Cloud Phone

Best Practices

Issue	04
Date	2021-09-01

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 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**.
3. In the navigation pane on the left, choose **Servers**. In the upper right corner, click **Buy Server**.
4. Complete the basic configuration as prompted.

Table 1-1 Parameter description

Parameter	Description	Example Value
Billing Mode	Servers are billed only yearly/monthly.	Yearly/Monthly
Region	Cloud phones in different regions cannot communicate with each other over an intranet. For lower network latency and quicker resource access, select the nearest region. After a server is purchases, its region cannot be changed.	CN East-Shanghai1

Parameter	Description	Example Value
AZ	An AZ is a part of a region with its own independent power supplies and networks. AZs are physically isolated but can communicate through an internal network. • If you require high availability, buy servers in difference AZs. • If you require low network latency, buy servers in the same AZ.	AZ1
Server Type	Two options are available: **Cloud phone server** and **Cloud mobile gaming server**. For details, see **Specifications**.	Cloud phone server physical.rx1.xlarge
Instance Specifications	Set this parameter as required.	rx1.cp.c60.d10.e1v1
Phone Image	Only the Android OS is supported.	AOSP7.1.1
Quantity	• A maximum of 10 servers can be purchased at a time. • The required duration ranges from 1 month to 3 years.	**Quantity**: **1** Required duration: **6 months**

5. Click **Next: Configure Network** to configure the network as prompted.

   Network customization has been available in the CN Southwest-Guiyang1 region. You are advised to use this function.

   physical.rx1 servers do not support network customization. **Table 1-3** shows their system-defined network configuration.

   NOTE

   Cloud phone network can be configured in the following two ways:

   - Custom network: You can reuse the existing VPCs to manage cloud phone servers and reuse resources such as the shared bandwidth that you have purchased. This function is only available in the Southwest-Guiyang1 region.
   - Default network: The system automatically creates VPCs and configures bandwidths for your cloud phone servers. The existing VPCs and bandwidths cannot be reused.

Table 1-2 Custom network configuration

Parameter	Description	Example Value
Networking	Set **Network** by selecting an available VPC and subnet from the drop-down list and specifying a private IP address assignment mode. Cloud phones use networks provided by a VPC, including subnets and security groups. Select an existing VPC or create one.	None
Security Group Authorization	A **cph_admin_trust** agency will be created for you. This agency has the **VPC FullAccess** permission. To authorize the Cloud Phone service to create an agency for you, ensure that your login user has the **Security Administrator** permission or the fine-grained permission **iam:agencies:createAgency** for creating agencies. For more information, see **Permission Management**. The Cloud Phone service will use the agency to perform the following operations: • Create an elastic NIC, EIP, and virtual IP address for a general-purpose or gaming cloud phone. • Create a default security group for the general-purpose or gaming cloud phone server and set the port range for the security group. The port range will be mapped to that of each cloud phone so that the instance can open application access ports. **NOTE** By default, if an ECS and 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 the instructions provided in **What Are the Security Group Authorization Rules for Cloud Phones Using Custom Networks?**	None
EIP	• **Auto assign**: Buy a new EIP for the cloud phone.	Buy now

Parameter	Description	Example Value
Line	• Static BGP offers routing control and protects against route flapping, but cannot choose an optimal path in real time when a network connection fails. • Dynamic BGP provides automatic failover and chooses the optimal path when a network connection fails.	Dynamic BGP
Bandwidth	• **Dedicated**: You will be charged based on the total traffic you generate. • **Shared**: The bandwidth can be shared by multiple EIPs.	Shared
Bandwidth Size	Supported range: 1 Mbits/s to 2000 Mbit/s	300 Mbit/s
Bandwidth Name	If you set **Bandwidth** to **Shared**, select an existing shared bandwidth name from the drop-down list.	bandwidth-001

Table 1-3 Default network configuration

Parameter	Description	Example Value
Bandwidth Type	• **Dedicated**: Specifies the bandwidth used exclusively by each purchased server. Dedicated bandwidth is required for transmitting heavy data traffic. • **Shared**: Specifies the bandwidth shared by multiple servers of a user. Shared bandwidth can help reduce the cost of public network bandwidth. It is suitable for services where traffic peaks tend to be staggered.	Shared
Method	The method can be **Using existing** or **Create Shared Bandwidth**. If you use a shared bandwidth for the first time, select **Create Shared Bandwidth**.	Create Shared Bandwidth

Parameter	Description	Example Value
Billed By	The dedicated bandwidth is billed only by traffic, and the shared bandwidth is billed only by bandwidth. • **Traffic**: You specify a maximum bandwidth and pay for the total traffic you generate. • **Bandwidth**: You are charged based on the bandwidth size and usage duration. This price is not included in the price of a server.	Bandwidth
Bandwidth	Specifies the maximum bandwidth, which is fixed at **300 Mbit/s**.	300 Mbit/s
Bandwidth Name	• If you set **Method** to **Using existing**, select an existing shared bandwidth name from the drop-down list. The number of associated servers is displayed on the existing shared bandwidth, so you can choose the right one as needed. • If you set **Method** to **Create Shared Bandwidth**, enter the shared bandwidth name in the text box.	whole-bandwidth-xxxx
Bandwidth Size	This parameter is mandatory when you set **Method** to **Create Shared Bandwidth**. The larger the bandwidth, the higher the price. Set this parameter based on your service requirements.	5 Mbit/s

NOTE

The default network (VPC and bandwidth) configured in this step is used only by the Cloud Phone servers. The network details are not displayed on the **Network Console**.

6. Click **Next: Configure Advanced Settings**. Complete the advanced configuration as prompted.

Table 1-4 Parameter description

Parameter	Description	Example Value
Name	Specifies the name for the server and the cloud phones that are virtualized from the server. The name must be unique. 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 the ascending order. For example, if you purchased a server for virtualizing 60 cloud phones and entered **CPH** for **Name**, the server name is **CPH-1**, and the cloud phone names range from **CPH-1-00001** to **CPH-1-00060**.	CPH
Key Pair	A key pair is used for remote login authentication. • 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. • If no key pair has been created, you can click **Create Key Pair** to create a key pair. Go back to the **Configure Advanced Settings** page, refresh the drop-down list, and select the created key pair. The private key 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 **Creating a Key Pair**.	KeyPair-test
VNC Login	After this function is enabled, you can log in to your cloud phone using VNC. **NOTE** Cloud phones of certain specifications do not support this function. For details about the cloud phones that support this function, see **What Are the Cloud Phone Specifications that Support VNC Login?**	N/A

Parameter	Description	Example Value
Application Port	This parameter is available when you select **Application Port** for **Advanced Settings**. Select this parameter when your cloud phones need to provide services for external systems. • **Application name**: The name can contain letters. However, the values **ADB** and **VNC** in uppercase, lowercase, or mixed case are not allowed. • **Port number**: Ports from **0** to **65535** are supported. • **Internet access** – If this option is selected, the cloud phone application port can be accessed over the Internet without authentication. That is, the cloud phone port and the server port are exposed to the Internet. – If this option is not selected, the cloud phone can be accessed only over the Intranet. **CAUTION** • Ensure that security control has been performed before you select **Internet access**. • The Cloud Phone service does not perform security check for the ports you configured for Internet access.	key 10001 Do not select it.

7. Click **Next: Confirm** to check the order.
   – If the information 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 and Obtaining Its Screens

Airtest is a cross-platform UI automation compiler. Due to the shortcomings of the VNC protocol, such as high bandwidth requirements and great performance deterioration, Monbox-based cloud phones do not support VNC access. It is recommended that you use Airtest to quickly obtain the cloud phone screens.

Prerequisites

- You have purchased a Cloud Phone server and logged in a cloud phone using ADB. For details, see **Buying a Cloud Phone**.
- Airtest has been installed on the local PC.

  NOTE

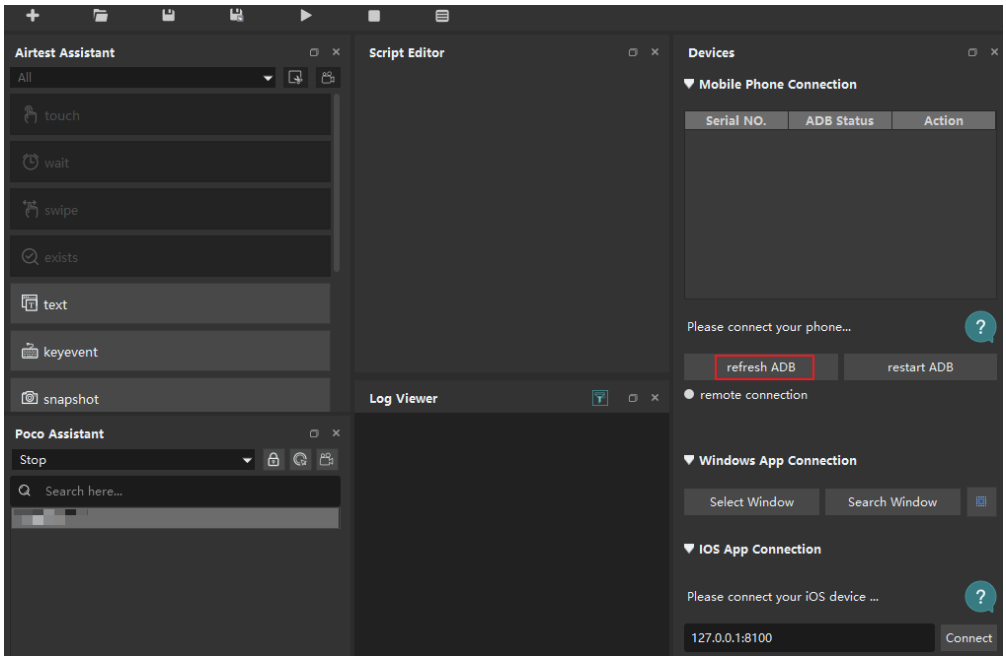
  Log in to the Airtest official website at **https://airtest.netease.com/**, and download the required version, and install it.
- The command-line interface (CLI) for the ADB connection has been closed, and the SSH tunnel has been successfully established.

Procedure

1. On the Airtest home page, click **refresh ADB**.

   The connected mobile phones are displayed.

   Figure 1-1 Airtest home page

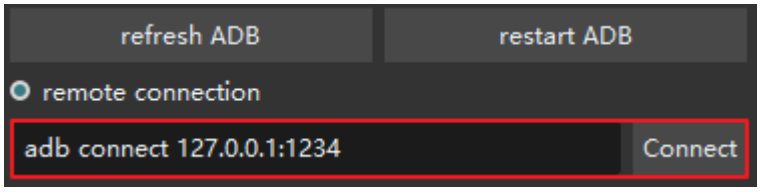


2. If the cloud phone you want to connect to is not displayed, select **remote connection** and enter the ADB command for connecting to the target cloud phone, as shown in **Figure 1-2**.

   adb connect 127.0.0.1:*1234*

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

   Figure 1-2 remote connection



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

CAUTION

Ensure that the CLI for the ADB connection is closed. Otherwise, the connection fails. In addition, ensure that the SSH tunnel is successfully established. Otherwise, **ADB Status** will be **offline** even if the cloud phone has been identified. As a result, the cloud phone screens cannot be obtained.

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

Figure 1-3 Mobile Phone Connection

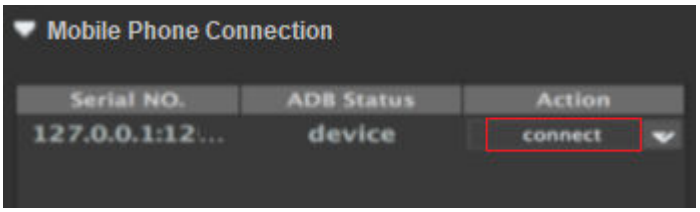
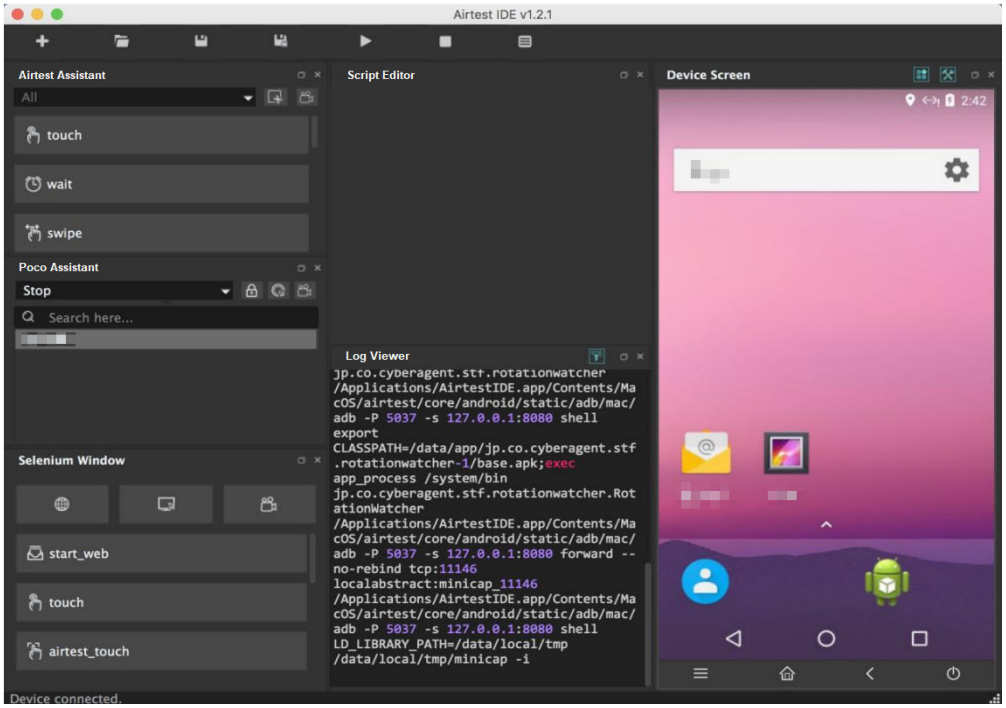
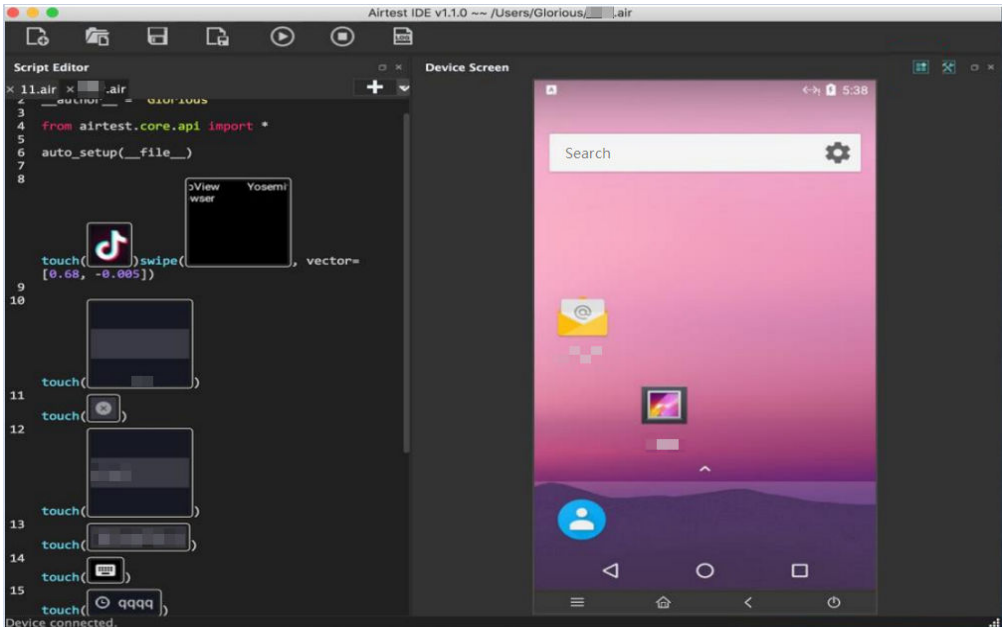


Figure 1-4 Device Screen



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 cloud phone screens



1.3 Deploying Applications on General-purpose Cloud Phones

1.3.1 Querying Cloud Phones

Obtain the cloud phone list by referring to **Querying the Cloud Phones** in the *Cloud Phone API Reference*.

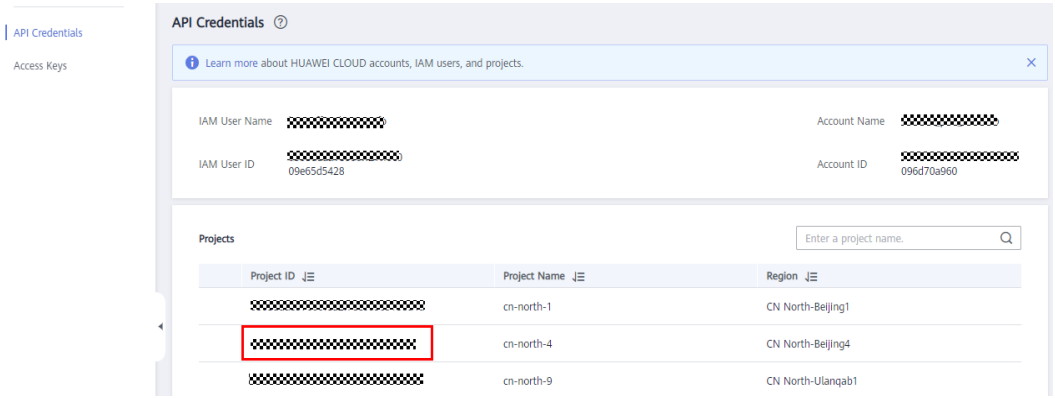
Example API

```
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}
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 North-Beijing4 region is **cph.cn-north-4.myhuaweicloud.com**.
- **project_id** indicates the project ID of the region to which the gaming cloud phone server belongs, 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 **获取token**.

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. Parameters such as **phone_name** and **server_id** are not specified.

1.3.2 Installing Applications on a Cloud Phone

Install applications on a cloud phone by referring to **Installing the APK** in the *Cloud Phone 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 located. (Upload the installation package by referring to **Step 6: Uploading an Object**.)
- The OBS bucket policies have been configured based on **Reading Data from an OBS Bucket**.

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": [
      "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
   ]
}
```

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, and **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 multiple 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, and **obs://yzw-apk-install/apk/com.hermes.bgame.apk** is the full path of the application installation package.
- Replace **${token}** with the actual token.

1.3.3 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 **Uploading Data to an OBS Bucket**.

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.

1.3.4 Deploying Applications

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 **${package_name}_${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 and obtain the server IDs by calling the API for **Querying the Cloud Phone Servers**.

Example

See **Pushing Shared Storage Files** in the *Cloud Phone API Reference*.

Follow-up Procedure

Reset all cloud phones in batches by referring to **Resetting Cloud Phones** in the *Cloud Phone API Reference*.

1.3.5 Updating the Version of an Application

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

Deleting the Application of the Old 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/${package_name}**, and **/data/media/0/Android/data/${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**, where **package_name** indicates the package name of the current application.
  - **server_ids** indicates the ID list of servers where the application is deployed. You can enter multiple server IDs and obtain the server IDs by calling the API for **Querying the Cloud Phone Servers**.

- Example:

  See **Deleting a Shared Storage File** in the *Cloud Phone API Reference*.

Deploying the Application of the New Version

See **Deploying Applications**.

2 Best Practices of Connecting to Gaming Phones

2.1 Buying a Server for Gaming Phones

Procedure

1. Log in to the management console.
2. On the **Service List** page, choose **Compute** > **Cloud Phone**.
3. In the navigation pane on the left, choose **Servers**. In the upper right corner, click **Buy Server**.
4. Complete the basic configuration as prompted.

Table 2-1 Parameter description

Parameter	Description	Example Value
Billing Mode	Servers are billed only yearly/ monthly.	Yearly/Monthly
Region	Cloud phones in different regions cannot communicate with each other over an intranet. For lower network latency and quicker resource access, select the nearest region. After a server is purchases, its region cannot be changed.	CN East-Shanghai1

Parameter	Description	Example Value
AZ	An AZ is a part of a region with its own independent power supplies and networks. AZs are physically isolated but can communicate through an internal network. • If you require high availability, buy servers in difference AZs. • If you require low network latency, buy servers in the same AZ.	AZ1
Server Type	Two options are available: **Cloud phone server** and **Cloud mobile gaming server**. For details, see **Specifications**.	Cloud phone server physical.rx1.xlarge
Instance Specifications	Set this parameter as required.	rx1.cp.c60.d10.e1v1
Phone Image	Only the Android OS is supported.	AOSP7.1.1
Quantity	• A maximum of 10 servers can be purchased at a time. • The required duration ranges from 1 month to 3 years.	**Quantity**: **1** Required duration: **6 months**

5. Click **Next: Configure Network** to configure the network as prompted.

   Network customization has been available in the CN Southwest-Guiyang1 region. You are advised to use this function.

   physical.rx1 servers do not support network customization. **Table 2-3** shows their system-defined network configuration.

   NOTE

   Cloud phone network can be configured in the following two ways:

   - Custom network: You can reuse the existing VPCs to manage cloud phone servers and reuse resources such as the shared bandwidth that you have purchased. This function is only available in the Southwest-Guiyang1 region.
   - Default network: The system automatically creates VPCs and configures bandwidths for your cloud phone servers. The existing VPCs and bandwidths cannot be reused.

Table 2-2 Custom network configuration

Parameter	Description	Example Value
Networking	Set **Network** by selecting an available VPC and subnet from the drop-down list and specifying a private IP address assignment mode. Cloud phones use networks provided by a VPC, including subnets and security groups. Select an existing VPC or create one.	None
Security Group Authorization	A **cph_admin_trust** agency will be created for you. This agency has the **VPC FullAccess** permission. To authorize the Cloud Phone service to create an agency for you, ensure that your login user has the **Security Administrator** permission or the fine-grained permission **iam:agencies:createAgency** for creating agencies. For more information, see **Permission Management**. The Cloud Phone service will use the agency to perform the following operations: • Create an elastic NIC, EIP, and virtual IP address for a general-purpose or gaming cloud phone. • Create a default security group for the general-purpose or gaming cloud phone server and set the port range for the security group. The port range will be mapped to that of each cloud phone so that the instance can open application access ports. **NOTE** By default, if an ECS and 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 the instructions provided in **What Are the Security Group Authorization Rules for Cloud Phones Using Custom Networks?**	None
EIP	• **Auto assign**: Buy a new EIP for the cloud phone.	Buy now

Parameter	Description	Example Value
Line	• Static BGP offers routing control and protects against route flapping, but cannot choose an optimal path in real time when a network connection fails. • Dynamic BGP provides automatic failover and chooses the optimal path when a network connection fails.	Dynamic BGP
Bandwidth	• **Dedicated**: You will be charged based on the total traffic you generate. • **Shared**: The bandwidth can be shared by multiple EIPs.	Shared
Bandwidth Size	Supported range: 1 Mbits/s to 2000 Mbit/s	300 Mbit/s
Bandwidth Name	If you set **Bandwidth** to **Shared**, select an existing shared bandwidth name from the drop-down list.	bandwidth-001

Table 2-3 Default network configuration

Parameter	Description	Example Value
Bandwidth Type	• **Dedicated**: Specifies the bandwidth used exclusively by each purchased server. Dedicated bandwidth is required for transmitting heavy data traffic. • **Shared**: Specifies the bandwidth shared by multiple servers of a user. Shared bandwidth can help reduce the cost of public network bandwidth. It is suitable for services where traffic peaks tend to be staggered.	Shared
Method	The method can be **Using existing** or **Create Shared Bandwidth**. If you use a shared bandwidth for the first time, select **Create Shared Bandwidth**.	Create Shared Bandwidth

Parameter	Description	Example Value
Billed By	The dedicated bandwidth is billed only by traffic, and the shared bandwidth is billed only by bandwidth. • **Traffic**: You specify a maximum bandwidth and pay for the total traffic you generate. • **Bandwidth**: You are charged based on the bandwidth size and usage duration. This price is not included in the price of a server.	Bandwidth
Bandwidth	Specifies the maximum bandwidth, which is fixed at **300 Mbit/s**.	300 Mbit/s
Bandwidth Name	• If you set **Method** to **Using existing**, select an existing shared bandwidth name from the drop-down list. The number of associated servers is displayed on the existing shared bandwidth, so you can choose the right one as needed. • If you set **Method** to **Create Shared Bandwidth**, enter the shared bandwidth name in the text box.	whole-bandwidth-xxxx
Bandwidth Size	This parameter is mandatory when you set **Method** to **Create Shared Bandwidth**. The larger the bandwidth, the higher the price. Set this parameter based on your service requirements.	5 Mbit/s

NOTE

The default network (VPC and bandwidth) configured in this step is used only by the Cloud Phone servers. The network details are not displayed on the **Network Console**.

6. Click **Next: Configure Advanced Settings**. Complete the advanced configuration as prompted.

Table 2-4 Parameter description

Parameter	Description	Example Value
Name	Specifies the name for the server and the cloud phones that are virtualized from the server. The name must be unique. 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 the ascending order. For example, if you purchased a server for virtualizing 60 cloud phones and entered **CPH** for **Name**, the server name is **CPH-1**, and the cloud phone names range from **CPH-1-00001** to **CPH-1-00060**.	CPH
Key Pair	A key pair is used for remote login authentication. • 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. • If no key pair has been created, you can click **Create Key Pair** to create a key pair. Go back to the **Configure Advanced Settings** page, refresh the drop-down list, and select the created key pair. The private key 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 **Creating a Key Pair**.	KeyPair-test
VNC Login	After this function is enabled, you can log in to your cloud phone using VNC. **NOTE** Cloud phones of certain specifications do not support this function. For details about the cloud phones that support this function, see **What Are the Cloud Phone Specifications that Support VNC Login?**	N/A

Parameter	Description	Example Value
Application Port	This parameter is available when you select **Application Port** for **Advanced Settings**. Select this parameter when your cloud phones need to provide services for external systems. • **Application name**: The name can contain letters. However, the values **ADB** and **VNC** in uppercase, lowercase, or mixed case are not allowed. • **Port number**: Ports from **0** to **65535** are supported. • **Internet access** – If this option is selected, the cloud phone application port can be accessed over the Internet without authentication. That is, the cloud phone port and the server port are exposed to the Internet. – If this option is not selected, the cloud phone can be accessed only over the Intranet. **CAUTION** • Ensure that security control has been performed before you select **Internet access**. • The Cloud Phone service does not perform security check for the ports you configured for Internet access.	key 10001 Do not select it.

7. Click **Next: Confirm** to check the order.
   – If the information 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**.

2.2 Deploying Games

2.2.1 Querying Cloud Phones

Obtain the cloud phone list by referring to **Querying the Cloud Phones** in the *Cloud Phone API Reference*.

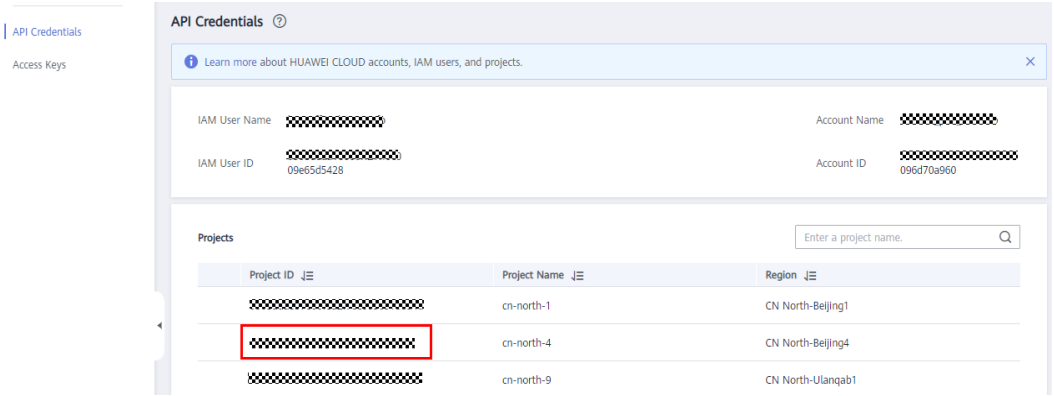
Example API

```
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}
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 North-Beijing4 region is **cph.cn-north-4.myhuaweicloud.com**.
- **project_id** indicates the project ID of the region to which the gaming cloud phone server belongs, 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 2-1 Obtaining the project ID



- The part after the question mark (**?**) in the URL is optional.
- **$token** indicates the response of the API for **获取token**.

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. Parameters such as **phone_name** and **server_id** are not specified.

2.2.2 Installing Applications on a Cloud Phone

Install applications on a cloud phone by referring to **Installing the APK** in the *Cloud Phone 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 located. (Upload the installation package by referring to **Step 6: Uploading an Object**.)
- The OBS bucket policies have been configured based on **Reading Data from an OBS Bucket**.

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": [
      "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
   ]
}
```

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, and **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 multiple 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, and **obs://yzw-apk-install/apk/com.hermes.bgame.apk** is the full path of the application installation package.
- Replace **${token}** with the actual token.

2.2.3 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 **Uploading Data to an OBS Bucket**.

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.

2.2.4 Deploying Applications

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 **${package_name}_${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 and obtain the server IDs by calling the API for **Querying the Cloud Phone Servers**.

Example

See **Pushing Shared Storage Files** in the *Cloud Phone API Reference*.

Follow-up Procedure

Reset all cloud phones in batches by referring to **Resetting Cloud Phones** in the *Cloud Phone API Reference*.

2.2.5 Updating the Version of an Application

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

Deleting the Application of the Old 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/${package_name}**, and **/data/media/0/Android/data/${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**, where **package_name** indicates the package name of the current application.
- **server_ids** indicates the ID list of servers where the application is deployed. You can enter multiple server IDs and obtain the server IDs by calling the API for **Querying the Cloud Phone Servers**.

- Example:

  See **Deleting a Shared Storage File** in the *Cloud Phone API Reference*.

Deploying the Application of the New Version

See **Deploying Applications**.

2.3 Accessing Games

To help you quickly develop cloud mobile gaming services, Huawei launches the Software Development Kit (SDK) for cloud mobile gaming. SDK encapsulates APIs provided for the cloud mobile gaming services to simplify user development. You can directly call API functions provided by the cloud mobile gaming SDK to access cloud games.

The Android SDK provides the following access methods:

- AAR package access: Based on the JAR package access, the SDK integrates the activities required for running cloud games. The Activity lifecycle is maintained in the SDK.
- JAR package access: The SDK provides the capability of cloud game access. The customer provides the activity or fragmentation component. When the game is started, the viewgroup component for image rendering is transferred to the SDK.

Complete the lifecycle management of a game, such as initializing the game, starting the game, configuring the game image, and exiting the game by referring to the AAR or JAR API examples.

Importing the SDK

Import the .aar package compiled by the SDK to the **app/libs** directory.

Import the .jar package compiled by the SDK to the **app/libs** directory and the .so file to the **app/jniLibs** directory.

Using the AAR SDK

Import the following information to the class file that invokes the SDK APIs:

import com.huawei.cloudappsdk.manager.CloudAppManager;

Set **ApplicationContext** before invoking the APIs.

CloudAppManager.defaultManager().setApplicationContext(getApplicationContext());

Using the JAR SDK

Import the following information to the class file that invokes the SDK APIs:

import com.huawei.cloudgame.api.CloudGameManager;

import com.huawei.cloudgame.api.ICloudGame;

2.4 Scheduling Cloud Phones

After **Accessing Games**, you can design and develop a system for scheduling gaming phones. The scheduling system should provide the following functions:

- When receiving a new access request, the system allocates an idle gaming phone to the user and updates the gaming phone status to allocated.
- After a user accesses a game on a gaming phone and the game is started, the system updates the gaming phone status to occupied.
- After a user finishes playing the game and exits the gaming phone, the system updates the gaming phone status to idle.

To sum up, you need to maintain the idle, allocated, and occupied states of gaming phones to allow users to access or exit games.

To help you maintain the gaming phone statuses, the gaming media engine performs authentication and event notification between the gaming phones and your applications when users access the gaming phones, start games, and exit games, so that you can know and maintain the gaming phone statuses.

Use the following APIs:

NOTE

The following parameters are only examples.

Entering the Game Event

- Example request

```
{
"event_type": "app",
"event": {
"phone_id": "a7f3a1c5258347d6b6f1def79e11f2bc",
"status": 0,
"session_id": "856f5555806443e98b7ed04c5a9d6a9a",
```


"ticket": "5558064856f5555806443e98b7ed04c5a9d6a9ab7ed04c5a9d6a806443e98",
"time": "2019-11-14T19:38:49Z",
"app_id": "856f5555806443e98b7ed04c5a9d6a9a",
"error_msg": null
}
}

- Example response

{
"request_id": "6837531fd3f54550927b930180a706bf"
}

Quitting the Game Event

- Example request

{
"event_type": "app",
"event": {
"phone_id": "a7f3a1c5258347d6b6f1def79e11f2bc",
"status": 1,
"session_id": "856f5555806443e98b7ed04c5a9d6a9a",
"ticket": "5558064856f5555806443e98b7ed04c5a9d6a9ab7ed04c5a9d6a806443e98",
"time": "2019-11-14T19:38:49Z",
"run_time": 41000,
"app_id": "856f5555806443e98b7ed04c5a9d6a9a",
"error_msg": null
}
}

- Example response

{
"request_id": "6837531fd3f54550927b930180a706bf"
}

Heartbeat Reporting

- Example request

{
"event_type": "heartbeat",
"event": {
"phone_id": "a7f3a1c5258347d6b6f1def79e11f2bc",
"app_id": "282fdfd613e04651a48225f1b2034de7",
"session_id": "36728337b89f22122af4a6e08bb1e7c0",
"time": "2019-11-14T19:38:49Z"
}
}

- Example response

{
"request_id": "6837531fd3f54550927b930180a706bf"
}

2.5 Performing O&M for Gaming Phones

Operations for Gaming Phone Data

During the operations of cloud mobile games, to help you obtain operations data of gaming phones and meet your requirements for data-driven operations, the cloud mobile games provide the following APIs:

- Registering cloud gaming data listener

This API is used to register the cloud game data listener to obtain the cloud game data sent to the mobile phone.

- Obtaining the network latency

  This API is used to obtain the network round-trip time (RTT).

- Obtaining the number of received frames

  This API is used to obtain the number of video frames received in a certain time.

- Obtaining the size of the received audio and video data

  This API is used to obtain the size of the received audio and video data.

O&M for Gaming Phones

- **Restarting a Server**
- **Restarting Cloud Phones**
- **Resetting Cloud Phones**
- **Updating Cloud Phone Attributes**
- **Querying the Task Execution Status**
- **Viewing Cloud Phone Metrics**

3 Installing an App on Cloud Phones in Batches

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

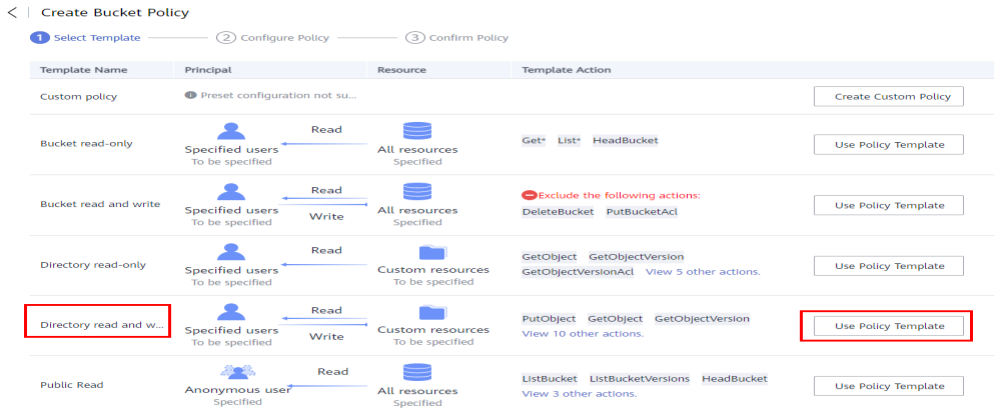
NOTE

Assume that the cloud phone has the app installed is 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



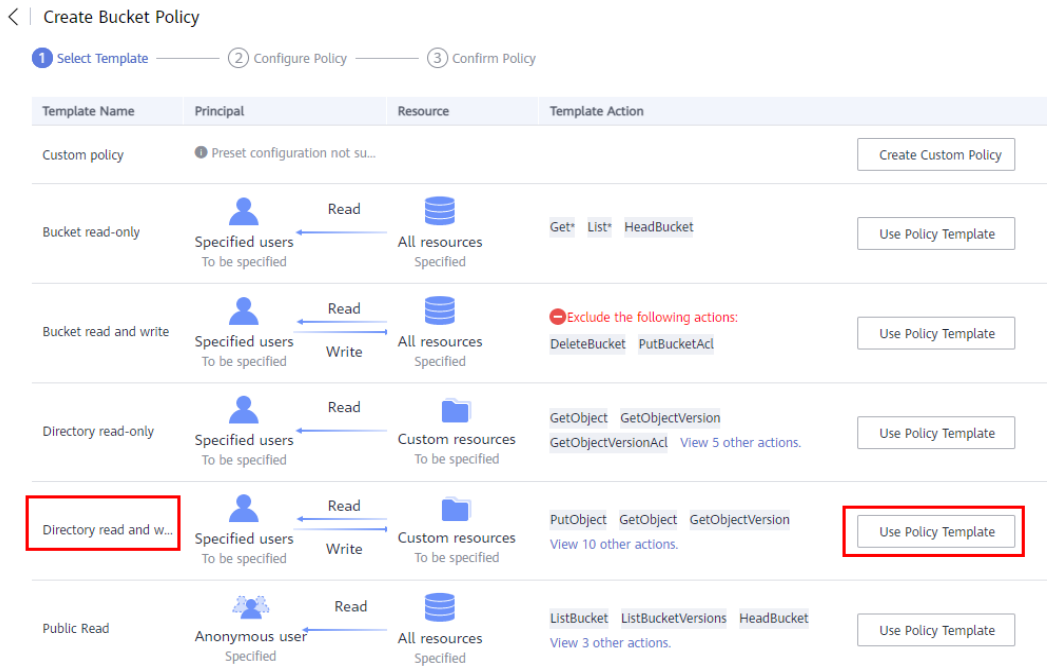
- The seed cloud phone must be a cloud phone that has not been operated on. If there is any operation on the cloud phone, **reset it**.

Procedure

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

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

Figure 3-2 Configuring the bucket policy



2. Connect to the seed cloud phone in ADB mode and install required app on it.

   For details, see **How Do I Install Apps on a Cloud Phone?**

3. Call the API to export the seed cloud phone data, pack the data, and upload the data package to the OBS bucket created in **1**.

   The curl command is as follows:

```
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 exported data
   - **object_path**: OBS path for storing the exported data

CAUTION

If you want to pack all data of the seed cloud phone app, 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}**

The following is an 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 all files packed and uploaded in **3** are successfully uploaded. If yes, go to the next step.
5. Call an API to push files stored in the OBS bucket to the shared storage directories of the servers.

   The following is an 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**: ID list of servers that receive the file push If multiple server IDs are specified, the app can be installed on cloud phones on multiple servers.

   NOTE

   For details about this API, see **Pushing Shared Storage Files**.

6. Log in to the Cloud Phone console, click the name of the server that has received the files pushed. In the **Instances** area, select all cloud phones on which the preceding app needs to be installed, and click **Reset**.

> **CAUTION**
>
> This step is a reset operation, not a restart operation.

Execution Result

The app on the seed cloud phone has been installed on all other cloud phones.

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 logged in a cloud phone using ADB. For details, see **Buying a Cloud Phone**.

Procedure

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

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

   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 Cloud Phones to Build a Cloud Mobile Gaming System

Cloud mobile gaming servers are suitable for the cloud mobile gaming scenario. One cloud mobile gaming server can virtualize 15 cloud phones. This topic describes how to build an entry-level cloud mobile gaming system in four steps.

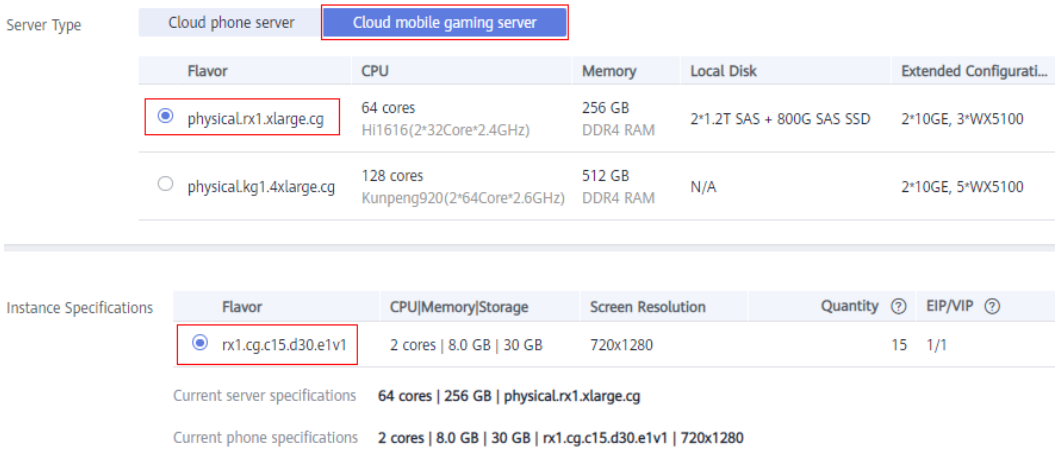
1. Streamlining the Integration and Interconnection Solution of the Cloud Mobile Gaming System

First, you need to sort out your requirements for the cloud mobile gaming system, specify the application scenarios and evolution directions of services, and specify the interaction modes between your operations and O&M systems and cloud phones. The interconnection solutions vary depending on service requirements. You can use the audio and video interfaces provided by cloud phones to develop your own control programs, network transmission, scheduling systems, and clients.

2. Purchasing a Cloud Mobile Gaming Server

When purchasing a server, select **Cloud mobile gaming server** for **Server Type**, that is, purchase 15 cloud phones.

Figure 5-1 Selecting the cloud phone specifications



3. Obtaining the Audio and Video SDKs of the Cloud Mobile Gaming System

Contact your account manager to obtain the audio and video SDKs of the cloud mobile gaming system.

4. Developing the Access, Touch, and Scheduling Systems of the Cloud Mobile Gaming System

After obtaining the SDK document, you can use audio and video interfaces to develop your own control programs, network transmission, dispatching systems, and clients. Cloud Phone will provide a simpler access mode in the future.

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 really use, debug, and test those devices on the cloud. This section describes how to deploy STF components on an ECS to quickly manage cloud phones in batches.

Restrictions and Limitations

- According to the test, STF can manage about 160 cloud phones at the same time. For larger-scale access management, secondary development is required.
- The smooth running of STF depends on a stable network. 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(40GB)

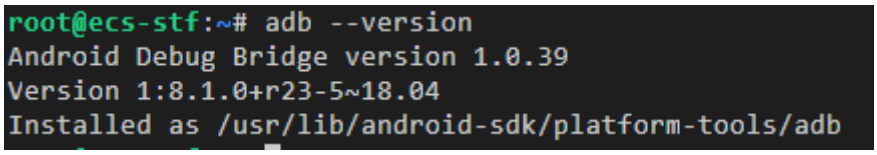
Procedure

Deploy components that STF depends on 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
```


Figure 6-1 Successful installation of ADB with **--version** displayed in the command output

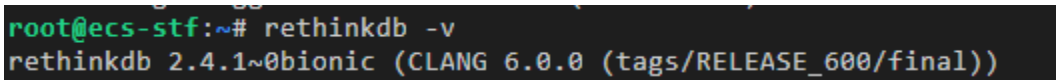


2. Update the source and install RethinkDB to store STF data.

```
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

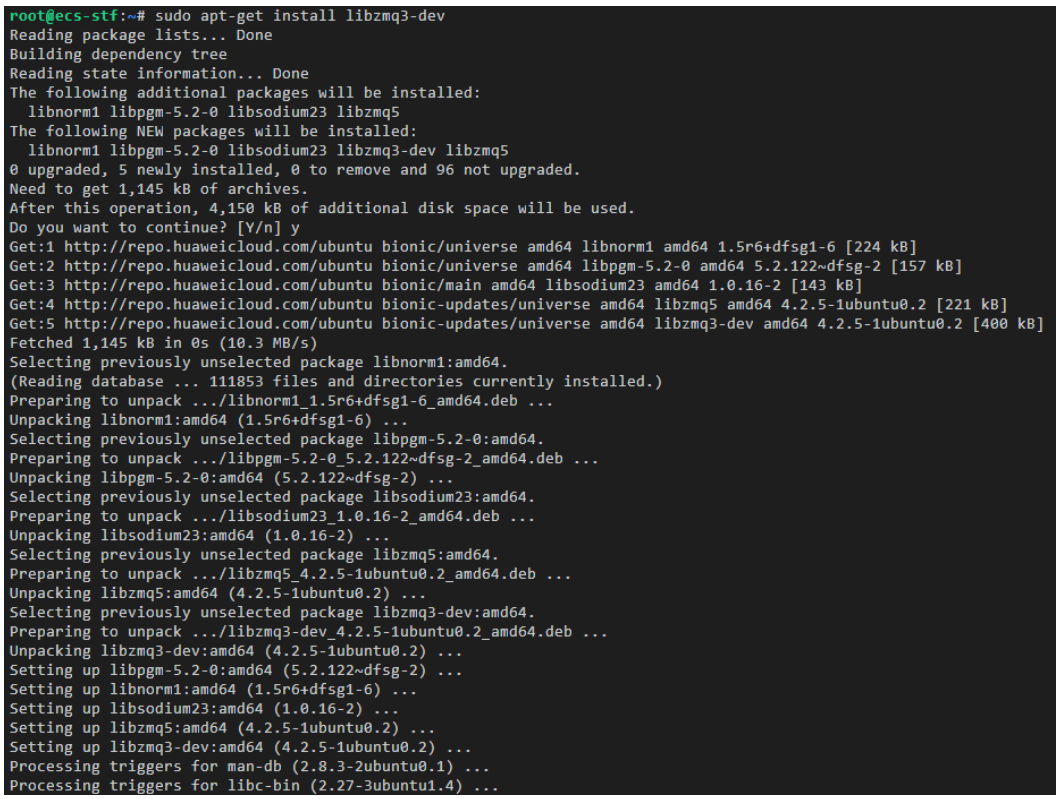


RethinkDB provides official support for the x86 architecture while experimental support for the ARM architecture.

3. Install ZeroMQ to transfer messages.

```
sudo apt-get install libzmq3-dev
```

Figure 6-3 Successful installation of ZeroMQ



4. Install Protocol Buffers as the data format for message transfer. If **--version** is displayed in the command output, the installation is successful.


```
sudo apt-get install libprotobuf-dev protobuf-compiler
protoc --version
```

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 images.

```
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 Node.js.

```
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 the dependent library 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 Node.js 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

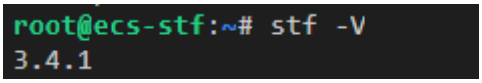
```
root@ecs-stf:~# node -v
v8.17.0
root@ecs-stf:~# npm -v
6.13.4
```


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

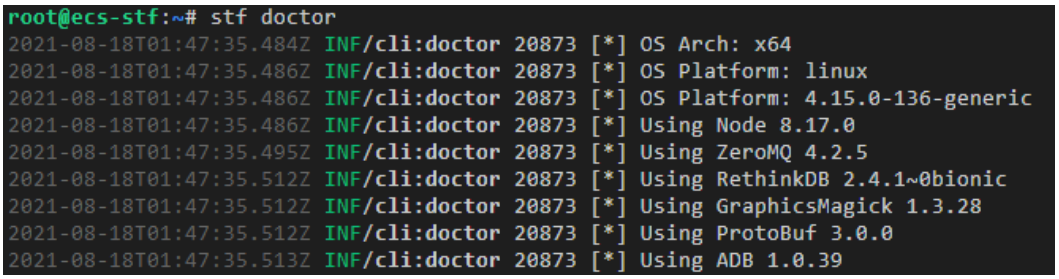


10. Check whether the environment STF depends on is available.

```
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

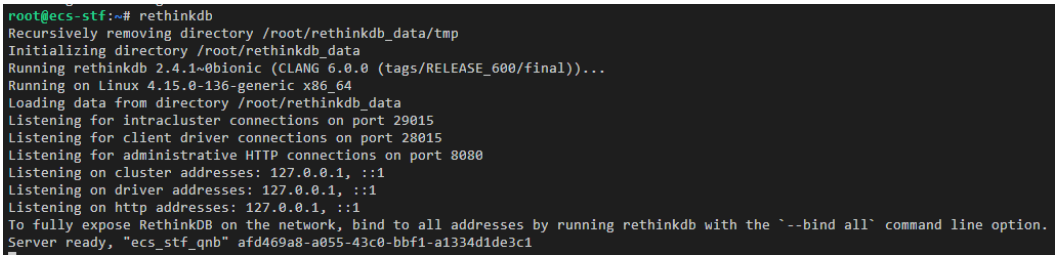


11. Use ADB to connect to the cloud phone. For details, see **ADB (Internet)**.
12. Start RethinkDB.

```
rethinkdb
```

If information similar to that shown in **Figure 6-11** is displayed, RethinkDB is started successfully.

Figure 6-11 Starting RethinkDB



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 Enter the default username and password of STF.

Figure 6-13 Cloud phone

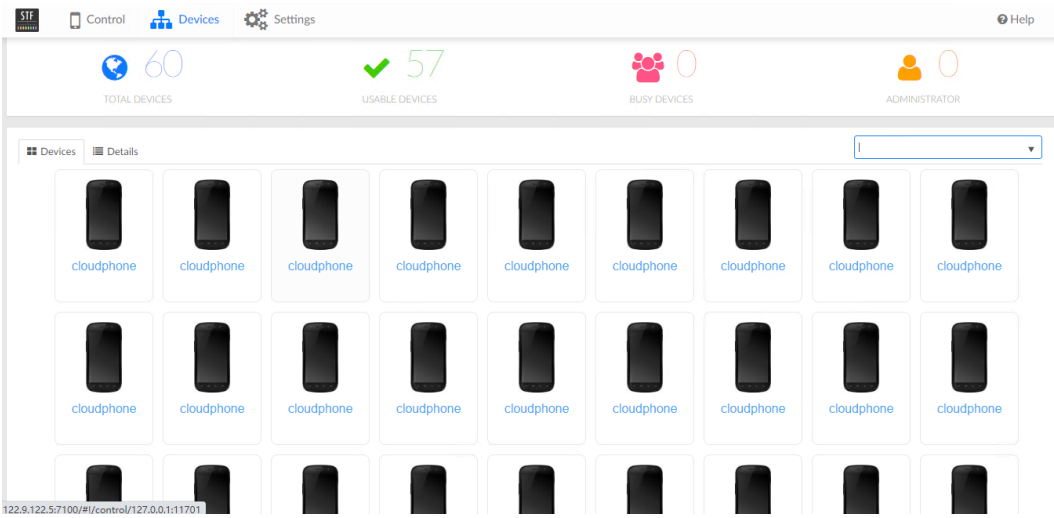
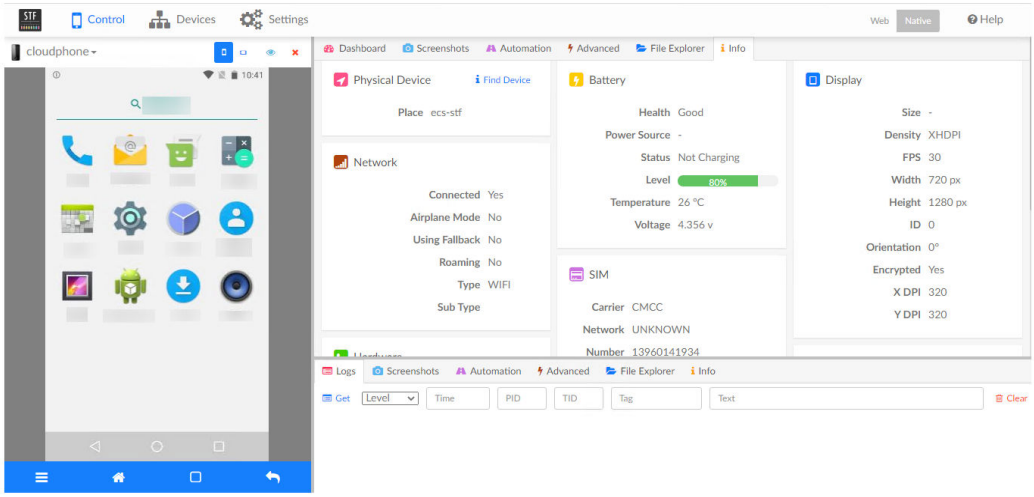


Figure 6-14 Cloud phone **Control** screen



A Reading Data from an OBS Bucket

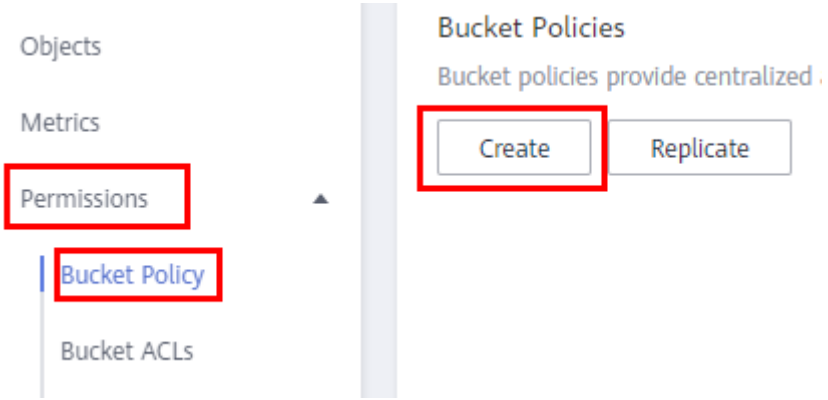
Prerequisites

The APK installation package, .tar packages for games, or other resources exist in the OBS bucket. Otherwise, you need to manually upload the game installation packages. For details, see **Step 6: Uploading an Object**.

Procedure

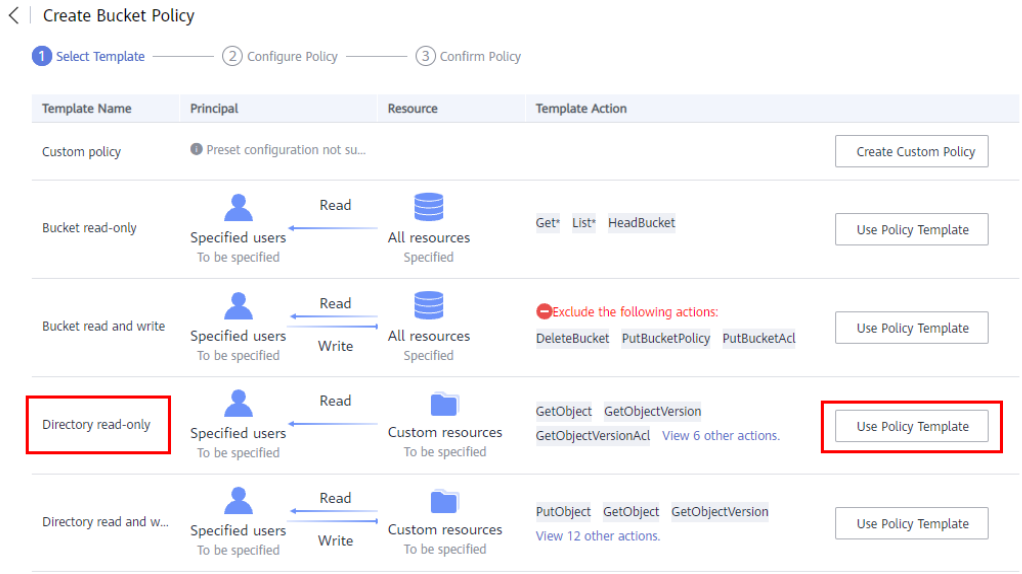
1. In the navigation pane on the left, choose **Permissions** > **Bucket Policy**. On the displayed page, click **Create**.

   Figure A-1 Creating a bucket policy

2. Select **Directory read-only** to grant only the read permission of the specified directory in the OBS bucket to the Cloud Phone built-in account. Click **Use Policy Template**.

Figure A-2 Select Template



3. For the **Configure Policy** step, configure the required parameters and click **Next**.
   - **Principal**: Select **Other account**.
   - **Account ID**: Enter the Cloud Phone built-in account.

     CAUTION

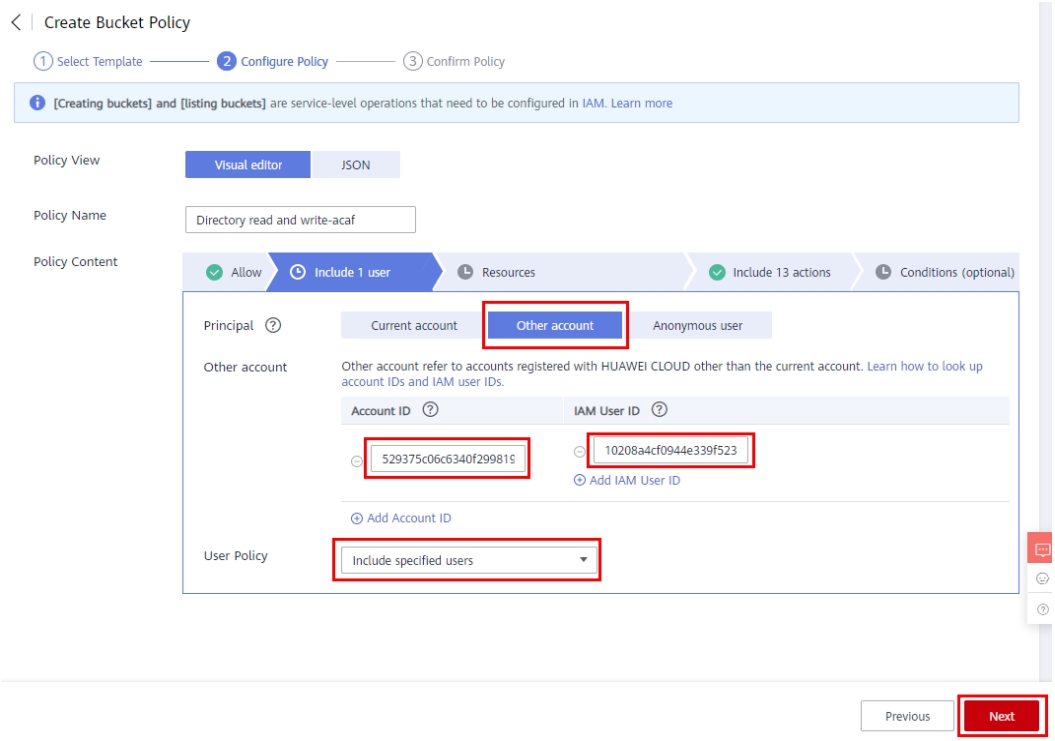
     The Cloud Phone built-in account is mandatory and must contain the following information. You cannot enter the ID of your own account.

     Account ID: **529375c06c6340f299819082b3051225**

     IAM User ID: **10208a4cf0944e339f523d9943ba02d3**

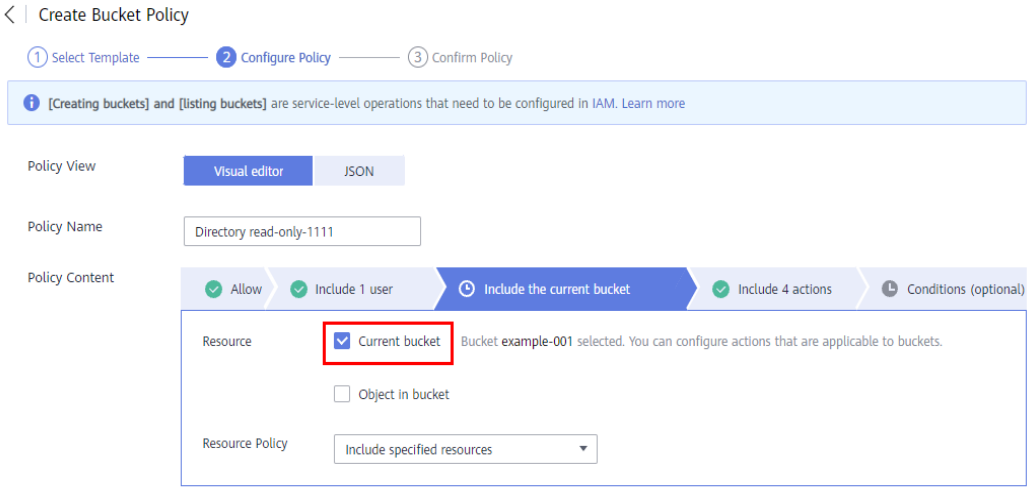
   - **User Policy**: Select **Include specified users**.

Figure A-3 Configure Policy



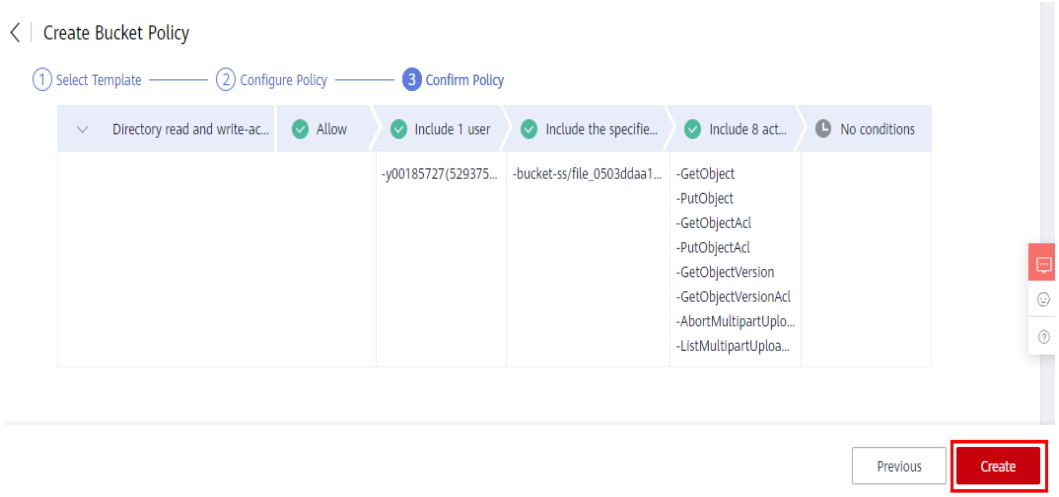
4. Select **Current bucket** for **Resource** and click **Next**.

Figure A-4 Resource



5. Confirm the bucket policy and click **Create**.

Figure A-5 Confirm Policy



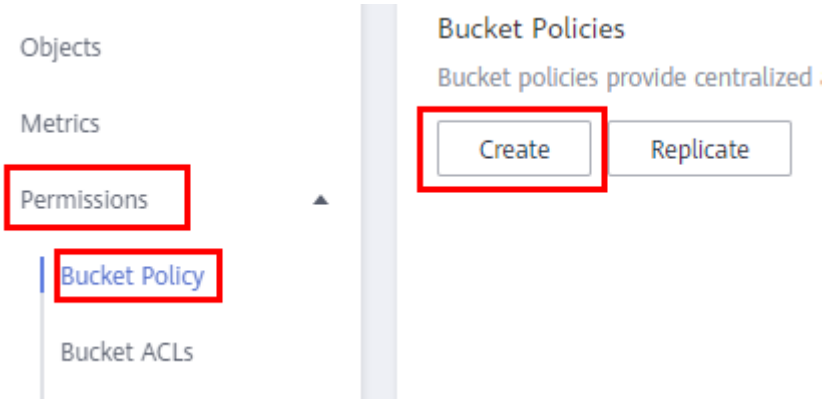
6. Check whether all cloud phones can only access the **game_tar** directory in the **cloud-app-game** bucket.

B Uploading Data to an OBS Bucket

Procedure

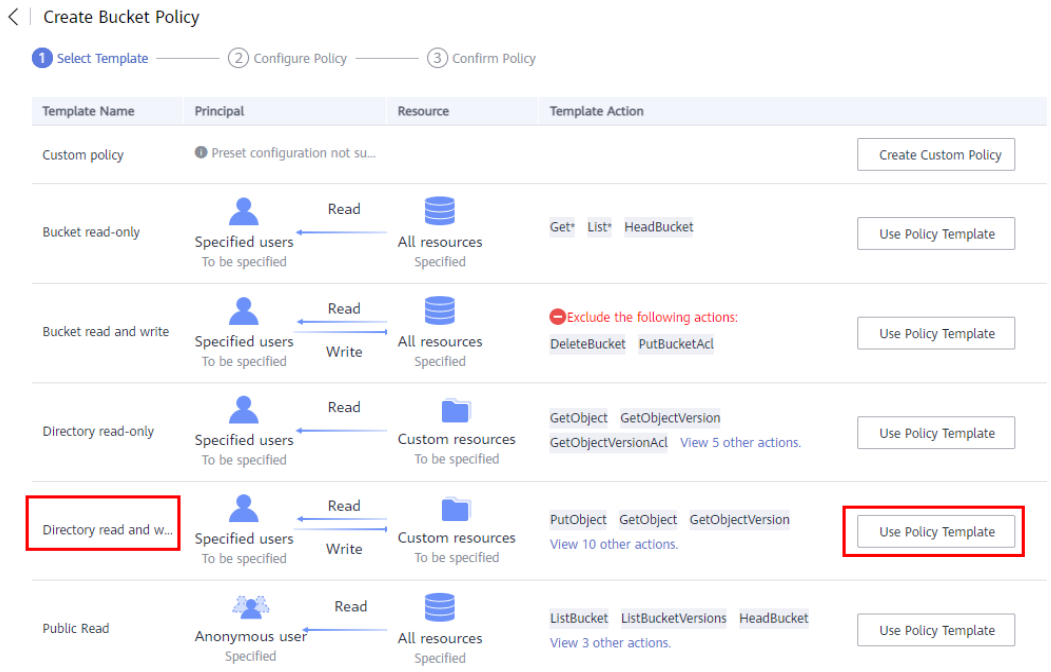
1. In the navigation pane on the left, choose **Permissions** > **Bucket Policy**. On the displayed page, click **Create**.

   Figure B-1 Creating a bucket policy



2. Select **Directory read and write** to grant the read and write permissions of the specified directory in the OBS bucket to the Cloud Phone built-in account. Click **Use Policy Template**.

Figure B-2 Select Template



3. For the **Configure Policy** step, configure the required parameters and click **Next**.
   - **Principal**: Select **Other account**.
   - **Account ID**: Enter the Cloud Phone built-in account.

     CAUTION

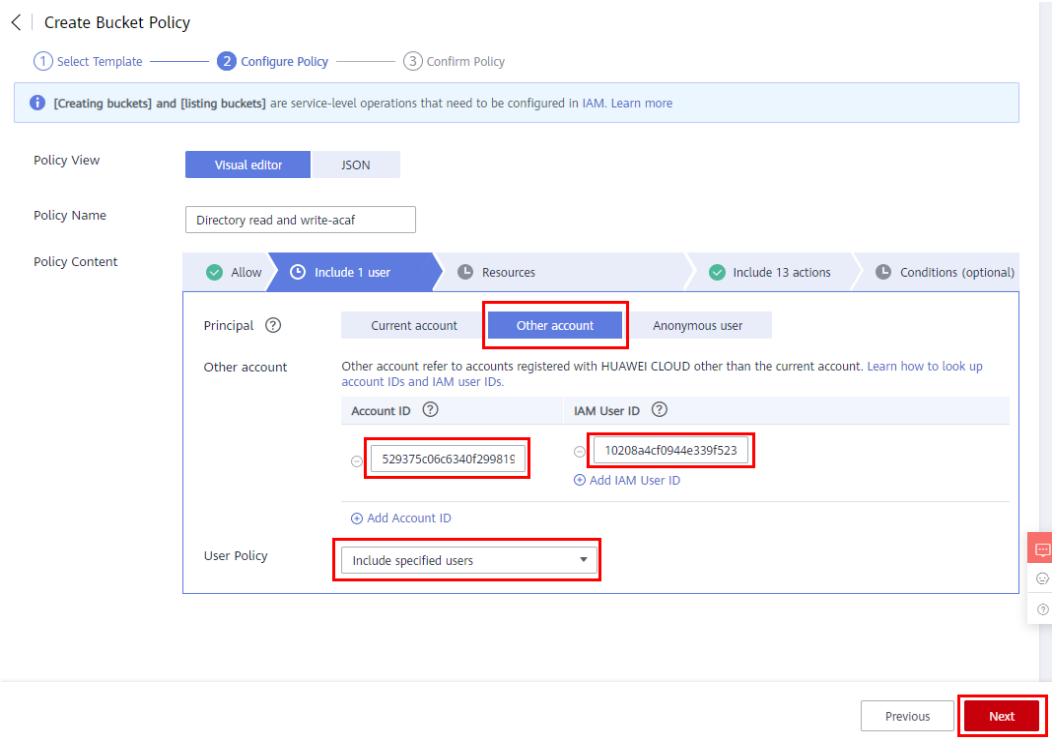
     The Cloud Phone built-in account is mandatory and must contain the following information. You cannot enter the ID of your own account.

     Account ID: **529375c06c6340f299819082b3051225**

     IAM User ID: **10208a4cf0944e339f523d9943ba02d3**

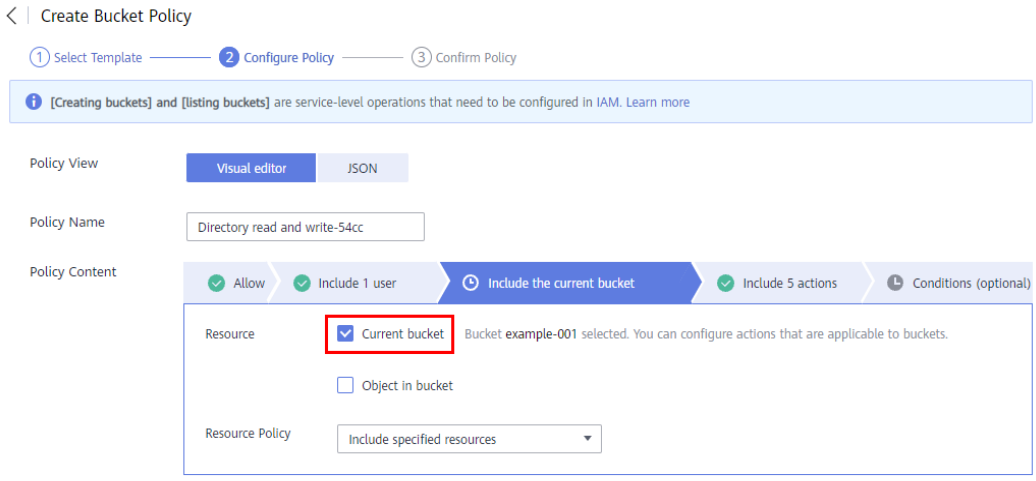
   - **User Policy**: Select **Include specified users**.

Figure B-3 Configure Policy



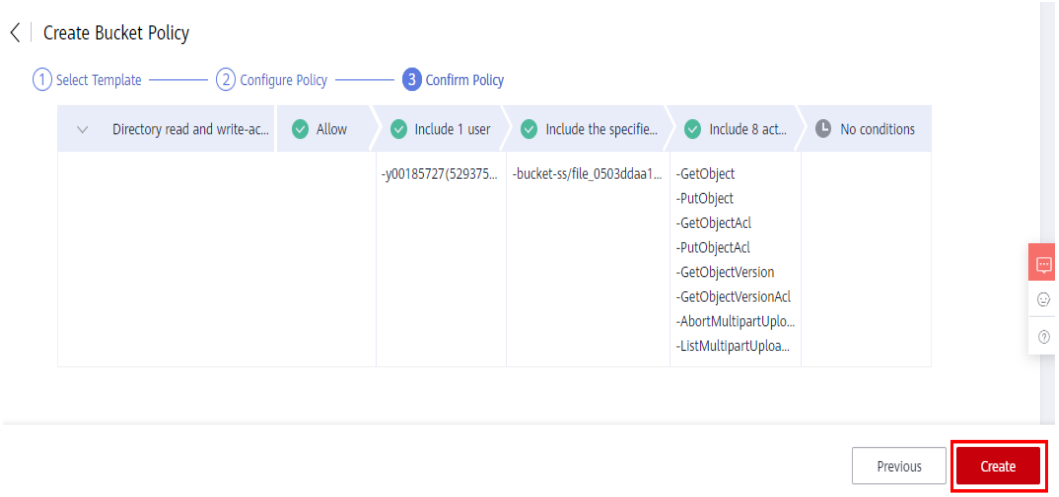
4. Select **Current bucket** for **Resource** and click **Next**.

Figure B-4 Resource



5. Confirm the bucket policy and click **Create**.

Figure B-5 Confirm Policy



6. Check whether all cloud phones can upload data to the **game_tar** directory in the **cloud-app-game** bucket.

C Change History

Released On	Description
2021-09-01	This issue is the fourth official release, which incorporates the following change: Added **Using STF to Manage Cloud Phones in Batches**.
2020-11-06	This issue is the third official release, which incorporates the following change: Added Sharing and Installing Apps from a Cloud Phone to Multiple Cloud Phones.
2020-02-18	This issue is the second official release, which incorporates the following changes: • Added **Using Cloud Phones to Build a Cloud Mobile Gaming System**. • Adjusted the document structure and optimized the content.
2019-08-30	This issue is the first official release.

