name can be resolved to the corresponding IP address through which applications on edge nodes can access each other.
- Gateways: A gateway exposes edge applications deployed using IEF to external applications and forwards traffic through virtual services.

As shown in the following figure, if you access *{IP address of the edge node where the gateway application is located}:{Gateway port}/URI*, the traffic can be forwarded to the corresponding Service, which then forwards the traffic to the corresponding backend application instance.

Figure 7-1 Gateway traffic forwarding path



 Copyright © Huawei Technologies Co., Ltd.

Creating a Service

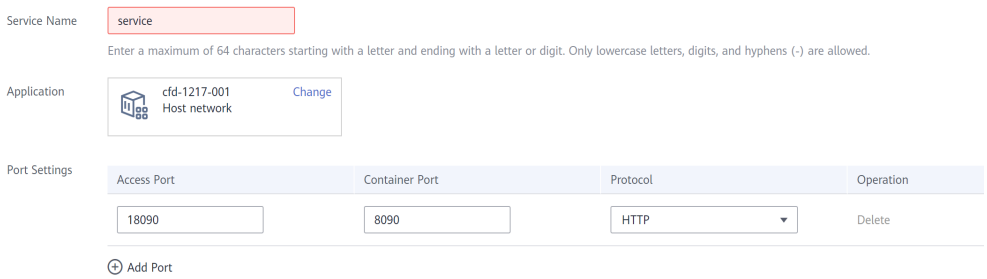
Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Application Mesh** > **Service List**. On the page displayed, click **Create Service** in the upper right corner.

Step 3 Set the parameters.

- **Service Name**: Enter a Service name.
- **Application**: Select the application that the Service is bound to.
- **Port Settings**:
  - **Access Port**: indicates the port used to access the containerized application.
  - **Container Port**: indicates the port on which the containerized application listens.

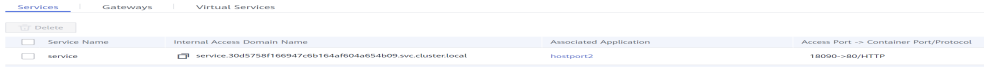
Figure 7-2 Creating a Service



Step 4 Click **Create**.

After the creation is complete, you can view the internal access domain name in the Service list.

Figure 7-3 Viewing the Service access domain name



----End

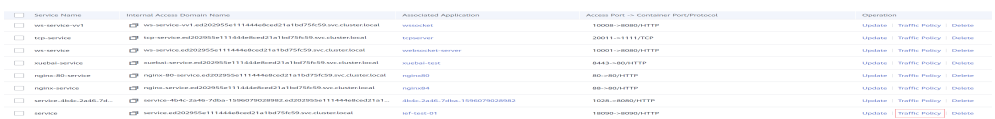
Configuring a Traffic Policy

A traffic policy defines how service traffic is forwarded to multiple application instances.

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Application Mesh** > **Service List**. Click **Traffic Policy** at the row where the Service resides.

Figure 7-4 Service list



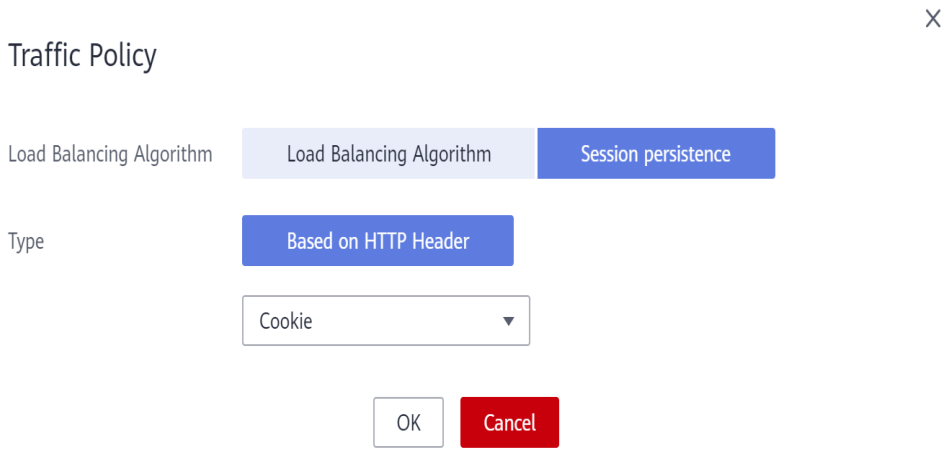
 Copyright © Huawei Technologies Co., Ltd.

Step 3 On the page displayed, select a traffic policy type.

Currently, the following traffic policy types are supported:

- **Load Balancing Algorithm**: Two forwarding modes, that is, **ROUND_ROBIN** and **RANDOM**, are supported.
- **Session persistence**: Three HTTP header-based forwarding modes, that is, **Cookie**, **User-Agent**, and **Custom**, are supported.

Figure 7-5 Selecting a traffic policy



Step 4 Click **OK**.

----End

Deploying an Edge Gateway Application

Edge gateway applications must be deployed on edge nodes. IEF has made an edge gateway image available on the edge marketplace. You can apply for the edge gateway image and deploy it to your edge nodes.

Step 1 Log in to the **Edge Application Center**, find the application named **Edge Gateway-x86_64**, and click **Apply for authorization**.

Step 2 After the authorization is complete, click **Deploy**.

Step 3 Select the edge node where the gateway is to be deployed and click **Next**. Then, configure the other parameters and click **Create**.

For details on how to deploy an application, see **Containerized Applications**.

An edge gateway application can listen on multiple IP addresses of an edge node. To enable an application to listen on multiple IP addresses, set the **NIC**, **INCLUDE_IP**, and **EXCLUDE_IP** environment variables during application deployment by following the description in **Table 7-1**. The edge gateway application filters IP addresses based on **INCLUDE_IP** and then **EXCLUDE_IP**. If **INCLUDE_IP** and **EXCLUDE_IP** contain the same IP address, the application will not listen on this IP address.

 Copyright © Huawei Technologies Co., Ltd.

Table 7-1 Environment variables for implementing listening on multiple IP addresses

Environment Variable	Description	Default Value	Example
NIC	List of NICs to be listened on. Use commas (,) to separate multiple NIC names. If this parameter is left blank or set to *****, information about all NICs will be listened on. If none of the specified NICs exist, the application will not listen on any IP address.	* (indicating that information about all NICs will be listened on)	eth0,lo
INCLUDE_IP	List of IP addresses bound to NICs to be listened on. The value can be an IP address or an IP address segment. Use commas (,) to separate multiple IP addresses. If this parameter is left blank or set to *****, all IP addresses bound to the specified NICs will be listened on.	* (indicating that all IP addresses bound to the specified NICs will be listened on)	192.168.0.1/24, 127.0.0.1
EXCLUDE_IP	List of IP addresses to be excluded from the IP address list obtained from **NIC** and **INCLUDE_IP**. The value can be an IP address or an IP address segment. Use commas (,) to separate multiple IP addresses. If this parameter is left blank, no IP address will be excluded.	There is no default value.	192.168.0.1/24, 127.0.0.1

----End

Creating a Gateway

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Application Mesh** > **Gateways**. On the page displayed, click **Create Gateway** in the upper right corner.

 Copyright © Huawei Technologies Co., Ltd.

Step 3 Specify the gateway information.

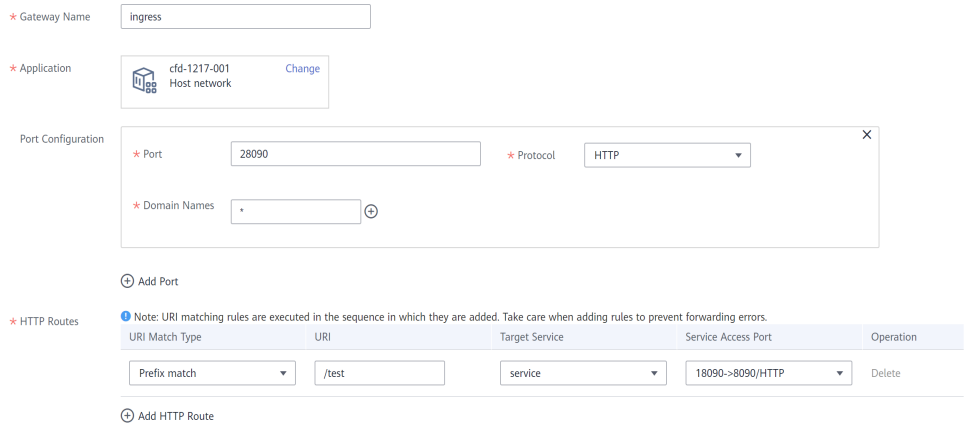
- **Gateway Name**: Enter a gateway name.
- **Application**: Select the gateway application deployed in **Deploying an Edge Gateway Application**.
- **Backend Configuration**

  Port: Enter the port number used by the gateway to access backend applications.

  Protocol: Select the protocol used by the gateway to access backend applications.

  Domain Name List: Enter the domain names that can be accessed. Regular expression matching is supported. For example, * indicates all domain names.
- **HTTP Routes**
  - **URI Match Type**: **Prefix match**, **Exact match**, and **Regular expression** are supported.
  - **URI**: Enter a URI.
  - **Target Service**: Select the Service to which the rule matches, that is, the Service to which the traffic from the URI is forwarded.
  - **Service Access Port**: Select the port used by the Service to access the application.

Figure 7-6 Gateway configuration



----End

 Copyright © Huawei Technologies Co., Ltd.

8 Edge-Cloud Messages

8.1 Edge-Cloud Message Overview

IEF provides the function of routing messages exchanged between the edge and cloud. Based on configured routes, IEF forwards messages to the corresponding endpoint. In this way, messages can be forwarded based on specified paths, enhancing flexibility in data routing control and improves data security.

- Message endpoint: a party that sends or receives a message. It can be an end device or a cloud service.
- Message route: a message forwarding path.

Message Endpoints

IEF provides the following default message endpoints:

- **SystemEventBus**: MQTT broker on an edge node, which can communicate with other endpoints on behalf of the edge node. It can function as a source endpoint to send data to the cloud, or as a destination endpoint to receive messages from the cloud. MQTT topics on the edge node are used as endpoint resources of the MQTT broker.
- **SystemREST**: a REST gateway interface in the cloud. It can function as a source endpoint to send REST requests to the edge. REST request paths are used as endpoint resources of SystemREST.
- **SysEdgedb**: time series database at the edge. It can only function as a source endpoint to send data to Systsdb in the cloud.
- **Systsdb**: time series database in the cloud. It can only function as a destination endpoint to send data to the cloud service CloudTable.

You can also create the following message endpoints:

- **Service Bus**: an endpoint deployed on an edge node to process transaction requests. It can function as a destination endpoint to process file upload requests. REST request paths are used as endpoint resources of Service Bus.
- **DIS**: data injection service. It can function as a destination endpoint to receive data forwarded by IEF. DIS streams created in DIS are used as endpoint resources.

 Copyright © Huawei Technologies Co., Ltd.

- **API Gateway**: API gateway service. It can function as a destination endpoint to receive data forwarded by IEF. API URLs created in API Gateway are used as endpoint resources.

Routes

Currently, IEF supports the following message forwarding paths:

- SystemREST -> ServiceBus: The REST gateway interface in the cloud is called to obtain file services on edge nodes. The collaboration with the edge file service in the edge marketplace is required.
- SystemREST -> SystemEventBus: The REST gateway interface in the cloud is called to send messages to SystemEventBus (MQTT broker) on edge nodes.
- SystemEventBus -> DIS/API Gateway: You can publish end device data to a custom topic in the MQTT broker of an edge node. IEF forwards the device data to a DIS stream or an API Gateway address. Then, you can extract the data for processing and analysis. You must customize an MQTT topic when creating the route. For details about custom topics, see **Custom Topics**.
- SystsEdgedb -> Systsdb: SystsEdgedb calls the REST interface of the time series database at the edge to upload the time series data of edge nodes to CloudTable. The collaboration with the time series database in the edge marketplace is required.

To use the message routing function, **create endpoints** and then **create routes**.

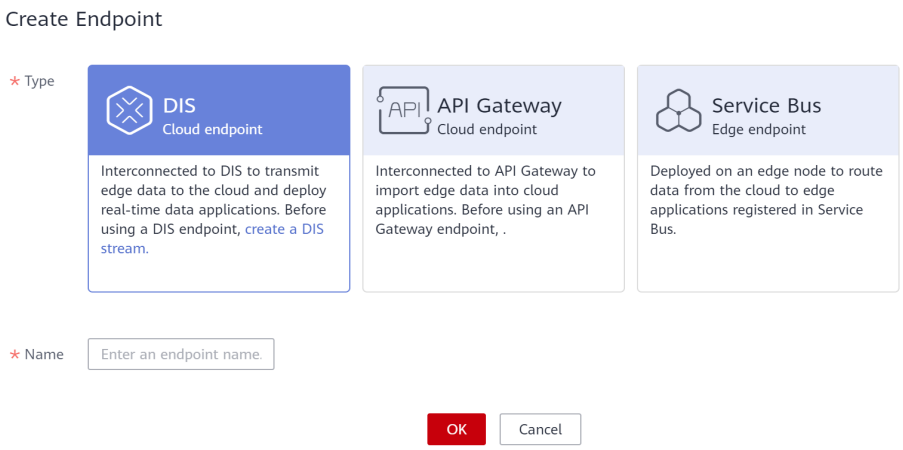
Creating an Endpoint

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge-Cloud Messages** > **Endpoints**.

Step 3 Click **Create Endpoint** in the upper right corner, and set the endpoint parameters.

Figure 8-1 Creating a message endpoint



- **Type**: Select an endpoint type. DIS, API Gateway, and Service Bus are supported.

 Copyright © Huawei Technologies Co., Ltd.

- **Name**: Enter an endpoint name.
- **Service Port**: This parameter is available only for endpoints of the **Service Bus** type. It ranges from 1024 to 65535.

Step 4 Click **OK**. The endpoint is successfully created and the endpoint list page is displayed.

----End

Creating a Route

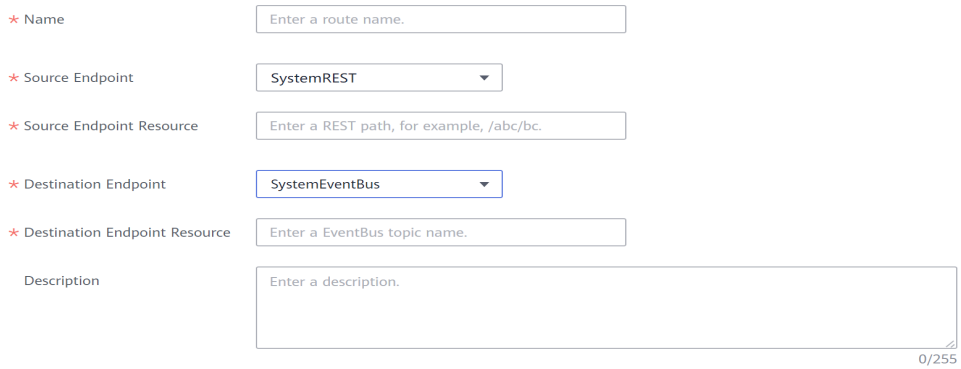
Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge-Cloud Messages** > **Routes**.

Step 3 Click **Create Route** in the upper right corner.

Step 4 Set the route parameters.

Figure 8-2 Creating a route



- **Source Endpoint**: Select a source endpoint type.
- **Source Endpoint Resource**: Select a source endpoint resource.
- **Destination Endpoint**: Select a destination endpoint type.
- **Destination Endpoint Resource**: Select a DIS stream.

Step 5 Click **Create**. The route is successfully created and the route list page is displayed.

IEF forwards messages sent to the specified resource of the source endpoint to the specified resource of the destination endpoint based on the route.

----End

Disabling or Enabling a Route

- To disable a route, click **Disable** in the row where the route resides.

  After the route is disabled, IEF will not forward messages destined for the specified resource of the source endpoint.
- To enable a route, click **Enable** in the row where the route resides.

  After the route is enabled, IEF will forward messages destined for the specified resource of the source endpoint.

 Copyright © Huawei Technologies Co., Ltd.

8.2 Cloud Delivering Messages to Edge Nodes

Scenario

IEF can deliver messages from SystemREST in the cloud to SystemEventBus (MQTT broker) on edge nodes.

To be specific, IEF calls the open REST gateway interface to send messages to SystemEventBus on an edge node based on the node ID and custom topic. Your end devices can receive messages from the cloud by subscribing to custom topics.

The procedure is as follows:

1. **Creating a Route**
2. **Sending a Message**
3. **Receiving a Message**

Creating a Route

SystemREST and SystemEventBus are default endpoints and do not need to be created. SystemREST indicates the REST gateway interface of IEF in the region. SystemEventBus indicates the MQTT broker of an edge node.

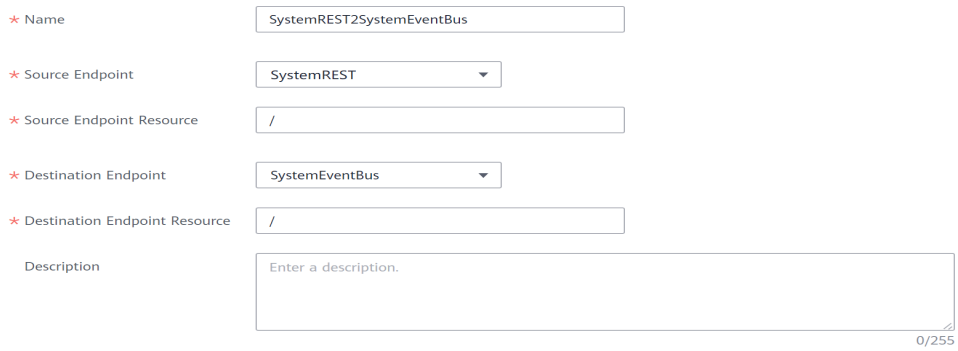
Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge-Cloud Messages** > **Routes**.

Step 3 Click **Create Route** in the upper right corner.

Step 4 Set parameters.

Figure 8-3 Creating a route



- **Source Endpoint**: Select **SystemREST**.
- **Source Endpoint Resource**: Enter a URL path starting with a slash (/). You can use {} for fuzzy match. For example, **/aaa/{any-str}/bbb** can match **/aaa/ccc/bbb** or **/aaa/ddd/bbb**. The specified source resource is used as a matching field for calling the REST interface.
- **Destination Endpoint**: Select **SystemEventBus**.
- **Destination Endpoint Resource**: Enter the topic name prefix used when the message is forwarded to the MQTT broker.

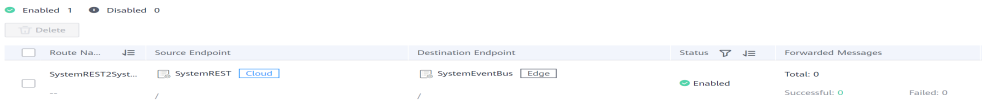
 Copyright © Huawei Technologies Co., Ltd.

> **NOTE**
>
> If both **Source Endpoint Resource** and **Destination Endpoint Resource** are set to **/**, IEF will forward all messages sent to the REST interface to the MQTT broker of the corresponding edge nodes. The topic name contained in the messages is the same as the request URL.

Step 5 Click **Create**.

The created route is displayed in the message route list.

Figure 8-4 Routes



----End

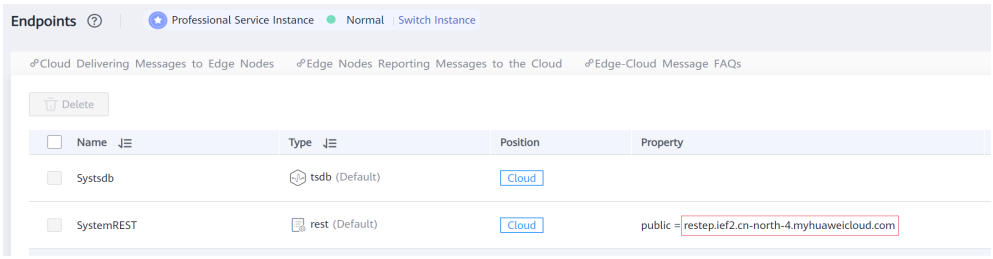
Sending a Message

To deliver a message from SystemREST in the cloud to SystemEventBus on an edge node, obtain the SystemREST endpoint property, construct a request, and send the request to SystemEventBus through a route.

Step 1 Obtain the SystemREST endpoint property.

1. Log in to the IEF console.
2. In the navigation pane, choose **Edge-Cloud Messages** > **Endpoints**. In the row where SystemREST is located, the value of **public** in the **Property** column is the endpoint of SystemREST, as shown in the following figure.

Figure 8-5 SystemREST endpoint property



> **NOTE**
>
> The SystemREST endpoint supports the following addresses. All these addresses support AK/SK authentication and token authentication. You are advised to use the following addresses:
>
> - CN South-Guangzhou: **ief-stargate.cn-south-1.myhuaweicloud.com**
> - CN North-Beijing1: **ief-stargate.cn-north-1.myhuaweicloud.com**
> - CN North-Beijing4: **ief-stargate.cn-north-4.myhuaweicloud.com**
> - CN East-Shanghai1: **ief-stargate.cn-east-3.myhuaweicloud.com**
> - CN East-Shanghai2: **ief-stargate.cn-east-2.myhuaweicloud.com**

Step 2 Construct a request and send it.

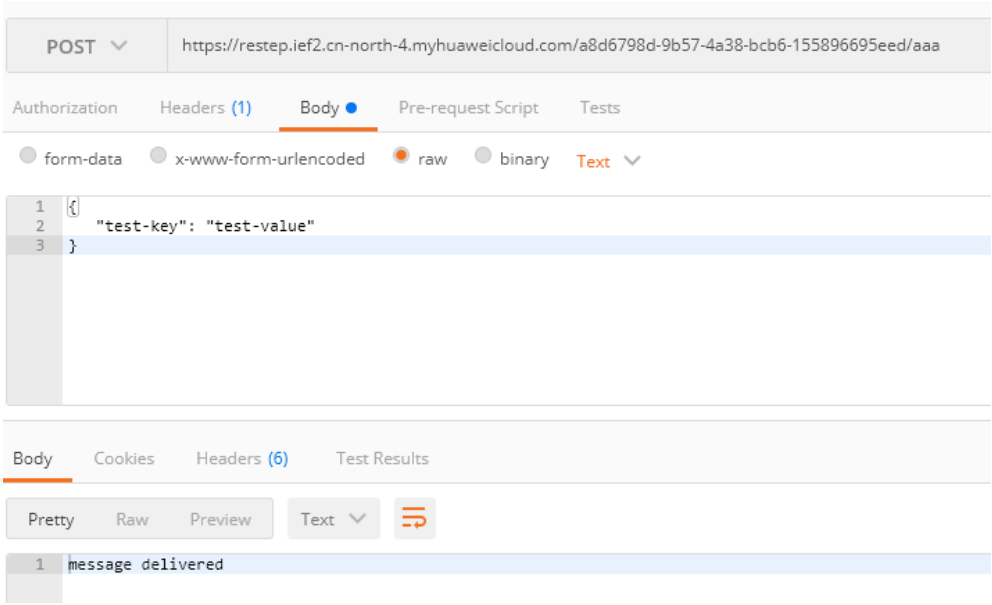
 Copyright © Huawei Technologies Co., Ltd.

An HTTPS request needs to be constructed and sent to SystemREST. The details are as follows:

- Method: POST
- URL: **https://{*rest_endpoint*}/{*node_id*}/{*topic*}**. In the URL, *{rest_endpoint}* is the endpoint obtained in **Step 1**, *{node_id}* is the edge node ID, and *{topic}* is the message topic, that is, the source endpoint resource defined in **Creating a Route**.
- Body: content of the message to be sent, which is user-defined.
- Header: **X-Auth-Token:** *Token* (a valid token obtained by the user from IAM). For details on how to obtain a token, see **Obtaining a User Token**.
- For details about AK/SK authentication, see **AK/SK Authentication**.

For example, use Postman to send the following message:

Figure 8-6 Sending a message



----End

Receiving a Message

Step 1 Log in to the edge node.

Step 2 Use the MQTT client to receive messages.

Currently, edge nodes support two types of MQTT brokers.

- Built-in MQTT broker (using port 8883): To use this type of MQTT broker, edge nodes must pass certificate authentication and subscribe to corresponding topics. For details about certificate authentication, see **Performing Security Authentication Using Certificate**.
- External MQTT broker (using port 1883): To use this type of MQTT broker, install a third-party MQTT broker on edge nodes and enable edge nodes to subscribe to corresponding topics.

 Copyright © Huawei Technologies Co., Ltd.

This section describes how to use an external MQTT broker, for example, Mosquitto, to receive messages. The procedure is as follows:

- In the Ubuntu OS, run the following commands to install Mosquitto:
```
apt-get install mosquitto
systemctl start mosquitto
systemctl enable mosquitto
```
- In the CentOS OS, run the following commands to install Mosquitto:
```
yum install epel-release
yum install mosquitto
systemctl start mosquitto
systemctl enable mosquitto
```

After the installation is complete, run the topic subscription command. If a message is sent after the subscription, the edge node will receive the message. In the following command, # indicates that any topic is subscribed to. You can replace **#** with a specified topic, for example, **/aaa** or **/bbb**.

```
[root@ief-node ~]# mosquitto_sub -t '#' -d
Client mosq-m02iwjsp4j2ISMw6rw sending CONNECT
Client mosq-m02iwjsp4j2ISMw6rw received CONNACK (0)
Client mosq-m02iwjsp4j2ISMw6rw sending SUBSCRIBE (Mid: 1, Topic: #, QoS: 0, Options: 0x00)
Client mosq-m02iwjsp4j2ISMw6rw received SUBACK
Subscribed (mid: 1): 0
Client mosq-m02iwjsp4j2ISMw6rw received PUBLISH (d0, q0, rQ, mQ, '/aaa', ... (31 bytes))
{
   "test-key": "test-value"
}
```

----End

8.3 Edge Nodes Reporting Messages to the Cloud

Scenario

IEF allows edge nodes to report messages to the cloud.

You can publish data to a custom topic in SystemEventBus (MQTT broker) of edge nodes. IEF forwards these data to a DIS stream or an API Gateway address. Then, you can extract the data for processing and analysis.

This section uses the DIS endpoint as an example. The method of using the API Gateway endpoint is similar. The procedure is as follows:

1. **Creating an Endpoint**
2. **Buying a DIS Stream**
3. **Creating a Route**
4. **Sending a Message**

Creating an Endpoint

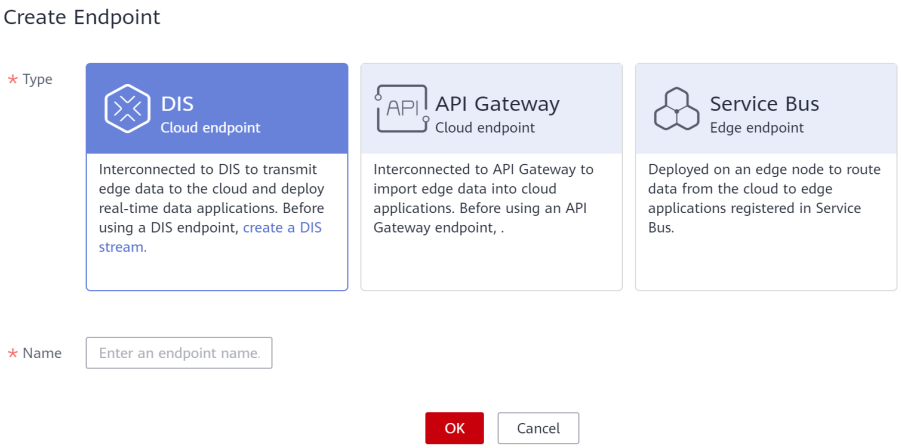
Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge-Cloud Messages** > **Endpoints**.

Step 3 Click **Create Endpoint** in the upper right corner. Select **DIS** for **Type**, and enter an endpoint name.

 Copyright © Huawei Technologies Co., Ltd.

Figure 8-7 Creating an endpoint



Step 4 Click **OK**.

----End

Buying a DIS Stream

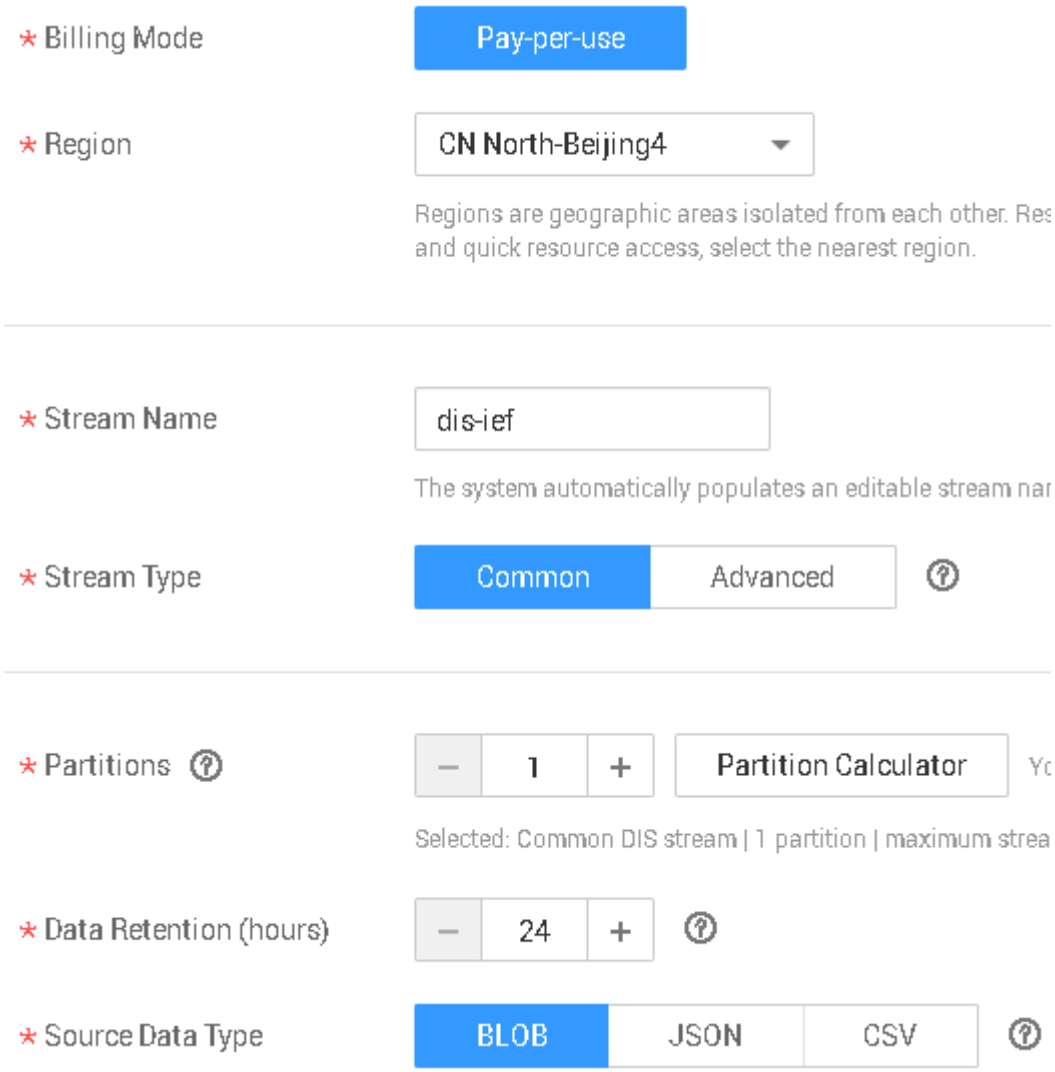
Before sending messages to DIS, buy a DIS stream.

Step 1 Log in to the DIS console.

Step 2 Click Buy Stream in the right corner and set parameters.

 Copyright © Huawei Technologies Co., Ltd.

Figure 8-8 Buying a DIS stream



Step 3 Click **Next** and **Submit**.

----End

Creating a Route

Step 1 Log in to the IEF console.

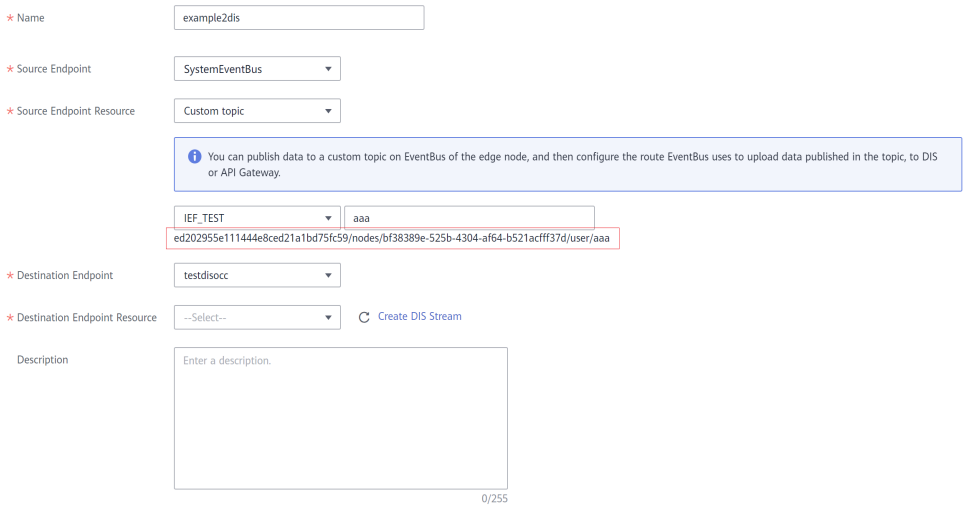
Step 2 In the navigation pane, choose **Edge-Cloud Messages** > **Routes**.

Step 3 Click **Create Route** in the upper right corner.

Step 4 Set parameters.

 Copyright © Huawei Technologies Co., Ltd.

Figure 8-9 Creating a route



- **Source Endpoint**: Select **SystemEventBus**.
- **Source Endpoint Resource**: Select **Custom topic**, select the edge node that will send messages, and enter a topic name.
- **Destination Endpoint**: Select the endpoint created in **Creating an Endpoint**.
- **Destination Endpoint Resource**: Select the DIS stream purchased in **Buying a DIS Stream**.

NOTE

Record the topic, as shown in the red box in the preceding figure.

Step 5 Click **Create**.

----End

Sending a Message

Use the MQTT client on the edge node to send messages.

In this example, the message needs to be published to the topic specified in **Creating a Route**. As shown in the following figure, mosquitto_pub is used to send a message.

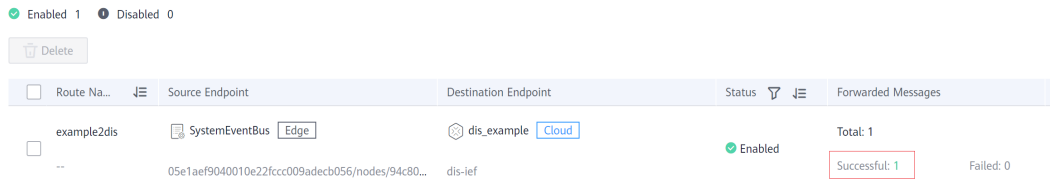
Figure 8-10 Sending a message

```
[root@ief-node ~]# mosquitto_pub -t 'a4a5d4098fb4474fa22cd05f897d6b99/nodes/a8d6798d-9b57-4a38-bcb6-155896695eed/user/aaa' -d
 -m '{ "edgemsg": "msgToCloud"}'
Client mosq-2khhOGMUUAHedlpzBt sending CONNECT
Client mosq-2khhOGMUUAHedlpzBt received CONNACK (0)
Client mosq-2khhOGMUUAHedlpzBt sending PUBLISH (d0, q0, r0, m1, 'a4a5d4098fb4474fa22cd05f897d6b99/nodes/a8d6798d-9b57-4a38-bc
b6-155896695eed/user/aaa', ... (26 bytes))
Client mosq-2khhOGMUUAHedlpzBt sending DISCONNECT
[root@ief-node ~]#
```

After the message is sent, you can view the **Forwarded Messages** column in the route list to check whether the message is successfully forwarded by the route.

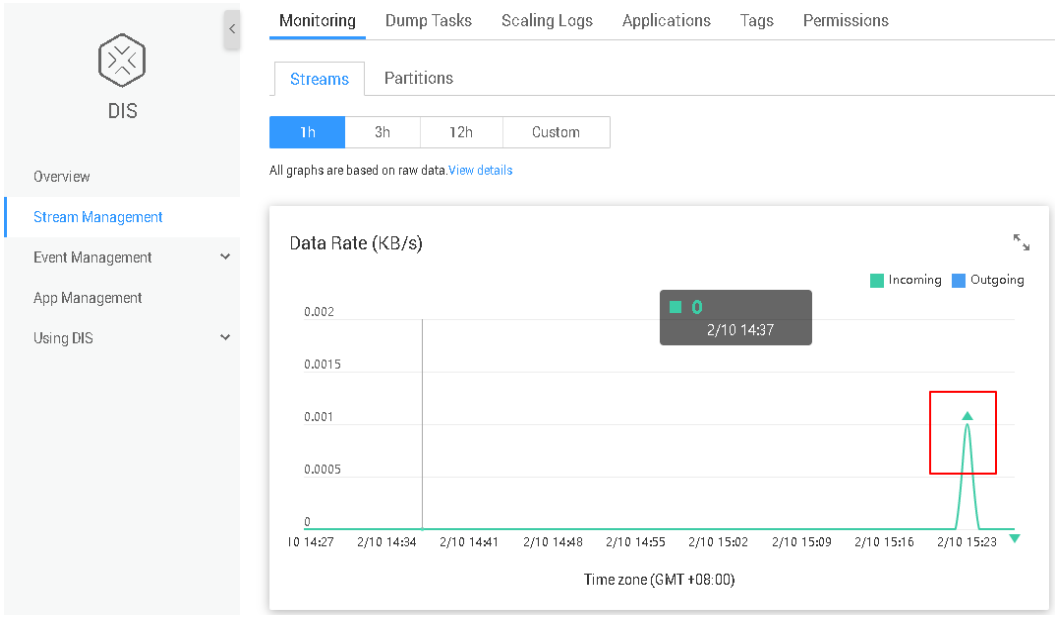
 Copyright © Huawei Technologies Co., Ltd.

Figure 8-11 Number of forwarded messages



You can view incoming messages on the DIS console. For details on how to obtain data from DIS, see **Obtaining Data from DIS**.

Figure 8-12 DIS data monitoring



8.4 System Subscriptions

Scenario

IEF provides the system subscription function for you to subscribe to IEF resource change events. When a resource is created, updated, or deleted, IEF will send a message to the specified API Gateway endpoint so that you can obtain resource changes in a timely manner.

System subscription is the special implementation of edge-cloud messages. IEF sends event messages about specific resources to a specified topic and calls an API of API Gateway to notify you of resource changes.

Events That Can Be Subscribed To

You can subscribe to the following events from IEF.

 Copyright © Huawei Technologies Co., Ltd.

Table 8-1 Events that can be subscribed to

System Event	Topic	Resource Type	Operation
Instance creation	$hw/events/instance/+/created	Application instance	Create
Instance update	$hw/events/instance/+/updated	Application instance	Update
Instance deletion	$hw/events/instance/+/deleted	Application instance	Delete
Application deletion	$hw/events/deployment/+/deleted	Containerized application	Delete
Application creation	$hw/events/deployment/+/created	Containerized application	Create
Application update	$hw/events/deployment/+/updated	Containerized application	Update
Node being brought online	$hw/events/edgeNode/+/online	Edge node	Bring online
Node being brought offline	$hw/events/edgeNode/+/offline	Edge node	Bring offline

System Subscription Process

The system subscription process is as follows:

1. Create an API on API Gateway for IEF to call.
2. Create an API Gateway endpoint on IEF.
3. Create a system subscription on IEF.

Creating an API

Create an API on API Gateway. For details, see **API Gateway Introduction**.

This API is called by IEF to send system event messages.

Creating an API Gateway Endpoint

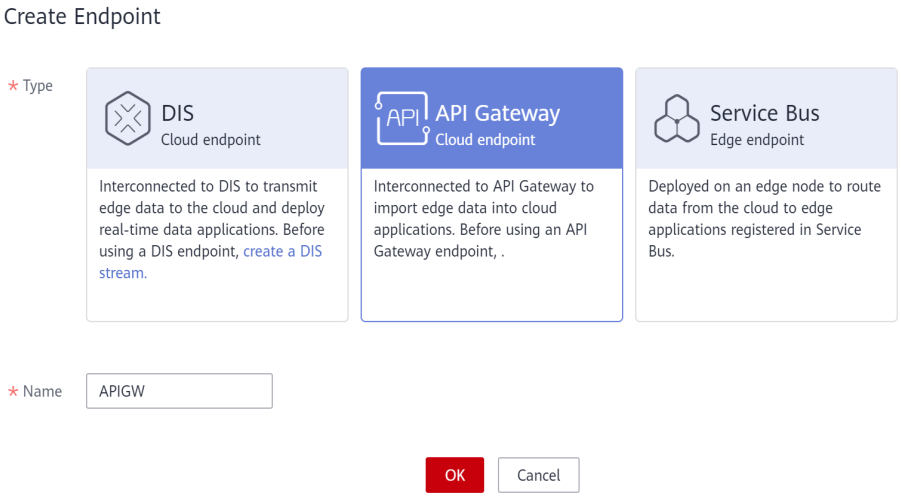
Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge-Cloud Messages** > **Endpoints**.

Step 3 Click **Create Endpoint** in the upper right corner, and set the endpoint parameters.

 Copyright © Huawei Technologies Co., Ltd.

Figure 8-13 Creating an endpoint



- **Type**: Select **API Gateway**.
- **Name**: Enter an endpoint name.

Step 4 Click **OK**. The endpoint is successfully created and the endpoint list page is displayed.

----End

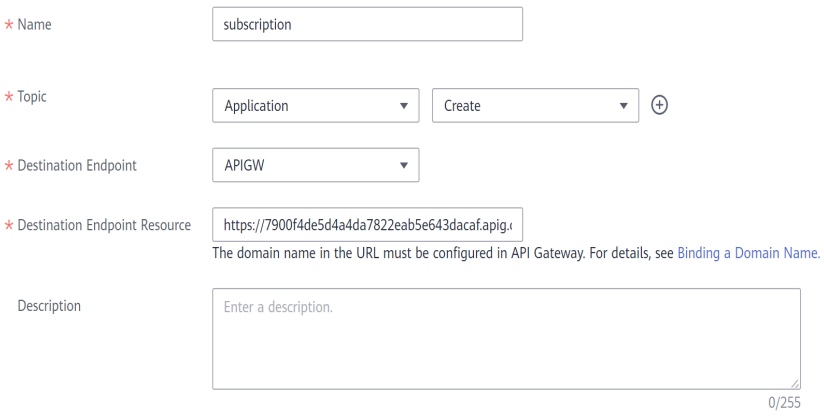
Creating a System Subscription

Step 1 Log in to the IEF console.

Step 2 In the navigation pane, choose **Edge-Cloud Messages** > **System Subscriptions**.

Step 3 Click **Create System Subscription** in the upper right corner, and set the system subscription parameters.

Figure 8-14 Creating a system subscription



- **Topic**: Select the resource and operation to be subscribed, for example, creating a containerized application. For details about the events that can be subscribed to, see **Table 8-1**.

 Copyright © Huawei Technologies Co., Ltd.

- **Destination Endpoint**: Select the endpoint created in **Creating an API Gateway Endpoint**.
- **Destination Endpoint Resource**: Select the API created in **Creating an API**.

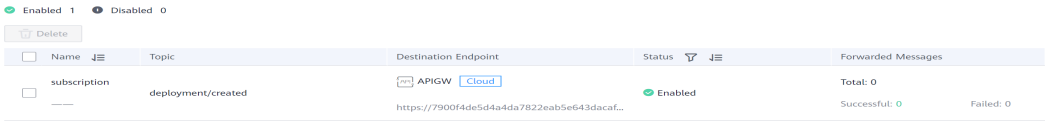
Step 4 Click **Create**.

----End

Message Forwarding After Subscription

After a system subscription is created, IEF will forward a message when a system event occurs. The statistics about message forwarding can be viewed on the IEF console.

Figure 8-15 Number of forwarded messages



Meanwhile, IEF will call the API registered on API Gateway. The request body is as follows: (The request body uses the standard CloudEvents format. For details about CloudEvents, click **here**.)

```
{
   "data": {
      "event_type": "instance",
      "operation": "created",
      "timestamp": 134567677,
      "topic": "$hw/events/instance/+/created",
      "name": "xxxx",
      "attributes": {"ID":"x"}
   },
      "datacontenttype": "application/json",
   "source": "sysevents",
   "id": "xxxx",
   "time": "2020-11-5 xxx"
}
```

- **data**: system event data.
  - **event_type**: resource type. The value is a character string.
  - **operation**: operation performed on the resource. The value is a character string.
  - **topic**: topic to which a message is sent. The value is a character string.
  - **timestamp**: time when an event occurs. The value is of the UInt64 type.
  - **name**: resource name. The value is a character string.
  - **attributes**: resource attributes. This parameter is not contained in the request for deleting a resource. The value is of the Object type.
- **datacontenttype**: format of the system event data content. The value is a character string.
- **source**: source of the system event. The value is a character string.
- **id**: system event ID, which is a character string.
- **time**: time when the system event occurs. The value is a character string.

 Copyright © Huawei Technologies Co., Ltd.

The following table lists the resource types, operations, and topics supported by IEF.

Table 8-2 Events that can be subscribed to

System Event	Topic	Resource Type	Operation
Instance creation	$hw/events/instance/+/created	Application instance	Create
Instance update	$hw/events/instance/+/updated	Application instance	Update
Instance deletion	$hw/events/instance/+/deleted	Application instance	Delete
Application deletion	$hw/events/deployment/+/deleted	Containerized application	Delete
Application creation	$hw/events/deployment/+/created	Containerized application	Create
Application update	$hw/events/deployment/+/updated	Containerized application	Update
Node being brought online	$hw/events/edgeNode/+/online	Edge node	Bring online
Node being brought offline	$hw/events/edgeNode/+/offline	Edge node	Bring offline

 Copyright © Huawei Technologies Co., Ltd.

9 Batch Management

 Copyright © Huawei Technologies Co., Ltd.

10 Edge Marketplace

IEF allows more than 20 AI models such as stream processing, video analysis, Optical Character Recognition (OCR), and image recognition, to be deployed on edge nodes. It also provides synergy between edge applications and cloud services. You can view the application details in **Edge Application Center** and deploy the application on edge nodes. If 

is displayed on the application details page, the application is available.

IEF provides a software and hardware integrated solution, which offers users with low-cost, out-of-the-box, and centralized on-cloud O&M services. It uses Huawei general-purpose servers and AI hardware and is deeply integrated with Huawei Ascend chips to provide high-performance, low-cost edge AI inference computing power. You can view the edge hardware provided by IEF on the **Edge Hardware Center** page.

In addition, you can click **Become a Partner** to apply for becoming an IEF partner in the edge marketplace. After becoming a partner, you can release your applications, hardware, and solutions in the edge marketplace, thereby enhancing your brand influence through HUAWEI CLOUD sales, channel, and online promotion.

 Copyright © Huawei Technologies Co., Ltd.

11 Auditing

11.1 IEF Operations Supported by CTS

Cloud Trace Service (CTS) records operations on cloud service resources, allowing you to query, audit, and backtrack the resource operation requests initiated from the IEF console or open APIs as well as responses to the requests.

Table 11-1 IEF operations that can be recorded by CTS

Operation	Resource Type	Trace Name
Registering an edge node	EdgeNode	createEdgeNode
Updating an edge node	EdgeNode	updateEdgeNode
Deleting an edge node	EdgeNode	deleteEdgeNode
Creating a device template	DeviceTemplate	createDeviceTemplate
Updating a device template	DeviceTemplate	updateDeviceTemplate
Deleting a device template	DeviceTemplate	deleteDeviceTemplate
Registering a device	Device	createDevice
Updating a device	Device	updateDevice
Deleting a device	Device	deleteDevice
Deploying an Edge-Connector application	Device	deployConnector
Updating device twins	Device	updateDeviceTwin
Updating the device access configuration	Device	updateDeviceAccessConfig

 Copyright © Huawei Technologies Co., Ltd.

Operation	Resource Type	Trace Name
Creating an application template	Application	createApplication
Updating an application template	Application	updateApplication
Deleting an application template	Application	deleteApplication
Creating an application template version	AppVersion	createAppVersion
Updating an application template version	AppVersion	updateAppVersion
Deleting an application template version	AppVersion	deleteAppVersion
Creating a ConfigMap	ConfigMap	createConfigMap
Updating a ConfigMap	ConfigMap	updateConfigMap
Deleting a ConfigMap	ConfigMap	deleteConfigMap
Creating a containerized application	Deployment	createDeployment
Updating a containerized application	Deployment	updateDeployment
Deleting a containerized application	Deployment	deleteDeployment
Querying a list of containerized applications	Deployment	getDeploymentList
Querying a containerized application	Deployment	getDeployment
Creating an endpoint	Endpoint	createEndpoint
Deleting an endpoint	Endpoint	deleteEndpoint
Creating a function	Function	createFunction
Updating a function	Function	deleteFunction
Deleting a function	Function	assignFunction
Deassociating a function with an edge node	Function	removeFunction
Starting a function	Function	startFunction
Stopping a function	Function	stopFunction

 Copyright © Huawei Technologies Co., Ltd.

Operation	Resource Type	Trace Name
Deploying a function	Function	deployFunction
Creating a product group	Group	createGroup
Deleting a product group	Group	deleteGroup
Creating a dependency	Library	createLibrary
Updating a dependency	Library	updateLibrary
Deleting a dependency	Library	deleteLibrary
Associating a dependency with a function	Library	assignLibrary
Deassociating a dependency with a function	Library	removeLibrary
Adding a node certificate	NodeCert	AddNodeCert
Deleting a node certificate	NodeCert	deleteNodeCert
Upgrading the firmware	NodeFirmware	UpgradeNodeFirmware
Querying a list of application instances	Pod	getPods
Querying an instance	Pod	getPod
Creating a product	Product	createProduct
Deleting a product	Product	deleteProduct
Querying quotas	Quota	getQuota
Updating a quota	Quota	updateQuota
Creating a rule	Rule	createRule
Deleting a rule	Rule	deleteRule
Updating a rule	Rule	updateRule
Creating a secret	Secret	createSecret
Updating a secret	Secret	updateSecret
Deleting a secret	Secret	deleteSecret
Filtering resources by tag	Tags	filterTags
Adding or deleting tags in batches	Tags	batchAddDeleteTags
Adding a tag	Tags	addTag

 Copyright © Huawei Technologies Co., Ltd.

Operation	Resource Type	Trace Name
Deleting a tag	Tags	deleteTag

11.2 Viewing Tracing Logs

Scenario

After CTS is enabled, CTS starts recording operations on cloud resources. Operation records for the last seven days can be viewed on the CTS console.

This section describes how to query operation records of the last seven days on the CTS console.

Procedure

Step 1 Log in to the management console.

Step 2 Click in the upper left corner and select a region.

Step 3 Click **Service List**, and choose **Management & Deployment** > **Cloud Trace Service**.

Step 4 In the navigation pane, choose **Trace List**.

Step 5 Specify the filters used for querying traces. The following filters are available:

- **Trace Type**: The options are **Management** and **Data**.
- **Trace Source**, **Resource Type**, and **Search By**

  Select the desired filter criteria from the drop-down lists. Select **IEF** from the **Trace Source** drop-down list.

  If you select **Trace name** from the **Search By** drop-down list, specify a trace name.

  If you select **Resource ID** from the **Search By** drop-down list, select or enter a resource ID.

  If you select **Resource name** from the **Search By** drop-down list, select or enter a resource name.
- **Operator**: Select an operator (at user level rather than account level).
- **Trace Status**: Select **All trace statuses**, **Normal**, **Warning**, or **Incident**.

Step 6 Click on the left of a trace to expand its details.

Step 7 Click **View Trace** in the **Operation** column. A dialog box is displayed to show trace structure details.

----End

 Copyright © Huawei Technologies Co., Ltd.

12 Permissions Management

12.1 Creating a User and Granting Permissions

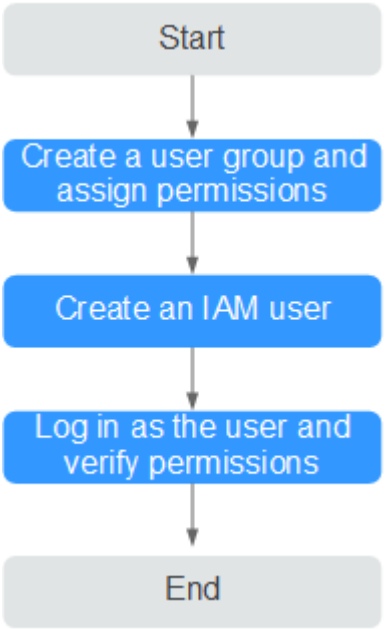
If your HUAWEI CLOUD account does not need individual IAM users, then you may skip over this chapter.

IAM users need to be granted the **Tenant Administrator** policy. This policy has system administrator permissions and has the operation permissions on all cloud services except IAM.

This section describes how to use a group to grant the **Tenant Administrator** policy to a user. **Figure 12-1** shows the process of granting permissions.

Process Flow

Figure 12-1 Process for granting IEF permissions



 Copyright © Huawei Technologies Co., Ltd.

1. **Create a user group and assign permissions** to it.

   Create a user group on the IAM console, and attach the **Tenant Administrator** policy to the group.

2. **Create an IAM user**.

   Create a user on the IAM console and add the user to the group created in **1**.

3. **Log in** and verify permissions.

   Log in to the IEF console by using the user created in **2**, and verify that the user has the administrator permissions for IEF.

12.2 IEF Custom Policies

IEF allows you to create custom policies.

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Edit JSON policies from scratch or based on an existing policy.

For details, see **Creating a Custom Policy**. The following section contains examples of common IEF custom policies.

Example Custom Policies

Allowing users to create and update applications and application templates

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ief:deployment:create",
            "ief:appVersion:update",
            "ief:deployment:update",
            "ief:application:create"
         ],
         "Condition": {
            "StringEquals": {
               "ief:AssumeUserName": [
                  "test"
               ]
            }
         },
         "Resource": [
            "ief:*:*:deployment:*",
            "ief:*:*:appVersion:*",
            "ief:*:*:application:*"
         ]
      }
   ]
}
```

12.3 IEF Resources

A resource is an object that exists within a service. IEF resources include product, node, group, deployment, batchjob, application, appVersion, and IEFInstance. You can select these resources by specifying their paths.

 Copyright © Huawei Technologies Co., Ltd.

Table 12-1 IEF resources and their paths

Resource	Resource Name	Path
product	Product	IEF:*:*:product:
node	Edge node	IEF:*:*:node:
group	Edge node group	IEF:*:*:group:
deployment	Deployment	IEF:*:*:deployment:
batchjob	Job	IEF:*:*:batchjob:
application	Application template	IEF:*:*:application:
appVersion	Application template version	IEF:*:*:appVersion:
IEFInstance	IEF instance	IEF:*:*:IEFInstance:

12.4 IEF Request Conditions

Request conditions are useful in determining when a custom policy takes effect. A request condition consists of a condition key and operator. Condition keys are either global or service-level and are used in the Condition element of a policy statement. **Global condition keys** (starting with **g:**) are available for operations of all services, while service-level condition keys (starting with a service name such as **ief:**) are available only for operations of a specific service. An operator is used together with a condition key to form a complete condition statement.

IEF has a group of predefined condition keys that can be used in IAM. For example, to define an "Allow" permission, you can use the condition key *ief:AssumeUserName* to filter matching requesters by username. The following table lists the predefined condition keys of IEF.

 Copyright © Huawei Technologies Co., Ltd.

Table 12-2 Predefined condition keys of IEF

Condition Key	Operator	Description
ief:AssumeUserName	StringEndWithAnyOfIfExists StringStartWithAnyOfIfExists StringEndWithIfExists StringStartWithIfExists StringNotLikeAnyOfIfExists StringLikeAnyOfIfExists StringNotEqualsIgnoreCaseAnyOfIfExists StringEqualsIgnoreCaseAnyOfIfExists StringNotEqualsAnyOfIfExists StringEqualsAnyOfIfExists StringNotLikeIfExists StringLikeIfExists StringNotEqualsIgnoreCaseIfExists StringEqualsIgnoreCaseIfExists StringNotEqualsIfExists StringEqualsIfExists IsNullOrEmpty StringEndWithAnyOf StringStartWithAnyOf StringEndWith StringStartWith StringNotLikeAnyOf StringLikeAnyOf StringNotEqualsIgnoreCaseAnyOf StringEqualsIgnoreCaseAnyOf	Used for matching username

 Copyright © Huawei Technologies Co., Ltd.

Condition Key	Operator	Description
	StringNotEqualsAnyOf StringEqualsAnyOf StringNotLike StringLike StringNotEqualsIgnoreCase StringEqualsIgnoreCase StringNotEquals StringEquals	

Example

This policy can be used only when the username is **test**.

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "ief:deployment:create",
            "ief:appVersion:update",
            "ief:deployment:update",
            "ief:application:create"
         ],
         "Condition": {
            "StringEquals": {
               "ief:AssumeUserName": [
                  "test"
               ]
            }
         },
         "Resource": [
            "ief:*:*:deployment:*",
            "ief:*:*:appVersion:*",
            "ief:*:*:application:*"
         ]
      }
   ]
}
```

 Copyright © Huawei Technologies Co., Ltd.