

**Cloud Phone Host**

# **Best Practices**

**Issue**            04  
**Date**             2023-10-31



**Copyright © Huawei Technologies Co., Ltd. 2023. 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**



HUAWEI 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](mailto:support@huawei.com)

# Security Declaration

## Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process*. For details about this process, visit the following web page:

<https://www.huawei.com/en/psirt/vul-response-process>

For vulnerability information, enterprise customers can visit the following web page:

<https://securitybulletin.huawei.com/enterprise/en/security-advisory>

---

# Contents

---

<b>1 Best Practices of Connecting to General-Purpose Cloud Phones.....</b>	<b>1</b>
1.1 Buying a Server for General-purpose Cloud Phones.....	1
1.2 Connecting to a Cloud Phone to Obtain Its Screen.....	8
1.3 Keeping Management Programs in the Cloud Phone Alive.....	11
1.4 Deploying Applications on General-purpose Cloud Phones.....	15
1.4.1 Querying Cloud Phones.....	15
1.4.2 Installing Applications on a Cloud Phone.....	16
1.4.3 Generating the TAR Package of the Application and Pushing It to the OBS Bucket.....	17
1.4.4 Deploying Applications.....	18
1.4.5 Updating the Version of an Application.....	20
<b>2 Best Practices of Cloud Phone Application Sharing.....</b>	<b>22</b>
2.1 Overview of Application Sharing.....	22
2.2 Usage Guide to Application Sharing.....	22
2.2.1 Purchasing a Cloud Phone Server That Supports Application Sharing.....	22
2.2.2 Preparing the TAR Package of an Application.....	23
2.2.3 Pushing the TAR Package of an Application to a Cloud Phone Server.....	24
2.2.4 Using the appctrl Command to Manage Shared Applications.....	24
2.2.5 Deleting an Application.....	26
2.2.6 Presetting Configuration Files by Sharing Applications.....	27
2.3 appctrl Commands.....	27
<b>3 Installing an Application on Cloud Phones in Batches.....</b>	<b>28</b>
<b>4 Modifying the Cloud Phone GPS Location.....</b>	<b>32</b>
<b>5 Using the Cloud Phone Camera.....</b>	<b>33</b>
<b>6 Using STF to Manage Cloud Phones in Batches.....</b>	<b>36</b>
<b>7 Allowing a Cloud Phone Server to Access a Public Network Outside the Chinese Mainland.....</b>	<b>41</b>
<b>8 Delegating CPH to Operate OBS Buckets.....</b>	<b>47</b>
<b>9 Changing the AOSP Version of a Cloud Phone.....</b>	<b>50</b>
<b>A Change History.....</b>	<b>52</b>

# 1 Best Practices of Connecting to General-Purpose Cloud Phones

## 1.1 Buying a Server for General-purpose Cloud Phones

### Procedure

1. Log in to the management console.
2. On the **Service List** page, choose **Compute > Cloud Phone Host**.
3. In the navigation pane on the left, choose **Servers**. In the upper right corner, click **Buy Server**.
4. Configure required parameters.

**Table 1-1** Basic server settings

Parameter	Description	Example Value
Billing Mode	CPH is billed only yearly/ monthly.	Yearly/Monthly
Region	Cloud phone servers in different regions cannot communicate with each other over a private network. For lower latency and quicker access, select the nearest region.  After a server is purchased, its region cannot be changed.	CN-Hong Kong

Parameter	Description	Example Value
AZ	<p>An AZ is a part of a region and has its own independent power supplies and networks. AZs can communicate with each other over an internal network and are physically isolated.</p> <ul style="list-style-type: none"> <li>• If you require high availability, buy servers in different AZs.</li> <li>• If you require low network latency, buy servers in the same AZ.</li> </ul>	AZ1
Server Type	<p>Two options are available: <b>Cloud phone server</b> and <b>Cloud mobile gaming server</b>. For details, see <a href="#">Servers for General-Purpose Cloud Phones</a> and <a href="#">Cloud Mobile Gaming Servers</a>.</p>	Cloud phone server physical.rx1.xlarge
Instance Specifications	Select the required cloud phone specifications.	rc1.se
Phone Image	<p>Only the Android is supported.</p> <p><b>NOTE</b> To view private images, you must have the <b>ims:images:list</b> permissions.</p>	AOSP7.1.1
Quantity	<ul style="list-style-type: none"> <li>• A maximum of 10 servers can be purchased at a time.</li> <li>• The required duration ranges from 1 month to 3 years.</li> </ul>	<b>Quantity: 1</b> Required duration: <b>6 months</b>

5. Click **Next: Configure Network** to configure the network for the cloud phone server.

You are advised to use the custom network described in [Table 1-2](#).

**Table 1-2** Configuring a custom network

Parameter	Description	Example Value
Network	<p>Select an available VPC and subnet from the drop-down list and specify the method for assigning a private IP address.</p> <p>Ensure that you have the <b>VPC ReadOnlyAccess</b> permissions at least. This VPC, including its subnets and security groups will be used to isolate the cloud phone server from the internet. You can also create a VPC.</p> <ul style="list-style-type: none"> <li> <b>IPv6 not required/Automatically-assigned IPv6 address:</b> This parameter is available only for the cloud phone servers with specific flavors and deployed in a VPC with IPv6 enabled in given regions. For details about how to enable IPv6 on a subnet, see <a href="#">IPv4 and IPv6 Dual-Stack Network</a>. By default, the system assigns IPv4 addresses. If you select <b>Automatically-assigned IPv6 address</b>, the system assigns both an IPv4 address and an IPv6 address.         </li> <li> <b>Do not configure/</b>Select the required shared bandwidth: In the same VPC, the cloud phone server can use the IPv6 address to access dual-stack servers. To access the Internet, select a shared bandwidth from the drop-down list and add the IPv6 address to the shared bandwidth. Then, the cloud phone server can access the IPv6 network on the Internet through the IPv6 address. If you do not select a shared bandwidth when creating a cloud phone server, you can manually add your IPv6 address to a shared bandwidth by referring to <a href="#">(Optional) Step 3: (Optional) Step 3: Buy a Shared Bandwidth and Add the IPv6 Address to It</a>.         </li> </ul>	N/A

Parameter	Description	Example Value
	<p><b>NOTE</b></p> <ul style="list-style-type: none"><li>• If you want to use IPv6 addresses, select <b>Automatically-assigned IPv6 address</b> here, and this option cannot be modified after your server is purchased. If you select <b>IPv6 not required</b> and want to use IPv6 addresses later, you can only buy a new server.</li><li>• Due to VPC restrictions, IPv4/IPv6 dual stack is not supported in the CN East-Shanghai 2 region.</li><li>• IPv6 addresses cannot be added to a dedicated bandwidth.</li><li>• By default, a maximum of 20 IPv6 addresses can be added to a shared bandwidth. A dual-stack cloud phone server has the same number of IPv6 addresses and virtual IP addresses. If you want to purchase a cloud phone server with multiple virtual IP addresses, such as e0v100, apply for a higher shared bandwidth quota in advance.</li><li>• RX1 servers do not support IPv6 addresses.</li></ul>	



Parameter	Description	Example Value
Agency	<p>Authorize CPH to create a <b>cph_admin_trust</b> agency that can assign <b>VPC FullAccess</b> permissions.</p> <p>To authorize CPH to create an agency for you, ensure that you have the <b>Security Administrator</b> permissions.</p> <p>For more information, see <a href="#">Permissions Management</a>.</p> <p>CPH will use the agency to perform the following operations:</p> <ul style="list-style-type: none"> <li>• Create an elastic NIC, EIP, and virtual IP address for a cloud phone.</li> <li>• Apply the default security group for the cloud phone server. The default security group defines the port range allowed to access the server. The port range will be mapped to that of each cloud phone. Then the cloud phones can access applications through the mapped ports.</li> </ul> <p><b>NOTE</b> By default, if an ECS and a cloud phone are in the same VPC, the ECS cannot access the cloud phone through ports 1 to 9999. If you want to allow such access, add a security group rule with a higher priority by following instructions provided in <a href="#">What Are the Security Group Authorization Rules for Cloud Phones Using Custom Networks?</a></p>	N/A
EIP	<ul style="list-style-type: none"> <li>• <b>Auto assign:</b> Buy a new EIP for the server.</li> <li>• <b>Using existing:</b> An existing EIP will be assigned to the server.</li> </ul>	Auto assign
EIP Type	<ul style="list-style-type: none"> <li>• Static BGP offers routing control and protects against route flapping, but cannot choose an optimal path in real time when a network connection fails.</li> <li>• Dynamic BGP enables automatic failovers and chooses the optimal path when a network connection fails.</li> </ul>	Dynamic BGP

Parameter	Description	Example Value
Billed By	The following options are available only when a new EIP is purchased: <ul style="list-style-type: none"> <li>• Traffic: You will be charged based on the total traffic your applications generate.</li> <li>• Shared bandwidth: You will be charged by the bandwidth shared by multiple EIPs.</li> </ul>	Shared bandwidth
Bandwidth Size	Value range: 1 Mbit/s to 2,000 Mbit/s	300 Mbit/s
Bandwidth Name	If you set <b>Bandwidth to Shared bandwidth</b> , select an existing shared bandwidth name from the drop-down list.	bandwidth-001

6. Click **Next: Configure Advanced Settings**.

**Table 1-3** Parameters for advanced settings

Parameter	Description	Example Value
Name	Specifies a unique name for the server and its cloud phones.  Naming rule: The system automatically adds a hyphen followed by a one-digit incremental number to the end of each server name. For the names of the cloud phones that are virtualized from the server, the system automatically adds a 5-digit number suffix in ascending order.  For example, if you purchased a server that can virtualize 60 cloud phones and entered <b>CPH</b> for <b>Name</b> , the server name is <b>CPH-1</b> , and the cloud phone names vary from <b>CPH-1-00001</b> to <b>CPH-1-00060</b> .	CPH

Parameter	Description	Example Value
Key Pair	<p>A key pair is used for remote login authentication.</p> <ul style="list-style-type: none"> <li>If you have created a key pair and stored the private key file (in .pem format) locally, you can select it from the drop-down list.</li> <li>If no key pair is available, click <b>Create Key Pair</b> to create one. Then go back to the <b>Configure Advanced Settings</b> page, refresh the drop-down list, and select the created key pair.</li> </ul> <p>The private key file is used for identity authentication during remote login. For security purposes, the private key file (in .pem format) can be downloaded only once. Keep it secure. For more information about key pairs, see <a href="#">(Recommended) Creating a Key Pair on the Management Console</a>.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>Ensure that your account has the <b>ecs:serverKeypairs:list</b> permissions to query key pairs.</li> <li>If you need to create a key pair, ensure that your account has the <b>ecs:serverKeypairs:create</b> permissions.</li> </ul>	KeyPair-test

Parameter	Description	Example Value
Application Port	<p>Enable this parameter when your cloud phones need to provide services for external systems.</p> <ul style="list-style-type: none"> <li>• <b>Application name:</b> The name can contain letters. However, <b>ADB</b> in uppercase, lowercase, or mixed case are not allowed.</li> <li>• <b>Port number:</b> Ports from <b>0</b> to <b>65535</b> are supported.</li> <li>• <b>Internet access</b> <ul style="list-style-type: none"> <li>– If this option is selected, the cloud phone application port can be accessed over the Internet without authentication. The cloud phone port and the server port are accessible from the Internet.</li> <li>– If this option is not selected, cloud phones can be accessed only over a private network.</li> </ul> </li> </ul> <p><b>CAUTION</b></p> <ul style="list-style-type: none"> <li>• Ensure that security control has been performed before you select <b>Internet access</b>.</li> <li>• CPH does not perform security check for ports you configured to be accessible from the Internet.</li> </ul>	<p>key 10001</p> <p>Do not select it.</p>

7. Click **Next: Confirm** to check the configuration.
  - If the configuration is correct, click **Buy Now**.
  - To modify the configuration, click **Previous**.

8. Complete the payment as prompted.

After the payment, it takes the system about 20 to 30 minutes to automatically create cloud phones.

The cloud phones are available when their statuses change to **Running**.

## 1.2 Connecting to a Cloud Phone to Obtain Its Screen

a cross-platform UI automation compiler, to obtain the cloud phone's screen.

### Prerequisites

- You have purchased a cloud phone server and connected to a cloud phone using ADB. For details, see [Buying a Cloud Phone Server \(Without Detailed Parameter Description\)](#).
- Airtest has been installed on your local PC.

**NOTE**

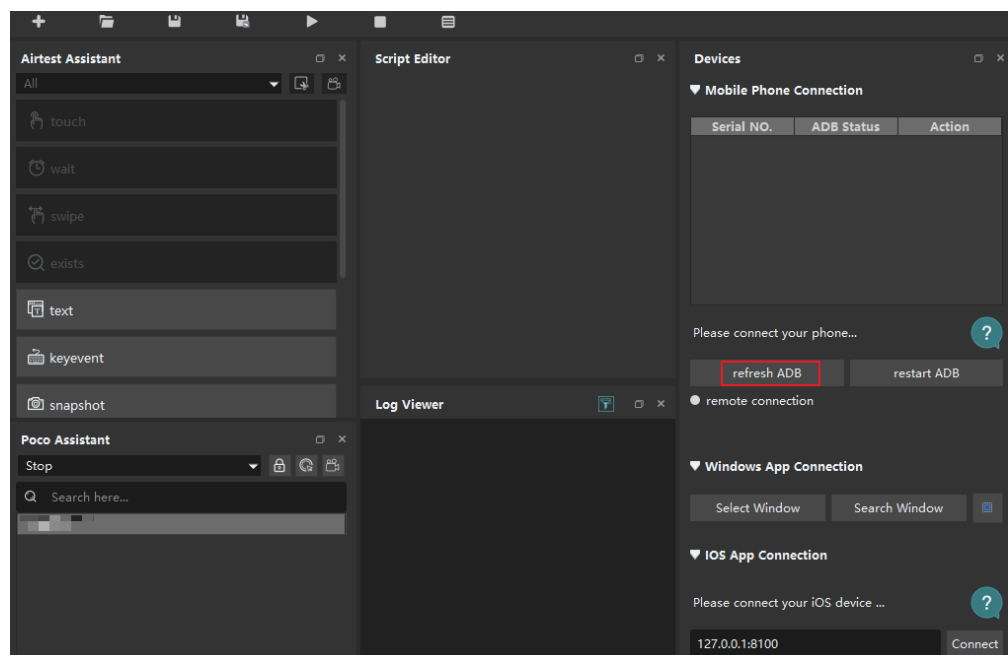
Airtest is available at the [Airtest official website](#). Log in to the website, download the required version, and install it.

- The command-line interface (CLI) for the ADB connection has been closed, and an SSH tunnel has been successfully established.

**Procedure**

1. On the Airtest homepage, click **refresh ADB**.  
Connected mobile phones are displayed.

**Figure 1-1** Airtest homepage

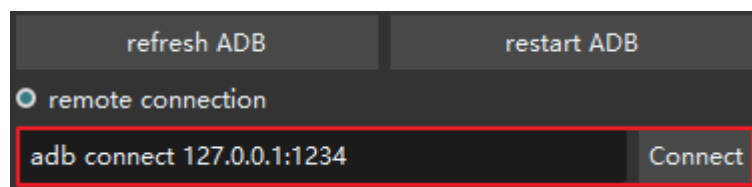


2. If the cloud phone to which you want to connect is not displayed, select **remote connection** and enter the ADB command for connecting to the target cloud phone, as shown in [Figure 1-2](#).

**adb connect 127.0.0.1:1234**

1234 is the local idle port used for establishing the SSH tunnel.

**Figure 1-2** Establishing a remote connection to a cloud phone



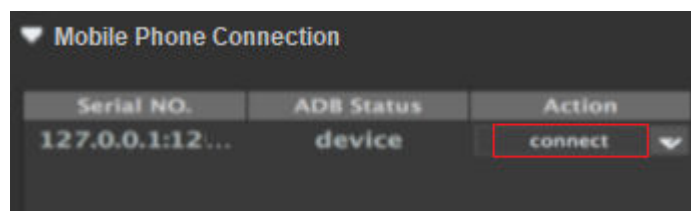
Click **Connect** on the right. The cloud phone to be connected will be displayed in the **Mobile Phone Connection** list.

**CAUTION**

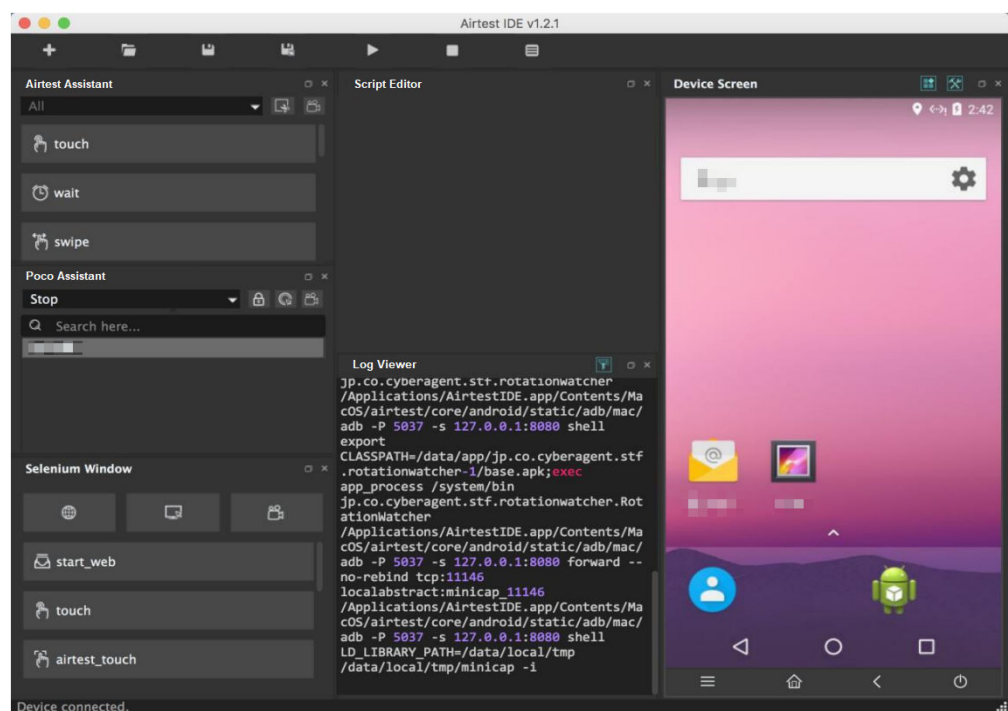
Ensure that the CLI for the ADB connection has been closed. Otherwise, the connection will fail in this step. Ensure that the SSH tunnel has been successfully established. Otherwise, **ADB Status** will be **offline** and the cloud phone screen cannot be obtained even if the cloud phone has been identified.

3. In the list of identified mobile phones, click **connect** on the right of the target cloud phone to obtain its screen.

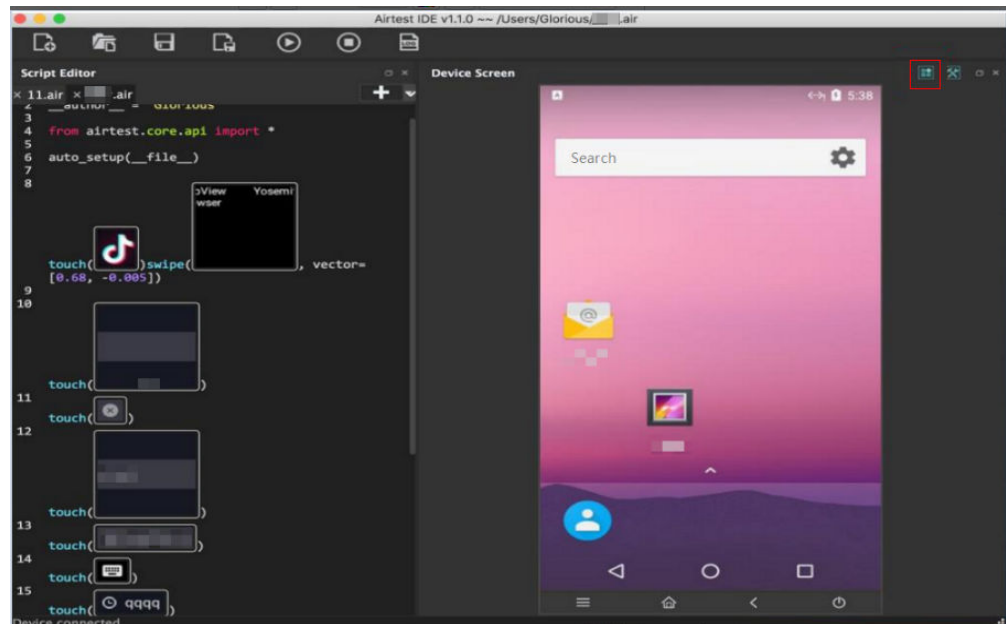
**Figure 1-3** Mobile Phone Connection



**Figure 1-4** Device Screen



4. If you have connected to multiple cloud phones through ADB, click the switch icon in the upper right corner to switch between screens.

**Figure 1-5** Switching between cloud phone screens

## 1.3 Keeping Management Programs in the Cloud Phone Alive

### Management Programs

The management programs in a cloud phone are classified into the following types:

Android APK Service programs: Android services running in the background of the cloud phone without a UI

Android JNI Native programs: executable binary program developed using Android Java Native Interface (JNI)

### Methods for Keeping Management Programs Alive

The `extend_custom.sh` hook script needs to be built into the cloud phone to keep its management problems alive.

When the cloud phone changes to the `boot_complete` state, it checks whether the `extend_custom.sh` script exists in the `/data/local/tmp` directory. If it does, the cloud phone executes the script. You can use this script to operate and move your own files, and start and manage your own programs. The script execution timeout interval is 10s.

- **Keeping Android APK Service programs alive**

Due to system mechanism restrictions, management programs of the Android APK Service type are in the stopped state if they are not started after the first installation. The broadcast indicating that the startup is complete uses `FLAG_EXCLUDE_STOPPED_PACKAGES`. As a result, the stopped applications

cannot receive the broadcast. The installation and initial startup of the Android APK Service management programs depend on the built-in **extend\_custom.sh** script in the **/data/local/tmp** directory of the cloud phone. After the cloud phone is restarted, the programs can be started by receiving the startup completion broadcast.

You can configure the **AndroidManifest.xml** file to receive the startup broadcast and start the Android APK Service programs in the corresponding broadcast processing.

```
<!-- example code-->
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<!--Applicable to 8.0 and later versions-->
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>

<receiver android:name=".DemoServiceReceiver">
  <intent-filter android:priority="1000">
    <action android:name="android.intent.action.BOOT_COMPLETED" />
    <category android:name="android.intent.category.DEFAULT" />
  </intent-filter>
</receiver>
```

Process the startup broadcast and start an Android APK Service management program.

```
// example code
public class DemoServiceReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    if (!intent.getAction().equals("android.intent.action.BOOT_COMPLETED")) {
      return;
    }
    Intent serviceIntent = new Intent(context, DemoService.class);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
      context.startForegroundService(serviceIntent);
    } else {
      context.startService(serviceIntent);
    }
  }
}
```

The keepalive of the Android APK Service management programs can return **START\_STICKY** in the onStartCommand function of the corresponding Service type. In this way, the Android APK Service management programs can be restarted after being killed by the system.

```
// example code
public class DemoService extends Service {
  @Override
  public int onStartCommand(Intent intent, int flags, int startId) {
    //other todo...

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) { //8.0 or later
      String channel_id = "MyTestService-id";
      String channel_name = "MyTestService-name";
      NotificationManager manager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
      NotificationChannel Channel = new NotificationChannel(channel_id, channel_name,
NotificationManager.IMPORTANCE_HIGH);
      if (manager != null) {
        manager.createNotificationChannel(Channel);
      }
      Notification notification = new
Notification.Builder(this).setChannelId(channel_id).setSmallIcon(R.mipmap.ic_launcher).build();
      startForeground(100, notification);
    }
    return START_STICKY;
  }
}
```



```
}  
}
```

The following is an example of the installation and initial startup of Android APK Service management programs that depend on the **extend\_custom.sh** script. (The default APK file is also stored in the **/data/local/tmp** directory.)

```
# example code  
  
#!/system/bin/sh  
  
# APK name  
ApkName=service.apk  
# APK package name  
PackageName=com.huawei.myapplication  
# Service  
ServiceName=com.huawei.myapplication/.MyService  
  
InstallService() {  
    count=`pm list packages | grep $PackageName |wc -l`  
    if [ $count -le 0 ]; then  
        echo "$ApkName is not installed, now install."  
        ret=`pm install $ApkName`  
        Succ="Success"  
        if [ "$ret" == "$Succ" ]; then  
            echo "install succeed."  
        else  
            echo "install failed."  
            exit 1  
        fi  
    else  
        echo "$ApkName is already installed."  
        echo "Service start by boot_complete broadcast."  
        exit 1  
    fi  
    return 0  
}  
  
StartService() {  
    ret=`am startservice -n $ServiceName`  
    Err="Error: Not found; no service started."  
    contain=$(echo $ret | grep "${Err}")  
    if [[ "$contain" != "" ]]; then  
        echo "Service start failed."  
        exit 1  
    else  
        echo "Service start succeed."  
    fi  
    return 0  
}  
  
main() {  
    # Installation  
    InstallService  
  
    echo "To start when first installed."  
    # Start  
    StartService  
}
```

- **Keepalive scheme of management programs of the Android JNI Native type**

The Android JNI Native management programs must be stored in the **/system/bin** directory of the cloud phone and granted the execute permissions. The binary **.so** libraries on which the Android JNI Native management programs depend must be stored in the **system/lib** and **system/lib64** directories.

The Android JNI Native management programs and binary .so libraries can be pushed to the **data** directory of cloud phones by shared storage or application. You can use the **extend\_custom.sh** script to place your files in the corresponding directory. The following methods are available to meet different keepalive requirements:

### - System-level keepalive

System-level keepalive can edit and move the **init\_custom.rc** file to the **/data/local/** directory. You need to restart the cloud phone, then the system will scan the **init\_custom.rc** file. After the system is in the **boot\_completed** state, it starts NativeDemo (the name of the Android JNI Native management program in the example). If the NativeDemo process exits abnormally or is killed, the system restarts it again.

```
on property:sys.boot_completed=1
    start NativeDemo

service NativeDemo /system/bin/NativeDemo
    user root
    group root
    disabled
    writepid /dev/cpuset/system-background/tasks
```

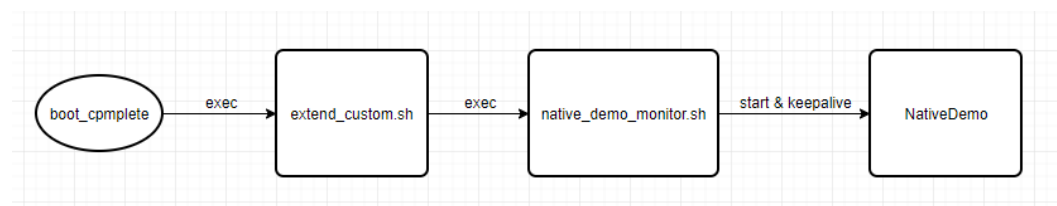
The system-level keepalive has the following advantages and disadvantage:

Advantages: System-level automatic startup and keepalive is highly reliable and of high real-time performance.

Disadvantage: The setting takes effect only after the cloud phone is restarted.

### - User-level keepalive

If you need both automatic startup and management program keepalive, and you do not want to restart the cloud phones for the settings to take effect, you can run the **native\_demo\_monitor.sh** script in **extend\_custom.sh** to check whether the management program is running. The **native\_demo\_monitor.sh** script implements the startup and keepalive of the NativeDemo management program.



Example of the **extend\_custom.sh** hook script

```
# example code
# !/system/bin/sh
[[ -f /data/local/tmp/native_demo_monitor.sh ]] && sh /data/local/tmp/native_demo_monitor.sh &
```

Example of the **native\_demo\_monitor.sh** script

```
# example code
# !/system/bin/sh
file_dir="/data/demo"
CopyFile() {
```

```
cp -rf $file_dir/NativeDemo /system/bin/  
chmod 755 /system/bin/NativeDemo  
  
cp $file_dir/lib*.so /system/lib64/  
}  
  
CheckIfCopyFile() {  
    if [ -s $file_dir ]; then  
        echo "file exist"  
        CopyFile  
    else  
        echo "file not exist"  
    fi  
}  
  
Start() {  
    nohup NativeDemo &  
}  
  
KeepAlive() {  
    while do  
        echo "check NativeDemo proc"  
        proc_count=`ps | grep NativeDemo | wc -l`  
        if [ $proc_count -le 0 ]; then  
            echo "start NativeDemo"  
            Start  
        else  
            echo "NativeDemo already started"  
        fi  
        sleep 5  
    done  
}  
  
main() {  
    CheckIfCopyFile  
    KeepAlive  
}  
  
main
```

The user-level keepalive has the following advantage and disadvantage:

Advantage: You do not need to restart your cloud phone for the settings to take effect.

Disadvantage: The real-time performance of the startup is not as good as that at the system level. The startup depends on the check interval.

## 1.4 Deploying Applications on General-purpose Cloud Phones

### 1.4.1 Querying Cloud Phones

Obtain the cloud phone list by referring to [Querying Cloud Phones](#) in the *Cloud Phone Host API Reference*.

#### Example API

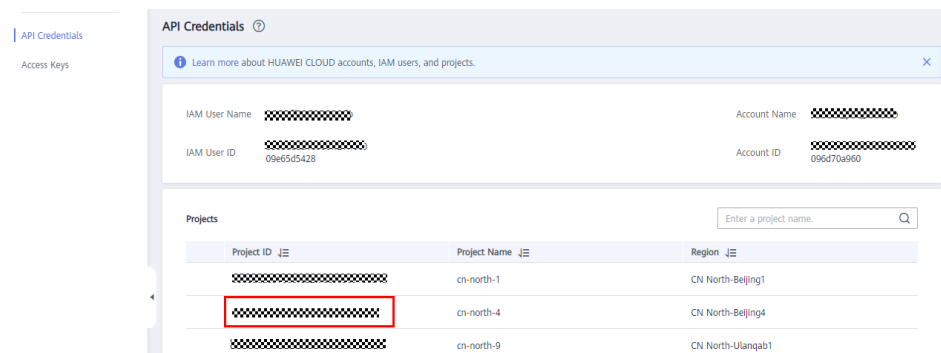
```
GET https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones?  
phone_name={phone_name}&server_id={server_id}&status={status}&offset={offset}&limit={limit}&type={type}  
e}  
Header:
```

```
Content-Type: application/json
X-Auth-Token: ${token}
```

Parameter descriptions:

- **CPH Endpoint** indicates the CPH endpoint in each region in the endpoint list. For example, the CPH endpoint in the CN-Hong Kong region is **cph.ap-southeast-1.myhuaweicloud.com**.
- **project\_id** indicates the project ID of the region where the gaming cloud phone server is deployed, for example, **083e9f825e80f50c2f96c0045edc70e8**. The project ID can be obtained by performing the following operations:
  - a. Log in to the management console.
  - b. Click the username in the upper right corner of the page, and choose **My Credentials** from the drop-down list.
  - c. On the **API Credentials** page, obtain the project ID in the project list.

**Figure 1-6** Obtaining the project ID



- The part after the question mark (?) in the URL is optional.
- **\$token** indicates the response of the API for [Obtaining a User Token Through Password Authentication](#).

## API Calling Example

```
GET https://cph.cn-north-4.myhuaweicloud.com/v1/083e9f825e80f50c2f96c0045edc70e8/cloud-phone/phones
Header:
Content-Type: application/json
X-Auth-Token: ${token}
```

### NOTE

Replace **\${token}** with the actual token.

## 1.4.2 Installing Applications on a Cloud Phone

Install applications on a cloud phone by referring to [Installing the APK](#) in the *Cloud Phone Host API Reference*.

### Prerequisites

- The required Android Package (APK) has been stored in the Object Storage Service (OBS) bucket in the region where the cloud phone server is deployed. For details about how to upload the installation package, see [Scenario 2: Uploading and Downloading Files Through OBS Browser+](#).

- The OBS bucket policies have been configured based on [Delegating CPH to Operate OBS Buckets](#).

## Example API

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/commands
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
  "command": "install",
  "content": "-t -r obs://{bucket_name}/{object_path}",
  "phone_ids": [
    "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  ]
}
```

Parameter descriptions:

- Obtain the values of parameters such as **CPH Endpoint**, **project\_id**, and **token** by referring to [Querying Cloud Phones](#).
- **bucket\_name** indicates the OBS bucket name. **object\_path** indicates the path for storing the application installation package.
- **phone\_ids** indicates the ID of the cloud phone on which the application is to be installed. (Obtain the cloud phone ID by referring to [Querying Cloud Phones](#). You can enter multiple cloud phone IDs, and the application will be installed on all these cloud phones.)

## API Calling Example

```
POST https://cph.cn-east-3.myhuaweicloud.com/v1/081ceeb7fb800f0c2f4cc004bb39c2f7/cloud-phone/phones/commands
Content-Type: application/json
X-Auth-Token: ${token}
{
  "command": "install",
  "content": "-t -r obs://yzw-apk-install/apk/com.hermes.bgame.apk",
  "phone_ids": [
    "bdc2f2e960164dd9a2765374afeea300"
  ]
}
```

- **yzw-apk-install** is the OBS bucket name.  
**apk/com.hermes.bgame.apk** is the path of the application installation package.  
**obs://yzw-apk-install/apk/com.hermes.bgame.apk** is the full path of the application installation package.
- Replace **token** with the actual token.

## 1.4.3 Generating the TAR Package of the Application and Pushing It to the OBS Bucket

### Prerequisites

- The required application has been installed on the cloud phone.
- The OBS bucket policies have been configured based on [Delegating CPH to Operate OBS Buckets](#).

## Example API

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/batch-storage
Header:
Content-Type: application/json
X-Auth-Token: ${token}
Body:
{
  "storage_infos": [{
    "phone_id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "include_files": [
      "/data/app/${package_name}-1",
      "/data/app/${package_name}-2",
      "/data/data/${package_name}",
      "/data/media/0/Android/data/${package_name}"
    ],
    "bucket_name": "${bucket_name}",
    "object_path": "apk/${package_name}_${version_name}.tar"
  ]
}
```

Parameter descriptions:

- Obtain the values of parameters such as **CPH Endpoint**, **project\_id**, **\${token}**, **bucket\_name**, and **object\_path** by referring to [Installing Applications on a Cloud Phone](#).
- **phone\_id** indicates the ID of the cloud phone on which the application is installed.
- The four elements of **include\_files** are four fixed paths.
- If the installation package is a .xapk package, add **/data/media/obb/\${package\_name}** to **include\_files**.
- **object\_path** indicates the path to which the .tar package is uploaded.

### NOTICE

**apk** is any existing folder. In **\${package\_name}\_\${version\_name}.tar**, **package\_name** and **version\_name** need to be modified as required.

- **package\_name** and **version\_name** indicate the package name and version number of the current application.

## 1.4.4 Deploying Applications

### Servers of Storage 2.0 (Recommended)

Push the .tar package to servers. That is, push **apk/\${package\_name}\_\${version\_name}.tar** to the shared application of **\${server\_id1}** and **\${server\_id2}**.

- Example API  
POST https://{CPH Endpoint}/v1/{project\_id}/cloud-phone/phones/share-apps  
Header:  
Content-Type: application/json  
X-Auth-Token: \${token}  
Body:  
{  
 "package\_name": "\${package\_name}"  
 "bucket\_name": "\${bucket\_name}",  
 "object\_path": "apk/\${package\_name}\_\${version\_name}.tar",  
 "server\_ids": [  
 ]  
}

```
"${server_id1}",
"${server_id2}"
]
}
```

Parameter descriptions:

- Obtain the values of parameters such as **CPH Endpoint**, **project\_id**, **token**, **bucket\_name**, and **object\_path** by referring to [Installing Applications on a Cloud Phone](#).
- **package\_name** indicates the package name of the application in Android, for example, **com.miniteck.miniworld**.
- **object\_path** indicates the path to which the .tar package is uploaded.
- **package\_name** and **version\_name** indicate the package name and version number of the current application.

#### NOTE

**apk** is any existing folder. In **apk/\${package\_name}\_\${version\_name}.tar**, **package\_name** and **version\_name** need to be modified as required.

- **server\_ids** indicates IDs of servers where the application is to be deployed. You can enter multiple server IDs. To obtain the server IDs, call the API for [Querying Cloud Phone Servers](#).

- Example

For details, see [Pushing a Shared Application](#) in the *Cloud Phone Host API Reference*.

## Servers of Storage 1.0

Push the .tar package to the server. That is, push **apk/\${package\_name}\_\${version\_name}.tar** to the shared storage of **server\_id1** and **server\_id2**.

- Example API

```
POST https://{CPH Endpoint}/v1/{project_id}/cloud-phone/phones/share-files
```

Header:

Content-Type: application/json

X-Auth-Token: \${token}

Body:

```
{
  "bucket_name": "${bucket_name}",
  "object_path": "apk/${package_name}_${version_name}.tar",
  "server_ids": [
    "${server_id1}",
    "${server_id2}"
  ]
}
```

Parameter descriptions:

- Obtain the values of parameters such as **CPH Endpoint**, **project\_id**, **token**, **bucket\_name**, and **object\_path** by referring to [Installing Applications on a Cloud Phone](#).
- **object\_path** indicates the path to which the .tar package is uploaded.
- **package\_name** and **version\_name** indicate the package name and version number of the current application.

#### NOTE

**apk** is any existing folder. In **apk/\${package\_name}\_\${version\_name}.tar**, **package\_name** and **version\_name** need to be modified as required.

- **server\_ids** indicates the ID list of servers where the application is deployed. You can enter multiple server IDs. To obtain the server IDs, you can call the API for [Querying Cloud Phone Servers](#).
- Example  
For details, see [Pushing Shared Storage Files](#) in the *Cloud Phone Host API Reference*.
- Follow-up procedure  
Reset all cloud phones in batches by referring to [Resetting Cloud Phones](#) in the *Cloud Phone Host API Reference*.

## 1.4.5 Updating the Version of an Application

### Servers of Storage 2.0 (Recommended)

Uninstall an application from your cloud phone.

Push the latest application.

Run the **appctrl start** command to start the application of the latest version.

### Servers of Storage 1.0

To update the version of an application, delete the application of the earlier version from the server and deploy the application of the new version.

### Deleting the Application of an Earlier Version

- Example API  
POST [https://{CPH Endpoint}/v1/{project\\_id}/cloud-phone/phones/share-files](#)  
Header:  
Content-Type: application/json  
X-Auth-Token: \${token}  
Body:  

```
{
  "file_paths": [
    "/data/app/${package_name}-1",
    "/data/app/${package_name}-2",
    "/data/data/${package_name}",
    "/data/media/0/Android/data/${package_name}"
  ],
  "server_ids": [
    "${server_id1}",
    "${server_id2}"
  ]
}
```

Delete the following files from the shared storage on servers **\${server\_id1}** and **\${server\_id2}**:

**/data/app/\${package\_name}-1**

**/data/app/\${package\_name}-2**

**/data/data/\${package\_name}**

**/data/media/0/Android/data/\${package\_name}**

Parameter descriptions:

- Obtain the values of parameters such as **CPH Endpoint**, **project\_id**, and **\${token}** by referring to [Installing Applications on a Cloud Phone](#).



- The content of **file\_paths** is the same as that of **include\_files** in [Generating the TAR Package of the Application and Pushing It to the OBS Bucket](#). **package\_name** indicates the package name of the current application.
- **server\_ids** indicates IDs of servers where the application is deployed. You can enter multiple server IDs. To obtain the server IDs, call the API for [Querying Cloud Phone Servers](#).
- Example  
For details, see [Deleting a Shared Storage File](#) in the *Cloud Phone Host API Reference*.

## Deploying the Application of the New Version

See [Deploying Applications](#).

# 2 Best Practices of Cloud Phone Application Sharing

---

## 2.1 Overview of Application Sharing

Both application sharing and shared storage enable batch installation of applications on cloud phones. However, compared with shared storage, application sharing saves more storage space.

In shared storage, the TAR package pushed by you is stored in the shared directory and then copied to cloud phones. In application sharing, the TAR package is stored in the shared directory and then directly mapped to cloud phones, so the TAR package does not occupy the space of the cloud phones.

## 2.2 Usage Guide to Application Sharing

### 2.2.1 Purchasing a Cloud Phone Server That Supports Application Sharing

All specifications support application sharing. You only need to select the required shared space and purchase it.

Note that if you want to use application sharing, set the size of the shared storage to a value greater than 0. The value **0** indicates that the server does not have shared application space.

Figure 2-1 Shared storage

Instance Specifications

Flavor	vCPUs   ROM   RAM	Screen Resolution	Quantity <sup>?</sup>
<input type="radio"/> rc2.pro_max	20 vCPUs   32 GB	1920x1080	20
<input type="radio"/> rs2.max	8 vCPUs   12 GB	1280x720	84
<input type="radio"/> rs2.pro	5 vCPUs   10 GB	1280x720	100
<input type="radio"/> rs2.plus	4 vCPUs   8 GB	1280x720	124
<input checked="" type="radio"/> rc2.max	16 vCPUs   24 GB	1920x1080	40
<input type="radio"/> rc2.pro	8 vCPUs   16 GB	1280x720	60
<input type="radio"/> rc2.plus	8 vCPUs   12 GB	1280x720	80
<input type="radio"/> rc2.se	5 vCPUs   10 GB	1280x720	100

Current phone specifications **Cloud phone | 16 vCPUs | 24 GB | rc2.max | 1920x1080**

Current server specifications **128 cores | 512 GB | physical.kg1.4xlarge.cp**

Phone Storage      
Phone storage (used to store the phone OS and data)

Shared Storage      
If this value is not 0, the shared storage can be used by all cloud phones on the server.

## 2.2.2 Preparing the TAR Package of an Application

### Querying Cloud Phones

The cloud phone list is sorted by creation time in descending order. You can specify **offset** and **limit**. If no cloud phone exists, an empty list is returned.

For details, see [Querying Cloud Phones](#) in the *Cloud Phone Host API Reference*.

### Installing Applications on a Cloud Phone

Install an APK on a cloud phone. The system downloads the specified APK file and installs it on the cloud phone.

A single APK application or multiple APK applications can be installed.

You can run the **install** command to install a single APK. Only one APK can be installed at a time.

You can run the **install-multiple** command to install multiple APKs (a single APK is split into multiple APKs). Only APKs of the same application can be installed at a time.

For details, see [Installing the APK](#) in the *Cloud Phone Host API Reference*.

## Generating the TAR Package of an Application and Pushing It to the OBS Bucket

Prerequisites

- The required application has been installed on the cloud phone.
- The OBS bucket policies have been configured based on [Delegating CPH to Operate OBS Buckets](#).

Call an API to generate an application version TAR package and push it to the OBS bucket. For details about the API example, see [Generating the TAR Package of the Application and Pushing It to the OBS Bucket](#).

## 2.2.3 Pushing the TAR Package of an Application to a Cloud Phone Server

When pushing or updating an application package for the first time, you need to call the API for pushing shared applications to push the TAR package of the application in the OBS bucket to the cloud phone server.

Example of the **curl** command

```
curl -i -k -X POST "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/share-apps" -H "Content-Type: application/json" -H "X-Auth-Token: $token" -d '{
  "package_name": "com.miniteck.miniworld",
  "bucket_name": "your-bucket-name",
  "object_path": "your/dir/miniworld.tar",
  "pre_install_app": 1,
  "server_ids": ["1678567b8bab40f93711234cb8","1234567b8bab40ffb711234cb"]
}'
```

Parameter descriptions:

- **bucket\_name** and **object\_path** are the same as those in [3](#).
- **server\_ids** indicates IDs of servers that receive the file pushed. If multiple server IDs are specified, the application can be installed on cloud phones on multiple servers.
- If **pre\_install\_app** is set to **1**, applications are preinstalled. If **pre\_install\_app** is set to **0**, applications are not preinstalled. After a cloud phone is restarted or reset, if pre-installation is not enabled, the shared application will be "lost". In this case, run **appctrl install** or **appctrl start** again to use the application. If pre-installation is enabled, you can directly use the application.
- The push task created by this API is executed asynchronously. You need to invoke the [Querying the Task Execution Status List](#) API to check whether the task is successfully executed.

For details about this API, see [Pushing a Shared Application](#).

## 2.2.4 Using the appctrl Command to Manage Shared Applications

You can run **appctrl** commands on a cloud phone to manage shared applications on the cloud phone.

You can

- Use ADB to connect to the cloud phone and run the **appctrl** commands. For details, see [ADB \(Recommended\)](#).
- Call an API to run the **appctrl** commands. For details, see [Running the Asynchronous ADB shell Commands](#).

## Running the `appctrl start` Command to Start an Application

Scenario: The `appctrl start` command is used to install an application on a cloud phone and start the application.

Prerequisites: The TAR package of the application has been pushed to the cloud phone server.

Usage guide: `appctrl start {package name} {launch_activity}`

Example: Start Subway Surfers.

```
appctrl start com.kiloo.subwaysurf  
com.idsky.android.impl.ui.IdskySplashActivity
```

You'd better transfer the startup activity name of the application. If the name cannot be obtained, you can transfer only the startup package name and let `appctrl` to obtain the startup item. That is, run the `appctrl start package_name` command to start the application.

```
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ # appctrl start com.kiloo.subwaysurf com.idsky.android.impl.ui.IdskySplashActivity  
Starting: Intent { cmp=com.kiloo.subwaysurf/com.idsky.android.impl.ui.IdskySplashActivity }  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #
```

## Running the `appctrl uninstall` Command to Uninstall an Application

Scenario: When an application is no longer used on a cloud phone, you can run `appctrl uninstall` to uninstall it from the cloud phone. You are advised to uninstall an application each time you finish using it. This is to ensure data security and use the latest application version in the future.

Prerequisites: The application has been installed on the cloud phone by running `appctrl install` or `appctrl start`.

Usage guide: Run the `appctrl uninstall {package name}` command on the cloud phone.

For example, uninstall Subway Surfers.

```
appctrl uninstall com.kiloo.subwaysurf
```

```
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ # appctrl uninstall com.kiloo.subwaysurf  
Success  
Broadcasting: Intent { act=com.huawei.action.CPH_REQUEST_GC }  
Broadcast completed: result=0  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #
```

## Running the `appctrl clear` Command to Clear Application Data

Scenario: When cloud phones are allocated to different users, run the `appctrl clear` command to clear all non-preinstalled applications on the cloud phones

each time the cloud phones are restarted or before the cloud phones are allocated to new users.

Prerequisites: The cloud phone has been restarted or reset and is ready to be allocated to a new user.

Usage guide: Run **appctrl clear** on the cloud phone.

```
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ # appctrl clear  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #  
HWSTF:/ #
```

## Updating the Version of an Application

Prerequisites: The TAR package of the application of the new version has been prepared. The application of the earlier version has been uninstalled from the cloud phone.

Operation guide

1. Push the TAR package of the latest version to the shared storage.
2. Run the **appctrl start** or **appctrl install** command, and the new version will be automatically installed.

### 2.2.5 Deleting an Application

Scenario: If an application is no longer used and needs to be taken offline, you can delete it.

Prerequisites: The application has been uninstalled from all cloud phones of the cloud phone server.

Usage guide: Invoke the DeleteShareApps API ([https://support.huaweicloud.com/intl/en-us/api-cph/cph\\_api\\_0547.html](https://support.huaweicloud.com/intl/en-us/api-cph/cph_api_0547.html)) to delete the application.

Example of the **curl** command

```
curl -i -k -X DELETE "https://{CPH Endpoint}/v1/{projectId}/cloud-phone/phones/share-apps" -H "Content-Type: application/json" -H "X-Auth-Token: $token" -d '{  
  "package_name": "com.miniteck.miniworld",  
  "server_ids": ["1678567b8bab40f93711234cb8","1234567b8bab40ffb711234cb"]  
}'
```

**server\_ids** indicates IDs of servers that receive file push. You can specify multiple server IDs.

The push task created by this API is executed asynchronously. You need to invoke the [Querying the Task Execution Status List](#) API to check whether the task is successfully executed.

## 2.2.6 Presetting Configuration Files by Sharing Applications

Scenario: One or more files, such as configuration files or scripts, need to be placed on cloud phones in batches.

Procedure

1. Compress the files into the **com.cph.config.tar** package based on the required directory structure.

For example, the following package contains a script file and a text file.

```
HWSTF:/data/local #  
HWSTF:/data/local #  
HWSTF:/data/local # tar -tvf ./com.cph.config.tar  
-rw-rw-rw- root/root      0 2022-10-24 20:36:15 data/local/tmp/test1.sh  
-rw-rw-rw- root/root      0 2022-10-24 20:37:03 data/local/user_1024/config.txt  
HWSTF:/data/local #
```

- a. Push the TAR package to the OBS bucket by referring to [Preparing the TAR Package of an Application](#).
- b. Set **pre\_install\_app** to **1** by referring to [Pushing the TAR Package of an Application to a Cloud Phone Server](#).
- c. Restart the cloud phone where you want to place the file.

After the operations are complete, the files in the TAR package are copied to all restarted cloud phones on the target cloud phone server.

## 2.3 appctrl Commands

### Command Overview

**appctrl** is a new CPH command line. It supports quick start, installation, uninstallation, and clearance of applications only through the PushShareApps API.

### Example Commands

```
usage: appctrl [start] [install] [uninstall] [clear]  
appctrl start package_name package.activity_name  
appctrl install package_name  
appctrl uninstall package_name  
appctrl clear
```

### Descriptions of Parameters in the Commands

**start**: Start the application that has been successfully pushed based on the value of *package\_name* and *package.activity\_name*.

**install**: Install the application that has been successfully pushed to the shared storage based on the value of *package name*.

**uninstall**: Uninstall an application based on the value of *package name*. Only applications started by running the **appctrl start** or **install** command are supported.

**clear**: Uninstall or clear all non-preinstalled applications that are started or installed by running the **appctrl start** command.

# 3 Installing an Application on Cloud Phones in Batches

After you install an application on a cloud phone, you can share and install the application on other cloud phones by calling an API.

 **NOTE**





The cloud phone that has the application installed is regarded as a seed cloud phone.

## Restrictions and Limitations

- This solution applies only to cloud phones whose specifications name does not contain **qemu**.

**Figure 3-1** Specifications

Instance Specifications

Flavor	vCPUs   ROM   RAM	Screen Resolution	Quantity 	EIP/VIP 
 rx1.cg.c15.d30.e1v1.a200	4 vCPUs   8 GB   30 GB	1280x720	15	1/1
 rx1.cg.c15.d30.e1v1	4 vCPUs   8 GB   30 GB	1280x720	15	1/1

- The seed cloud phone must be a cloud phone that has not been operated on. Otherwise, [reset it](#).

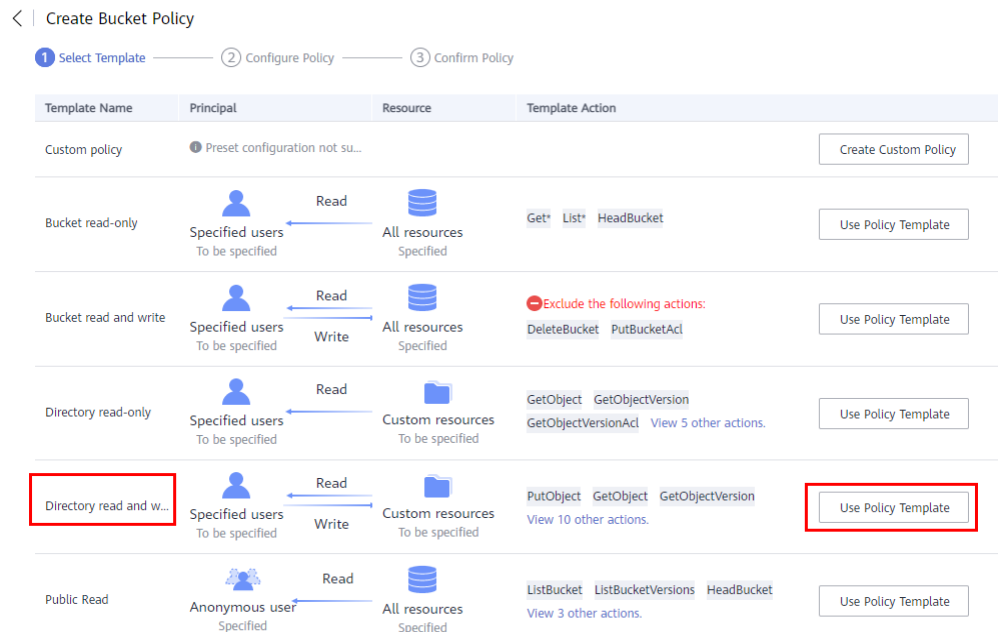
## Procedure

1. Log in to the management console and choose **Storage > Object Storage Service**. Create an OBS bucket and configure policies for accessing the bucket. For details, see [Managing Cloud Phones in Batches](#).

To successfully upload files to the bucket, select **Directory read and write** when configuring the bucket policy.



**Figure 3-2** Create Bucket Policy



2. Connect to the seed cloud phone in ADB mode and install the required application on it.  
For details, see [How Do I Install Applications on a Cloud Phone?](#)
3. Call an API to export data of the seed cloud phone, pack the data, and upload the data package to the OBS bucket created in 1.

**curl** command

```
curl -i -k -X POST "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/batch-storage" -H
"Content-Type: application/json" -H "X-Auth-Token: $token" -d '
{
  "storage_infos": [
    {
      "phone_id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
      "include_files": [
        "/data/app/${package-name}-1",
        "/data/data/${package-name}",
        "/data/media/0/Android/data/${package-name}"
      ],
      "bucket_name": "${bucket_name}",
      "object_path": "${your_dir}/${package-name}.tar"
    }
  ]
}'
```

Parameter descriptions:

- **phone\_id**: ID of the seed cloud phone
- **bucket\_name**: name of the OBS bucket for storing the data exported
- **object\_path**: OBS path for storing the data exported

 **CAUTION**

If you want to pack all data of applications on the seed cloud phone, the following three paths must be included:

- **/data/app/\${package-name}-1**  
**\${package-name}** may not be followed by **-1**. Configure this parameter as required.
- **/data/data/\${package-name}**
- **/data/media/0/Android/data/\${package-name}**

**Example**

```
curl -i -k -X POST "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/batch-storage" -H
"Content-Type: application/json" -H "X-Auth-Token: $token" -d '
{
  "storage_infos": [
    {
      "phone_id": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
      "include_files": [
        "/data/app/com.taptap-1",
        "/data/data/com.taptap",
        "/data/media/0/Android/data/com.taptap"
      ],
      "bucket_name": "your-bucket-name",
      "object_path": "your/dir/taptap.tar"
    }
  ]
}'
```

4. View the OBS bucket created in **1** and check whether the file packed and uploaded in **3** is successfully uploaded. If yes, go to the next step.
5. Differentiate storage 1.0 and storage 2.0 servers in this step.

- Servers of storage 2.0 (recommended)

Call an API to push files stored in the OBS bucket to the server.

**Example of the curl command**

```
curl -i -k -X POST "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/share-apps" -H
"Content-Type: application/json" -H "X-Auth-Token: $token" -d '
{
  "package_name": "com.miniteck.miniworld",
  "bucket_name": "your-bucket-name",
  "object_path": "your/dir/miniworld.tar",
  "server_ids": ["1678567b8bab40f93711234cb8","1234567b8bab40ffb711234cb"]
}'
```

**Parameter descriptions:**

- **bucket\_name** and **object\_path** are the same as those in **3**.
- **server\_ids**: IDs of servers that receive the file pushed. If multiple server IDs are specified, the application can be installed on cloud phones on multiple servers.

 **NOTE**

For details about this API, see [Pushing a Shared Application](#).

- Servers of storage 1.0

Call an API to push files stored in the OBS bucket to the shared storage directory of the server.

### Example of the **curl** command

```
curl -i -k -X POST "https://${CPH Endpoint}/v1/${projectId}/cloud-phone/phones/share-files" -H  
"Content-Type: application/json" -H "X-Auth-Token: $token" -d '  
{  
  "bucket_name": "your-bucket-name",  
  "object_path": "your/dir/taptap.tar",  
  "server_ids": ["1678567b8bab40f93711234cb8","1234567b8bab40ffb711234cb"]  
}'
```

#### Parameter descriptions:

- **bucket\_name** and **object\_path** are the same as those in [3](#).
- **server\_ids**: IDs of servers that receive the file pushed. If multiple server IDs are specified, the application can be installed on cloud phones on multiple servers.

#### NOTE

For details about this API, see [Pushing Shared Storage Files](#).

#### 6. Differentiate storage 1.0 and storage 2.0 servers in this step.

- Servers of storage 2.0 (recommended)

Run the **apptctl install** *package\_name* command on the cloud phone to install the application. *package\_name* corresponds to the value of **package\_name** in [5](#), for example, **com.miniteck.miniworld**.

- Servers of storage 1.0

Log in to the CPH console, click the name of the server that has received the files pushed. In the **Cloud Phones** area, select all cloud phones on which the application needs to be installed, and click **Reset**.

---

#### CAUTION

This step is a reset operation, not a restart operation.

---

## Execution Results

For servers that use storage 1.0, the applications have been installed after the cloud phones are reset.

For servers that use storage 2.0, the applications can be directly started by running **apptctl start** *package\_name package.activity\_name*.

# 4 Modifying the Cloud Phone GPS Location

---

The GPS location of a cloud phone is its longitude and latitude obtained by simulating the GPS satellite. It is expressed in decimal numbers and the unit is degree. The GPS location follows the international conventions, in which east longitude is positive, west longitude is negative, north latitude is positive, and south latitude is negative. This topic describes how to modify the GPS location of a cloud phone.

## Prerequisites

You have purchased a cloud phone server and connected to a cloud phone using ADB. For details, see [Buying a Cloud Phone Server \(Without Detailed Parameter Description\)](#).

## Procedure

Assume that the location to be modified is longitude 114.055939 degree East and latitude 22.657501 degrees North.

In the ADB installation directory of the local device, run the following command to modify the GPS location:

```
adb -s 127.0.0.1:Local idle port shell "echo 'longitude=114.055939:latitude=22.657501' > /data/gps/fifo"
```

### NOTE

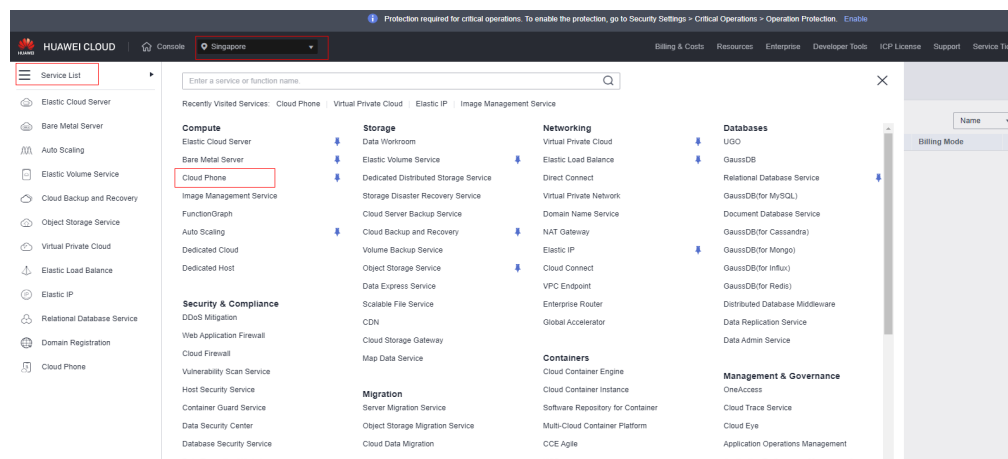
*Local idle port* is the local idle port used for establishing the SSH tunnel.

The modification takes effect immediately after the command is executed. You can use map or social software to view the modification result. For example, if you use social software to create a moment, you can view the GPS location when adding a location.

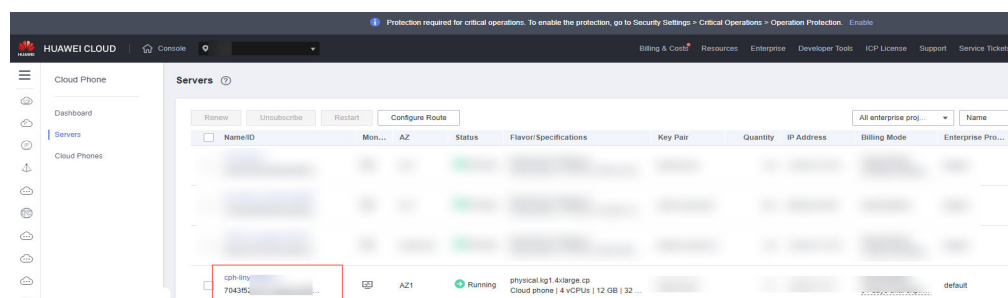
# 5 Using the Cloud Phone Camera

## Step 1: Replace the Image of a Cloud Phone

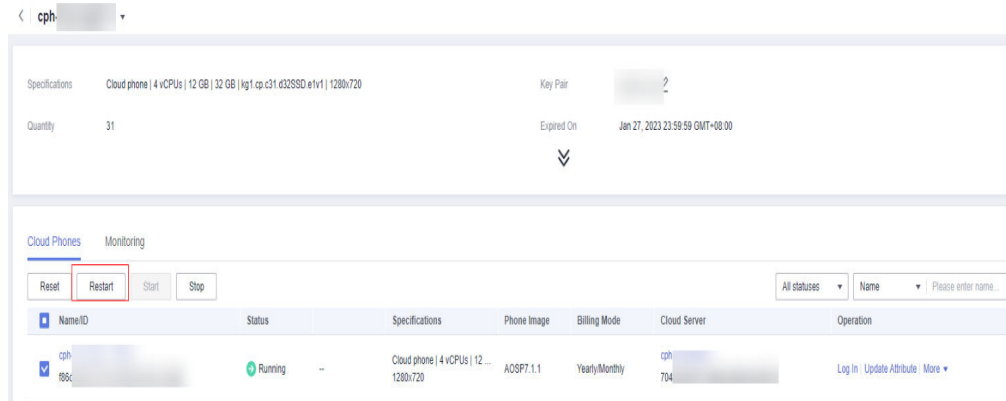
1. View [AOSP 7 Cloud Phone Image Change History](#), select an image released on or after October 9, 2020, and copy the image ID.
2. Log in to the management console, switch to the region where your resources are deployed, and choose **Compute > Cloud Phone Host**.



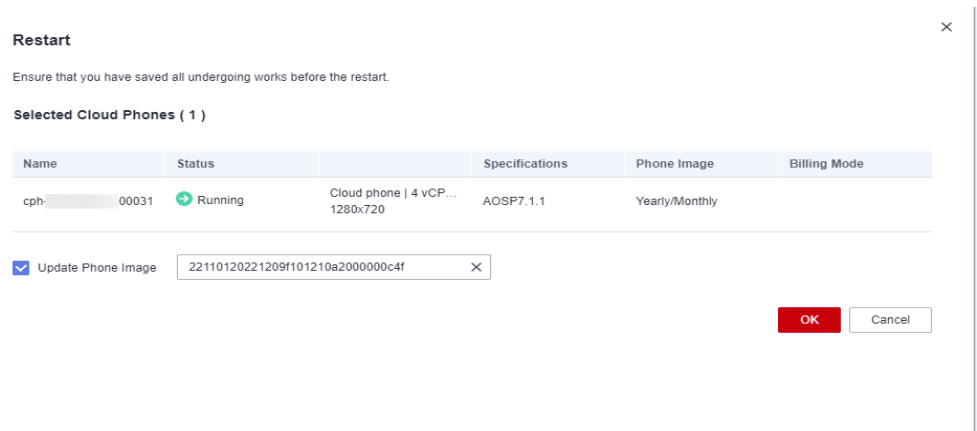
3. Choose **Servers** to view servers.



4. Click the name of a server to go to the server details page, select a cloud phone, and click **Restart**.



5. Select the **Update Phone Image** checkbox and enter the image ID selected in 1.



6. Click **OK**.

## Step 2: Upload a Picture to the Cloud Phone

Upload a picture to the `/data/local/tmp/` directory of the cloud phone in either of the following ways. The following uses `pic.jpeg` in the `/path/to/local` directory as an example.

- Method 1: Run the **adb push** command to push the picture.  
Connect to the cloud phone through ADB, and run the following commands:  
**adb push </path/to/local/pic.jpeg> /data/local/tmp/pic.jpeg**  
**adb shell chmod 644 /data/local/tmp/pic.jpeg**
- Method 2: Call the CPH API to push the picture.  
See [Pushing Files](#).

#### NOTICE

- The size of the picture to be uploaded must be 480 (width) x 640 (height). If the size is not 480 x 640, the picture may be zoomed in or out on the camera.
- Only JPEG and PNG pictures stored in the `/data/local/tmp/` path are supported.
- Picture permissions must be at least 644 (rw-r--r--).

### Step 3: Configure Cloud Phone Attributes

You can configure cloud phone attributes in either of the following ways:

Method 1: Use ADB to connect to the cloud phone and run the `adb` command.

```
adb shell setprop com.cph.cam_local_pic_path /data/local/tmp/pic.jpeg
```

If you use this method, the cloud phone attributes become invalid after the cloud phone is restarted.

Method 2: Call the CPH API to configure attributes.

Set `com.cph.cam_local_pic_path` to `/data/local/tmp/pic.jpeg` for the cloud phone. For details, see [Updating Cloud Phone Attributes](#). The attributes will always be valid even after the cloud phone is restarted.

### Step 4: Test the Camera

Install an application that needs to use the camera, open the application, and check whether the picture you uploaded is displayed in the viewfinder.

#### NOTE

CPH supports only the rear camera.

# 6 Using STF to Manage Cloud Phones in Batches

---

## Scenarios

Smartphone Test Farm (STF) is an open-source web-based application used for managing and controlling mobile devices. STF uses a browser to control and manage Android devices, enabling you to use, debug, and test those devices on the cloud. This section describes how to deploy STF components on an ECS to manage cloud phones in batches.

## Restrictions and Limitations

- STF can manage about 160 cloud phones at the same time. For larger-scale access management, secondary development is required.
- STF needs a stable network to run smoothly. When the network status is poor, the operation latency of cloud phones increases significantly.

## Prerequisites

- A cloud phone server that has an EIP bound is available.
- An ECS that has an EIP bound is available.

### NOTE

The following cloud phone server and ECS specifications are only examples.

- Cloud phone server: physical.kg1.4xlarge.cp | kg1.cp.c60.d16SSD.e1v1
- ECS: general computing | s6.large.2 | 2 vCPUs | 4 GiB | Ubuntu 18.04 server 64bit (40 GB)

## Procedure

Deploy components on which STF depends on the ECS, use the ADB tool to connect to the cloud phone, and access the STF address through a browser to manage cloud phones in batches.

1. Install ADB and check the installation result.

```
sudo apt install android-tools-adb android-tools-fastboot  
adb --version
```

If **--version** is displayed, the installation is successful.



**Figure 6-1** Successful installation of ADB

```
root@ecs-stf:~# adb --version
Android Debug Bridge version 1.0.39
Version 1:8.1.0+r23-5~18.04
Installed as /usr/lib/android-sdk/platform-tools/adb
```

2. Update the yum source and install RethinkDB to store STF data.  
source /etc/lsb-release && echo "deb https://download.rethinkdb.com/repository/ubuntu-\$DISTRIB\_CODENAME \$DISTRIB\_CODENAME main" | sudo tee /etc/apt/sources.list.d/rethinkdb.list  
wget -qO- https://download.rethinkdb.com/repository/raw/pubkey.gpg | sudo apt-key add -  
sudo apt-get update  
sudo apt-get install rethinkdb  
rethinkdb -v

If **-v** is displayed, the installation is successful.

**Figure 6-2** Successful installation of RethinkDB

```
root@ecs-stf:~# rethinkdb -v
rethinkdb 2.4.1~0bionic (CLANG 6.0.0 (tags/RELEASE_600/final))
```

RethinkDB provides official support for the x86 architecture while experimental support for the Arm architecture.

3. Install ZeroMQ to transfer messages.  
sudo apt-get install libzmq3-dev

**Figure 6-3** Successful installation of ZeroMQ

```
root@ecs-stf:~# sudo apt-get install libzmq3-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnorm1 libpgm-5.2-0 libsodium23 libzmq5
The following NEW packages will be installed:
  libnorm1 libpgm-5.2-0 libsodium23 libzmq3-dev libzmq5
0 upgraded, 5 newly installed, 0 to remove and 96 not upgraded.
Need to get 1,145 kB of archives.
After this operation, 4,150 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://repo.huaweicloud.com/ubuntu bionic/universe amd64 libnorm1 amd64 1.5r6+dfsg1-6 [224 kB]
Get:2 http://repo.huaweicloud.com/ubuntu bionic/universe amd64 libpgm-5.2-0 amd64 5.2.122~dfsg-2 [157 kB]
Get:3 http://repo.huaweicloud.com/ubuntu bionic/main amd64 libsodium23 amd64 1.0.16-2 [143 kB]
Get:4 http://repo.huaweicloud.com/ubuntu bionic-updates/universe amd64 libzmq5 amd64 4.2.5-1ubuntu0.2 [221 kB]
Get:5 http://repo.huaweicloud.com/ubuntu bionic-updates/universe amd64 libzmq3-dev amd64 4.2.5-1ubuntu0.2 [400 kB]
Fetched 1,145 kB in 0s (10.3 MB/s)
Selecting previously unselected package libnorm1:amd64.
(Reading database ... 111853 files and directories currently installed.)
Preparing to unpack .../libnorm1_1.5r6+dfsg1-6_amd64.deb ...
Unpacking libnorm1:amd64 (1.5r6+dfsg1-6) ...
Selecting previously unselected package libpgm-5.2-0:amd64.
Preparing to unpack .../libpgm-5.2-0_5.2.122~dfsg-2_amd64.deb ...
Unpacking libpgm-5.2-0:amd64 (5.2.122~dfsg-2) ...
Selecting previously unselected package libsodium23:amd64.
Preparing to unpack .../libsodium23_1.0.16-2_amd64.deb ...
Unpacking libsodium23:amd64 (1.0.16-2) ...
Selecting previously unselected package libzmq5:amd64.
Preparing to unpack .../libzmq5_4.2.5-1ubuntu0.2_amd64.deb ...
Unpacking libzmq5:amd64 (4.2.5-1ubuntu0.2) ...
Selecting previously unselected package libzmq3-dev:amd64.
Preparing to unpack .../libzmq3-dev_4.2.5-1ubuntu0.2_amd64.deb ...
Unpacking libzmq3-dev:amd64 (4.2.5-1ubuntu0.2) ...
Setting up libpgm-5.2-0:amd64 (5.2.122~dfsg-2) ...
Setting up libnorm1:amd64 (1.5r6+dfsg1-6) ...
Setting up libsodium23:amd64 (1.0.16-2) ...
Setting up libzmq5:amd64 (4.2.5-1ubuntu0.2) ...
Setting up libzmq3-dev:amd64 (4.2.5-1ubuntu0.2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
```

4. Install Protocol Buffers as the data format for message transfer.  
sudo apt-get install libprotobuf-dev protobuf-compiler  
protoc --version

If **--version** is displayed, the installation is successful.

**Figure 6-4** Successful installation of Protocol Buffers

```
root@ecs-stf:~# protoc --version
libprotoc 3.0.0
```

5. Install GraphicsMagick to read, write, and operate pictures.

```
sudo apt-get install graphicsmagick
gm version
```

If **version** is displayed, the installation is successful.

**Figure 6-5** Successful installation of GraphicsMagick

```
root@ecs-stf:~# gm version
GraphicsMagick 1.3.28 2018-01-20 Q16 http://www.GraphicsMagick.org/
Copyright (C) 2002-2018 GraphicsMagick Group.
Additional copyrights and licenses apply to this software.
See http://www.GraphicsMagick.org/www/Copyright.html for details.
```

6. Install pkg-config to compile the third-party library of Nodejs.

```
sudo apt-get install pkg-config
pkg-config --version
```

If **--version** is displayed, the installation is successful.

**Figure 6-6** Successful installation of pkg-config

```
root@ecs-stf:~# pkg-config --version
0.29.1
```

7. Install yasm to compile dependent libraries of STF.

```
sudo apt-get install yasm
yasm --version
```

If **--version** is displayed, the installation is successful.

**Figure 6-7** Successful installation of yasm

```
root@ecs-stf:~# yasm --version
yasm 1.3.0
Compiled on Apr  3 2018.
Copyright (c) 2001-2014 Peter Johnson and other Yasm developers.
Run yasm --license for licensing overview and summary.
```

8. Install Nodejs to deploy the STF runtime environment.

```
##STF supports only Node.js 8.x.
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
sudo apt-get install -y nodejs
node -v
npm -v
```

If **-v** is displayed, the installation is successful.

**Figure 6-8** Successful installation of node and npm

```
root@ecs-stf:~# node -v
v8.17.0
root@ecs-stf:~# npm -v
6.13.4
```

9. Install STF.

```
sudo npm install -g cnpm --registry=https://registry.npm.taobao.org
sudo cnpm install -g stf
stf -V
```

If **-V** is displayed, the installation is successful.

**Figure 6-9** Successful installation of STF

```
root@ecs-stf:~# stf -V
3.4.1
```

10. Check whether the environment on which STF depends is available.

```
stf doctor
```

If the version of each component is displayed in the command output, the environment is available.

**Figure 6-10** Checking the STF startup environment

```
root@ecs-stf:~# stf doctor
2021-08-18T01:47:35.484Z INF/cli:doctor 20873 [*] OS Arch: x64
2021-08-18T01:47:35.486Z INF/cli:doctor 20873 [*] OS Platform: linux
2021-08-18T01:47:35.486Z INF/cli:doctor 20873 [*] OS Platform: 4.15.0-136-generic
2021-08-18T01:47:35.486Z INF/cli:doctor 20873 [*] Using Node 8.17.0
2021-08-18T01:47:35.495Z INF/cli:doctor 20873 [*] Using ZeroMQ 4.2.5
2021-08-18T01:47:35.512Z INF/cli:doctor 20873 [*] Using RethinkDB 2.4.1~0bionic
2021-08-18T01:47:35.512Z INF/cli:doctor 20873 [*] Using GraphicsMagick 1.3.28
2021-08-18T01:47:35.512Z INF/cli:doctor 20873 [*] Using ProtoBuf 3.0.0
2021-08-18T01:47:35.513Z INF/cli:doctor 20873 [*] Using ADB 1.0.39
```

11. Use ADB to connect to the cloud phone. For details, see [ADB \(Internet\)](#).

12. Start RethinkDB.

```
rethinkdb
```

If information similar to that in the following is displayed, RethinkDB is started successfully.

**Figure 6-11** Starting RethinkDB

```
root@ecs-stf:~# rethinkdb
Recursively removing directory /root/rethinkdb_data/tmp
Initializing directory /root/rethinkdb_data
Running rethinkdb 2.4.1~0bionic (CLANG 6.0.0 (tags/RELEASE_600/final))...
Running on Linux 4.15.0-136-generic x86_64
Loading data from directory /root/rethinkdb_data
Listening for intracluster connections on port 29015
Listening for client driver connections on port 28015
Listening for administrative HTTP connections on port 8080
Listening on cluster addresses: 127.0.0.1, ::1
Listening on driver addresses: 127.0.0.1, ::1
Listening on http addresses: 127.0.0.1, ::1
To fully expose RethinkDB on the network, bind to all addresses by running rethinkdb with the '--bind all' command line option.
Server ready, "ecs_stf_qnb" afd469a8-a055-43c0-bbf1-a1334d1de3c1
```

13. Start STF in local mode and access STF using a browser.

```
## Set EIP to the EIP bound to the ECS.
stf local --public-ip {EIP} --allow-remote
## Access method
http://{EIP}:7100/
```

Figure 6-12 Entering the default username and password of STF



Figure 6-13 Cloud phones

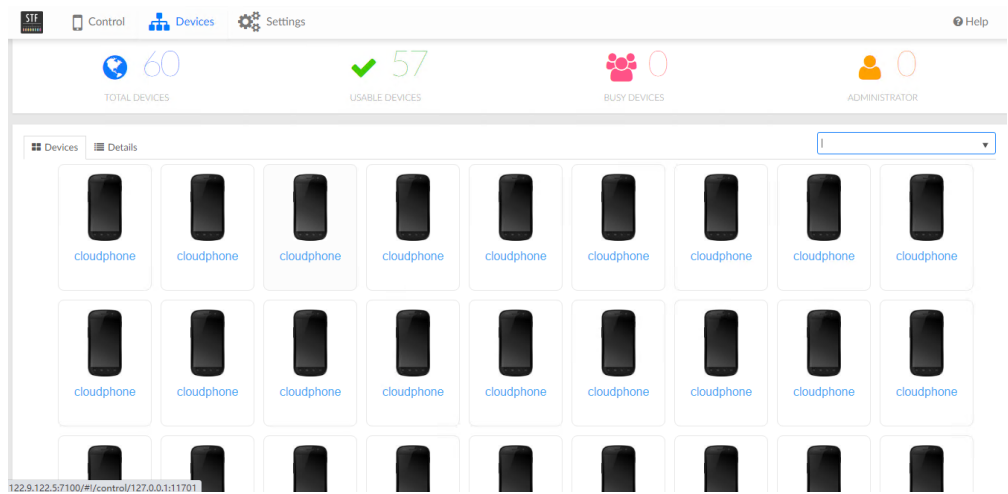
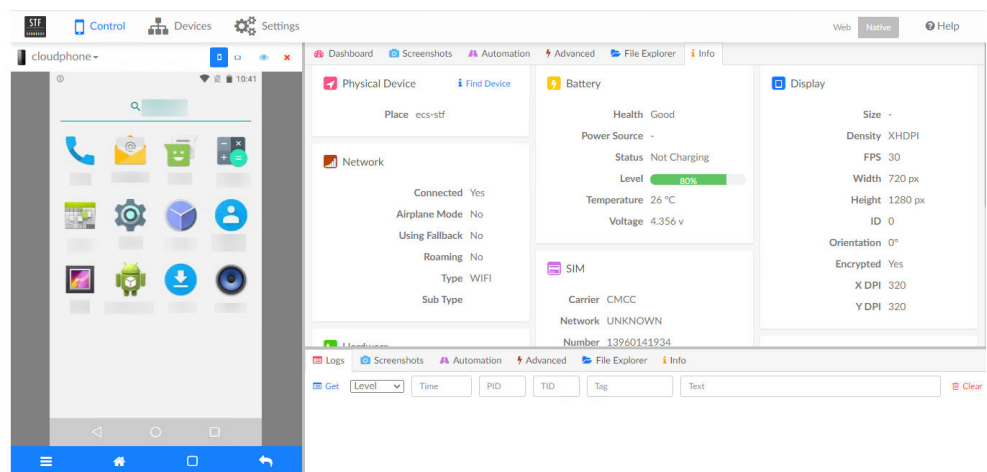
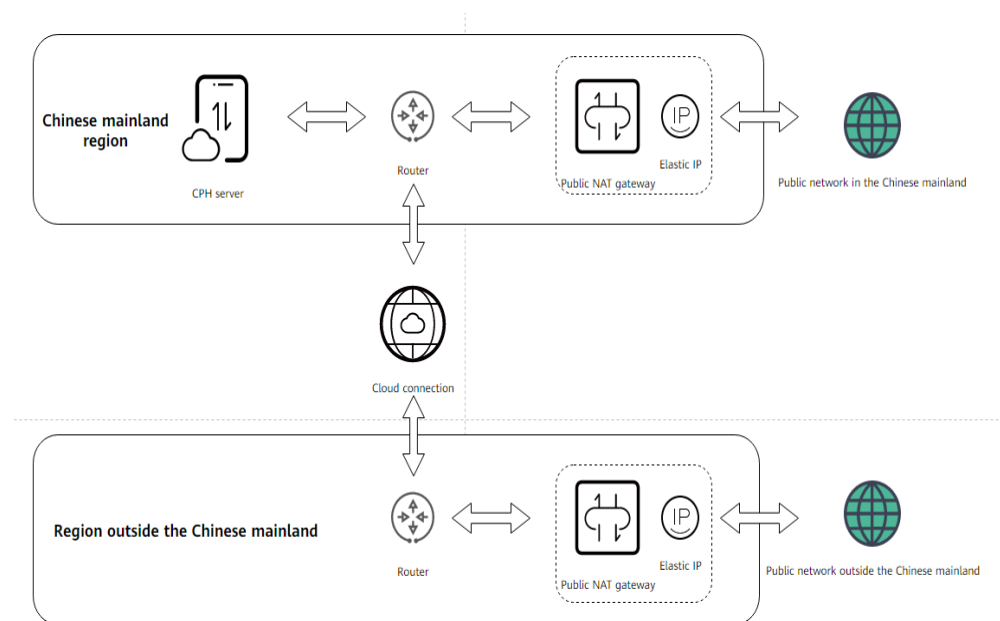


Figure 6-14 Cloud phone Control screen



# 7 Allowing a Cloud Phone Server to Access a Public Network Outside the Chinese Mainland

The following figure shows how to allow a cloud phone server to access a public network outside the Chinese mainland.



## Restrictions and Limitations

- This practice applies only to cloud phone servers without EIPs. That is, the number of EIPs in the cloud phone specifications must be 0. The number of virtual IP addresses is not limited.

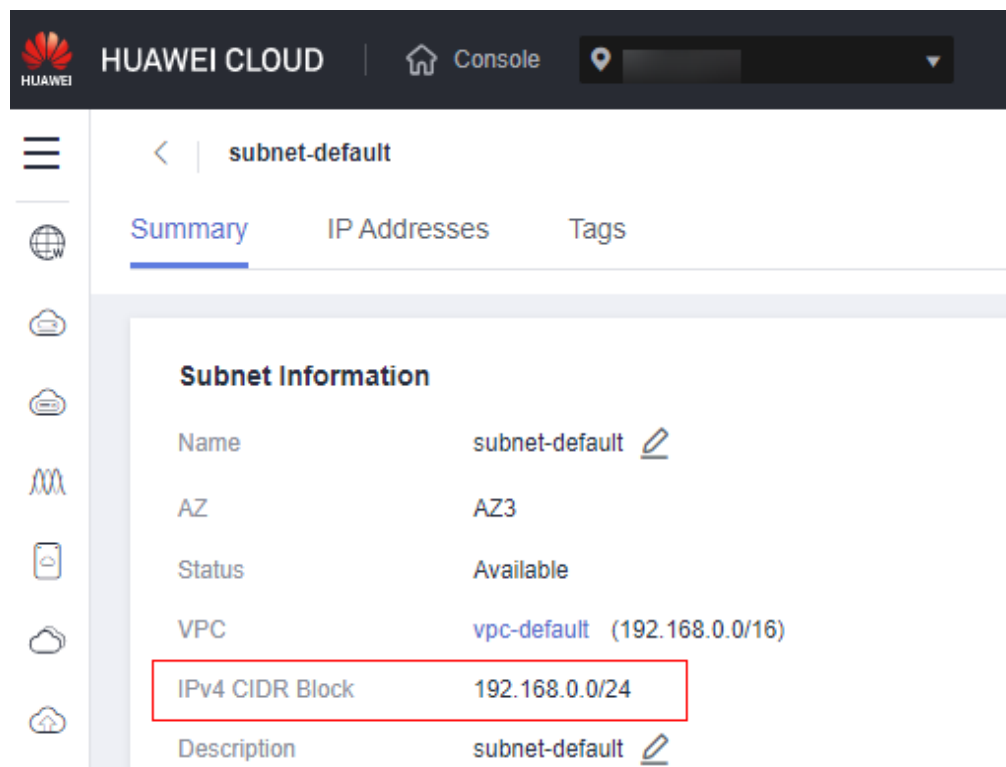
Flavor	vCPUs   ROM   RAM	Screen Resolution	Quantity ?	EIP/VIP ?
<input type="radio"/> rx1.cp.c15.d46.e1v1	4 vCPUs   16 GB   46 GB	1280x720	15	1/1
<input checked="" type="radio"/> rx1.cp.c60.d10.e0v60	2 vCPUs   3.5 GB   10 GB	1280x720	60	0/60

## Procedure

1. Apply for a cross-border permit. For details, see [Cross-Border Permits](#). Proceed with the next step only if you have obtained the cross-border permit.
2. Log in to the CPH console, choose **Servers**, and click the cloud phone server whose traffic is to be diverted. On the server details page, locate **Subnet**.



3. Click the subnet name. On the subnet details page, find **IPv4 CIDR Block** and record the subnet CIDR block of the cloud phone server, for example, **192.168.0.0/24**.



4. Choose **Networking > NAT Gateway**.
5. Select the region where you want your cloud phone server to access the public network outside the Chinese mainland, for example, **CN-Hong Kong**.
6. Purchase an EIP and a public NAT gateway. Add an SNAT rule. Add a route with 0.0.0.0/0 as the destination and the public NAT gateway as the next hop. For details, see [Configuring SNAT Rules to Enable Servers to Access the Internet](#).

When you add the SNAT rule, select **Direct Connect/Cloud Connect** for **Scenario** and enter the subnet CIDR block of the cloud phone server recorded in **3**.

### Add SNAT Rule

**i**

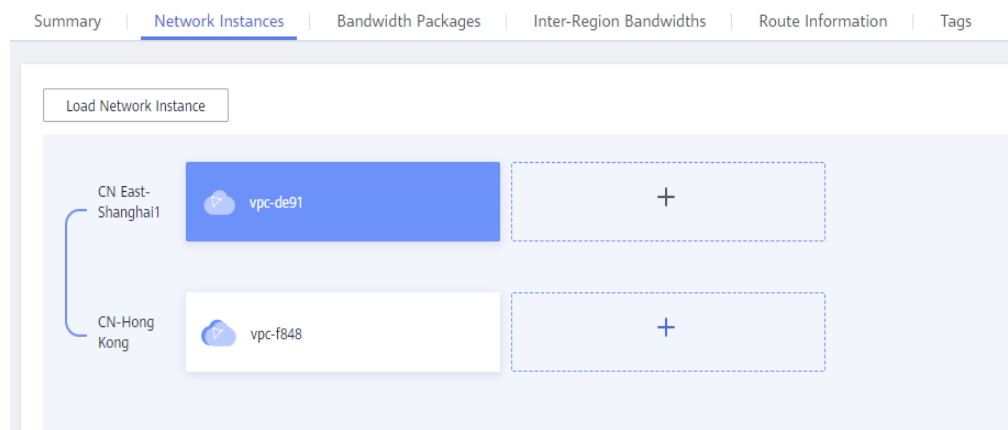
- If both an EIP and a NAT gateway are configured for a server, data will be forwarded through the EIP. [View restrictions](#)
- It is not recommended that an SNAT rule and a DNAT rule share the same EIP because there may be service conflicts.
- An SNAT rule cannot share an EIP with a DNAT rule with Port Type set to All ports.

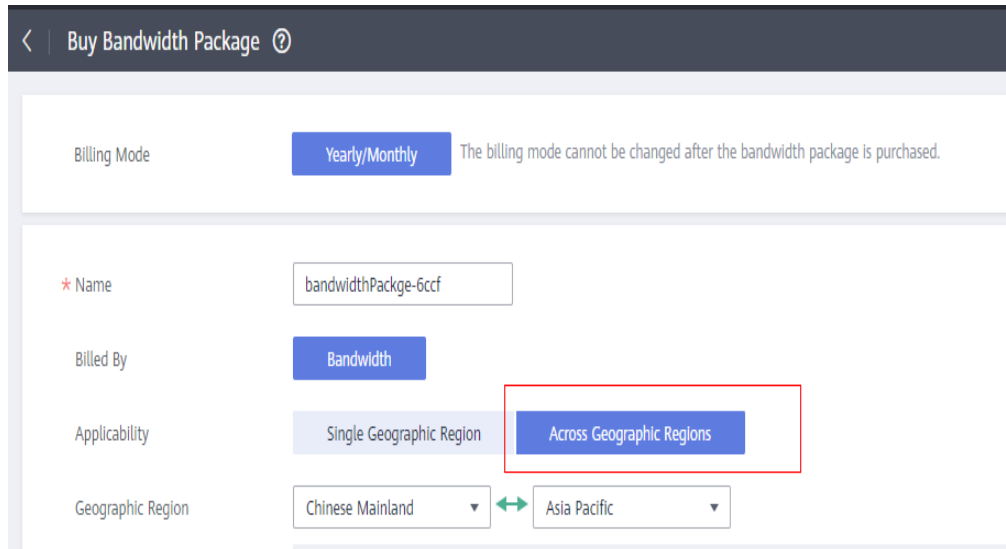
Public NAT Gateway Name: test

\* Scenario: VPC Direct Connect/Cloud Connect

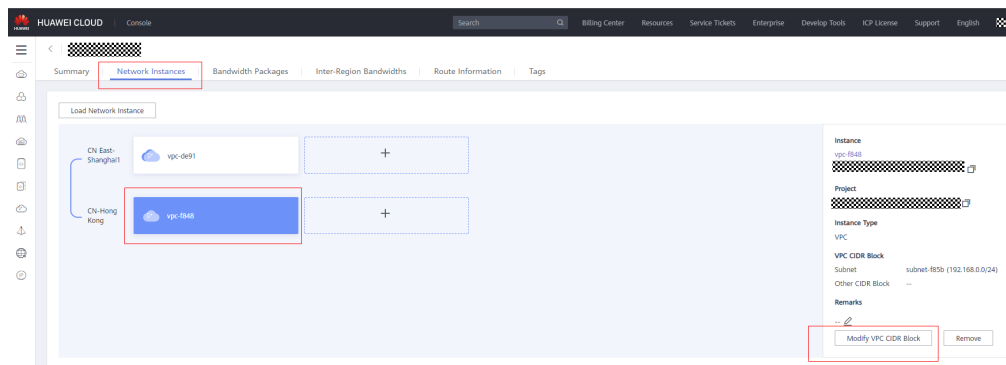
192 . 168 . 0 . 0 / 24 ?

- Choose **Networking > Cloud Connect**.
- Create a cloud connection, and load the VPC where the cloud phone server is deployed and the VPC where the public NAT gateway is deployed to the cloud connection. When loading the VPCs, select the subnets where the cloud phone server and the public NAT gateway purchased in **6** are deployed. For details, see [Connecting VPCs in the Same Account](#). Purchase a bandwidth package for communications between geographic regions, in this example, from the Chinese mainland to Asia Pacific. Bind the bandwidth package to the cloud connection.



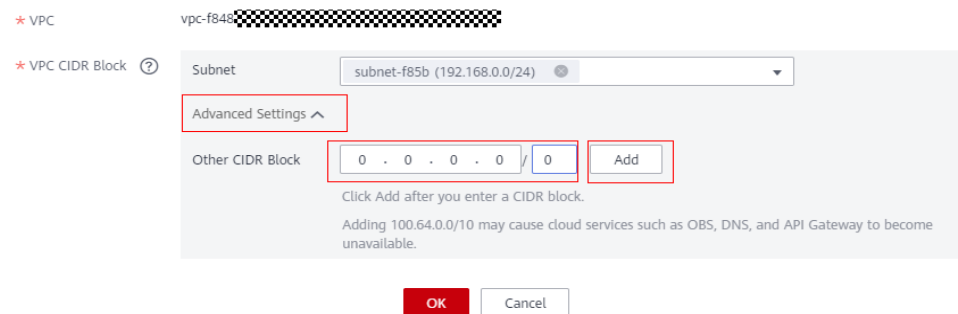


9. On the **Network Instances** page of the cloud connection, select the loaded VPC in the CN-Hong Kong region and click **Modify VPC CIDR Block**.



In the displayed **Modify VPC CIDR Block** dialog box, click **Advanced Settings**, enter **0.0.0.0/0**, click **Add**, and click **OK**.

#### Modify VPC CIDR Block



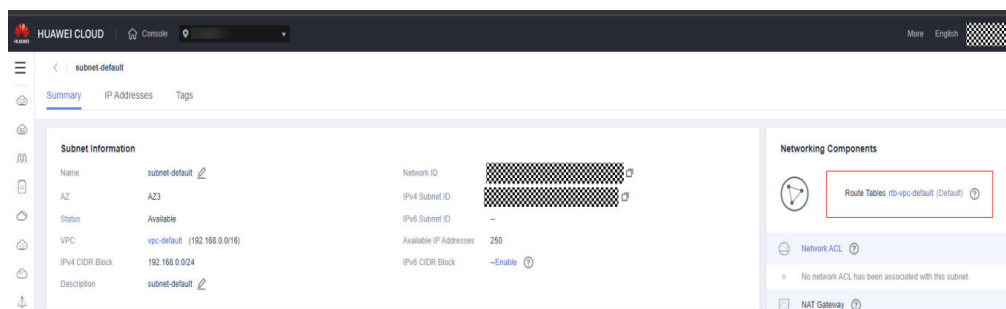
Now your cloud phone server can access the public network outside the Chinese mainland. All traffic from this server is diverted to the cloud connection and then to the public NAT gateway in the CN-Hong Kong region, so this server can use the EIP bound to the public NAT gateway to access the public network outside the Chinese mainland. To verify the above configurations, you can use your cloud phone server to access the public network outside the Chinese mainland.



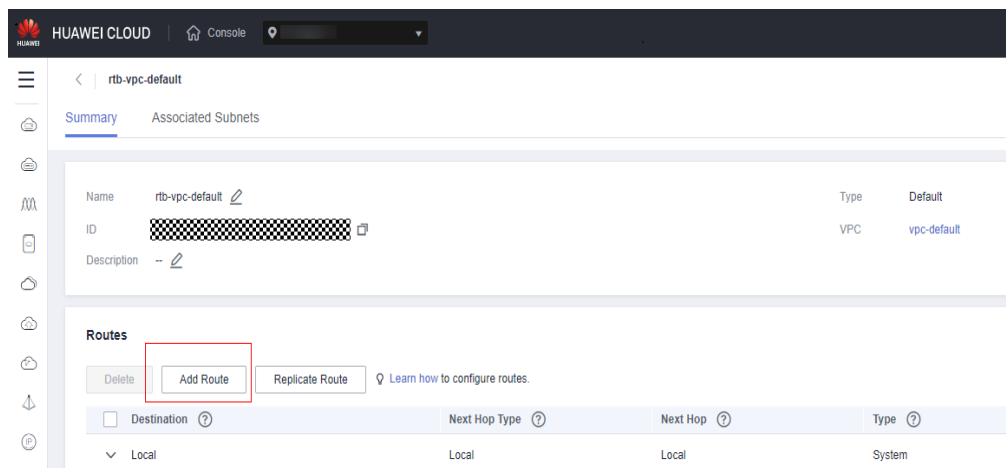
If you do not need your cloud phone server to access a public network inside the Chinese mainland, skip the following steps.

### (Optional) Allowing a Cloud Phone Server to Access a Public Network Inside the Chinese Mainland

1. Purchase an EIP and a public NAT gateway in the region where the cloud phone server is deployed. Add an SNAT rule. For details, see 6. You do not need to add a route with 0.0.0.0/0 as the destination and the public NAT gateway as the next hop.
2. Repeat 2 and 3 to view the subnet of the cloud phone server and the route table of the subnet.



3. Click the name of the route table. On the displayed page, click **Add Route**.



4. In the dialog box that is displayed, set **Destination** to the IP address or CIDR block where the cloud phone server traffic is to be diverted, set **Next Hop Type** to **NAT gateway**, set **Next Hop** to the public NAT gateway purchased in 1, and click **OK**.

#### Add Route

Route Table rtb-vpc-default(Default)

Destination ?	Next Hop Type ?	Next Hop ?	Description
114.114.114.114/32	NAT gateway	test(114.114.114.114/32)	

+ Add Route

OK
Cancel

5. Repeat [4](#) if traffic to your cloud phone server needs to be diverted to other IP addresses or CIDR blocks.

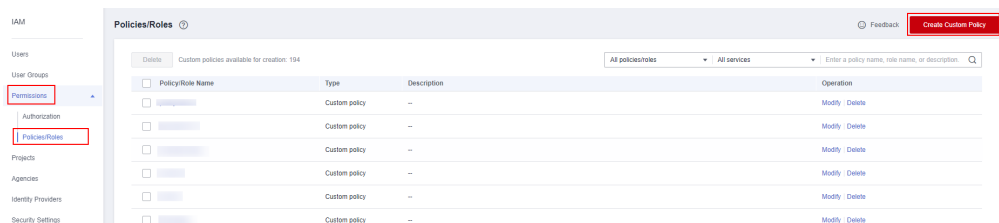
Now when you access the IP address configured with traffic diversion from the cloud phone server, the traffic will be diverted from the EIP configured for the NAT gateway purchased in the Chinese mainland region, and other traffic will be diverted to the cloud connection from the EIP configured for the NAT gateway purchased outside the Chinese mainland region.

# 8 Delegating CPH to Operate OBS Buckets

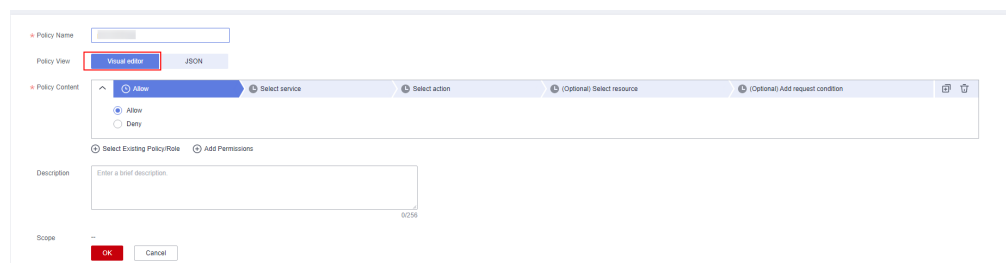
The administrator can create custom policies on IAM, assign custom policies to OBS buckets for refined access control, and delegate the policies to CPH to back up and restore cloud phone data and install applications.

## Procedure

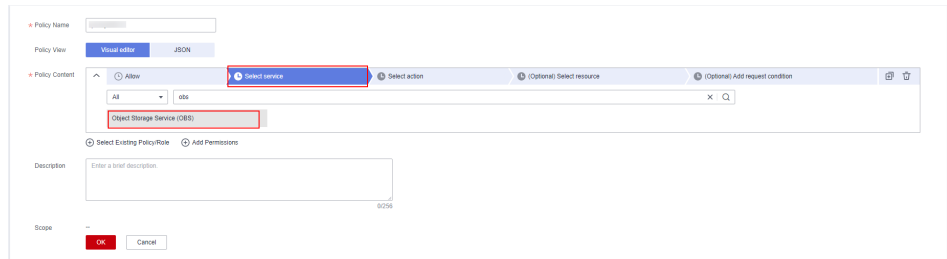
- Create a custom policy.
  1. Log in to the management console.
  2. On the **Service List** page, choose **Management & Governance > Identity and Access Management**.
  3. On the IAM console, choose **Permissions > Policies/Roles** from the navigation pane, and click **Create Custom Policy** in the upper right corner.



4. Enter a policy name.  
Select **Visual editor** for **Policy View**.



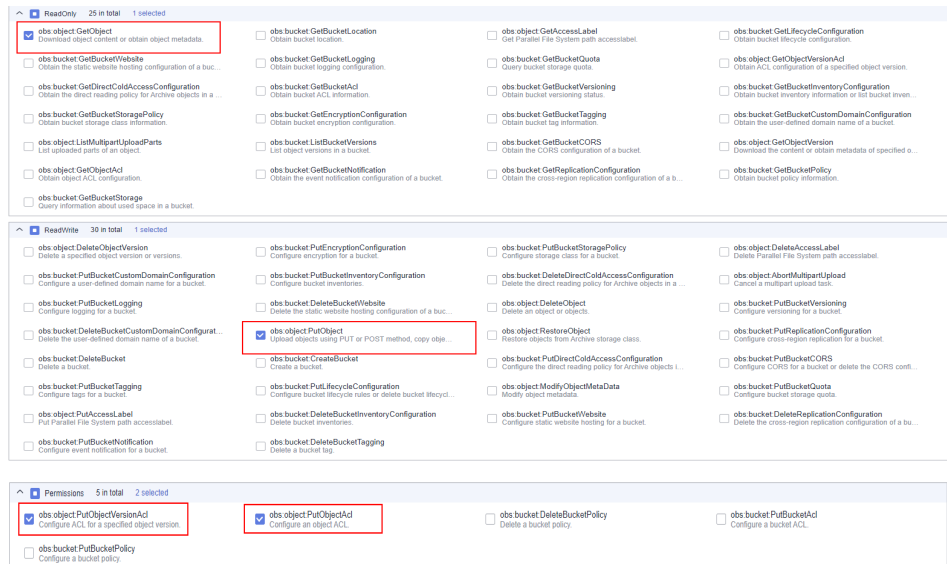
5. Set the policy content.
  - a. Select **Allow**.
  - b. Click **Select service** and select **Object Storage Service (OBS)**.



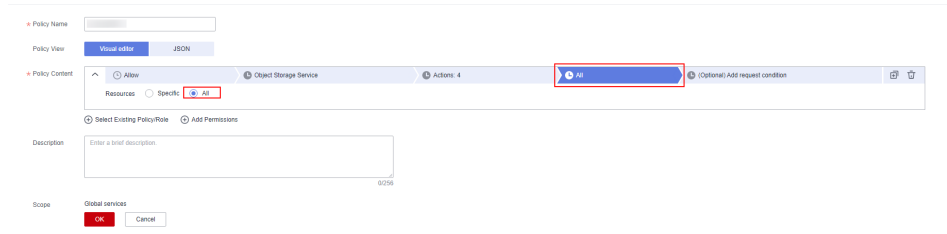
- c. For **Actions**, select actions as required. For details about the actions, see [Object-Related Actions](#).

To delegate CPH to perform operations on OBS buckets, at least the following four actions must be selected:

- obs:object:GetObject**
- obs:object:PutObject**
- obs:object:PutObjectVersionAcl**
- obs:object:PutObjectAcl**



- d. Select **All** for **Resources**. For details about how to select specific resources, see descriptions of specific resources in [Creating a Custom Policy](#).



6. Click **OK**.
- Create an agency.
1. Log in to the IAM console.

In the navigation pane on the left, choose **Agencies** and click **Create Agency** in the upper right corner.

Agency Name	Delegated Party	Validity Period	Create	Description	Operation
Cloud service	Cloud Phone Service (CPH)	Unlimited	Mar 23, 2023 18:56:13 GMT+08:00		Authorize Modify Delete
Account	Account	Unlimited	Mar 21, 2023 18:07:03 GMT+08:00		Authorize Modify Delete
Account	Account	Unlimited	Feb 22, 2023 17:19:00 GMT+08:00		Authorize Modify Delete
Account	Account	Unlimited	Feb 22, 2023 11:58:57 GMT+08:00		Authorize Modify Delete
Account	Account	Unlimited	Feb 08, 2023 10:00:37 GMT+08:00		Authorize Modify Delete
Cloud service	Cloud service	Unlimited	Feb 01, 2023 09:22:30 GMT+08:00		Authorize Modify Delete

2. Configure parameters shown in the following figure and click **Next**.

**Agencies / Create Agency**

\* Agency Name:

\* Agency Type:  Account  
 Cloud service  
 Delegate another HUAWEI CLOUD account to perform operations on your resources.  
 Delegate a cloud service to access your resources in other cloud services.

\* Cloud Service:

\* Validity Period:

Description:

0/255



The agency name must be **cph\_obs\_agency**.

3. Select the custom policy created in **Create a custom policy** and click **Next**.

**Authorize Agency**

1 Select Policy/Role 2 Select Scope 3 Finish

Assign selected permissions to cph\_obs\_agency.

View Selected (1) Copy Permissions from Another Project

Policy/Role Name	Type
<input checked="" type="checkbox"/> Custom policy	Custom policy

4. Select **Global services** and click **OK**.

**Authorize Agency**

1 Select Policy/Role 2 Select Scope 3 Finish

The following are recommended scopes for the permissions you selected. Select the desired scope requiring minimum authorization.

Scope

All resources

Enterprise projects

Global services

After authorization, users can use resources of the global service based on their permissions.

Show Less

# 9 Changing the AOSP Version of a Cloud Phone

---

This section describes how to change the AOSP version of a cloud phone.

## Prerequisites

The image version of cloud phones on the target server is AOSP 7.

## Precautions

1. A version change is a high-risk operation. To learn how to back up user data, refer to [Exporting Data from Cloud Phones in Batches](#). To learn how to roll back the image to the original version, refer to [Restoring Cloud Phone Data](#).
2. During a version change, the values of attributes such as **ro.build.version.release** and **ro.build.fingerprint**, which define the Android version, are automatically changed to values that identify the target image version.

## Upgrading the AOSP Version

### Method 1 (User data retained)

Call the API for [restarting cloud phones](#) to change the AOSP version and retain user data.

Notes:

- The restart API can only change the image from an earlier version to a later one.
- If you do not need to retain user data, use method 2 to change to the AOSP version, which reduces application incompatibility issues.

### Method 2 (User data not retained)

To change the AOSP version, call the API for [resetting cloud phones](#).

Notes:

- You can call the restart API to upgrade an image version or roll back the image to an earlier version.

## Rolling Back the AOSP Image to an Earlier Version

For compatibility reasons, user data can be retained only when an earlier version is changed to a later version. To roll back to an earlier version, you can only call the API for [resetting cloud phones](#) that does not retain user data.

### NOTE

If you roll back to an earlier image version by calling the API for resetting cloud phones, to view the screens of the cloud phones, run the **adb disconnect ip:port** command on the ADB clients of cloud phones of some versions and then run the **adb connect ip:port** command again.

# A Change History

Released On	Description
2023-10-31	<p>This issue is the fourth official release, which incorporates the following change:</p> <ul style="list-style-type: none"> <li>Added <a href="#">Changing the AOSP Version of a Cloud Phone</a>.</li> </ul>
2023-08-24	<p>This issue is the third official release, which incorporates the following changes:</p> <ul style="list-style-type: none"> <li>Updated the example value of <b>Instance Specifications</b> in <a href="#">Buying a Server for General-purpose Cloud Phones</a>.</li> <li>Updated <a href="#">Purchasing a Cloud Phone Server That Supports Application Sharing</a>.</li> <li>Updated restrictions and limitations in <a href="#">Allowing a Cloud Phone Server to Access a Public Network Outside the Chinese Mainland</a>.</li> </ul>
2023-03-31	<p>This issue is the second official release, which incorporates the following changes:</p> <ul style="list-style-type: none"> <li>Added <a href="#">Delegating CPH to Operate OBS Buckets</a>.</li> <li>Deleted sections "Reading Data from an OBS Bucket" and "Uploading Data to an OBS Bucket."</li> </ul>
2022-12-30	<p>This issue is the first official release.</p>