百度智能云

ARMCM 文档

目录

产品动态

镜像

新功能发布：2025-04-14

镜像ID	镜像名称	功能概述
507666707417677828	3588_aosp15_v1.0.0(2025-04-02)	1. 新系统发布，支持最新安卓15
8452004025491873823	8550_aosp13_v1.24.1(2025-03-13)	1. 升级dg5.13 2. 优化DuFS清理逻辑以及文件校验
4488836397410750482	3588_aosp12_v3.28.1(2025-03-13)	
4488836397410750477	3588_aosp10_v2.25.1(2025-03-13)	
4488836397410750483	845_aosp10_v1.33.1(2025-03-13)	
507666707417677838	845_aosp8.1_v2.25.0(2025-04-02)	1. 支持后台仿真服务授权管理功能 2. 优化配置导致虚拟机故障问题

发布时间：2025-03-14

镜像ID	镜像名称	功能概述
4488836397410750467	88550_aosp13_v1.22.0(2025-02-24)	1. 优化虚拟机启动逻辑及事件 2. 优化虚拟机通讯链路 3. 优化快照挂载状态 4. 优化快照卸载超时处理 5. 优化恢复出厂设置接口，解决偶现失败的情况 6. 支持摄像头1080P分辨率 7. 支持基于对象存储的备份还原
8452004025491873796	3588_aosp12_v3.26.0(2025-02-24)	
4488836397410750465	3588_aosp10_v2.23.0(2025-02-24)	
4488836397410750468	845_aosp10_v1.31.0(2025-02-24)	
4488836397410750469	845_aosp8.1_v2.23.0(2025-02-24)	

发布时间：2025-02-14

镜像ID	镜像名称	功能概述
6479414955299864588	8550_aosp13_v1.19.0(2025-01-06)	1. 优化设备DuFS开关 2. lmkd支持psi cgroupV1 3. 升级镜像GBoard输入法为64位
4930163260468854811	3588_aosp12_v3.22.0(2025-01-06)	1. 优化设备DuFS开关 2. 修复三键导航sdk控制开关 3. 升级dg5.12
4930163260468854810	3588_aosp10_v2.20.0(2025-01-06)	1. 优化设备DuFS开关 2. 修复三键导航sdk控制开关 3. 升级dg5.12

发布时间：2025-01-17

镜像ID	镜像名称	功能概述
4930163260468854788	3588_aosp12_v3.20.0(2024-12-16)	1. 优化IaaS软件模块信息采集规范 2. 修复同物理机并发内置应用问题 3. 优化sim配置、一键新机融合、proxy更新
561671636425011202	3588_aosp10_v2.18.0(2024-12-16)	1. 优化IaaS软件模块信息采集规范 2. 优化一键新机融合优化数据同步逻辑 3. 优化ROOT unix domain socket访问权限 4. 优化installApp创建目录的权限 5. 支持系统镜像SDK开启/关闭三键导航 6. 支持云手机设置指定dns 7. 修复mgr内置接口问题
561671636425011216	845_aosp10_v1.28.0(2024-12-27)	1. 修复mgr内置接口问题 2. 支持系统镜像SDK开启/关闭三键导航 3. 支持云手机设置指定dns 4. 修复重置后截图不实时
4930163260468854797	845_aosp8.1_v2.20.0(2024-12-25)	1. 支持禁用外网 2. 支持软件包按模块更新 3. 优化IaaS软件模块信息采集规范 4. 优化一键新机融合优化数据同步逻辑 5. 支持镜像定时服务 6. 优化ROOT unix domain socket访问权限 7. 支持云手机设置指定dns 8. 支持系统镜像SDK开启/关闭三键导航 9. 修复mgr内置接口问题

发布时间：2024-12-03

镜像ID	镜像名称	功能概述
6353272831439675415	3588_aosp12_v3.14.0(2024-10-29)	
6353272831439675406	3588_aosp10_v2.11.0(2024-10-14)	1. 去除个别模块，防止识别为异常环境。 2. 内置proxy插件更新，优化dns设置和特殊语言启动失败问题。 3. 修复dg dump location中gps显示异常问题。 4. 支持network/fuse location定位模拟。 5. 解决aosp10状态栏在重置后首次启动无法完整下拉的问题。

发布时间：2024-10-29

镜像ID	镜像名称	功能概述
3128683890169999368	3588_aosp10_v2.9.0(2024-09-10)	1. 限速、存储隔离问题优化 2. 日志管理优化；规范软件日志存放路径 3. DuFS状态优化
6848655575367237645	3588_aosp12_v3.10.0(2024-09-10)	1. Dufs状态优化；支持取消备份；错误码优化 2. 推流问题优化；日志规范化；ROOT开关优化
4011358924465319941	845_aosp8.1_v2.9.0(2024-06-18)	1. 限速功能、存储隔离优化 2. 优化版本记录

产品介绍

什么是云手机实例

云手机实例

百度云手机基于自主知识产权的磐玉蜂巢服务器和容器虚拟化软件技术，通过在云端虚拟的原生安卓系统实例，为客户提供高性价比、弹性灵活、稳定可靠的IaaS+PaaS服务。百度云手机产品已广泛应用于文娱、教育、电商、金融、新零售、餐饮、旅游等行业，基于端云同构、原生应用兼容、信息数据安全、规格灵活调整、资源弹性伸缩等技术优势，为客户提供云端原生安卓工作环境，助力客户业务的自动化和智能化实现。

产品优势

高性价比

百度磐玉蜂巢服务器支持多款主流芯片，高密设计架构可支持单台服务器虚拟上百台云手机，每台云手机均具有原生操作系统，更高效资源利用、更低运营成本

效能提升

百度云手机提供功能强大的管理后台，支持批量运维，统一管控，具有丰富SDK接口，可实现业务灵活配置、资源调度，提升运营效能

业务安全

百度云手机运行的客户数据的计算和存储在云端，数据不落地，业务运营环境统一配置管控，保护业务数据安全

应用场景

面向游戏行业

- 云手机可实现游戏免下载试玩、云手游等，赋予游戏全新的互动体验方式。

面向政企、金融行业

- 为企业提供更加安全高效的移动办公解决方案，公私数据分离，信息安全更有保障。

面向新媒体行业

- 面向私域营销场景，为企业提供云手机私域营销算力与SCRM系统

落地场景

云游戏

场景介绍：

面向云游戏和云试玩场景，目标客户包括游戏开发商、运营商、视频网站等泛游戏类客户。

场景特点：

主打多(支持游戏大作多)、快(即点即玩)、好(4K高清和高达60FPS帧率的极致游戏体验)、省(订阅模式)等优势。

云直播

场景介绍：

面向云直播和一播多场景，目标客户包括运营商、直播基地等。

场景特点：

主打直播监播、离线文件转直播、虚拟偶像直播、无人直播、一机多播等直播带货新玩法，节省运营成本（支持24小时在线，节省主播数量及终端成本，自动化处理直播素材）。

安全办公云手机

场景介绍：

面向政企办公场景，目标客户包括公检法司、央企等业务安全要求高的客户。

场景特点：

主打办公数据保密（数据可用不可取，端外一律不可见；云端调取数据，终端显示信息；统一部署发布，分级设置权限）、工作和生活终端分离。

实例规格

实例规格

百度云手机的计费模式为**预付费和按量后付费**模式。具体规则如下：

- 预付费需要提前一次性支付几个月或者几年的费用。
- 按量后付费提交试用申请，然后提交工单联系客服。
- 购买实例前需保证账户无欠款。

实例规格详情如下：

以下为单片云手机物理机规格，单片物理机可自定义虚拟多个云手机实例：

商品	说明
云手机 G392 旗舰型	基于高通845，配置8核CPU8G内存128G存储，适用于云手游场景
云手机 R296 旗舰型	基于RK3588，配置8核CPU16G内存256G存储，适用于云游戏、云托管、云营销等场景
云手机 G272 旗舰型	基于高通8550，配置8核CPU24G内存512G存储，适用于云游戏、数字人直播、AI大模型实训等场景

通用部分

产品术语

- 磐玉蜂巢服务器

  磐玉蜂巢服务器是百度自主研发的基于ARM架构打造的具有仿生算力、高能效比、高性价比、高密设计的云边协同算力矩阵。为企业级数据中心和边缘计算场景提供高密服务器，广泛应用于安卓云算力和通用算力场景。

- 云手机Android系统

  基于AOSP（Android Open-Source Project）开源源码进行定制研发，可在ARM板卡运行，并且具备虚拟化能力，同一物理设备可运行多个Android实例，云手机Android系统与真机一致，并具备高度定制能力，满足多类业务需求。

- PaaS服务

  提供在线手机服务的能力，使虚拟机在安全、可视、可监管的环境下运行，使用视频传输的方式将系统画面传输至用户终端产品上，为用户提供传统手机的云端操作模式；同时通过管理平台以租户的形式对内对外提供服务，各租户由各自的管理员进行监控与授权。

- SaaS服务

  在云游戏、云直播等场景，百度致力于与合作伙伴共同建设行业解决方案，促进和推动生态发展。

- 全平台接入SDK

  提供Android SDK和H5 SDK接入方式，为开发者接入云手机、云游戏产品提供全平台接入方式。

什么是云手机（速享版）

1 产品概述

1.1 服务介绍

云手机（速享版）是面向以运营能力为主的客户，提供全套云手机端能力+运营管理平台的产品，旨在让客户用较少的技术投入，快速搭建自己的云手机产品，轻松打造自主云手机品牌。

1.2 与云手机服务的差异

- **客户端能力：**云手机（速享版）产品提供PC客户端、安卓APP、H5端及管理平台，客户端提供完成的UI交互功能，可满足C端用户各场景需要。
- **SDK能力：**针对自有业务的商户，云手机（速享版）产品提供SDK，支持商户在自己的APP中集成SDK，便捷集成，快速入驻商户开放平台。
- **运营能力：**商户可根据业务规划，自主定义商品套餐在自主品牌的客户端APP中进行销售获取盈利。同时，云手机（速享版）产品提供商户开放平台为商户提供云手机平台管理服务，支持商户自主管理用户、云手机、商品、订单、权限等操作，商户可根据业务需要发起运营活动。
- **应用市场能力：**云手机（速享版）产品镜像中提供应用市场，商户可内置应用，支持应用一键安装。

产品优势

- 功能丰富

  提供完整客户端、SDK等接入方式可选，已经支持常用云手机平台所需要的各项功能，对齐一线云手机平台。

- 快速接入

  无需漫长开发及测试周期，成品APP及管理平台支持快速商用。

- 应用兼容

  兼容市面常见应用，用户体验与主流平台保持一致。

- 极低投入

  无需研发人员及测试人员投入，大幅降低上线成本。

应用场景

云游戏

免安装，即点即玩，减少游戏软件与手机硬件的兼容性问题，低配手机玩高配游戏。

云托管

支持设备24h在线，设备稳定，数据安全，一键托管，省心无忧。

快速入门

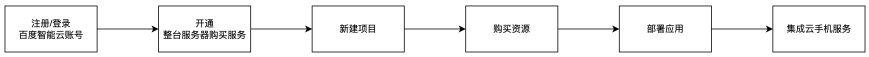
快速入门流程

概述

百度云手机是基于自研虚拟化技术，具有安卓实例及手机功能的云服务器。您可以远程实时控制云手机，实现安卓App的云端运行。基于云手机，可高效搭建应用，如云营销、云游戏、云直播和安全办公云手机等场景。

云手机接入流程

云手机接入流程如下图所示：



1.注册及实名认证：注册百度智能云账号，并完成企业实名认证。
2.开通云手机服务：申请免费试用国内/海外云手机，然后进入管理控制台，再进入默认项目测试试用服务。
2.购买云手机实例：进入管理控制台的国内业务/海外业务，新建项目，进入项目的物理机管理列表，通过购买入口下单国内/海外云手机。
3.使用云手机实例：进入管理控制台的项目，通过实例管理列表进行实例操作，点击“更多管理功能”进行应用管理、镜像管理、SDK下载等更多操作。

如需要批量购买云手机，或需要提供定制化服务，可提交工单联系我们。

注册账号

首先您需要注册/登录百度智能云账号，并完成企业实名认证。

开通云手机服务

1.申请免费试用：打开云手机的产品介绍页，点击“申请免费试用”，提交您的需求，您的需求会在5分钟内完成审核，并且百度智能云商务人员会在24小时内与您建立联系。如未收到，可提交工单联系我们。
2.试用云手机：访问云手机管理控制台，进入业务管理，进入已开通的默认POC项目，查看并试用免费云手机资源和服务。

购买云手机实例

1.新建项目：访问云手机管理控制台，左侧导航栏选择“业务管理”，进入国内业务/海外业务，点击“创建项目”，提交项目基本信息。
2.下单购买：进入刚创建的项目，左侧导航栏选择“物理机管理”，通过物理机列表页面的“购买”入口，进入下单页。下单页选择云手机规格、付费方式、创建实例个数、下单数量、时长等信息，然后提交订单支付即可。

（温馨提醒：因云手机配套带宽为后付费商品，下单前请确认账号最少有500元余额。如批量下单云手机时发现库存不足，可提交工单联系我们。）

使用云手机实例

1.部署应用：购买资源后，回到相应项目，左侧导航栏选择“更多管理功能”，进入业务管理平台，上传应用文件，完成应用安装部署，为后续集成客户端 SDK 时，获取指定实例资源的推流信息做准备。
2.集成云手机服务：如需接入SDK（Android、Web/H5）、OPEN API，开发定制化的客户端，集成云手机服务，请进入相应项目，左侧导航栏选择“更多管理功能”，进入业务管理平台，平台右上角“帮助文档”自行下载相应文件。

如您在云手机服务接入过程中遇到问题，欢迎提交工单联系我们。

购买云手机实例

概述

本文将向您介绍如何通过控制台购买云手机实例。

备注：如您对以下操作有疑问，请提交工单联系我们。

前提条件

- **注册百度智能云账号并实名认证**

  云手机仅限于为企业用户提供服务，使用百度智能云云手机BAC前，需要拥有一个百度智能云账号并完成企业实名认证，具体操作如下：

  1. 注册并登录百度智能云平台，请参考 注册 和 登录。
  2. 完成实名认证，操作细节请参考 实名认证，完成企业实名认证后才可购买百度智能云云手机BAC。

购买云手机

百度云手机支持物理机灵活购买和整台服务器购买，整台服务器购买可提交工单联系我们。 购买流程图：

> **申请接入百度云手机-->接入审核-->审核通过-->购买实例**

1. 新用户初次登录云手机 BAC 管理控制台，进入云手机BAC控制台。
2. 自动弹出“欢迎接入百度云手机”的申请表单，同时我们为新用户提供免费试用的限时限量福利，表单填写试用需求。
3. 填写申请表单，点击“确认”，进入到审核阶段，提交后预计10分钟内完成审核。
4. 审核通过后，回到云手机 BAC 管理控制台，点击**业务管理 > 国内业务/海外业务 > 创建项目 > 进入项目 > 物理机管理 > 物理机列表**，通过“购买”进入下单界面。
5. 根据需求选择所需要的配置后，点击**确定**，进入“确认订单”页面。
6. 确认选择的云手机套餐和价格后，点击**确定**，进入支付环节。
7. 点击**确认支付**，完成支付。支付成功后，系统在后台进行云手机实例的创建。
8. 购买成功后，回到**管理控制台**，进入“实例列表”界面，可以查看到对应购买实例列表。

> 温馨提示：目前云手机BAC产品支持物理机购买，单片物理机支持灵活多开。即下单数量为物理机，配置创建实例数量为云手机多开实例数。

计费模式

百度云手机包括国内云手机和海外云手机，云手机服务费用包括云手机资源服务费和带宽费用。

云手机资源支持预付费和后付费两种计费方式；带宽支持按带宽月95峰值、按带宽月峰值、按带宽日峰值和按实际流量四种后付费计费方式。可根据业务需要，选择不同的计费方式订购云手机服务。

云手机资源

- **预付费模式**

定义：先付费后使用服务。您可通过预充值账户余额直接下订单购买云手机，或通过百度智能云支持的支付方式直接下订单购买，下单时扣费。足额付费且成功订购后，百度智能云才开始为您提供服务。

预付费方式式支持包年包月预留资源，享有更优惠的价格，适合中长期稳定业务需求。

- **后付费模式**

定义：先使用服务后付费。您购买百度智能云按订购量计费的服务。您可先开通和使用云手机服务，百度智能云按实际使用时长计算费用、出具账单，并按账单金额扣减您的百度智能云账户余额。

后付费模式支持以天为单位进行计费，适合有短期弹性云资源需求的业务场景。

不同计费模式的对比说明如下：

计费模式	计费周期	适用场景	时间限制
包年包月	包年/包月预付费	适合中长期稳定业务需求	单次购买时间1个月起购
按量计费（按天）	以天为单位后付费	适合短期弹性云资源需求	无起购时间限制

带宽流量

带宽支持按带宽上限计费和按实际流量计费两种模式，其中，带宽上限计费模式支持多种计费方式。

不同计费方式的对比说明如下：

计费模式	计费方式	适用场景	费用计算
带宽	按带宽月95峰值	适用于流量较大或较稳定的场景	按当前业务实际使用的公网带宽月95峰值计费，按月结算
	按带宽月峰值	适用于流量较大或较稳定的场景	按当前业务实际使用的公网带宽月峰值计费，按月结算
	按带宽日峰值	适用于流量波动较大的场景	按当前业务实际使用的公网带宽日峰值计费，按天结算，隔两日出账 例：2024年5月1号的日峰值带宽用量，将统计在2024年5月2日的账单周期，于2024年5月3号出具账单再进行扣费
流量	按实际流量	适用于流量较小的场景	按当前业务实际使用的公网实际流量计费，按日结算 例：2024年5月1号的实际流量用量，将统计在2024年5月2日的账单周期，于2024年5月3号出具账单再进行扣费

> 温馨提示：同个机房，带宽的计费方式只可开通一种，具体情况可联系百度智能云沟通，确认后再下单。

起购限制

云手机服务开启后，会同步开通实时音视频传输功能。在使用过程中，会产生相应的带宽用量，并按周期从您的账户余额中扣除相关费用。为避免由于账户余额不足导致服务中断，申请购买云手机服务时，您的账户总资产必须达到500元以上。如果账户余额低于此限额，将无法完成购买。

续费说明

在您的百度智能云余额不足以支付您下一个服务周期的服务费、当前余额不足以支持当前服务，或您此时的服务周期即将到期时，我们会通过站内信/邮件/短信的方式对您进行提醒，避免影响您的服务体验。

提醒类型	提醒方式	提醒时间及频次	操作
预付费延期续费	站内信/邮件/短信	提前7/3/1天和当天，每次发送一次提醒	若您开通了自动延期续费，请前往控制台确认延期时间。我们在延期续费节点进行自动延期续费时，默认优先使用代金券，您可在支付时进行取消优先使用代金券的延期续费方式。
后付费余额不足	站内信/邮件/短信	提前3/2/1天和当天，每次发送一次提醒	在您在提醒周期内没有及时进行充值，我们会终止您的服务，对您的数据及信息进行统一释放。

到期回收规则

当您购买了包年包月的资源，到期后状态变化如下：

1. 到期7天内，到期资源进入赎回期，到期资源会变为过期关停状态，云手机实例无法继续使用，平台会继续保留您的云手机数据，期间续费成功后即可恢复正常使用。

2. 到期超过7天，云手机实例将会被回收释放，资源回收后相关数据也将被清空。

退订说明

如果您在购买后有任何不满意，均可通过提交工单联系我们。

1. 云手机预付费资源，如遇到产品不可用问题（如云手机连接闪退、云手机系统不兼容应用等），购买后7天内可退订，费用以系统实际计费天数为准来收取，超过7天不支持退订。退订申请通过后，您的云手机实例将会被立即回收释放，资源回收后相关数据也将被清空。
2. 云手机后付费资源，如需退订，请提前30天提交工单发送退订申请，如退订申请通过，则自工单发出30个自然日后该退订申请生效，退订生效后，您的云手机实例将会被立即回收释放，资源回收后相关数据也将被清空。

请您在仔细了解并试用云手机后再进行购买。

在您退订申请通过后，预付费对应金额返回至您的百度智能云账户中。

> 温馨提示：云手机预付费资源，非产品不可用问题，不支持退订，百度智能云负责对本协议进行解释说明。如出现疑似异常/恶意退货，百度智能云有权拒绝您的退订申请

解释权说明

购买指南的最终解释权归百度智能云云手机所有。

使用云手机实例

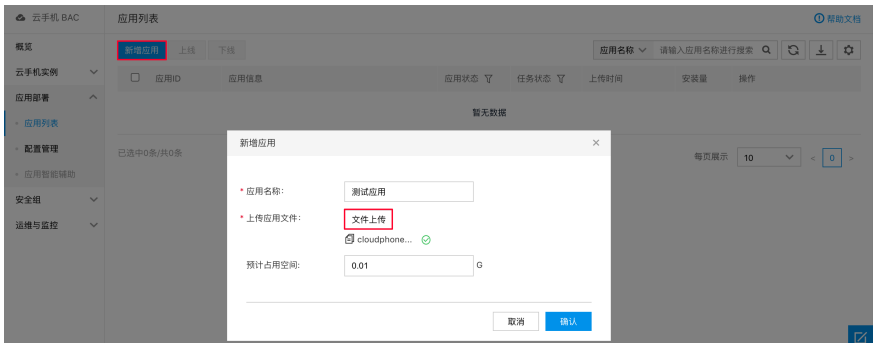
概述

您可以通过控制台使用和管理云手机应用与实例。

使用云手机实例

新增应用

1. 登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。
2. 在左侧导航栏点击**应用部署 > 应用列表**，进入应用列表页面。
3. 点击**新增应用**按钮，用户可以根据个人需要上传所需要使用的应用。



4. 填写应用名称，选择上传的应用文件后，点击**确认**即可完成应用上传。
5. 上传成功后，可以在应用列表查看该应用的“应用ID、应用信息、应用状态、任务状态、上传时间、安装量”等信息，并且可以对该应用进行“安装”、“卸载”、“更新版本”等操作。

上线应用

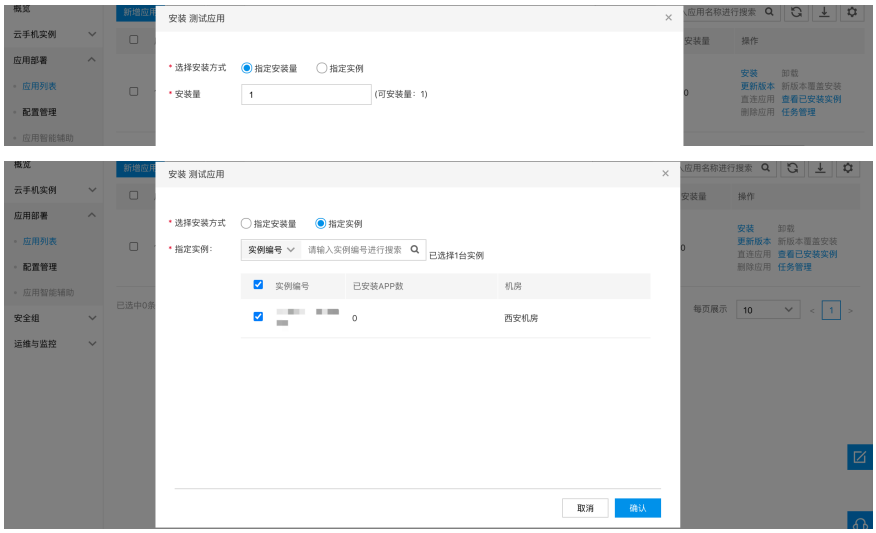
1. 登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。
2. 在左侧导航栏点击**应用部署 > 应用列表**，进入应用列表页面。
3. **勾选**应用列表中想要上线的目标**应用 ID**，点击**上线**按钮，在弹出的确认框中点击**确认**。

4. 上线后，该应用的“应用状态”由“下线”变为 “**上线**”。

安装应用

1. 登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。
2. 在左侧导航栏点击**应用部署 > 应用列表**，进入应用列表页面。
3. 用户可以选择将需要使用的应用安装到购买的实例上。找到目标应用，在其操作栏点击**安装**按钮。

5. 在弹出的安装应用窗口可以选择**指定安装量**或**指定实例**的安装方式，填写**安装量**或选择**指定实例**后，点击**确认**，完成应用安装。
   - 指定安装量：指定选择录入实例设备数量。
   - 指定实例：指定选择录入实例设备号（1台或多台）进行安装。



6. 页面提示“安装操作已发起，预计耗时5分钟”，等待安装完成后，您可通过操作列的**任务管理**或**查看已安装实例**查看安装状态。

查看已安装实例

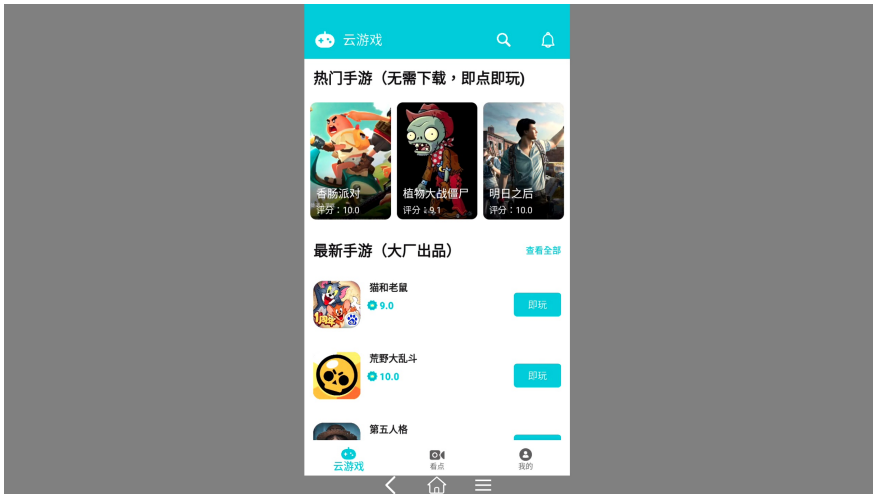
1. 登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。
2. 在左侧导航栏点击**应用部署 > 应用列表**，进入应用列表页面。

3. 找到目标应用，点击操作列的**查看已安装实例**按钮，进入查看已经安装列表页面，在此页面可以查看此应用安装的实例列表，以及实例编号、已经安装APP数、已经安装APP名称、故障状态、机房、应用版本等信息。

直连应用

1. 登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。
2. 在左侧导航栏点击**应用部署 > 应用列表**，进入应用列表页面。
3. 找到目标应用，点击操作列的**直连应用**按钮。

4. 应用可以连接任意或指定的云手机，页面跳转到云手机页面。



试玩云手机实例

百度云手机提供试玩模块，方便其快速了解和使用云手机

说明：

1. 公测用户即试用用户，申请公测成功后，有7天的试用时间，7天试用结束后回到新用户状态。
2. 剩余试玩时间：申请到的公测设备只能试用7天，7天后自动收回，这里展示剩余的试玩时间，倒计时精确到秒。
3. 立即试玩：申请公测成功后，可以在控制台安装应用并试玩。

镜像管理

镜像管理页面

功能开通说明

镜像功能为增值服务，欲开通此功能，至少需拥有20台云手机实例。如需开通，请提交工单，在工单中提供下述必要信息：

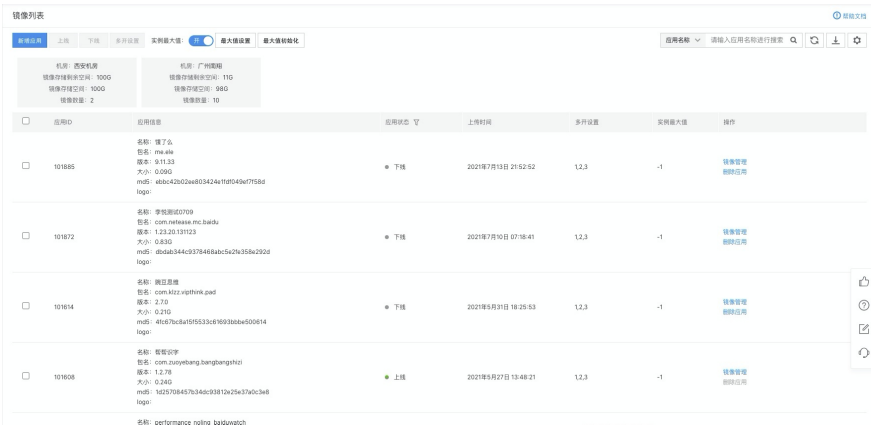
1. 业务场景
2. 公司年营收规模
3. 开通镜像功能的目的
4. 当前持有的云手机实例数量

镜像管理功能

镜像列表

1. 登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。

2. 在左侧导航栏点击镜像管理 > 镜像列表，进入镜像列表页面。

- 用户在镜像列表页面可以统一查看和管理镜像应用。
- 镜像管理以应用纬度展示列表，可点击对应应用的镜像管理
- 页面展示每个机房的镜像存储空间和镜像数量
- 

新增镜像应用

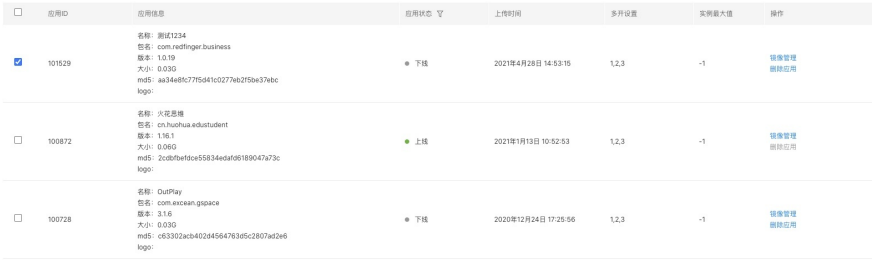
1. 登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。

2. 在左侧导航栏点击镜像管理 > 镜像列表，进入镜像列表页面。

3. 点击新增应用按钮，用户可以根据个人需要上传所需要使用的应用。

4. 填写应用名称，选择上传的应用文件后，点击确认即可完成应用上传。

上传成功后，可以在镜像列表查看该应用的“应用ID、应用信息、应用状态、任务状态、上传时间、多开设置、最大实例数”等信息，并且可以对该应用进行“镜像管理”、“删除应用”等操作。



上线应用及多开设置

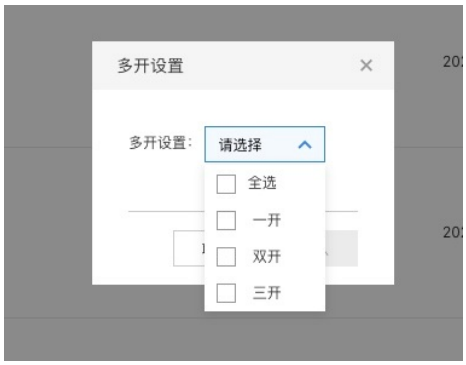
1. 登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。

2. 在左侧导航栏点击镜像管理 > 镜像列表，进入镜像列表页面。

3. 勾选应用列表中想要上线的目标应用 ID，点击上线按钮，在弹出的确认框中点击确认。

4. 上线后，该应用的“应用状态”由“下线”变为 “上线”。

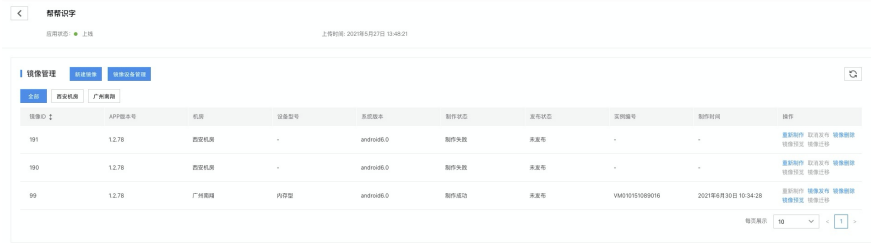
5. 点击多开设置可以控制安装镜像应用的设备类型



最大值设置 1、登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。 2、在左侧导航栏点击镜像管理 > 镜像列表，进入镜像列表页面。 3、点击列表上方的最大值设置，可以设置镜像应用的最大安装数量，初始默认值为"-1"，表示实例安装数量无限制，如需限制，设置调整"实例最大值"，可填写"0-999999"之间任意整数值，设置完成后，点击确定生效

镜像管理：

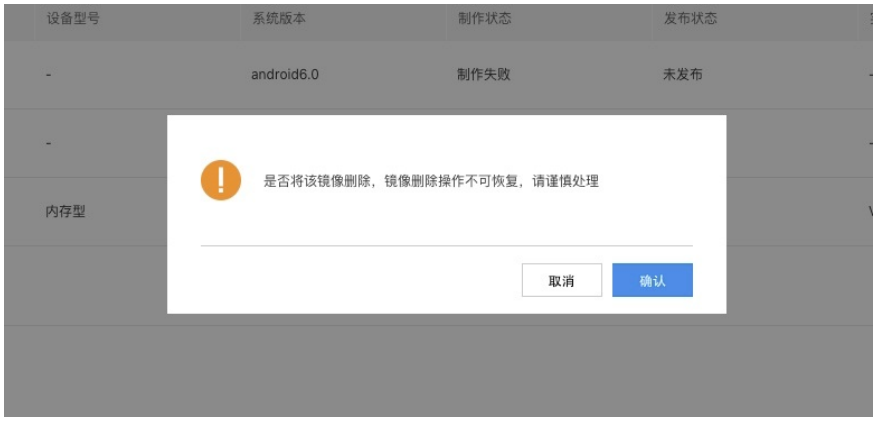
1. 登录云手机 BAC 管理控制台，进入云手机 BAC 概览页面。
2. 在左侧导航栏点击镜像管理 > 镜像列表，进入镜像列表页面。
3. 点击镜像管理后对应的会以机房为纬度展示每个机房的镜像列表类型的页面，如下图



操作栏中包含 5 个按钮，

- 镜像发布/取消发布（即镜像挂载）

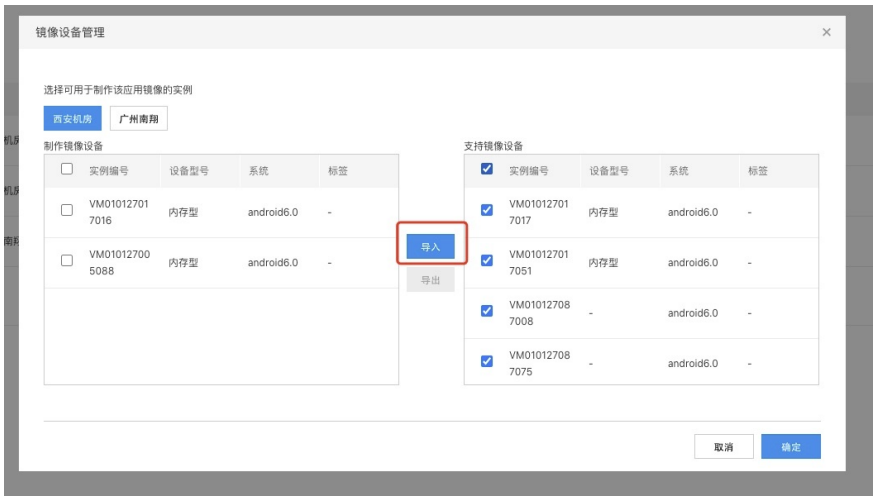
- 镜像删除（已发布的镜像不可删除，制作成功未发布的镜像，将镜像文件删除，未制作成功的镜像，将设备初始化并释放掉）



- 镜像迁移（本期仅保留前端入口，全部置灰不可点击）
- 重新制作（制作卡死或制作失败时可点击再次制作）
- 镜像预览（所有制作成功的镜像，均可进行镜像直连预览，随机选择一台实例进行镜像的预览）

镜像设备管理

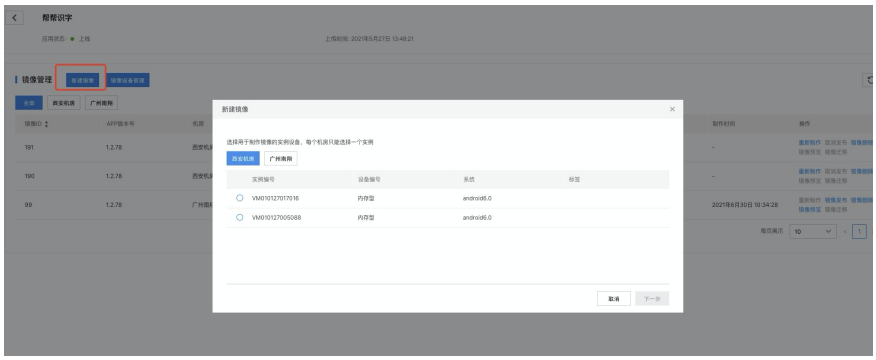
1. 点击页面镜像设备管理，可选择可用于制作该应用镜像的实例
2. 用户从所有可制作镜像的实例中，选择可用于制作该应用镜像的实例



镜像制作

点击页面新建镜像，出现弹窗

1. 用户自定义选择母版机
2. 选择用于制作镜像的实例设备，每个机房只能选择一个实例。
3. 列表仅展示当前可用于制作该应用镜像的实例设备。
4. 切换tab，可选择不同机房的实例
5. 在未勾选实例设备时，”下一步“按钮处于不可点击状态；在单选实例设备后，”下一步“可点击。



1. 点击”下一步“后，选择制作完成后是否清理应用。
2. 配置清理目录，按钮后方有蓝色问号提示按钮，点击后提示文案：“请谨慎填写，该操作可能会导致某些功能无法正常使用,目录示例：目录示例：data/data/com.tencent.mm/MicroMsg”。

点击后开始制作镜像并弹窗关闭，页面弹出loading加载页面

安装结束后，会弹窗提示：是否确认制作镜像

产品定价

计费概述

云手机服务开启后，会同步开通实时音视频传输功能。在使用过程中，会产生相应的带宽用量。云手机服务费用 = 云手机资源服务费 + 带宽费用。

计费规则

计费项	计费方式	配置	计费周期	说明
云手机资源	预付费	• 8vCPU | 8GB内存 | 128GB存储 • 8vCPU | 16GB内存 | 256GB存储	包月	• 先付费后使用的方式。下订单时，根据云手机数量、订购时长确定云手机服务费用，付费后资源生效可使用。 • 包月云手机资源服务费=单次订购实例数量*订购时长*包月单价（时长最小单位为1个月起）
	后付费	• 8vCPU | 8GB内存 | 128GB存储 • 8vCPU | 16GB内存 | 256GB存储	按日	• 客户订购当天开始计费，按日峰值订购量计算。 • 日云手机资源服务费=日订购云手机峰值数量*云手机按日单价。 • 按日为结算周期，每日出具前一日服务账单并按账单金额扣费
带宽流量	后付费	按月95峰值	按月	• 在一个自然月内，每5分钟统计同一时刻入网带宽和出网带宽数据较大值，每日得到288个采集值，然后将所有的采集值进行降序排序，其中Top 5%的采集值去掉，剩余最高值为月95峰值带宽。 • 每月带宽费用=月带宽95峰值*月95峰值带宽单价 • 按月为结算周期，每月前4个工作日内出具上一自然月服务账单并按账单金额扣费
	后付费	按月峰值	按月	• 在一个自然月内，每5分钟统计同一时刻入网带宽和出网带宽数据较大值，每日得到288个采集值，然后将所有的采集值进行降序排序，取最高值计量。 • 每月带宽费用=月带宽峰值*月峰值带宽单价 • 按月为结算周期，每月前4个工作日内出具上一自然月服务账单并按账单金额扣费
	后付费	按日峰值	按日	• 在一个自然日内，每5分钟统计同一时刻入网带宽和出网带宽数据较大值，每日得到288个采集值，然后将所有的采集值进行降序排序，取最高值计量。 • 每日带宽费用=日带宽峰值*日峰值带宽单价。 • 按日为结算周期，每日出具前一日服务账单并按账单金额扣费
	后付费	按实际流量	按日	• 在一个自然日内，产生的所有流量之和。 • 每日流量费用=日实际流量*流量单价 • 按日为结算周期，每日出具前一日服务账单并按账单金额扣费

> 按月带宽计费的特殊说明:

1. 计费周期：按自然月计费
2. 结算规则：按照当月带宽费用乘以有效因子计费

- 有效天数：按月95峰值计费生效日开始到该月月末的天数。例如2021年11月5日为按月95峰值计费生效时间，则2021年11月的有效天数为26天。次月起，有效天数为整个自然月的天数。
- 有效因子：以一个自然月的天数为分母、该自然月内的有效天数为分子计算得到的小数。假设2021年11月的有效天数为26天，则有效因子为26/30=0.8666666667。

服务关停

预付费到期关停

- 预付费实例到期日前，可操作续费，成功续费即能继续正常使用。
- 未及时操作续费的，预付费实例到期日后超过7天，回收云手机实例资源，资源回收后相关数据将被清空。

账号欠费关停

- 账户可用余额不足或账户可用余额小于0被扣为负值即为欠费。
- 账号欠费后7天内补缴所有欠费账单的，可正常使用云手机服务。欠费后7天内仍未能补缴所有欠费账单的，将关停云手机所有服务（包括实例和带宽），关停后云手机服务将无法使用，充值可恢复正常。欠费7天后回收云手机实例资源和带宽资源。
    - 回收云手机实例资源后，实例列表将不可见相关实例。
    - 回收带宽资源后，实例将无法进行公网访问。

退订说明

- 云手机预付费资源和后付费资源的具体退订规则请参见[购买云手机实例]的退订规则说明。
- 以单个百度智能云账户维度发起云手机预付费实例退订申请，费用以系统实际计费天数为准来收取。
- 在您申请退订时，若相应产品/服务在购买时参与“活动”或使用“代金券”，优先适用当时活动或代金券相关规则。
- 退订申请通过后，相关退款金额返回至您的百度智能云账户中。

如百度智能云认为您的退订退款申请为疑似异常/恶意退订，百度智能云有权要求您配合提供合理理由以审核您的退订退款申请或拒绝您的退订退款申请。

关于价格咨询，欢迎提交工单联系我们。

技术文档

云手机Android SDK

百度云手机Android-SDK接入说明

1.综述

云手机：用户在运营平台购买实例后，通过接入SDK，指定进入云手机模式。

2.1 接入准备

2.1.2环境要求

- minSdkVersion 21（android 5.0）及以上
- GLES 2.0及以上
- gradle版本6.1.1
- sdk只支持arm架构android环境（支持32及64位系统）

2.2 SDK集成

2.2.1 aar集成方法

将sdk（.aar文件）拷贝到项目libs(**没有libs目录手动创建一个**)目录下，并在项目模块的build.gradle中配置相应参数。

```
 android {
   defaultConfig {
      ndk {
         abiFilters "arm64-v8a","armeabi-v7a"
      }
   }
   repositories {
      //libs 目录
      flatDir {
         dirs "libs"
      }
   }
   dependencies {
      implementation fileTree(include: ['*.aar'], dir: 'libs')
      ...
   }
}
```

2.2.2 sdk所需权限

```
 <uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<!---------------- 下面两项按需添加 ---------------------->
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

2.3 混淆规则

集成sdk后如需混淆，需要在相应的proguard-rules.pro文件中添加如下混淆规则

```
 -keep class com.mci.** { *; }
-keep class com.baidu.armvm.mciwebrtc.** { *; }
```

2.4 布局组件

在布局文件中添加**SdkView**组件

```
<com.baidu.armvm.api.SdkView
    android:id="@+id/sdk_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_centerInParent="true"
    android:gravity="center"/>
```

2.5 Activity中接入SDK

2.5.1 接入流程图

连接实例流程图

断开连接实例流程图

2.5.2 预加载，初始化UI

接口：preLoad

示例代码：

```
public class DemoApplication extends Application {
  private static final String TAG = "DemoApplication";
  public static boolean sdkPreLoadSuccess = false;

  @Override
  public void onCreate() {
    super.onCreate();

    // 1、initialization environment
    HashMap<String, Object> params = new HashMap<>();
    params.put("preLoadListener",
          (PreLoadListener) (code, msg) -> {
            Log.d(TAG, "preload so code:" + code + ", msg:" + msg);
            if (code == 0) {
              sdkPreLoadSuccess = true;
            }
          });
    BgsSdk.preLoad(this, params);
  }
}

public class VideoPlayActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // 推荐设置Activity界面为全屏沉浸式效果
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
          WindowManager.LayoutParams.FLAG_FULLSCREEN);

    if (!DemoApplication.sdkPreLoadSuccess) {
      createErrorDialog(this, 100001, "PreLoadFail",
            (dialog, which) -> VideoPlayActivity.this.finish());
      return;
    }

    setContentView(R.layout.activity_video_play);

    mSdkView = findViewById(R.id.sdk_view);
  }
}
```

2.5.3 创建sdk对象、初始化实例播放参数

接口：initPhone

示例代码：

```
 bgsSdk = new BgsSdk(VideoPlayActivity.this);

HashMap<String, Object> params = new HashMap<>();
params.put("sdkView", mSdkView);
params.put("uuid", mUuid);
params.put("width", 720);
params.put("height", 1280);
params.put("bitrate", 8192);
params.put("fps", 30);
params.put("sdkCallback", myPlaySdkCallback);
// 用SDK采集视频
params.put("useSdkCollectVideo", true);
// 用SDK采集音频
params.put("useSdkCollectAudio", true);

bgsSdk.initPhone(params);
```

2.5.4 获取ServerToken

通过访问接口server-token去服务端获取serverToken。

2.5.5 连接实例

接口：startPhone

示例代码：

```
 if (bgsSdk != null) {
   bgsSdk.startPhone(serverToken);
}
```

2.5.6 断开实例

接口：stopPhone

示例代码：

```
 if (bgsSdk != null) {
   bgsSdk.stopPhone();
}
```

2.5.7 解绑实例

在SDK回调onStopped后，可视情况通过业务后台调用openApi接口disconnect去服务端解绑实例。

2.5.8 完整示例

```
 public class DemoApplication extends Application {
   private static final String TAG = "DemoApplication";
   public static boolean sdkPreLoadSuccess = false;

   @Override
   public void onCreate() {
      super.onCreate();

      // 1、initialization environment
      HashMap<String, Object> params = new HashMap<>();
      params.put("preLoadListener",
            (PreLoadListener) (code, msg) -> {
               Log.d(TAG, "preload so code:" + code + ", msg:" + msg);
               if (code == 0) {
                  sdkPreLoadSuccess = true;
               }
            });
      BgsSdk.preLoad(this, params);
   }
}

public class VideoPlayActivity extends AppCompatActivity {
   public static final int REQUEST_CAMERA_CODE = 5;
   public static final int REQUEST_MIC_CODE = 6;
   private static final String[] PERMISSION_CAMERA = new String[]{
         Manifest.permission.CAMERA,
   };
   private static final String[] PERMISSION_MIC = new String[]{
         Manifest.permission.RECORD_AUDIO,
   };
   // 需要启动的应用ID
   private static final int APP_ID = 6;
```

```
private MyPlaySdkCallback myPlaySdkCallback;

private static final int INIT_PHONE = 1;
private static final int STOPED_PHONE = 2;

private BgsSdk bgsSdk;

private SdkView mSdkView; // video渲染用view
private View mBackView;

private String mUuid; // 用户唯一id，不能为空

private HandlerThread mHandlerThread;
private Handler mHandler;
private String serverToken;
private HandlerNetwork handlerNetwork;

/**
 * 获取uuid，此方法在部分手机上（例如小米11 android 11）可能获取不到，
 * 商用时获取uuid谨慎使用，此处仅用于演示
 *
 * @param application
 * @return
 */
public String getUUID(Application application) {
    String uuid = null;
    try {
        uuid = Settings.System.getString(application.getContentResolver(),
                Settings.Secure.ANDROID_ID);
    } catch (Exception e) {
        e.printStackTrace();
    }

    if (TextUtils.isEmpty(uuid)) {
        uuid = "mciDemoDefaultUuid";
    }
    return uuid;
}

private void initPhone() {
    myPlaySdkCallback = new MyPlaySdkCallback(VideoPlayActivity.this);

    HashMap<String, Object> params = new HashMap<>();
    params.put("sdkView", mSdkView);
    params.put("uuid", mUuid);
    params.put("width", 720);
    params.put("height", 1280);
    params.put("bitrate", 8192);
    params.put("fps", 30);
    params.put("sdkCallback", myPlaySdkCallback);
    // 用SDK采集视频
    params.put("useSdkCollectVideo", true);
    // 用SDK采集音频
    params.put("useSdkCollectAudio", true);
    bgsSdk.initPhone(params);
}

private void getServerToken() {
    if (bgsSdk == null) {
        return;
    }

    try {
        handlerNetwork.setCallBack(new HandlerNetwork.CallBack() {
            @Override
            public void onSuccess(String result) {
                Gson gson = new Gson();
                Response response = gson.fromJson(result, Response.class);

                if (response == null || response.getData() == null
                        || response.getData().getSuccessList() == null
                        || response.getData().getSuccessList().size() < 1
                        || response.getData().getSuccessList().get(0) == null
                        || TextUtils.isEmpty(response.getData()
                        .getSuccessList().get(0).getServerToken())) {
                    onDisconnected(222001, "获取serverToken失败");
                    return;
                }
                serverToken = response.getData()
                        .getSuccessList().get(0).getServerToken();

                if (bgsSdk != null) {
                    bgsSdk.startPhone(serverToken);
```

```
                }
            }

            @Override
            public void onFail(String error) {
                onDisconnected(222001, "获取serverToken失败");
            }
        });

        String[] instanceCodes = new String[] {"VM010151051179"}; //实列列表
        String onlineTime = String.valueOf(60 * 60 * 24 *30); // 在线时长1个月
        handlerNetwork.getServerToken(mUuid, instanceCodes,
                onlineTime);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void stopPhone() {
    if (mHandler != null) {
        mHandler.removeCallbacksAndMessages(null);
        mHandler = null;
    }

    if (mHandlerThread != null) {
        mHandlerThread.quitSafely();
        mHandlerThread = null;
    }
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // set  full screen
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);

    if (!DemoApplication.sdkPreLoadSuccess) {
        createErrorDialog(this, 100001, "PreLoadFail",
                (dialog, which) -> VideoPlayActivity.this.finish());
        return;
    }

    setContentView(R.layout.activity_video_play);
    mSdkView = findViewById(R.id.sdk_view);
    mBackView = findViewById(R.id.menu_view);
    bgsSdk = new BgsSdk(VideoPlayActivity.this);
    mUuid = getUUID(getApplication());
    handlerNetwork = new HandlerNetwork();

    mHandlerThread = new HandlerThread("NetworkRequestThread");
    mHandlerThread.start();
    mHandler = new Handler(mHandlerThread.getLooper()) {
        @Override
        public void handleMessage(@NonNull Message msg) {
            super.handleMessage(msg);
            switch (msg.what) {
                case INIT_PHONE:
                    initPhone();
                    break;
                case STOPED_PHONE:
                    stopPhone();
                    break;
                default:
                    break;
            }
        }
    };
    mHandler.sendEmptyMessage(INIT_PHONE);
}

@Override
protected void onResume() {
    super.onResume();
    // resume game
    if (bgsSdk != null) {
        bgsSdk.resume();
    }
}

@Override
protected void onStop() {
```

```
protected void onStop() {
    super.onStop();
    // pause game, 应用彻底不可见时暂停云端推流
    if (bgsSdk != null) {
        bgsSdk.pause();
    }
}

@Override
public void onBackPressed() {
    if (mBackView != null) {
        mBackView.setVisibility(View.VISIBLE);
    }
}

@Override
protected void onDestroy() {
    super.onDestroy();
    // stop game and release resources
    if (bgsSdk != null) {
        // stop game
        bgsSdk.stopPhone();
    }

    myPlaySdkCallback = null;
}

public void onDisconnected(int errorCode, String msg) {
    createErrorDialog(VideoPlayActivity.this, errorCode, msg,
            (dialog, which) -> VideoPlayActivity.this.finish());
}

public void onInitSuccess() {
    getServerToken();
}

public void onInitFail(int code, String msg) {
    createErrorDialog(VideoPlayActivity.this, code, msg,
            (dialog, which) -> VideoPlayActivity.this.finish());
}

public void onConnectSuccess() {
}

public void onStopped() {
    if (mHandler != null) {
        mHandler.sendEmptyMessage(STOPED_PHONE);
    }
}

public void onClick(View view) {
    int keyCode = -1;
    switch (view.getId()) {
        case R.id.menu_btn:
            keyCode = MCIKeyEvent.KEYCOED_MENU;
            break;
        case R.id.home_btn:
            keyCode = MCIKeyEvent.KEYCOED_HOME;
            break;
        case R.id.back_btn:
            keyCode = MCIKeyEvent.KEYCOED_BACK;
            break;
        case R.id.quit_btn:
            // 退出
            finish();
            break;
        default:
            break;
    }

    if (keyCode > 0 && bgsSdk != null) {
        bgsSdk.sendKeyEvent(MCIKeyEvent.ACTION_UNDEFINED, keyCode);
    }

    if (mBackView != null) {
        mBackView.setVisibility(View.GONE);
    }
}

private void createErrorDialog(Context context, int errCode, String errMsg,
                               DialogInterface.OnClickListener onClickListener) {
    runOnUiThread(new Runnable() {
        @Override
```

```
            public void run() {
                AlertDialog.Builder b = new AlertDialog.Builder(context);
                b.setTitle("返回错误提示");
                b.setMessage("CallBack返回了" + errCode + " ; errMsg : " + errMsg);
                b.setNegativeButton("确定", onClickListener);
                b.setPositiveButton("取消", onClickListener);
                AlertDialog dialog = b.create();
                dialog.setCanceledOnTouchOutside(false);
                dialog.show();

            }
        });
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == REQUEST_CAMERA_CODE) {
            if (bgsSdk != null) {
                bgsSdk.openCamera();
            }
        } else if (requestCode == REQUEST_MIC_CODE) {
            if (bgsSdk != null) {
                bgsSdk.openMic();
            }
        }
    }

    public void onRequestPermission(String s) {
        if (Manifest.permission.RECORD_AUDIO.equals(s)) {
            checkPermission(VideoPlayActivity.this, PERMISSION_MIC, REQUEST_MIC_CODE);
        }
        if (Manifest.permission.CAMERA.equals(s)) {
            checkPermission(VideoPlayActivity.this, PERMISSION_CAMERA, REQUEST_CAMERA_CODE);
        }
    }

    public static boolean checkPermission(Activity activity, String[] permission, int requestCode) {
        if (isPermissionGranted(activity, permission)) {
            return true;
        } else {
            // 如果没有设置过权限许可，则弹出系统的授权窗口
            ActivityCompat.requestPermissions(activity, permission, requestCode);
            return false;
        }
    }

    // 每个权限是否已授权
    public static boolean isPermissionGranted(Activity activity, String[] permission) {
        if (Build.VERSION.SDK_INT >= 23) {
            for (int i = 0; i < permission.length; i++) {
                int checkPermission = ContextCompat.checkSelfPermission(activity, permission[i]);
                /***
                 * checkPermission返回两个值
                 * 有权限: PackageManager.PERMISSION_GRANTED
                 * 无权限: PackageManager.PERMISSION_DENIED
                 */
                if (checkPermission != PackageManager.PERMISSION_GRANTED) {
                    return false;
                }
            }
        }
        return true;
    }
}

public class MyPlaySdkCallback extends BgsSdkCallback {
    private static final String TAG = "MyPlaySdkCallback";
    private VideoPlayActivity videoPlayActivity;

    public MyPlaySdkCallback(VideoPlayActivity videoPlayActivity) {
        this.videoPlayActivity = videoPlayActivity;
    }
    @Override
    public void onInitSuccess() {
        if (videoPlayActivity != null) {
            videoPlayActivity.onInitSuccess();
        }
    }

    @Override
    public void onInitFail(int i, String s) {
```

```
        if (videoPlayActivity != null) {
            videoPlayActivity.onInitFail(i, s);
        }
    }

    @Override
    public void onConnectSuccess() {
        if (videoPlayActivity != null) {
            videoPlayActivity.onConnectSuccess();
        }
    }

    @Override
    public void onConnectFail(int i, String s) {
        if (videoPlayActivity != null) {
            videoPlayActivity.onDisconnected(i, s);
        }
    }

    @Override
    public void onStopped() {
        if (videoPlayActivity != null) {
            videoPlayActivity.onStopped();
        }
    }

    @Override
    public void onDisconnected(int i) {
        Log.d(TAG, "onDisconnected i = " + i);
        if (videoPlayActivity != null) {
            videoPlayActivity.onDisconnected(i, "onDisconnected");
        }
    }

    @Override
    public void onPlayInfo(String s) {
        Log.d(TAG, "onPlayInfo s = " + s);
    }

    @Override
    public void onRenderedFirstFrame(int i, int i1) {
        Log.d(TAG, "onRenderedFirstFrame i = " + i + ", i1 = " + i1);
    }

    @Override
    public void onVideoSizeChanged(int i, int i1) {
        Log.d(TAG, "onVideoSizeChanged i = " + i + ", i1 = " + i1);
    }

    @Override
    public void onRequestPermission(String s) {
        if (videoPlayActivity != null) {
            videoPlayActivity.onRequestPermission(s);
        }
    }
}
```

函数说明：

1. 视情况调用pause暂停实例推流；
2. 视情况调用resume恢复实例推流。

3 SDK高级功能

3.1 切换分辨率，比特率，帧率

4 SDK开发指南

sdk的api函数都封装到了com.mci.commonplaysdk.BgsSdk.java类中，具体函数描述如下

4.1 构造函数

- 函数原型：

```
public BgsSdk(Activity activity)
```

- 函数说明：BgsSdk构造函数。

4.2 类函数

4.2.1 preLoad

- **函数原型：**

```
public static void preLoad(Application application, HashMap<String, Object> params)
```

- **函数说明：**预加载SO，初始化SDK运行时环境。
- **参数说明：**

参数	类型	说明
application	Application	非空Application对象
params	HashMap<String, Object>	加载SO的参数，见下表

- **params参数说明：**

关键字	值的类型	必填项	说明
preLoadListener	PreLoadListener	非必填	默认值为null。（SO加载回调，详见PreLoadListener）
cloudLoadSo	boolean	非必填	默认值为false。（true：加载网络SO， false：加载本地SO）
soFile	String	非必填	默认值为null。（SDK 中加载so库文件地址，不设置使用SDK中默认so库）
logLevel	int	非必填	BsgSdk.LOG_DEFAULT。（日志级别，具体定义参考log等级）
writeLogToFile	boolean	非必填	默认值为true。（标记是否把SDK中日志保存到设备本地，true ：保存；false：不保存）

- **返回结果：**

通过**PreLoadListener**接口中的**void onLoad(int code, String msg)**回调SO加载结果。

code：0，表示加载成功。

code：非0，表示加载失败，msg变量中给出相应的异常信息。

4.3 成员函数

initPhone

特别说明：初始化操作要确保在SO加载成功后再进行。

- **函数原型：**

```
public int initPhone(HashMap<String, Object> params)
```

- **函数说明:**用户信息参数、云手机定制参数。
- **params参数说明：**

关键字	值的类型	必填项	说明
sdkView	SdkView	必填	SdkView对象，云手机显示的画布，需要布局到您的应用
uuid	String	必填	设置用户uuid，uuid是用来区分真实用户的唯一标识，由应用测来定义，请根据业务场景来决定
width	int	必填	推流视频分辨率宽，如：720（推流请求是视频是竖屏width < height）
height	int	必填	推流视频分辨率高，如：1280
bitrate	int	必填	推流视频码率，单位kBps，如：4092
fps	int	必填	推流视频帧率，如：30
instanceCode	string	非必填	设备编号
sdkCallback	BgsSdkCallback	非必填	默认值为null。(SDK回调接口，详见BgsSdkCallback)
useSWDecode	boolean	非必填	默认值为false。（Video解码方式，true：软解；false：硬解）
noVideoDataTimeout	int	非必填	默认值为30。（单位s，设置首次启动视频推流超时时间，若超过设定超时时间还没有视频流返回，会报10006错误码，并断开与云端连接。）
gameScreenRotate	boolean	非必填	默认值为false。（控制Sdk是否允许画面依据手机重力方向做180度旋转。true，开启此功能；false，关闭此动能）
autoSwitchDecodeMode	boolean	非必填	默认值为false。（控制Sdk硬解异常后自动切换到软解。 false：不切换； true：切换）
forcePortrait	boolean	非必填	默认值为false。（设置强制横屏显示，设置后，竖屏应用会被强制横屏显示。针对电视、盒子等大屏设备，普通手机无需设置。）
defaultRotation	boolean	非必填	默认值为false。（设置应用默认屏幕方向，false：竖屏， true：横屏）
useSdkCollectVideo	boolean	非必填	默认值为false。（true，由sdk采集摄像头数据； false, 由客户端自己处理）
useSdkCollectAudio	boolean	非必填	默认值为false。（true，由sdk采集麦克风数据, false, 由客户端自己处理）
logSource	String	非必填	默认值为null。（日志来源，通常传appkey，同时用于控制是否上报sdk日志，默认不上报）
foregroundTimeOut	int	非必填	默认值为0。（前台超时时间，单位s）
backgroundTimeOut	int	非必填	默认值为0。（后台超时时间，单位s）
autoControlQuality	boolean	非必填	默认值为false。（控制Sdk是否允许依据网络情况自动切换码率及帧率，true，开启此功能；false，关闭此动能），功能开启后需要同时设置videoLevels才会生效，详见自动调整画质等级）
videoLevels	VideoParam[]	非必填	默认值null，(画质等级数组，只有autoControlQuality设置为true才会生效，开启后sdk会依据网络情况及传入的画质等级参数自动适配一组参数请求推流，详见自动调整画质等级）
h265StreamMode	int	非必填	默认值为1。（H265的设置方式：1-自动、2-手动、3-白名单模式）**2.6.0以上的版本才支持**
forceEncodeType	int	非必填	默认值为-1。在h265StreamMode等于2的情况下有效。（手动设置的推流模式：2-H264、10-H265）**2.6.0以上的版本才支持**

- **返回结果：**

通过BgsSdkCallback类中的**void onInitSuccess()**回调初始化成功。

通过BgsSdkCallback类中的**void onInitFail(int code, String msg)**回调初始化失败信息。

code：初始化失败错误码，msg变量中给出相应的初始化异常信息。

- **失败错误码**

错误码	错误描述
330102	初始化时uuid为空
330103	初始化时sdkView为空
330106	初始化时width为空
330107	初始化时height为空
330108	初始化时bitrate为空
330109	初始化时fps为空

云手机播放

> 特别说明：播放操作要确保在initGame成功后再进行。

startPhone

特别说明：播放操作要确保在initGame成功后再进行。

- **函数原型：**

```
public void startPhone(String serverToken)
```

- **函数说明:**连接并播放云手机。
- **返回结果：**

通过BgsSdkCallback类中的**void onConnectSuccess()**回调设备连接成功。

通过BgsSdkCallback类中的**void onConnectFail(int code, String msg)**回调连接和播放失败信息。

code：连接和播放失败错误码，msg变量中给出相应的连接和播放异常信息。

- **失败错误码**

错误码	错误描述
330101	serverToken为空
330201	连接设备失败
330202	调用startPhone接口时，initGame接口还没有回调onInitSuccess
330204	连接设备失败，服务端不可用，http status != 200的情况
330205	连接设备失败，服务端可用，http status == 200的情况，申请失败

pause

- **函数原型：**

```
public void pause()
```

- **函数说明：**暂停连接云手机，只有在调用startPhone后才生效，与resume对应。

resume

- **函数原型：**

```
public void resume()
```

- **函数说明：**恢复连接云手机，只有在调用startPhone后才生效，与pause对应。

stopPhone

- **函数原型：**

```
public void stopPhone()
```

- **函数说明：**释放云手机。

画质

setStreamConfig

- **函数原型：**

```
public void setStreamConfig(int width, int height, int bitrate, int fps)
```

- **函数说明：**设置视频宽高，帧率，码率，**须在第一帧画面返回后设置才能生效**。
- **参数说明：**

参数	类型	说明
width	int	压码视屏宽
height	int	压码视频高
bitrate	int	视频码率，单位kBps
fps	int	视频帧率

数据透传到云手机

sendKeyEvent

- **函数原型：**

```
public void sendKeyEvent(int action, int keyCode)
```

- **函数说明：**透传键值事件到云手机。
- **参数说明：**

参数	类型	说明
action	int	事件行为，按下或抬起，对应 KeyEvent.ACTION_DOWN和KeyEvent.ACTION_UP
keyCode	int	具体按键值，传linux键值，对应KeyEvent.getScanCode()

sendSensorData

- **函数原型：**

```
public int sendSensorData(int type, float[] data)
```

- **函数说明：**透传传感器数据到云手机。
- **参数说明：**

参数	类型	说明
type	int	传感器类型，详见传感器类型
data	float[]	传感器数据

- **返回结果：**0成功，其他失败。

sendLocationData

- **函数原型：**

```
public int sendLocationData(float longitude,
                            float latitude,
                            float altitude,
                            float floor,
                            float horizontalAccuracy,
                            float verticalAccuracy,
                            float speed,
                            float direction,
                            String timestamp)
```

- **函数说明：**发送位置数据到云手机。
- **参数说明：**

参数	类型	说明
longitude	float	纬度(度)
latitude	float	经度(度)
altitude	float	高度(米)
floor	float	楼层
horizontalAccuracy	float	水平精度(米)
verticalAccuracy	float	垂直精度(米)
speed	float	速度(米/秒)
direction	float	方向
timestamp	String	时间戳

- **返回结果：**0成功，其他失败。

sendTransparentMsgReq

- **函数原型：**

```
public int sendTransparentMsgReq(int type, String data, String binderService)
```

- **函数说明：**透传信息到云手机。
- **参数说明：**

参数	类型	说明
type	int	自定义类型
data	String	具体数据
binderService	String	云端binder service名称

- **返回结果：**0成功，其他失败。

sendJoystickInput

- **函数原型：**

```
public int sendJoystickInput(int index, int pressed, int buttons,
                             int lx, int ly, int rx, int ry)
```

- **函数说明：**透传手柄键值到云手机。
- **参数说明：**

参数	类型	说明
index	int	手柄编号，从0开始递增
pressed	int	按键事件，抬起按下行为， 参见自定义手柄键值中ACTION_DOWN及ACTION_UP
buttons	int	按键值，参见自定义手柄键值
lx	int	左摇杆x坐标
ly	int	左摇杆y坐标
rx	int	右摇杆x坐标
ry	int	右摇杆y坐标

- **返回结果：**0成功，其他失败。

setExtraData

- **函数原型：**

```
public void setExtraData(int type, String info)
```

- **函数说明：**透传账号信息到云手机。
- **参数说明：**

参数	类型	说明
type	int	账号信息自定义类型
info	String	账号数据

sendAVData

- **函数原型：**

```
public int sendAVData(int avType, int frameType, byte[] data)
```

- **函数说明：**发送音视频数据到云手机，目前只支持H264及aac。
- **参数说明：**

参数	类型	说明
avType	int	用于区分音频（211）及视频（212）， 具体值参考传感器类型
frameType	int	音视频不同的编码类型，详情参考音视频编码帧类型
data	byte[]	具体音视频编码帧数据

- **返回结果：**0成功，其他失败。

copyToRemote

- **函数原型：**

```
public void copyToRemote(byte[] value)
```

- **函数说明：**透传剪切板数据到云手机的剪切板中，间接实现客户端到云手机的粘贴。
- **参数说明：**

参数	类型	说明
value	byte[]	剪切板的字符串数据要转为byte[]

sendInputString

- **函数原型：**

```
public void sendInputString(byte[] value)
```

- **函数说明：**透传字符串数据到云手机。
- **参数说明：**

参数	类型	说明
value	byte[]	字符串数据要转为byte[]

获取设备编号

getPadCode

- **函数原型：**

```
public String getPadCode()
```

- **函数说明：**获取设备编号，调用startGame后可获取。
- **返回结果：**返回设备编号。

音视频采集

setUseSdkCollectVideo

- **函数原型 ：**

```
public void setUseSdkCollectVideo(boolean isSdkCollectVideo)
```

- **函数说明：**控制是否由SDK采集摄像头数据并上传。
- **参数说明：**

参数	类型	说明
isSdkCollectVideo	boolean	true，由sdk采集摄像头数据； false, 由客户端自己处理

setAVEncodeParams

- **函数原型 ：**

```
public void setAVEncodeParams(AVEncodeParamsBean paramsBean)
```

- **函数说明：**设置sdk采集摄像头数据压码参数。
- **参数说明：**

参数	类型	说明
paramsBean	AVEncodeParamsBean	摄像头采集参数

openCamera

- **函数原型 ：**

```
public void openCamera()
```

- **函数说明：**打开摄像头并采集数据上传到云端，通常不需要主动调用此函数，只有当用户未授权摄像头权限，sdk回调通知用户去授权且等用户授权后才需要调用，权限请求接口回调详见BgsSdkCallback中**onRequestPermission**

setUseSdkCollectAudio

- **函数原型 ：**

```
public void setUseSdkCollectAudio(boolean isSdkCollectAudio)
```

- **函数说明：**控制是否由SDK采集麦克风数据并上传。
- **参数说明：**

参数	类型	说明
isSdkCollectAudio	boolean	true，由sdk采集麦克风数据, false, 由客户端自己处理

openMic

- **函数原型 ：**

```
public void openMic()
```

- **函数说明：**打开麦克风并采集数据上传到云端，通常不需要主动调用此函数，只有当用户未授权麦克风权限，sdk回调通知用户去授权且等用户授权后才需要调用，权限请求接口回调详见BgsSdkCallback中**onRequestPermission**

其他

getVersion

- **函数原型：**

```
public String getVersion()
```

- **函数说明：**获取sdk版本号。
- **返回结果：**返回sdk版本号。

audioPauseOrResume

- **函数原型 ：**

```
public void audioPauseOrResume(boolean isResume)
```

- **函数说明：**控制sdk是否静音及恢复播放声音。
- **参数说明：**

参数	类型	说明
isResume	boolean	true，恢复播放声音；false，设置静音

OPEN API参考

API文档

更新历史

发布时间	发布内容
2024-12-25	新增： 云手机相关接口 查询实例的所有绑定信息 解绑实例的所有绑定信息 基础能力相关接口 实例限速 分页查询应用列表 修改： 基础能力相关接口 分页查询实例信息 新增出参： instanceStatus deviceStatus
2024-10-30	新增： 基础能力相关接口 实例分辨率调整 服务回调相关接口 实例分辨率调整通知 进程死亡告警通知
2024-10-16	新增： 基础能力相关接口 获取同步截图url 销毁已获取的同步截图url 系统镜像相关接口 删除自定义镜像
2024-09-25	新增： 基础能力相关接口 分辨率调整
	新增： 基础能力相关接口 应用安装 应用卸载 系统镜像相关接口

2024-09-05	查询公共镜像 查询自定义镜像 上传自定义镜像 获取自定义镜像上传结果 创建更新镜像任务 查询更新镜像结果
2023-12-06	新增： 基础能力相关接口 实例切换会话控制权 实例设置内存限额
2023-11-02	新增： 基础能力相关接口 桌面推荐图标刷新 服务回调相关接口 桌面推荐图标刷新通知
2023-09-14	新增： 基础能力相关接口 实例授权连接 云手机相关接口 获取授权serverToken
2023-08-17	新增： 基础能力相关接口 应用上传(新版) 应用更新
2023-03-23	新增： 基础能力相关接口 实例信息查询[/resources/instance/infos] 下架： 基础能力相关接口 旧版分页查询实例信息接口[distribute/pad/infos.html]
2023-03-09	新增： 基础能力相关接口 客户实例汇总数据查询
2023-02-23	新增： 基础能力相关接口 创建流量查询任务接口 获取流量查询任务结果接口
2023-01-12	新增： 基础能力相关接口 机房已绑定实例数信息 实例执行脚本 上传文件到实例 服务回调相关接口： 实例执行脚本通知 上传文件到实例通知
2022-10-24	新增： 云手机相关接口 获取serverToken 新增入参： multiTerminal merchantPoolNo 实例指令相关接口 实例重启 新增入参： merchantPoolNo 实例重置 新增入参： merchantPoolNo
2022-10-20	新增： 基础能力相关接口 应用镜像按实例发布 应用镜像按机房发布 获取应用镜像按机房发布结果 服务回调相关接口

	应用镜像机房发布完成通知
2022-08-18	新增接口： 基础能力相关接口 分页查询实例信息 云手机相关接口 获取severToken 实例解绑
2022-07-15	实例指令相关接口 实例解绑 服务回调相关接口 实例解绑回调
2022-06-10	新增接口 获取每个机房可用实例数量和总的可用实例数量

API概述

最近更新时间：2024年12月25日 12:00:00

云手机相关接口

接口名称	接口功能
获取serverToken	获取serverToken后在两小时内可以用来申请实例。申请实例后可以用serverToken解绑已成功申请的实例。
获取授权serverToken	获取授权serverToken，通过serverToken连接的方式将云手机授权给其他用户使用
实例解绑	根据申请实例时传参的serverToken来释放指定实例。
实例切换会话控制权	根据serverToken切换已绑定实例会话控制权。
查询实例的所有绑定信息	根据instanceCode查询实例的所有绑定信息。
解绑实例的所有绑定信息	根据instanceCode解绑实例的所有绑定信息。

基础能力相关接口

接口名称	接口功能
分页查询实例信息	根据查询条件分页获取实例列表信息。
客户实例汇总数据查询	客户可以通过该接口来实时查询当前所有机房下的总实例数、可用实例数和已绑定的实例数。
机房未绑定实例数信息	获取每个机房可用且未绑定实例数量和总的可用且未绑定实例数量。
机房已绑定实例数信息	获取每个机房可用且已绑定实例数量和总的可用且已绑定实例数量。
实例截图	对实例进行截图，支持设置截图质量
实例截图结果查询	查询截图结果，如果截图成功，提供截图下载url
创建流量查询任务	创建流量查询任务
获取流量查询任务结果	获取流量查询任务的结果
上传文件到实例	上传文件到百度云游戏平台
实例执行脚本	让实例执行传入的脚本内容
实例授权连接	将已连接的云手机授权给其他用户使用
实例设置内存限额	用于修改实例可以使用的内存上限值
应用上传(新版)	上传应用到百度云游戏平台
应用更新	更新应用文件，应用信息不变更
应用上传	上传应用到百度云游戏平台。
应用启停	根据实例编号和应用包名对实例进行应用启停的操作。执行的结果可以根据返回的任务ID,通过获取应用启停结果接口查看。
桌面推荐图标刷新	用于刷新并展示实例桌面推荐应用，只能操作未绑定实例
任务结果查询	根据任务id查询任务执行结果
实例调度	对实例进行实例池之间的调度
应用安装	为单台或多台云手机实例同时安装应用
应用卸载	为单台或多台云手机实例同时卸载应用
预置应用安装	为单台或多台云手机实例同时安装预置应用
预置应用卸载	为单台或多台云手机实例同时卸载预置应用
分页查询应用列表	分页获取应用列表信息。
实例分辨率调整	调整实例分辨率参数
实例限速	设置实例限速

实例指令相关接口

接口名称	接口功能
实例重启	重启指定实例编码的实例
实例重置	将指定的实例恢复出厂设置

服务回调相关接口

回调名称	功能
应用上传通知	应用上传完成后的通知。
应用启动通知	应用启动完成后的通知。
应用停止通知	应用停止完成后的通知。
实例解绑通知	实例解绑完成后的通知。
实例重置通知	实例重置完成后的通知。
实例重启通知	实例重启完成后的通知。
实例执行脚本通知	实例完成执行脚本后的通知
上传文件到实例通知	文件上传到实例完成后的通知
分辨率调整结果通知	分辨率调整完成后的通知
进程死亡告警通知	实例运行的应用进程非正常死亡后的告警通知

调用方式

请求结构

Headers参数说明

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		

公共参数

公共参数用于标识请求用户并进行签名校验和用户鉴权，在每次请求时均需要携带公共参数才能正常发起请求。
注：后续接口文档中不再对公共参数进行说明

名称	类型	是否必须	备注	示例值
appkey	String	是	百度云游戏平台分配给第三方用户的接口 key	50b13132bb394901f151bc12
auth_ver	String	是	api 的版本号	3
nonce	Long	是	当前有的时间戳，精确到毫秒	1600846168490
s	String	是	方法签名，根据 appkey、auth_ver 和 nonce 计算	efabcaca7c66d629a639cee37d924813

方法签名生成规则

```
s = MD5(appkey + appkey值 + auth_ver + auth_ver值 + nonce + nonce值 + appSecret值)
```

注：字符串内容统一用 utf-8 字符集

方法签名生成示例

Java

```
String paramsStr = "appkey" + appKey +
        "auth_ver" + authVer +
        "nonce" + System.currentTimeMillis() +
        appSecret;//接口签名密钥（可配置）
String sign = org.apache.commons.codec.digest.DigestUtils.md5Hex(paramsStr.getBytes(StandardCharsets.UTF_8));
```

URL拼接公共参数示例

```
https://platform.armvm.com/{resourcePath}?s={s}&appkey={appkey}&auth_ver={auth_ver}&nonce={nonce}
```

body参数

参数名称	参数值	是否必须	示例	备注
msg	见msg加密规则	是		
createTime	公共参数中的nonce	是		

msg加密规则

加密模式：3DES——CBC模式，填充模式PKCS5Padding

加密步骤：

> 1.desKey截取前24位转byte[]
> 2.时间戳nonce转byte[]
> 3.接口请求实际入参的Json串转byte[]
> 4.3DES对内容进行加密（CBC模式，填充模式PKCS5Padding）
> 5.使用base64对加密结果进行编码

msg加密示例

Java

```
 // 填装实际接口请求参数
Map param = new HashMap();
param.put("param1", "value1");
param.put("param2", "value2");
param.put("param3", "value3");

// 请求参数转化为JSON字符串
String requestParam = new Gson().toJson(param);

// 请求参数加密
Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
SecretKeyFactory skf = SecretKeyFactory.getInstance("DESede");
SecretKey secretKey = skf.generateSecret(new DESedeKeySpec(desKey.getBytes("utf-8")));
ByteBuffer byteBuffer = ByteBuffer.allocate(8);
byteBuffer.order(ByteOrder.BIG_ENDIAN);
byteBuffer.putLong(nonce);
byte[] iv = byteBuffer.array();
IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivParameterSpec);
byte[] cipherText = cipher.doFinal(requestParam.getBytes("utf-8"));
String msg = Base64.encodeBase64String(cipherText);
Map map = new HashMap();
map.put("createTime", nonce);
map.put("msg", msg);
// 转化为JSON字符串
String encryptBody = new Gson().toJson(map);
```

完整调用示例

Java

```
 package org.example.openapi;

import cn.hutool.json.JSONUtil;
import org.apache.commons.codec.digest.DigestUtils;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESedeKeySpec;
import javax.crypto.spec.IvParameterSpec;
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class OpenApiTest {
    public static void main(String[] args) throws Exception {
        // 百度云游戏平台分配给第三方用户的接口Key
        String appKey = "4a8311be034943cfaa6a357fdc9f0461";
        // 接口版本号，现在有2和3两种版本的接口，具体按照接口文档上的版本号
        String authVer = "3";
        // 当前的时间戳
        long nonce = System.currentTimeMillis();
        // 接口签名秘钥
        String appSecret = "01a258aff68e6b2f2513df7f7dc3cf18";
        // 生成方法签名
        String paramsStr = "appkey" + appKey +
                "auth_ver" + authVer +
                "nonce" + System.currentTimeMillis() +
```

```
            appSecret;//接口签名密钥（可配置）
        String sign = DigestUtils.md5Hex(paramsStr.getBytes(StandardCharsets.UTF_8));
        // desKey 3DES秘钥（可配置），根据商户平台实际生成的替换此值
        String desKey = "cba76b7b0ab1011179087092effe1b8d";
        // 拼接公共参数到URL
        String apiUrl = "https://platform.armvm.com/{resourcePath}?s=" + sign + "&appkey=" + appKey + "&auth_ver=" + authVer + "&nonce=" + nonce;

        // 填装实际接口请求参数
        Map<String, Object> param = new HashMap<>();
        param.put("param1", "value1");
        param.put("param2", "value2");
        param.put("param3", "value3");

        // 请求参数转化为JSON字符串
        String requestParam = JSONUtil.toJsonStr(param);

        // 请求参数加密
        Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
        SecretKeyFactory skf = SecretKeyFactory.getInstance("DESede");
        SecretKey secretKey = skf.generateSecret(new DESedeKeySpec(desKey.getBytes(StandardCharsets.UTF_8)));
        ByteBuffer byteBuffer = ByteBuffer.allocate(8);
        byteBuffer.order(ByteOrder.BIG_ENDIAN);
        byteBuffer.putLong(nonce);
        byte[] iv = byteBuffer.array();
        IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
        cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivParameterSpec);
        byte[] cipherText = cipher.doFinal(requestParam.getBytes(StandardCharsets.UTF_8));
        String msg = Base64.getEncoder().encodeToString(cipherText);
        Map<String, Object> map = new HashMap<>();
        map.put("createTime", nonce);
        map.put("msg", msg);
        // 转化为JSON字符串
        String encryptBody = JSONUtil.toJsonStr(map);

        // 构造请求
        URL url = new URL(apiUrl);
        HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
        // 连接超时时间
        httpURLConnection.setConnectTimeout(1000);
        // 读取结果超时时间
        httpURLConnection.setReadTimeout(1000);
        httpURLConnection.setDoInput(true);
        httpURLConnection.setDoOutput(true);
        httpURLConnection.setUseCaches(false);
        httpURLConnection.setRequestProperty("Content-type", "application/json;charset=utf-8");
        httpURLConnection.setRequestMethod("POST");

        // 得到请求的输出流对象
        OutputStreamWriter out = new OutputStreamWriter(httpURLConnection.getOutputStream());
        // 把数据写入请求的Body
        out.write(encryptBody);
        out.flush();
        out.close();

        String result;
        try (InputStream in = 200 == httpURLConnection.getResponseCode() ? httpURLConnection.getInputStream() : httpURLConnection.getErrorStream();) {
            byte[] buf = new byte[1024];
            int length;
            ByteArrayOutputStream bout = new ByteArrayOutputStream();
            while ((length = in.read(buf, 0, buf.length)) > 0) {
                bout.write(buf, 0, length);
            }
            bout.flush();
            result = new String(bout.toByteArray(), StandardCharsets.UTF_8);
        } finally {
            httpURLConnection.disconnect();
        }
        int responseCode = httpURLConnection.getResponseCode();
        if (responseCode == 200) {
            System.out.println("调用成功,接口返回结果:\n" + result);
        } else {
            System.out.println("调用失败");
        }
    }
}
```

Python

```python
import json
import time
import hashlib
import base64
import struct

import requests
from Crypto.Cipher import DES3
from Crypto.Util.Padding import pad


def _pad(data):
    padding_length = 8 - len(data) % 8
    padding = bytes([padding_length] * padding_length)
    return data + padding


def encrypt(key, iv, message):
    cipher = DES3.new(key[:24].encode('utf-8'), DES3.MODE_CBC, struct.pack('>Q', iv))
    padded_message = pad(message.encode('utf-8'), DES3.block_size)
    ciphertext = cipher.encrypt(padded_message)
    return ciphertext


def encrypt_with_base64(key, iv, message):
    ciphertext = encrypt(key, iv, message)
    return base64.b64encode(ciphertext).decode('utf-8')


if __name__ == '__main__':
    # 百度云游戏平台分配给第三方用户的接口Key
    app_key = "4a8311be034943cfaa6a357fdc9f0461"
    # 接口版本号，现在有2和3两种版本的接口，具体按照接口文档上的版本号
    auth_ver = "3"
    # 当前的时间戳
    nonce = int(time.time() * 1000)
    # 接口签名秘钥
    app_sec = "01a258aff68e6b2f2513df7f7dc3cf18"
    # 方法签名
    sign = "appkey%sauth_ver%snonce%d%s" % (app_key, auth_ver, nonce, app_sec)
    md5_sign = hashlib.md5(sign.encode('utf-8')).hexdigest()
    # desKey 3DES秘钥（可配置），根据商户平台实际生成的替换此值
    des_key = "cba76b7b0ab1011179087092effe1b8d"
    # 拼接公共参数到URL
    url = "https://platform.armvm.com/resourcePath?appkey=%s&auth_ver=%s&s=%s&nonce=%d" % (app_key, auth_ver, md5_sign, nonce)
    data = {
        "param1": "value1",
        "param2": "value2",
        "param3": "value3",
    }
    msg = encrypt_with_base64(des_key, nonce, json.dumps(data))
    post_data = {
        "msg": msg,
        "createTime": nonce
    }
    print('url = ', url)
    print('data = ', post_data)
    res = requests.post(url, json.dumps(post_data), headers={
        'Content-Type': 'application/json'
    })
    print(res.text)
```

Go

```go
package main

import (
 "bytes"
 "crypto/md5"
 "encoding/base64"
 "encoding/hex"
 "encoding/json"
 "fmt"
 "github.com/forgoer/openssl"
 "net/http"
 "time"
)

func encryptWithBase64(key string, iv int64, message string) (string, error) {
 ivBytes := make([]byte, 8)
 for i := 0; i < 8; i++ {
```

```
  ivBytes[7-i] = byte(iv >> (8 * i))
 }
 ciphertext, err := openssl.Des3CBCEncrypt([]byte(message), []byte(key)[:24], ivBytes, openssl.PKCS7_PADDING)
 if err != nil {
  return "", err
 }
 return base64.StdEncoding.EncodeToString(ciphertext), nil
}

func main() {
 appKey := "4a8311be034943cfaa6a357fdc9f0461"
 authVer := "3"
 nonce := time.Now().UnixNano() / int64(time.Millisecond)
 appSec := "01a258aff68e6b2f2513df7f7dc3cf18"

 sign := fmt.Sprintf("appkey%sauth_ver%snonce%d%s", appKey, authVer, nonce, appSec)
 md5Sum := md5.Sum([]byte(sign))
 md5Sign := hex.EncodeToString(md5Sum[:])

 desKey := "cba76b7b0ab1011179087092effe1b8d"

 url := fmt.Sprintf("https://platform.armvm.com/resourcePath?appkey=%s&auth_ver=%s&s=%s&nonce=%d", appKey, authVer, md5Sign, nonce)

 data := map[string]string{
  "param1": "value1",
  "param2": "value2",
  "param3": "value3",
 }

 jsonData, err := json.Marshal(data)
 if err != nil {
  fmt.Println("Error marshalling JSON:", err)
  return
 }

 msg, err := encryptWithBase64(desKey, nonce, string(jsonData))
 if err != nil {
  fmt.Println("Encryption error:", err)
  return
 }

 postData := map[string]interface{}{
  "msg":        msg,
  "createTime": nonce,
 }

 postDataJson, err := json.Marshal(postData)
 if err != nil {
  fmt.Println("Error marshalling post data:", err)
  return
 }

 fmt.Println("url = ", url)
 fmt.Println("data = ", string(postDataJson))

 req, err := http.NewRequest("POST", url, bytes.NewBuffer(postDataJson))
 if err != nil {
  fmt.Println("Error creating request:", err)
  return
 }
 req.Header.Set("Content-Type", "application/json")

 client := &http.Client{}
 resp, err := client.Do(req)
 if err != nil {
  fmt.Println("Error making request:", err)
  return
 }
 defer resp.Body.Close()

 var responseBytes bytes.Buffer
 _, err = responseBytes.ReadFrom(resp.Body)
 if err != nil {
  fmt.Println("Error reading response:", err)
  return
 }

 fmt.Println(responseBytes.String())
}
```

云手机相关接口

获取serverToken

基本信息

URL

> https://platform.armvm.com/auth/instance/cloud-phone-server-token

请求格式

> POST

版本号

> 3

使用备注

> 同一个uuid+instanceCode的组合，只允许拥有一个有效的serverToken，不允许同时为相同的uuid+instanceCode申请多个serverToken

请求参数

名称	类型	是否必须	默认值	备注	示例值
uuid	String	必须	无	用户唯一标识。uuid是用来区分真实用户的唯一标识	89e87993500e424ab0de340da6f3599e
instanceCodes	String[]	必须	无	实例编码集合，长度不能超过50	["VM192168169028","VM192168169029"]
onlineTime	Integer	必须	无	控制时长，单位：秒；1~2147483647	300
grantControl	String	必须	无	控制模式： CONTROL，可控制； WATCH：只能观看	WATCH

返回参数

名称	类型	是否必须	备注	示例值
code	number	非必须	响应码	0
data	object	非必须	返回信息	
├──successList	object[]	非必须	获取成功的列表	[{"instanceCode": "VM192168169028", "serverToken": "50b13132bb394901f151bc45"}]
├────instanceCode	String	非必须	实例编码	"VM192168169028"
├────serverToken	String	非必须	服务端token	"50b13132bb394901f151bc45"
├──errorList	object[]	非必须	获取失败的列表	[{"instanceCode": "VM192168169028", "errorMsg": "实例不存在"}]
├────instanceCode	String	非必须	实例编码	"VM192168169028"
├────errorMsg	String	非必须	错误信息	"实例不存在"
msg	string	非必须	响应消息	OK
ts	number	非必须	响应时间戳	1658393762049

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "msg": "OK",
  "data": {
    "successList": [
      {
        "instanceCode": "VM192168169027",
        "serverToken": "50b13132bb394901f151bc45"
      }
    ],
    "errorList": [
      {
        "instanceCode": "VM192168169028",
        "errorMsg": "实例不存在"
      },
      {
        "instanceCode": "VM192168169029",
        "errorMsg": "实例不存在"
      }
    ]
  },
  "ts": "1658455442690"
}
```


异常示例


```
{
  "code": 11101118,
  "msg": "商户类型错误,只允许云手机客户使用",
  "ts": "1658455442690"
}
```


获取授权serverToken

基本信息

URL

> https://platform.armvm.com/auth/instance/authorized-server-token

请求格式

> POST

版本号

> 3

使用场景

> 用于实例授权场景，将已连接的云手机授权给其他用户使用，支持主用户和授权用户可同时使用

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
uuid	String	必须		用户唯一标识	89e87993500e424ab0de340da6f3599e
instanceCode	String	必须		实例编号	VM192168169028
onlineTime	Integer	必须		授权使用时间，单位：秒	3600
grantControl	String	必须		控制模式： CONTROL，可控制； WATCH：只能观看	CONTROL

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
data	Object	必须			
├─serverToken	String	必须		服务端token	50b13132bb394901f151bc45
├─instanceCode	String	必须		实例编号	VM192168169028
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {
     "instanceCode": "VM192168169028",
     "serverToken": "c5039adcda344cebb748e94644fbc68f5"
  },
  "msg": "OK",
  "ts": "1694584934947"
}
```

异常示例

```
{
  "code": 11103270,
  "msg": "实例暂未被使用，不能对该实例申请授权",
  "ts": 1601192809490
}
```

```
{
  "code": 11103271,
  "msg": "当前实例授权连接个数已达上限",
  "ts": 1601192809490
}
```

实例解绑

基本信息

URL

> https://platform.armvm.com/auth/instance/disconnect

请求格式

> POST

版本号

> 3

使用场景

> 根据申请实例时传参的serverToken来释放实例。

请求参数

名称	类型	是否必须	默认值	备注	示例值
uuid	String	必须	无	用户唯一标识。uuid是用来区分真实用户的唯一标识，必须与申请serverToken时的uuid一致	89e87993500e424ab0de340da6f3599e
serverToken	String	必须	无	申请实例时传入的serverToken	50b13132bb394901f151bc45

返回参数

名称	类型	是否必须	备注	示例值
code	number	非必须	响应码	0
data	object	非必须	返回信息	
msg	string	非必须	响应消息	OK
ts	number	非必须	响应时间戳	1658393762049

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {},
  "msg": "OK",
  "ts": "1658455442690"
}
```

异常示例

```
{
  "code": 11102027,
  "msg": "serverToken无效",
  "ts": "1658455442690"
}
```

新增实例到期时间

基本信息

URL

> https://platform.armvm.com/resources/instance/expire-time-increase

请求格式

> POST

版本号

> 3

使用场景

> 按分钟数新增实例的到期时间

请求参数

名称	类型	是否必须	默认值	备注	示例值
serverTokens	String[]	必须	无	绑定的serverToken值	['021c6bee611b4d27aeb3de240d7144055']
time	Integer	必须	无	新增的时长，单位：分钟，范围：[1, 5256000]	1000

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		返回状态，0-成功	0
data	Object	非必须			
├── failList	String[]	必须		续时失败的serverToken值	
msg	String	必须		返回信息	success
ts	Long	必须		接口返回时时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例

```
{
  "data": {
    "failList": []
  },
  "code": 0,
  "msg": "OK",
  "ts": "1702454509182"
}
```

异常示例

```
{
  "code": 11102027,
  "msg": "serverToken无效",
  "ts": "1702458796121"
}
```

更新实例到期时间

基本信息

URL

> https://platform.armvm.com/resources/instance/expire-time-update

请求格式

> POST

版本号

> 3

使用场景

> 更新实例的到期时间

请求参数

名称	类型	是否必须	默认值	备注	示例值
serverTokens	String[]	必须	无	绑定的serverToken值	['021c6bee611b4d27aeb3de240d7144055']
expireTime	Long	必须	无	新的到期时间（时间戳）	1704038400000

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		返回状态，0-成功	0
data	Object	非必须			
├── failList	String[]	必须		续时失败的serverToken值	
msg	String	必须		返回信息	success
ts	Long	必须		接口返回时时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例

```
{
 "data": {
   "failList": []
 },
 "code": 0,
 "msg": "OK",
 "ts": "1702454509182"
}
```

异常示例

```
{
 "code": 11102027,
 "msg": "serverToken无效",
 "ts": "1702458796121"
}
```

查询实例的所有绑定信息

基本信息

URL

> https://platform.armvm.com/resources/instance/bind-infos

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
instanceCodes	String[]	必须		实例编码集合,集合长度在200内（包含200）	["VM192168001001","VM192168001002"]

返回参数

名称	类型	是否必须	默认值	备注	示例值	
code	Integer	必须		响应编码	0	
data	Object[]	非必须				
├─instanceCode	String	非必须		实例编号	VM011101100190	
├─serverToken	String	非必须		serverToken	49b1f2943e04468383b10e4acafaad2d5	
├─uuid	String	非必须		用户uid	72f4d5bc0036	
├─businessType	String	非必须		绑定业务类型：cloud_phone-云手机；cloud_app-云游戏；oem-OEM	cloud_phone	
├─bindTime	Long	非必须		绑定时间	1734578634443	
├─expireTime	Long	非必须		到期时间	1735587244443	
├─controlStatus	String	非必须		受控状态：online-非受控，control-受控	online	
├─grantControl	String	非必须		控制权：CONTROL-可控制，WATCH-仅观看	CONTROL	
├─instanceControlTime	Long	非必须		最近受控时间	1734578795543	
msg	String	非必须		响应消息		
ts	Long	必须		时间戳	1598754678123	

请求示例

> 参考：调用方式

响应示例

```
{
 "data": [
  {
   "instanceCode": "VM011101100190",
   "serverToken": "49b1f2943e04468383b10e4acafaad2d5",
   "uuid": "72f4d5bc0036",
   "businessType": "cloud_phone",
   "bindTime": 1734578634443,
   "expireTime": 1735587244443,
   "controlStatus": "online",
   "grantControl": "CONTROL",
   "instanceControlTime": null
  }
 ],
 "code": 0,
 "msg": "OK",
 "ts": "1734578646712"
}
```

异常示例

```
{
 "code": 11103062,
 "msg": "实例不存在",
 "ts": "1734578806958"
}
```

解绑实例的所有绑定信息

基本信息

URL

> https://platform.armvm.com/auth/instance/disconnect-all

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
instanceCodes	String[]	必须		实例编码集合,集合长度在200内（包含200）	["VM192168001001","VM192168001002"]

返回参数

名称	类型	是否必须	默认值	备注	示例值	
code	Integer	必须		响应编码	0	
data	Object	非必须				
msg	String	非必须		响应消息		
ts	Long	必须		时间戳	1598754678123	

请求示例

> 参考：调用方式

响应示例

```
{
  "data": {},
  "code": 0,
  "msg": "OK",
  "ts": "1734578060146"
}
```

异常示例

```
{
  "code": 11103062,
  "msg": "实例不存在",
  "ts": "1734578078843"
}

{
  "code": 11103397,
  "msg": "实例存在云游戏或OEM的绑定信息，暂不支持调用此接口解绑",
  "ts": "1734578136525"
}
```

基础能力相关接口

分页查询实例信息

基本信息

URL

> https://platform.armvm.com/resources/instance/infos

请求格式

> POST

版本号

> 3

使用场景

> 根据查询条件分页获取实例列表信息。

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
page	Integer	非必须	1	页码，默认值1，最小1页（包含1页）	1
rows	Integer	非必须	200	每页记录数，默认值200，最小1条（包含1条），最大1000条（包含1000条）	10
instanceCodes	String[]	非必须		实例编码集合，集合长度在200内（包含200）	VM192168001001
merchantPoolNos	Long[]	非必须		实例池编号集合，集合长度在50内（包含50）	1,2

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
data	Object	非必须			
├──total	Integer	必须		总数	20
├──totalPage	Integer	必须		总页码	2
├──page	Integer	必须		页码	1
├──rows	Integer	必须		每页数量	10
├──pageData	Object[]	必须			
├────instanceCode	String	必须		实例编号	VM192168001001
├────instanceGrade	String	必须		实例开数	1
├────serverType	String	必须		实例服务器类型	R3
├────idcCode	String	必须		机房编码	GZ-IDC-TEST-01
├────availableStatus	String	必须		可分配状态 available 可分配 unavailable 不可分配	available
├────usableStatus	String	必须		可用状态 usable 可用 unusable 不可用	usable
├────bindStatus	String	必须		绑定状态 bindable 可绑定 bound 已绑定	bindable
├────controlStatus	String	必须		控制状态 online 在线 control 受控	online
├────instanceIp	String	必须		实例ip	11.11.198.77
├────deviceCode	String	必须		物理机编号	GZNX-IDC_DM011010099026
├────imageVersionId	Long	非必须		镜像版本ID	10001
├────memoryLimit	Integer	非必须		内存限额，单位：MB	1024
├────instanceStatus	String	非必须		实例在线状态，0-离线，1-在线	1
├────deviceStatus	String	非必须		物理机在线状态，0-离线，1-在线	1
msg	String	必须		响应消息	
ts	Long	必须		时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {
    "total": 1,
    "totalPage": 1,
    "page": 1,
    "rows": 10,
    "pageData": [
      {
        "instanceCode": "VM192168232001",
        "instanceGrade": "1",
        "serverType": "R3",
        "idcCode": "GZ-IDC-TEST-01",
        "availableStatus": "available",
        "usableStatus": "usable",
        "bindStatus": "bindable",
        "controlStatus": "online",
        "instanceIp": "11.11.198.77",
        "deviceCode": "GZNX-IDC_DM011010099026",
        "imageVersionId": 10001,
        "memoryLimit": 1024,
        "instanceStatus": "1",
        "deviceStatus": "1"
      }
    ]
  },
  "msg": "",
  "ts": 1601021167163
}
```


客户实例汇总数据查询

基本信息

URL

> https://platform.armvm.com/resources/instance/summary-data

请求格式

> POST

版本号

> 3

使用场景

> 客户可以通过该接口来实时查询当前所有机房下的总实例数、可用实例数和已绑定的实例数。

请求参数

名称	类型	是否必须	默认值	备注	示例值
merchantPoolNo	Long	非必须	无	实例池编号	1
includeSubPool	Boolean	非必须	true	是否包含子实例池	true-包含子实例池,false-不包含子实例池

merchantPoolNo与includeSubPool的使用

merchantPoolNo	includeSubPool	筛选实例范围
null	null 或 true	当前商户的所有可用实例
null	false	当前商户顶级实例池中的可用实例
100	null 或 true	当前商户实例池编号为100的实例池及下层子实例池中的可用实例
100	false	当前商户实例池编号为100的实例池中的可用实例

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		返回状态，0-成功	0
data	Object	必须			
├── totalData	Object	必须		实例总数信息	
├── ├── totalInstanceCount	Integer	必须		总数	10
├── ├── idcList	Object[]	必须		机房列表	
├── ├── ├── idcName	String	必须		机房名称	广州机房
├── ├── ├── idcCode	String	必须		机房编码	IDC-GZ
├── ├── ├── instanceCount	Integer	必须		数量	10
├── availableData	Object	必须		可用实例总数	
├── ├── totalInstanceCount	Integer	必须		总数	5
├── ├── idcList	Object[]	必须		机房列表	
├── ├── ├── idcName	String	必须		机房名称	广州机房
├── ├── ├── idcCode	String	必须		机房编码	IDC-GZ
├── ├── ├── instanceCount	Integer	必须		数量	5
├── boundData	Object	必须		已绑定实例总数	
├── ├── totalInstanceCount	Integer	必须		总数	
├── ├── idcList	Object[]	必须		机房列表	
├── ├── ├── idcName	String	必须		机房名称	广州机房
├── ├── ├── idcCode	String	必须		机房编码	IDC-GZ
├── ├── ├── instanceCount	Integer	必须		数量	5
msg	String	必须		返回信息	success
ts	Long	必须		接口返回时时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "msg": "success",
  "ts": 1658455442690,
  "data": {
    "totalData": {
      "totalInstanceCount": 10,
      "idcList": [
        {
          "idcName": "广州机房",
          "idcCode": "GZ-IDC-01",
          "instanceCount": 10
        }
      ]
    },
    "boundData": {
      "totalInstanceCount": 5,
      "idcList": [
        {
          "idcName": "广州机房",
          "idcCode": "GZ-IDC-01",
          "instanceCount": 5
        }
      ]
    },
    "availableData": {
      "totalInstanceCount": 5,
      "idcList": [
        {
          "idcName": "广州机房",
          "idcCode": "GZ-IDC-01",
          "instanceCount": 5
        }
      ]
    }
  }
}
```


机房未绑定实例数信息

基本信息

URL

> https://platform.armvm.com/monitor/pad/available-count.html

请求格式

> POST

版本号

> 3

请求参数

无

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
data	Object	非必须			
├──totalAvailableCount	Integer	非必须		可用实例总数	20
├──idcInfos	Object[]	非必须			
├────────idcName	String	必须		机房名称	广州机房
├────────idcCode	String	必须		机房编码	GZ-IDC-TEST-01
├────────availableCount	Integer	必须		可用数量	1001
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例


```
{
"code": 0,
"data": {
  "totalAvailableCount": 128,
  "idcInfos": [
    {
      "idcName": "北京机房",
      "idcCode": "BJ-1001",
      "availableCount": 20
    },
    {
      "idcName": "广州机房",
      "idcCode": "GZ-10012",
      "availableCount": 108
    }
  ],
  "msg": "",
  "ts": 1601021167163
}
}
```


异常示例

机房已绑定实例数信息

基本信息

URL

> https://platform.armvm.com/monitor/pad/enable-bind-count.html

请求格式

> POST

版本号

> 3

请求参数

无

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
data	Object	非必须			
├──totalEnableBindCount	Integer	非必须		客户已绑定的设备数	20000
├──idcInfos	Object[]	非必须			
├────────idcName	String	必须		机房名称	广州机房
├────────idcCode	String	必须		机房编码	GZ-IDC-TEST-01
├────────enableBindCount	Integer	必须		机房的已绑定的设备数	999
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
"code": 0,
"data": {
  "totalEnableBindCount": 2000,
  "idcInfos": [
    {
      "idcName": "北京机房",
      "idcCode": "BJ-1001",
      "enableBindCount": 1001
    },
    {
      "idcName": "广州机房",
      "idcCode": "GZ-10012",
      "enableBindCount": 999
    }
  ],
  "msg": "",
  "ts": 1601021167163
}
}
```


异常示例

实例截图

基本信息

URL

> https://platform.armvm.com/command/pad/screenshot.html

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
padCodes	String[]	必须			["VM192168043080", "VM192168043091"]
quality	Integer	非必须	3	枚举值：1,2,3，表示截图质量。 其中，1：高质量，2：中质量，3低质量	3

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	非必须			
├─taskId	Integer	必须		任务ID	1001
├─padCode	String	必须		实例编号	VM192168043091
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
"code": 0,
"data": [
  {
    "taskId": 1001,
    "padCode": "VM192168043080"
  },
  {
    "taskId": 1002,
    "padCode": "VM192168043091"
  }
],
"msg": "",
"ts": 1668572148662
}
```


异常示例


```
{
  "code":11103062,
  "msg":"实例不存在",
  "ts":"1668586771490"
}
```


实例截图结果查询

基本信息

URL

> https://platform.armvm.com/command/pad/screenshot-info.html

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
taskIds	Integer[]	必须		任务ID集合	[1001, 1002]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	非必须			
├─taskId	Integer	必须		任务ID	1001
├─padCode	String	必须		实例编号	VM192168043091
├─taskStatus	Integer	必须		截图任务状态，枚举值6，7，8。其中，6：截图中，7：截图成功，8：截图失败	7
├─url	String	必须		截图下载URL	
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
 "code": 0,
 "data": [
  {
   "taskId": 1001,
   "padCode": "VM192168043080",
   "taskStatus": 7,
   "url":"https://platform.armvm.com/files/screenshot/6a8030673f534b9c948c58b51c615553.png"
  },
  {
   "taskId": 1002,
   "padCode": "VM192168043091",
   "taskStatus": 7,
   "url":"https://platform.armvm.com/files/screenshot/6a8030673f534b9c948c58b51c615555.png"
  }
 ],
 "msg": "",
 "ts": 1668572148662
}
```


异常示例


```
{
 "code":11109015,
 "msg":"操作失败，请输入正确的任务ID",
 "ts":"1668584368093"
}
```


创建流量查询任务

基本信息

URL

> https://platform.armvm.com/distribute/task/query-flow

请求格式

> POST

版本号

> 3

使用场景

> 创建流量查询任务，查询给定实例集合的流量或宽带计费汇总，查询时间范围限制为只能查询2小时以前60天以内的数据。
>
> 具有最大任务创建数限制，超过限制时，需要等待前面的任务执行完毕后才能创建新的查询任务。

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
instanceCodes	String[]	必须		实例列表集合	["VM011021103175"]
startTime	String	必须		查询开始时间（必须为小时整点时间），格式为yyyy-MM-dd hh:mm:ss	2023-02-23 18:00:00
endTime	String	必须		查询结束时间（必须为小时整点时间），格式为yyyy-MM-dd hh:mm:ss，	2023-02-23 18:00:00
billingType	String	必须		计费类型，枚举类型：FLOW（流量计费）、PEEK_PERCENT_100（峰值带宽计费）、PEEK_PERCENT_95（95峰值带宽计费）	FLOW
taskDesc	String	非必须		任务描述	"2023年2月份流量计费汇总"

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		返回状态，0-成功	0
data	Object	必须			
├── taskId	String	必须		任务ID	"89695"
msg	String	必须		返回信息	
ts	Long	必须		接口返回时时间戳	

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "msg": "OK",
  "data": [
    {
      "taskId": "89695",
    },
  ],
  "ts": "1658455442690"
}
```

异常示例

```
{
  "code": 11100064,
  "msg": "查询开始时间不能早于前60天",
  "ts": 1601192809490
}
```

```
{
  "code": 11100065,
  "msg": "查询结束时间只能到2小时以前",
  "ts": 1601192809490
}
```

```
{
  "code": 11109021,
  "msg": "当前已有%s个查询任务正在执行，超过任务个数限制，请稍后再试",
  "ts": 1601192809490
}
```

```
{
  "code": 11100050,
  "msg": "系统异常，请联系客服人员",
  "ts": 1601192809490
}
```

获取流量查询任务结果

基本信息

URL

> https://platform.armvm.com/distribute/task/query-flow-result

请求格式

> POST

版本号

> 3

使用场景

> 创建流量查询任务后，用返回的taskId调用该接口获取流量查询任务的结果

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
taskId	String	必须		任务ID	"89695"

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		返回状态，0-成功	0
data	Object	必须			
├── taskId	String	必须		任务ID	"89695"
├── taskDesc	String	必须		创建任务传入的任务描述	"2023年2月份流量计费汇总"
├──billingType	String	必须		计费类型	"FLOW"
├──taskStatus	String	必须		任务状态，INIT 初始化, PROCESSING 处理中, SUCCESS 成功, NOT_EXIST 不存在, FAIL 失败;	"SUCCESS"
├── msg	String	非必须		任务失败时，描述失败原因	
├── billingValue	Double	必须		计费值，成功状态时有值，流量（单位byte）或带宽（单位bps）	100.0
├── bandwidthList	Object[]	非必须		带宽明细列表	
├── ├── recordTime	Long	必须		记录时间（毫秒时间戳）	1676020200000
├── ├── send	Double	必须		发送数据，单位：bps	50.0
├── ├── receive	Double	必须		接收数据，单位：bps	50.0
msg	String	必须		返回信息	
ts	Long	必须		接口返回时时间戳	

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "msg": "OK",
  "data": {
    "msg": "",
    "billingValue": 100.0,
    "taskStatus": "SUCCESS",
    "billingType": "FLOW",
    "taskId": "89695",
    "taskDesc": "2023年2月份流量计费汇总",
    "bandwidthList": [
      {
        "recordTime": 1676020200000,
        "receive": 50.0,
        "send": 50.0
      }
    ]
  },
  "ts": "1658455442690"
}
```

异常示例

```
{
  "code": 11108002,
  "msg": "请输入正确的任务ID",
  "ts": 1601192809490
}
```

```
{
  "code": 11100050,
  "msg": "系统异常，请联系客服人员",
  "ts": 1601192809490
}
```


上传文件到实例

基本信息

URL

> https://platform.armvm.com/resources/instance/file-upload

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
instanceCodes	String[]	必须		实例编号集合	["VM192168043080", "VM192168043091"]
fileUrl	String	必须		文件URL	http://platform.armvm.com/test.apk
fileName	String	必须		文件名称	test.apk
fileMd5	String	必须		文件MD5摘要	185da05e944e869ebf077d6217d3efe9
autoInstall	Integer	非必须		枚举值：0,1，表示是否选择自动安装，仅上传apk时生效。 其中，0：不自动安装，1：自动安装	0
customizeFilePath	String	非必须		上传到实例的自定义路径	/tmp/files/

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	非必须			
├─taskId	Long	必须		任务ID	1001
├─instanceCode	String	必须		实例编号	VM192168043091
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例


```
{
 "code": 0,
 "data": [
   {
     "taskId": 1001,
     "instanceCode": "VM192168043080"
   },
   {
     "taskId": 1002,
     "instanceCode": "VM192168043091"
   }
 ],
 "msg": "",
 "ts": 1668572148662
}
```

异常示例


```
{
  "code":11103062,
  "msg":"实例不存在",
  "ts":1668586771490
}
```

```
{
  "code":11103082,
  "msg":"自定义路径禁止包含[.]或空格",
  "ts":1668586771490
}
```


实例截图

基本信息

URL

> https://platform.armvm.com/command/pad/execute-script.html

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
padCodes	String[]	必须		实例编号集合	["VM192168043080", "VM192168043091"]
scriptContent	String	必须		脚本内容	

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	非必须			
├─taskId	Long	必须		任务ID	1001
├─padCode	String	必须		实例编号	VM192168043091
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
 "code": 0,
 "data": [
  {
   "taskId": 1001,
   "padCode": "VM192168043080"
  },
  {
   "taskId": 1002,
   "padCode": "VM192168043091"
  }
 ],
 "msg": "",
 "ts": 1668572148662
}
```


异常示例


```
{
  "code":11103062,
  "msg":"实例不存在",
  "ts":"1668586771490"
}
```


实例备份

基本信息

> 对云手机实例进行数据备份，生成一份快照数据保存到对象存储中，之后可以通过实例还原功能将快照数据恢复到云手机实例中。
> **备份还原会产生大量流量，请谨慎使用外网的对象存储服务。**

URL

> https://platform.armvm.com/resources/instance/backup

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
instanceCode	String	必须		实例编号	VM192168001001
snapshotName	String	必须		快照名称	VM192168001001_0925
ossConfig	Object	必须		对象存储配置，需要支持s3协议	
├─endpoint	String	必须		对象存储endpoint	s3.bj.bcebos.com
├─bucket	String	必须		对象存储bucket	test
├─accessKey	String	必须		对象存储accessKey	ATqjcosksxz
├─secretKey	String	必须		对象存储secretKey	sdfaweradfg
├─protocol	String	必须		对象存储访问协议：http, https	https
snapshotPath	String	非必须	/	快照数据保存在对象存储的路径，不传则默认保存在根目录"/"下	/snapshot/data/

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object	必须			
├─taskId	Long	必须		任务ID	100000
├─instanceCode	String	必须		实例编号	VM192168001001
├─snapshotId	Long	必须		快照id	7290017192180314133
├─snapshotName	String	必须		快照名称	VM192168001001_0925
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data":[
     {
        "taskId": 100000,
        "instanceCode": "VM192168001001",
       "snapshotId": 7290017192180314133,
       "snapshotName": "VM192168001001_0925"
     }
  ],
  "msg": "",
  "ts": 1601188620078
}
```

异常示例

```
{
 "code": 11103062,
 "msg": "实例不存在",
 "ts": 1601021167163
}
```

实例还原

基本信息

> 将已备份成功的快照数据恢复到指定的云手机实例。
> **备份还原会产生大量流量，请谨慎使用外网的对象存储服务。**

URL

> https://platform.armvm.com/resources/instance/restore

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
snapshotId	Long	必须		快照ID	7290017192180314133
instanceCodes	String[]	必须		实例编号集合，最多200个	["VM192168001001"]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	必须			
├─taskId	Long	必须		任务ID	100000
├─instanceCode	String	必须		实例编号	VM192168001001
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data":[
     {
        "taskId": 100000,
        "instanceCode": "VM192168001001"
     }
  ],
  "msg": "",
  "ts": 1601188620078
}
```

异常示例

```
{
 "code": 11108050,
 "msg": "实例快照未完成创建",
 "ts": 1601021167163
}
```

分页查询实例快照信息

基本信息

URL

> https://platform.armvm.com/resources/instance/snapshot-page

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
snapshotIds	Long[]	非必须		快照ID集合	[7290017192180314133]
snapshotName	String	非必须		快照名称，可模糊查询	
snapshotStatus	String	非必须		快照状态： wait_create-待创建 creating-创建中 create_success-创建成功 create_failed-创建失败	create_success
page	Integer	必须		页码	1
rows	Integer	必须		每页条数，最多200	10

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
data	Object	非必须			
├─total	Integer	必须		总数	20
├─totalPage	Integer	必须		总页码	2
├─page	Integer	必须		页码	1
├─rows	Integer	必须		每页数量	10
├─pageData	Object[]	必须			
├──────snapshotId	Long	必须		快照ID	7290017192180314133
├──────snapshotName	String	必须		快照名称	VM192168001001_0925
├──────snapshotStatus	String	必须		快照状态： wait_create-待创建 creating-创建中 create_success-创建成功 create_failed-创建失败	create_success
msg	String	必须		响应消息	
ts	Long	必须		时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例


```
{
  "code": 0,
  "data": {
    "total": 1,
    "totalPage": 1,
    "page": 1,
    "rows": 10,
    "pageData": [
      {
        "snapshotId": 7290017192180314133,
        "snapshotName": "VM192168001001_0925",
        "snapshotStatus": "create_success"
      }
    ]
  },
  "msg": "",
  "ts": 1601021167163
}
```


异常示例

实例授权连接

基本信息

URL

> https://platform.armvm.com/auth/instance/authorized-connect

请求格式

> POST

版本号

> 3

使用场景

> 用于实例授权场景，将已连接的云手机授权给其他用户使用，支持主用户和授权用户可同时使用

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
uuid	String	必须		用户唯一标识	89e87993500e424ab0de340da6f3599e
instanceCode	String	必须		实例编号	VM192168169028
onlineTime	Integer	必须		授权使用时间，单位：秒	3600
grantControl	String	必须		控制模式： CONTROL，可控制； WATCH：只能观看	CONTROL

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
data	Object	必须			
├─serverToken	String	必须		服务端token	50b13132bb394901f151bc45
├─instanceCode	String	必须		实例编号	VM192168169028
├─streamModeList	String[]	必须		推流方式配置	["1", "2"]
├─sessionId	String	必须		会话id	3WGVL2R7NQXUMKAAERANZXT003RNQRPZ
├─connData	String	必须		连接信息(json格式)	
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例


```
{
  "code": 0,
  "data": {
    "connData": {
      "controlList": [
        {
          "controlCode": "TEST-TCP-CONTROL-242-bk",
          "controlInfoList": [
            {
              "cmccLine": "",
              "controlIp": "test.armvm.com",
              "controlPort": 10038,
              "ctLine": "",
              "cuLine": "",
              "traceServer": ""
            }
          ]
        }
      ],
      "multiControlList": [
        {
          "multiControlCode": "multi-242",
          "multiControlInfoList": [
            {
```

```
            {
                "multiControlIp": "192.168.169.158",
                "multiControlPort": 8080,
                "traceServer": ""
            }
        ]
    }
],
"padCode": "VM192168169028",
"remoteList": [],
"screenList": [
    {
        "screenCode": "SCREEN-GZ-TEST001",
        "screenInfoList": [
            {
                "bitrate": "50",
                "fps": "20",
                "gop": "5",
                "resolution": "2048",
                "screenIp": "127.0.0.1",
                "screenPort": 80,
                "traceServer": ""
            }
        ]
    }
],
"sessionExpireTime": "1694588533428",
"sessionId": "I0YUSI3Y9EX0BWF1JZF7XG4GG68ZE6VR",
"sessionInitTime": "1694584934428",
"webControlList": [
    {
        "webControlCode": "TEST-TCP-CONTROL-242-bk",
        "webControlInfoList": [
            {
                "cmccLine": "",
                "controlIp": "paas101.armvm.com",
                "controlPort": 10040,
                "ctLine": "",
                "cuLine": "",
                "traceServer": ""
            }
        ]
    }
],
"webRtcControlList": [
    {
        "controlCode": "TEST-TCP-CONTROL-242-bk",
        "gateway": {
            "cmccLine": "",
            "ctLine": "",
            "cuLine": "",
            "gatewayIp": "paas101.armvm.com",
            "gatewayPort": 8989
        },
        "webRtcControlInfoList": [
            {
                "controlIp": "192.168.170.242",
                "controlPort": 10038,
                "traceServer": ""
            }
        ]
    }
],
"webRtcMode": "",
"webScreenList": [
    {
        "webScreenCode": "SCREEN-GZ-TEST001",
        "webScreenInfoList": [
            {
                "bitrate": "50",
                "fps": "20",
                "gop": "5",
                "resolution": "2048",
                "screenIp": "127.0.0.1",
                "screenPort": 10036,
                "traceServer": ""
            }
        ]
    }
]
},
"instanceCode": "VM192168169028",
"serverToken": "c5039adcda344cebb748e94644fbc68f5",
```

```
      "sessionId": "I0YUSI3Y9EX0BWF1JZF7XG4GG68ZE6VR",
      "streamModeList": [
        "1",
        "2"
      ]
    },
    "msg": "OK",
    "ts": "1694584934947"
}
```


异常示例


```
{
    "code": 11103270,
    "msg": "实例暂未被使用，不能对该实例申请授权",
    "ts": 1601192809490
}
```

```
{
    "code": 11103271,
    "msg": "当前实例授权连接个数已达上限",
    "ts": 1601192809490
}
```


实例切换会话控制权

基本信息

URL

> https://platform.armvm.com/resources/instance/session-control-switch

请求格式

> POST

版本号

> 3

使用场景

> 切换已绑定实例的会话控制权

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
watchServerTokens	String[]	非必须		需要切换成仅观看权限的serverToken集合；元素个数不能超过50	["39baa7d579034774af98da7cc449945c5","f209f26361bf487a9fdc88c88902753b5"]
controlServerTokens	String[]	非必须		需要切换成可控制权限的serverToken集合；元素个数不能超过50	["39baa7d579034774af98da7cc449945c5","f209f26361bf487a9fdc88c88902753b5"]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应码	0
data	Object[]	非必须		切换失败的serverToken和对应的失败信息	
├─serverToken	String	必须		切换失败的serverToken	50b13132bb394901f151bc45
├─errorMsg	String	必须		切换失败原因	serverToken无效
msg	String	必须		响应信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例


```
{
  "data": [
    {
      "serverToken": "39baa7d579034774af98da7cc449945c5",
      "errorMsg": "serverToken无效"
    },
    {
      "serverToken": "f209f26361bf487a9fdc88c88902753b5",
      "errorMsg": "同一个serverToken的权限切换期望不能同时为WATCH和CONTROL"
    }
  ],
  "code": 0,
  "msg": "OK",
  "ts": "1700721933096"
}
```


异常示例


```
{
    "code": 11100003,
    "msg": "watchServerTokens集合的元素个数不能大于50",
    "ts": "1700721933096"
}
```


实例设置内存限额

基本信息

URL

> https://platform.armvm.com/resources/instance/memory-limit

请求格式

> POST

版本号

> 3

使用场景

> 用于修改实例可以使用的内存上限值

请求参数

名称	类型	是否必须	默认值	备注	示例值
instanceCodes	String[]	必须		实例编号集合（最多200个）	["VM192168043080", "VM192168043091"]
memoryLimit	Integer	非必须		内存限额值，单位MB（isLimit为true时，必须设置限额值）	2048
isLimit	Boolean	必须		是否设置限额：true-限额 | false-不限额	true

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	非必须			
├─taskId	Long	必须		任务ID	1001
├─instanceCode	String	必须		实例编号	VM192168043091
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例


```
{
  "code": 0,
  "data": [
    {
      "taskId": 1001,
      "instanceCode": "VM192168043080"
    },
    {
      "taskId": 1002,
      "instanceCode": "VM192168043091"
    }
  ],
  "msg": "",
  "ts": 1668572148662
}
```


异常示例


```
{
 "code":11103292,
 "msg":"实例内存限额不能为空",
 "ts":"1668584368093"
}
```


实例分辨率调整

基本信息

URL

> https://platform.armvm.com/resources/instance/resolution

请求格式

> POST

版本号

> 3

注意

> 在使用1080p以上分辨率时，请确保您的设备的 manage-agent 版本为 4.27.0 或以上。您可以与 PMO 确认设备的 manage-agent 版本是否满足该要求。这样可以确保您的设备能够正常处理更高的分辨率。

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
instanceCodes	String []	必须		实例编号列表，最多200个	["VM192168001001"]
height	Integer	必须		高	范围：300~4096
width	Integer	必须		宽	范围：300~4096
fps	Integer	必须		帧率	枚举：30/60/90
dpi	Integer	必须		dpi	范围：120~480

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object []	必须			
├─taskId	Long	必须		任务ID	100000
├─instanceCode	String	必须		实例编号	VM192168001001
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data":[
     {
        "taskId": 100000,
        "instanceCode": "VM192168001001"
     }
  ],
  "msg": "",
  "ts": 1601188620078
}
```

异常示例

```
{
 "code": 11103062,
 "msg": "实例不存在",
 "ts": 1601021167163
}
```

应用安装

基本信息

URL

> https://platform.armvm.com/resources/instance/install-app

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
appId	Integer	必须		应用ID	10001
instanceCodes	String []	必须		云手机实例列表，最多200个	["VM192168001001"]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object []	必须			
├─taskId	Long	必须		任务ID	100000
├─instanceCode	String	必须		实例编号	VM192168001001
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data":[
     {
        "taskId": 100000,
        "instanceCode": "VM192168001001"
     }
  ],
  "msg": "",
  "ts": 1601188620078
}
```

异常示例

```
{
 "code": 11103072,
 "msg": "应用不存在",
 "ts": 1601021167163
}
```

应用卸载

基本信息

URL

> https://platform.armvm.com/resources/instance/uninstall-app

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
appId	Integer	必须		应用ID	10001
instanceCodes	String []	必须		云手机实例列表，最多200个	["VM192168001001"]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object []	必须			
├──	taskId	必须		任务ID	100000
├──	instanceCode	必须		实例编号	VM192168001001
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data":[
     {
        "taskId": 100000,
        "instanceCode": "VM192168001001"
     }
  ],
  "msg": "",
  "ts": 1601188620078
}
```

异常示例

```
{
 "code": 11103072,
 "msg": "应用不存在",
 "ts": 1601021167163
}
```

内置应用安装

基本信息

URL

> https://platform.armvm.com/resources/app/app-builtin-install

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
appId	Long	必须		应用ID	10001
instanceCodes	String []	必须		云手机实例列表，最多200个	["VM192168001001"]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object []	必须			
├─taskId	Long	必须		任务ID	100000
├─instanceCode	String	必须		实例编号	VM192168001001
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data":[
     {
        "taskId": 100000,
        "instanceCode": "VM192168001001"
     }
  ],
  "msg": "",
  "ts": 1601188620078
}
```

异常示例

```
{
 "code": 11103072,
 "msg": "应用不存在",
 "ts": 1601021167163
}
```

内置应用卸载

基本信息

URL

> https://platform.armvm.com/resources/app/app-builtin-uninstall

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
appId	Long	必须		应用ID	10001
instanceCodes	String []	必须		云手机实例列表，最多200个	["VM192168001001"]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object []	必须			
├──	taskId	必须		任务ID	100000
├──	instanceCode	必须		实例编号	VM192168001001
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data":[
     {
        "taskId": 100000,
        "instanceCode": "VM192168001001"
     }
  ],
  "msg": "",
  "ts": 1601188620078
}
```

异常示例

```
{
 "code": 11103072,
 "msg": "应用不存在",
 "ts": 1601021167163
}
```

应用启停

基本信息

URL

> https://platform.armvm.com/command/apps/app-operate.html

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
padCodes	String []	必须		实例编号，上限200个	VM192168169029
├──		非必须			
operateType	String	必须		枚举：start、stop	start
packageName	String	必须		应用包名	com.demo.app

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
data	Object []	必须			
├─padCode	String	必须		实例编号	VM192168001001
├─taskId	String	必须		任务id	"10001"
msg	String	非必须		响应消息	
ts	Long	必须		时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例


```
{
"code": 0,
"data": {
    "total": 1,
    "pageData": [
        {
            "padCode": "VM192168232001",
            "padStatus": "0",
            "idcCode": "GZ-IDC-TEST-01",
            "padGrade": "1",
            "apps": [
                1000016516
            ]
        }
    ],
    "totalPage": 1,
    "page": 1,
    "rows": 10
},
"msg": "",
"ts": 1601021167163
}
```


异常示例


```
{
"code": -1,
"msg": "实例[VM19216801100]不存在",
"ts": 1601021202058
}
```


应用上传(新版)

基本信息

URL

> https://platform.armvm.com/resources/app/new-version

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
appId	Long	必须		应用ID	1
appName	String	必须		应用名称，字符长度最长为64	appTest
appPackageName	String	必须		应用包名 ，字符长度最长为255	com.demo.app
appUrl	String	必须		应用下载地址	https://www.demo.com/demo/download

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
msg	String	非必须		响应消息	ok
data	Object	非必须		响应结果	
├── applicationVersionId	Long	必须		应用版本ID	
├── appId	Long	必须		应用ID	
ts	Long	必须		时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "msg": "ok",
  "ts": 1601021167163
}
```

异常示例

```
{
  "code": 11103153,
  "msg": "AppId不能重复",
  "ts": 1601021202058
}
```

应用更新

基本信息

URL

> https://platform.armvm.com/resources/app/upgrade

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
appId	Long	必须		应用ID	1
appName	String	必须		应用名称，字符长度最长为64	appTest
appPackageName	String	必须		应用包名 ，字符长度最长为255	com.demo.app
appUrl	String	必须		应用下载地址	https://www.demo.com/demo/download

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
msg	String	非必须		响应消息	ok
data	Object	非必须		响应结果	
├── applicationVersionId	Long	必须		应用版本ID	
├── appId	Long	必须		应用ID	
ts	Long	必须		时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例

```
{
"code": 0,
"msg": "ok",
"ts": 1601021167163
}
```

异常示例

```
{
"code": 11103153,
"msg": "AppId不能重复",
"ts": 1601021202058
}
```

应用上传

基本信息

URL

> https://platform.armvm.com/distribute/apps/uploads.html

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
apps	Object []	必须		应用数量，上限10个	
├── appId	Long	必须		应用ID	1
├── appName	String	必须		应用名称，字符长度最长为64	appTest
├── pkgName	String	必须		应用包名 ，字符长度最长为255	com.demo.app
├── url	String	必须		应用下载地址	https://www.demo.com/demo/download
├── md5sum	String	必须		应用文件的MD5	
taskId	Long	非必须		任务id，不填则使用系统生成的taskId	000001

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
msg	String	非必须		响应消息	ok
ts	Long	必须		时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "msg": "ok",
  "ts": 1601021167163
}
```

异常示例

```
{
  "code": 11103153,
  "msg": "AppId不能重复",
  "ts": 1601021202058
}
```

分页查询应用列表

基本信息

URL

> https://platform.armvm.com/resources/app/page

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
appId	Integer	非必须		应用ID	10001
appPackage	String	非必须		应用包名	com.example.app
page	Integer	必须		页码	1
rows	Integer	必须		每页条数	10

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应编码	0
data	Object	非必须			
├─total	Integer	必须		总数	20
├─totalPage	Integer	必须		总页码	2
├─page	Integer	必须		页码	1
├─rows	Integer	必须		每页数量	10
├─pageData	Object []	必须			
├────appId	Integer	必须		应用id	10001
├────appName	String	必须		应用名称	云手机
├────appPackage	String	必须		应用包名	com.example.app
├────versionName	String	必须		版本	1.0
├────syncStatus	Integer	必须		同步状态；0-未完成/终止同步，1-已完成同步	1
├────releaseMarketStatus	Integer	必须		是否已发布应用市场；0-未发布，1-已发布	1
msg	String	必须		响应消息	OK
ts	Long	必须		时间戳	1601021167163

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {
     "total": 20,
     "totalPage": 2,
     "page": 1,
     "rows": 10,
     "pageData": [
        {
           "appId": 10001,
           "appName": "云手机",
           "appPackage": "com.example.app",
           "versionName":"1.0",
           "syncStatus":1,
           "releaseMarketStatus":1
        }
     ]
  },
  "msg": "",
  "ts": 1601021167163
}
```

异常示例

```
{
 "code": 11103072,
 "msg": "应用不存在",
 "ts": 1601021167163
}
```

桌面推荐图标刷新

基本信息

URL

> https://platform.armvm.com/resources/instance/recommend-app-icon-refresh

请求格式

> POST

版本号

> 3

使用场景

> 用于刷新并展示实例桌面推荐应用。该接口会先重置桌面，因此限制只能操作未绑定实例
> 注：该功能只适用于OEM镜像的云手机

请求参数

名称	类型	是否必须	默认值	备注	示例值
instanceCodes	String[]	必须		实例编号集合	["VM192168043080", "VM192168043091"]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	非必须			
├─taskId	Long	必须		任务ID	1001
├─instanceCode	String	必须		实例编号	VM192168043091
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": [
    {
      "taskId": 1001,
      "instanceCode": "VM192168043080"
    },
    {
      "taskId": 1002,
      "instanceCode": "VM192168043091"
    }
  ],
  "msg": "",
  "ts": 1668572148662
}
```

异常示例

```
{
 "code":11103287,
 "msg":"已绑定实例[VM192168043092]不能更新桌面推荐应用图标",
 "ts":"1668584368093"
}
```

任务结果查询

基本信息

URL

> https://platform.armvm.com/command/pad/execute-task-info.html

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
taskIds	Long[]	必须		任务ID列表，最多200个	[1001,1002]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	非必须			
├─taskId	Long	必须		任务ID	1000
├─padCode	String	必须		实例编号	VM192168169028
├─taskStatus	Integer	必须		任务状态， 5-待执行 6-执行中 7-执行成功 8-执行失败	7
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
"msg": "OK",
"code": 0,
"data": [{
 "padCode": "VM011010067010",
 "taskId": 2813218757970419736,
 "taskStatus": 7
}],
"ts": 1672820359299
}
```

异常示例

```
{
 "code": 11108002,
 "msg": "请输入正确的任务ID",
 "ts": 1672820359299
}
```

获取ssh连接信息

基本信息

URL

> https://platform.armvm.com/resources/instance/ssh-info

请求格式

> POST

版本号

> 3

使用场景

> 获取ssh连接信息

请求参数

名称	类型	是否必须	默认值	备注	示例值
instanceCode	String	必须	无	实例编号	VM011101102170
connectType	Integer	必须	无	连接类型：1-ssh 2-adb	1
liveTime	Integer	必须	无	连接有效时间，单位：秒，范围：[1, 86400]	600

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		返回状态，0-成功	0
data	Object	非必须			
├── sshInfo	Object	必须		ssh连接信息	
├── ├── instanceCode	String	必须		实例编号	
├── ├── sshCommand	String	必须		连接命令	
├── ├── sshPwd	String	必须		连接秘钥	
├── expireTime	Long	必须		过期时间戳	
msg	String	必须		返回信息	success
ts	Long	必须		接口返回时时间戳	1598754678123

请求示例

> 参考：调用方式

响应示例

```
{
 "data": {
  "sshInfo": {
   "instanceCode": "VM011101102170",
   "sshCommand": "ssh 11.101.102.170__62485__15920850610__1702456492@192.168.170.235 -p 17706",
   "sshPwd": "XnoQiOuyAvTjL+YZG5pcM9qi4vOX1ZHga2EO4wBragMfD7NeDOJQPYjm4QRMJZkHDqFIVvufUYbpucGROTzvqg=="
  },
  "expireTime": "1702456492329"
 },
 "code": 0,
 "msg": "OK",
 "ts": "1702455888197"
}
```

异常示例

```
{
 "code": 11109015,
 "msg": "操作失败，实例不存在",
 "ts": "1702458767487"
}
```

实例调度

基本信息

URL

> https://platform.armvm.com/resources/instance-pool/allocate

请求格式

> POST

版本号

> 3

使用场景

> 对实例进行实例池之间的调度

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
merchantPoolNo	Long	必须		目标实例池Id	10000
autoReset	Boolean	非必须	false	调度到子商户实例池时，是否自动执行恢复出厂设置	true
instanceCodes	String[]	必须		需要进行调度的实例池编号；元素个数不能超过200	["VM192168001001","VM192168001002"]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		响应码	0
msg	String	必须		响应信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "msg": "OK",
  "ts": "1601188620078"
}
```

异常示例

```
{
  "code": 11103180,
  "msg": "实例池不存在",
  "ts": "1601188620078"
}
```

获取同步截图url

基本信息

> 获取新的同步截图url，访问该url将从云手机实时获取截图

URL

> https://platform.armvm.com/resources/instance/get-screenshot-url

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
instanceCodes	String[]	必须		实例编号，最多200个	["VM192168169028"]
fullQuality	Integer	非必须	100	截图质量数值 范围[1-100]	100
scale	Integer	非必须	288	缩放比，等比缩放截图 0为不缩放 最小值：0 最大值：720	288
rotate	Integer	非必须	0	旋转角度 枚举：0，90，180，270	0

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object	必须			
├──successList	Object[]	必须		成功的实例编号以及截图token	
├────────screenshotUrl	String	必须		截图地址	https://zc-sync-cap.armvm.com:9809/op-agent/pad/screenshot?time=1727581342254&route=%2BAiT1Lts4RASs6sBCRRYvg%3D%3D&screenshotToken=xxx
├────────instanceCode	String	必须		实例编号	VM192168169029
├──expireTime	Long	必须		token到期时间（时间戳）	1727667742254
├──failList	String[]	必须		失败的实例编号列表	["VM192168169028"]
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1727581342086

使用说明：

1. 每次获取的url有效期为10分钟，不可指定，可联系PMO调整每次获取url的有效期，最长不超过24小时

2. 使用GET请求screenshotUrl，将在云手机实时截取一张图片，并返回图片流

3. 截图状态码说明：

状态码	说明
200	成功
404、500	请求错误，包括：篡改过URL参数
400	无效的url
401	用户ID校验失败
402	url已过期
403	url已被销毁
501	执行截图程序失败，需要设备端排查
502	设备端程序端口不通
503	截图超时
504	设备ping不通
505	设备空间不足

4. 调用建议：将接口返回的screenshotUrl缓存起来，在到期或被销毁之前可一直请求，并且捕获402/403状态码以便重新获取新的同步截图url

请求示例

> 参考：调用方式

响应示例

```
{
  "data": {
    "successList": [
      {
        "instanceCode": "VM192168169029",
        "screenshotUrl": "https://zc-sync-cap.armvm.com:9809/op-agent/pad/screenshot?
time=1727581342254&route=%2BAiT1Lts4RASs6sBCRRYvg%3D%3D&screenshotToken=YEl6G8YsTUPcQW4cyauhXOt%2FpXz%2BTmlSrKQylGzYZs4B02G3HbdOUDSy5BDI(

      }
    ],
    "failList": [],
    "expireTime": 1727667742254
  },
  "code": 0,
  "msg": "OK",
  "ts": "1727581342086"
}
```


异常示例


```
{
  "data": {
    "successList": [],
    "failList": [
      "VM192168169029"
    ],
    "expireTime": 1727582003554
  },
  "code": 0,
  "msg": "OK",
  "ts": "1727582003554"
}
```


销毁已获取的同步截图url

基本信息

> 销毁云手机当前所有的同步截图url

URL

> https://platform.armvm.com/resources/instance/destroy-screenshot-url

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
instanceCode	String	必须		实例编号	VM192168169028

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object	必须			
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1727581342086

请求示例

> 参考：调用方式

响应示例


```
{
  "data": {},
  "code": 0,
  "msg": "OK",
  "ts": "1727581342086"
}
```


异常示例


```
{
  "code": 11105017,
  "msg": "截图url不存在或已销毁",
  "ts": "1727586992599"
}
```


实例限速

基本信息

URL

> https://platform.armvm.com/resources/instance/set-speed

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
instanceCodes	String[]	必须		实例编码集合,集合长度在200内（包含200）	["VM192168001001","VM192168001002"]
direction	String	必须		限速类型，up-上行限速，down-下行限速	up
speed	Double	必须		外网限速，范围为0.2-1024，也可传入0取消限速	0.20
intranetSpeed	Double	非必须		内网限速，范围为1-1024，也可传入0取消限速	1.00

返回参数

名称	类型	是否必须	默认值	备注	示例值	
code	Integer	必须		响应编码	0	
data	Object[]	非必须				
├─instanceCode	String	必须		实例编码	VM192168001001	
├─taskId	String	必须		任务编号	"2840204519134814256"	
msg	String	非必须		响应消息		
ts	Long	必须		时间戳	1598754678123	

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {
     "padCode": "VM192168041004",
     "taskId": "2840204353481548485",
   },
  "msg": "ok",
  "ts": 1663664302000
}
```


异常示例


```
{
  "code": -1,
  "msg": "实例编号不存在",
  "ts": 1663664302000
}
```


系统镜像相关接口

查询公共镜像

基本信息

URL

> https://platform.armvm.com/resources/instance-base-image/list

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
serverType	String	非必须		实例服务器类型： G4|G6|R1|R2|R3	G4
romVersion	String	非必须		系统版本： android6.0 android8.1 android10.0 android12.0 android13.0	android8.1
imageVersionIds	Long[]	非必须		镜像版本id，最多200个	[5]
page	Integer	必须		页码，最小值为1	1
rows	Integer	必须		每页条数，最小值为1，最大值为200	200

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	必须			
├── total	Integer	必须		总数	20
├── totalPage	Integer	必须		总页码	2
├── page	Integer	必须		当前页码	1
├── rows	Integer	必须		每页数量	10
├── pageData	Object[]	非必须			
├──── imageVersionId	Long	必须		镜像版本ID	5
├──── imageVersionName	Long	必须		镜像版本名称	5
├──── serverType	String	必须		实例服务器类型	G4
├──── romVersion	String	必须		系统版本	android8.1
├──── createTime	Long	必须		上传时间（时间戳）	1601188620078
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {
    "total": 1,
    "totalPage": 1,
    "page": 1,
    "rows": 10,
    "pageData": [
      {
        "imageVersionId": "5",
        "imageVersionName": "baidu_phone_845_aosp8.1_v2.4.0_20240130172159.tar.gz",
        "serverType": "G4",
        "romVersion": "android8.1",
        "createTime": 1716819459335
      }
    ]
  },
  "msg": "",
  "ts": 1601021167163
}
```

查询自定义镜像

基本信息

URL

> https://platform.armvm.com/resources/instance-image/list

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
page	Integer	非必须	1	页码，最小值为1	1
rows	Integer	非必须	200	每页条数，最小值为1，最大值为200	200
imageVersionId	Long	非必须		镜像版本id	10001
imageVersionIds	Long[]	非必须		镜像版本id，最多200个	[10001,10002]
serverType	String	非必须		实例服务器类型： G4|G6|R1|R2|R3	G4
romVersion	String	非必须		系统版本： android6.0 android8.1 android10.0 android12.0 android13.0	android8.1
imageVersionName	String	非必须		镜像版本名称，模糊查询	xxx

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object []	必须			
├── total	Integer	必须		总数	20
├── totalPage	Integer	必须		总页码	2
├── page	Integer	必须		当前页码	1
├── rows	Integer	必须		每页数量	10
├── pageData	Object[]	非必须			
├──── imageVersionId	Long	必须		镜像版本ID	10001
├──── imageUploadStatus	String	必须		镜像上传状态，wait_execute|executing:执行中|success:成功|fail:失败	success
├──── serverType	String	必须		实例服务器类型	G4
├──── romVersion	String	必须		系统版本	android8.1
├──── imageFiles	Object[]	必须		镜像文件信息	
├────── imageFileUrl	String	必须		镜像文件下载地址	http://xxx.img
├────── imageFileName	String	必须		镜像文件名称（带后缀）	root_aosp10_xx.img
├────── imageFileType	String	必须		odm_aosp root_aosp system_aosp vendor_aosp super_aosp	root_aosp
├────── imageFileMd5	String	必须		镜像文件md5值	xxxxxxx
├──── baseImageVersionId	Long	必须		镜像底包ID	5
├──── imageVersionName	String	必须		镜像版本名称	xxx
├──── describe	String	必须		镜像版本描述	xxx
├──── createTime	Long	必须		上传时间（时间戳）	1601188620078
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {
    "total": 1,
    "totalPage": 1,
    "page": 1,
    "rows": 10,
    "pageData": [
      {
        "imageVersionId": 10001,
        "imageUploadStatus": "success",
        "serverType": "G4",
        "romVersion": "android8.1",
        "imageFiles": [
          {
            "imageFileUrl": "https://baidu.com/armvm/root-v15.img",
            "imageFileName": "root_aosp-v15.img",
            "imageFileType": "root_aosp",
            "imageFileMd5": "20d41ecb84d86667525a8a02d0102f2b"
          },
          {
            "imageFileUrl": "https://baidu.com/armvm/system-v15.img",
            "imageFileName": "system_aosp-v15.img",
            "imageFileType": "system_aosp",
            "imageFileMd5": "22aca472f2d66551f197ff14de8618f3"
          }
        ],
        "baseImageVersionId": 5,
        "imageVersionName": "xxx",
        "describe": "xxx",
        "createTime": 1601021167163
      }
    ]
  },
  "msg": "",
  "ts": 1601021167163
}
```


上传自定义镜像

基本信息

URL

> https://platform.armvm.com/resources/instance-image/upload

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
imageFiles	Object[]	必须		镜像文件信息，最多20个	
├── imageFileUrl	String	必须		镜像文件下载地址	http://xxx.img
├── imageFileName	String	必须		镜像文件名称。 镜像文件名称需要以镜像文件类型为前缀，比如说root_aosp类型的就需要以root_aosp开头命名，如root_aosp_xx.img ； 如果是安卓12，在文件类型后面应该带f2fs，如system_aosp_f2fs_xxx.img	root_aosp10_xx.img
├── imageFileType	String	必须		odm_aosp root_aosp system_aosp vendor_aosp super_aosp	root_aosp
├── imageFileMd5	String	必须		镜像文件md5值	xxxxxxx
serverType	String	必须		实例服务器类型： G4|G6|R1|R2|R3	G4
romVersion	String	必须		系统版本： android6.0 android8.1 android10.0 android12.0 android13.0	android8.1
baseImageVersionId	Long	非必须		镜像底包ID	5
imageVersionName	String	必须		镜像版本名称，最多50个字符	xxx
describe	String	非必须		镜像版本描述，最多80个字符	xxx

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object	必须			
├── imageVersionId	Long	必须		镜像版本ID	10001
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {
      "imageVersionId": 10001
  },
  "msg": "",
  "ts": 1601021167163
}
```

获取自定义镜像上传结果

基本信息

URL

> https://platform.armvm.com/resources/instance-image/upload-info

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
imageVersionId	Long	必须		实例镜像版本ID	10001

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object	必须			
├── imageUploadStatus	String	必须		镜像上传状态，wait_execute|executing:执行中|success:成功|fail:失败	success
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
    "code": 0,
    "data": {
        "imageUploadStatus": "success"
    },
    "msg": "",
    "ts": 1601021167163
}
```

删除自定义镜像

基本信息

URL

> https://platform.armvm.com/resources/instance-image/remove

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
imageVersionIds	Long[]	必须		实例镜像版本ID	[10001]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object	必须			
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1728627866628

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {},
  "msg": "",
  "ts": 1728627866628
}
```

创建更新镜像任务

基本信息

URL

> https://platform.armvm.com/resources/instance-image/update

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
imageVersionId	Long	必须		镜像版本id	5
instanceCodes	String[]	必须		实例编号集合，最多200个	["VM010100240100"]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	必须			
├── instanceCode	String	必须		实例编号	VM010100240100
├── taskId	Long	必须		任务id	9001
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": [
    {
      "instanceCode": "VM010100240100",
      "taskId": 9001
    }
  ],
  "msg": "",
  "ts": 1601021167163
}
```


查询更新镜像结果

基本信息

URL

> https://platform.armvm.com/resources/instance-image/update-info

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
taskIds	Long[]	必须		任务ID集合，最多200个	[8001,8002]

返回参数

名称	类型	是否必须	默认值	备注	示例值
code	Integer	必须		错误码	0
data	Object[]	必须			
├─taskId	Long	必须		任务ID	8001
├─ instanceCode	String	必须		实例编号	VM010100240100
├─ imageUpdateStatus	String	必须		镜像更新状态，wait_execute:待执行|executing:执行中 success:成功 | fail:失败	success
msg	String	必须		错误信息	
ts	Long	必须		时间戳	1601188620078

请求示例

> 参考：调用方式

响应示例

```
{
 "code": 0,
 "data": [
  {
   "taskId": 8001,
   "instanceCode": "VM0101002401 00",
   "imageUpdateStatus": "success"
  },
  {
   "taskId": 8002,
   "instanceCode": "VM0101002401 01",
   "imageUpdateStatus": "executing"
  }
 ],
 "msg": "",
 "ts": 1601021167163
}
```


实例指令相关接口

实例重启

基本信息

URL

> https://platform.armvm.com/command/pad/reboot.html

请求格式

> POST

版本号

> 3

使用场景

> 重启指定实例编码的实例

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
padCodes	String[]	非必须		实例编码集合,集合长度在200内（包含200）	["VM192168001001","VM192168001002"]
merchantPoolNos	Long[]	非必须		实例池编号集合,集合长度在50内（包含50）	[1,2]

说明

padCodes与merchantPoolNos不能同时为空;
padCodes为空，merchantPoolNos不为空时则重启merchantPoolNos的所有实例;

返回参数

名称	类型	是否必须	默认值	备注	示例值	
code	Integer	必须		响应编码	0	
data	Object[]	非必须				
├─padCode	String	必须		实例编码	VM192168001001	
├─taskId	String	必须		任务编号	"2840204519134814256"	
msg	String	非必须		响应消息		
ts	Long	必须		时间戳	1598754678123	

请求示例

> 参考：调用方式

响应示例


```
{
  "code": 0,
  "data": {
     "padCode": "VM192168041004",
     "taskId": "2840204353481548485",
   },
  "msg": "ok",
  "ts": 1663664302000
}
```


异常示例


```
{
  "code": -1,
  "msg": "实例编号不存在",
  "ts": 1663664302000
}
```


实例重置

基本信息

URL

> https://platform.armvm.com/command/pad/reset.html

请求格式

> POST

版本号

> 3

请求参数

body

名称	类型	是否必须	默认值	备注	示例值
padCodes	String[]	非必须		实例编码集合,集合长度在200内（包含200）	["VM192168001001","VM192168001002"]
merchantPoolNos	Long[]	非必须		实例池编号集合,集合长度在50内（包含50）	[1,2]

说明

padCodes与merchantPoolNos不能同时为空;
padCodes为空，merchantPoolNos不为空时则重启merchantPoolNos的所有实例;

返回参数

名称	类型	是否必须	默认值	备注	示例值	
code	Integer	必须		响应编码	0	
data	Object[]	非必须				
├─padCode	String	必须		实例编码	VM192168001001	
├─taskId	String	必须		任务编号	"2840204519134814256"	
msg	String	非必须		响应消息		
ts	Long	必须		时间戳	1598754678123	

请求示例

> 参考：调用方式

响应示例

```
{
  "code": 0,
  "data": {
     "padCode": "VM192168041004",
     "taskId": "2840204353481548485",
   },
  "msg": "ok",
  "ts": 1663664302000
}
```

异常示例

```
{
  "code": -1,
  "msg": "实例编号不存在",
  "ts": 1663664302000
}
```

服务回调相关接口

回调接口使用示例

> 以实例重启回调为例，其他回调接口使用方法类似。

配置说明

> 需要客户在客户开放平台配置实例重启通知的回调地址，如接口地址为http://host:port/api/v1/instance/reboot-fb，则配置为http://host:port/api/v1/instance/reboot-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Long	必须		任务编号	1
padCode	String	必须		实例编码	VM123
msg	String	必须		返回信息	ok
result	Boolean	必须		上传结果的标识。true：成功，false：失败	true

客户服务端接收请求示例

Java

```
 // 配置好客户自己的desKey，可在开放平台的账号管理菜单下查看和重置
private String desKey = "d3b5c2b2e480e9e452c8b9f75fbe192b";
private String appSecret = "5e5b5f13052dfc53159e2069af12e6cf";

@PostMapping("http:127.0.0.1:8888/api/v1/instance/reboot-fb")
public Map<String, Object> instanceReboot(String appkey, String nonce, String s, String auth_ver, @RequestBody
(required = false) String body) throws Exception {
    // 校验签名
    StringBuilder paramsStr = new StringBuilder();
    paramsStr.append("appkey").append(appkey)
            .append("auth_ver").append(auth_ver)
            .append("nonce").append(nonce)
            .append(appSecret);//接口签名密钥（可配置）
    String sign = DigestUtils.md5Hex(org.apache.commons.codec.binary.StringUtils.getBytesUtf8(paramsStr.toString()));
    if (!sign.equals(s)) {
        // 签名校验不合法
    }

    Map<String, Object> bodyMap = new ObjectMapper().readValue(body, Map.class);
    Long bodyNonce = (Long) bodyMap.get("createTime");
    String msg = (String) bodyMap.get("msg");
    byte[] key = desKey.getBytes("UTF-8");
    ByteBuffer byteBuffer = ByteBuffer.allocate(8);
    byteBuffer.order(ByteOrder.BIG_ENDIAN);
    byteBuffer.putLong(bodyNonce);
    byte[] iv = byteBuffer.array();
    byte[] message = Base64.decodeBase64(msg);
    Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
    SecretKeyFactory skf = SecretKeyFactory.getInstance("DESede");
    SecretKey secretKey = skf.generateSecret(new DESedeKeySpec(key));
    IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
    cipher.init(Cipher.DECRYPT_MODE, secretKey, ivParameterSpec);
    byte[] bytes = cipher.doFinal(message);

    // 获取回调的内容
    String requestBody = new String(bytes, "UTF-8");
    logger.info("回调消息, request body : {}", requestBody);

    // 处理客户自己的业务逻辑

    // 返回响应信息
    Map<String, Object> responseHeader = new HashMap<>();
    responseHeader.put("time", System.currentTimeMillis());
    responseHeader.put("version", "2");
    responseHeader.put("status", 200);

    Map<String, Object> response = new HashMap<>();
    response.put("responseHeader", responseHeader);
    return response;
}
```

测试请求示例

Java

```
 // PaaS 平台分配给第三方用户的接口Key
String appkey = "50b13132bb394901f151bc45";
// 接口版本号，现在有2和3两种版本的接口
String auth_ver = "2";
// 当前的时间戳
long nonce = new Date().getTime();
// 方法签名，生成示例见'方法签名生成示例'
String s = "efabcaca7c66d629a639cee37d924813";
//desKey 3DES秘钥（可配置）,根据商户平台实际生成的替换此值
String desKey = "d3b5c2b2e480e9e452c8b9f75fb1qazs";
// 拼接公共参数到URI
String apiUrl = "http://host:port/api/v1/instance/reboot-fb?s=" + s + "&appkey=" + appkey + "&auth_ver=" + auth_ver + "&nonce=" + nonce;

// 填装请求参数
Map param = new HashMap();
param.put("taskId", 1);
param.put("result", true);
param.put("msg", "ok");
param.put("padCode", "VM123");

// 请求参数转化为JSON字符串
String requestParam = new Gson().toJson(param);

// 请求参数加密
Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
SecretKeyFactory skf = SecretKeyFactory.getInstance("DESede");
```

```
SecretKey secretKey = skf.generateSecret(new DESedeKeySpec(desKey.getBytes("utf-8")));
ByteBuffer byteBuffer = ByteBuffer.allocate(8);
byteBuffer.order(ByteOrder.BIG_ENDIAN);
byteBuffer.putLong(nonce);
byte[] iv = byteBuffer.array();
IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivParameterSpec);
byte[] cipherText = cipher.doFinal(requestParam.getBytes("utf-8"));
String msg = Base64.encodeBase64String(cipherText);
Map map = new HashMap();
map.put("createTime", nonce);
map.put("msg", msg);
// 转化为JSON字符串
String encryptBody = new Gson().toJson(map);

// 构造请求
URL url = new URL(apiUrl);
HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
// 连接超时时间
httpURLConnection.setConnectTimeout(1000);
// 读取结果超时时间
httpURLConnection.setReadTimeout(1000);
httpURLConnection.setDoInput(true);
httpURLConnection.setDoOutput(true);
httpURLConnection.setUseCaches(false);
httpURLConnection.setRequestProperty("Content-type", "application/json;charset=utf-8");
httpURLConnection.setRequestMethod("POST");
if ("https".equalsIgnoreCase(url.getProtocol())) {
    HttpsURLConnection husn = (HttpsURLConnection) httpURLConnection;
    husn.setSSLSocketFactory(new BaseHttpSSLSocketFactory());
    //解决由于服务器证书问题导致HTTPS无法访问的情况
    husn.setHostnameVerifier(new BaseHttpSSLSocketFactory.TrustAnyHostnameVerifier());
    httpURLConnection = husn;
}
// 得到请求的输出流对象
OutputStreamWriter out = new OutputStreamWriter(httpURLConnection.getOutputStream());
// 把数据写入请求的Body
out.write(encryptBody);
out.flush();
out.close();

String result = "";
try (InputStream in = 200 == httpURLConnection.getResponseCode() ? httpURLConnection.getInputStream() : httpURLConnection.getErrorStream();) {
    byte[] buf = new byte[1024];
    int length = 0;
    ByteArrayOutputStream bout = new ByteArrayOutputStream();
    while ((length = in.read(buf, 0, buf.length)) > 0) {
        bout.write(buf, 0, length);
    }
    bout.flush();
    result = new String(bout.toByteArray(), StandardCharsets.UTF_8);
} catch (Exception e) {
    throw e;
} finally {
    if (null != httpURLConnection) {
        httpURLConnection.disconnect();
    }
}
int responseCode = httpURLConnection.getResponseCode();
if (responseCode == 200) {
    System.out.println("调用成功,接口返回结果:\n" + result);
} else {
    System.out.println("调用失败");
}
```

请求参数示例

```
{
  "taskId": 1,
  "padCode": "VM123",
  "result": true,
  "msg": "ok"
}
```

应用上传通知

基本信息

URL

> http://host:port/api/v1/apps/upload-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云游戏应用上传接口，应用上传情况会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置保存应用通知的回调地址，如接口地址为http://host:port/api/v1/apps/upload-fb，则配置为http://host:port/api/v1/apps/upload-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Integer	必须		任务编号	1
padCode	String	必须		实例编码	VM123
packageName	String	必须		应用包名	com.demo
taskStatus	String	必须		任务结果：7： 成功 ； 8： 失败	7
apps	Object	必须		应用信息	
├─appId	Long	必须		应用ID	10001
├─appName	String	必须		应用名称	demo
├─pkgName	String	必须		包名	com.demo
├─result	boolean	必须		上传结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

应用启动通知

基本信息

URL

> http://host:port/api/v1/apps/start-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云游戏应用启动接口后，应用启动情况会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置应用启动通知的回调地址，如接口地址为http://host:port/api/v1/apps/start-fb，则配置为http://host:port/api/v1/apps/start-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Integer	必须		任务编号	1
padCode	String	必须		实例编码	VM123
packageName	String	必须		应用包名	com.demo
operateType	String	必须		startApp：应用启动	startApp
taskStatus	String	必须		任务结果：7： 成功 ； 8： 失败	7
apps	Object	必须		应用信息	
├─appId	Long	必须		应用ID	10001
├─padCode	String	必须		实例编码	VM123
├─appName	String	必须		应用名称	demo
├─pkgName	String	必须		包名	com.demo
├─result	boolean	必须		上传结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

应用停止通知

基本信息

URL

> http://host:port/api/v1/apps/stop-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云游戏应用停止接口后，应用停止情况会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置应用停止通知的回调地址，如接口地址为http://host:port/api/v1/apps/stop-fb，则配置为http://host:port/api/v1/apps/stop-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Integer	必须		任务编号	1
padCode	String	必须		实例编码	VM123
packageName	String	必须		应用包名	com.demo
operateType	String	必须		killApp：应用停止	killApp
taskStatus	String	必须		任务结果：7： 成功 ； 8： 失败	7
apps	Object	必须		应用信息	
├─appId	Long	必须		应用ID	10001
├─padCode	Long	必须		实例编码	VM123
├─appName	String	必须		应用名称	demo
├─pkgName	String	必须		包名	com.demo
├─result	boolean	必须		上传结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

实例解绑通知

基本信息

URL

> http://host:port/api/v1/instance/unbind-fb

请求格式

> POST

版本号

> 3

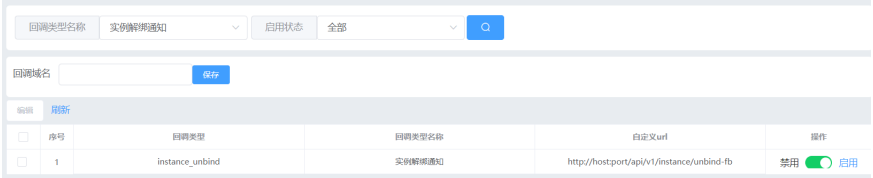
使用场景

> 客户调用百度云游戏实例解绑接口后，实例解绑情况会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置实例解绑通知的回调地址，如接口地址为http://host:port/api/v1/instance/unbind-fb，则配置为http://host:port/api/v1/instance/unbind-fb

配置示例



请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
uuid	String	必须		uuid	123456
padCode	String	必须		实例编码	VM123123123123
sessionId	String	必须		会话ID	JKWEUNBL6RX3WCBUQ92AGYCU9MLX0WTB
endTimestamp	Long	必须		完成解绑操作的时间戳	1669020092256

使用示例

> 参考：回调接口使用示例

实例重置通知

基本信息

URL

> http://host:port/api/v1/instance/reset-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云游戏实例重置接口后，实例重置情况会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置实例重置通知的回调地址，如接口地址为http://host:port/api/v1/instance/reset-fb，则配置为http://host:port/api/v1/instance/reset-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Integer	必须		任务编号	1
padCode	String	必须		实例编码	VM123
msg	String	必须		返回信息	ok
result	boolean	必须		上传结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

实例重启通知

基本信息

URL

> http://host:port/api/v1/instance/reboot-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云游戏实例重启接口后，实例重启情况会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置实例重启通知的回调地址，如接口地址为http://host:port/api/v1/instance/reboot-fb，则配置为http://host:port/api/v1/instance/reboot-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Integer	必须		任务编号	1
padCode	String	必须		实例编码	VM123
packageName	String	必须		应用包名	com.demo
operateType	String	必须		startApp：应用启动	startApp
taskStatus	String	必须		任务结果：7： 成功 ； 8： 失败	7
apps	Object	必须		应用信息	
├─appId	Long	必须		应用ID	10001
├─appName	String	必须		应用名称	demo
├─pkgName	String	必须		包名	com.demo
├─result	boolean	必须		上传结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

实例执行脚本通知

基本信息

URL

> http://host:port/api/v1/instance/execute-script-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云游戏实例执行脚本接口后，脚本执行情况会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置实例重启通知的回调地址，如接口地址为http://host:port/api/v1/instance/execute-script-fb，则配置为http://host:port/api/v1/instance/execute-script-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Long	必须		任务编号	1
padCode	String	必须		实例编码	VM123
msg	String	必须		返回信息	ok
result	Boolean	必须		上传结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

上传文件到实例通知

基本信息

URL

> http://host:port/api/v1/instance/file-upload-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云游戏上传文件到实例接口后，文件上传结果会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置实例重启通知的回调地址，如接口地址为http://host:port/api/v1/instance/file-upload-fb，则配置为http://host:port/api/v1/instance/file-upload-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Long	必须		任务编号	1
padCode	String	必须		实例编码	VM123
msg	String	必须		返回信息	ok
result	Boolean	必须		上传结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

桌面推荐图标刷新通知

基本信息

URL

> http://host:port/api/v1/instance/recommend-app-icon-refresh

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云桌面推荐图标刷新，刷新结果会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置桌面推荐图标刷新通知的回调地址，如接口地址为http://host:port/api/v1/instance/recommend-app-icon-refresh，则配置为http://host:port/api/v1/instance/recommend-app-icon-refresh

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Integer	必须		任务编号	1
padCode	String	必须		实例编码	VM192168043080
msg	String	必须		返回信息	ok
result	boolean	必须		上传结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

实例备份通知

基本信息

URL

> http://host:port/api/v1/instance/reset-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云实例备份接口后，实例备份的结果会通过该回调接口通知。

配置说明

> 需要在开放平台配置实例备份通知的回调地址，如接口地址为http://host:port/api/v1/instance/backup-fb，则配置为http://host:port/api/v1/instance/backup-fb

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Integer	必须		任务编号	1
padCode	String	必须		实例编码	VM123
msg	String	必须		返回信息	ok
result	boolean	必须		任务结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

实例还原通知

基本信息

URL

> http://host:port/api/v1/instance/restore-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用百度云游戏实例重置接口后，实例重置情况会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置实例重置通知的回调地址，如接口地址为http://host:port/api/v1/instance/restore-fb，则配置为http://host:port/api/v1/instance/restore-fb

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Integer	必须		任务编号	1
padCode	String	必须		实例编码	VM123
msg	String	必须		返回信息	ok
result	boolean	必须		上传结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

实例分辨率调整通知

基本信息

URL

> http://host:port/api/v1/instance/resolution-fb

请求格式

> POST

版本号

> 3

使用场景

> 调用实例分辨率调整接口后，调整结果会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置实例分辨率调整通知的回调地址，如接口地址为http://host:port/api/v1/instance/resolution-fb，则配置为http://host:port/api/v1/instance/resolution-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
taskId	Long	必须		任务编号	1
padCode	String	必须		实例编码	VM123
msg	String	必须		返回信息	ok
result	Boolean	必须		执行结果的标识。true：成功，false：失败	true

使用示例

> 参考：回调接口使用示例

进程死亡告警通知

基本信息

URL

> http://host:port/api/v1/monitor/process-died-alarm-fb

请求格式

> POST

版本号

> 3

使用场景

> 实例应用进程异常死亡后，进程死亡信息会通过该回调接口通知给客户。

配置说明

> 需要客户在客户开放平台配置进程死亡告警通知通知的回调地址，如接口地址为http://host:port/api/v1/monitor/process-died-alarm-fb，则配置为http://host:port/api/v1/monitor/process-died-alarm-fb

配置示例

请求参数

Body

名称	类型	是否必须	默认值	备注	示例值
instanceCode	String	必须		实例编码	VM123
content	Object	必须			
├──packageName	String	必须		应用包名	com.example.app
├──reason	String	必须		进程死亡原因；参考reason枚举表；	process other exception died
├──controlStatus	String	必须		实例受控状态；control：受控，online：非受控；	online
├──uuid	String	必须		实例绑定的uuid	
├──recordTime	String	必须		进程死亡时间	2024-10-30 12:00:00

reason枚举

reason	含义
anr	进程anr被终止
bg anr	后台进程anr被终止
crash	进程崩溃被终止
start timeout	进程启动超时被终止
depends on provider	所绑定的provider已销毁所以被杀
lmk-kill	进程被Low Memory Killer终止

使用示例

> 参考：回调接口使用示例

返回结果

版本一

返回参数

名称	类型	是否必须	默认值	备注	示例值
resultCode	Integer	必须		操作结果代码，0代表成功 -1代表失败 9999代表系统错误	0
resultInfo	String	必须		操作具体描述以及数据封装	

版本二

返回参数

名称	类型	是否必须	默认值	备注	示例值	
code	Integer	必须		响应编码	0	
data	Object[]	非必须				
msg	String	非必须		响应消息		
ts	Long	必须		十三位时间戳	1598754678123	

错误码

错误码	错误信息	备注
11100050	系统异常，请联系客服人员	
11101100	商户信息异常	
11101118	商户类型错误,只允许云手机客户使用	
11102027	serverToken无效	
11102032	serverToken无效，[serverToken]	
11103062	实例不存在	
11103078	实例不属于此用户	
11103103	无设备可绑定	
11103180	实例池不存在	
11103183	实例池中无实例	
11103184	实例编号和实例池编码不能同时为空	
11103185	实例池不存在或无实例	
11103190	实例池不存在，[实例池编号]	
11108003	工作任务不存在	
11108009	找不到脚本[脚本名称],请确定是否有配置	
11109002	内部服务通信失败	

百度云手机数据互通Android SDK文档

百度云手机账号互通Android SDK主要功能是为了打通用户手机与云手机之间数据传输，将用户信息同步到云手机，实现账号登录功能，以及将云手机上游戏支付订单发送到用户本地手机，完成订单支付。

[TOC]

运行环境

可运行于 **Android 4.0(API Level 14) 及 以 上 版 本** 。

数据互通Android SDK 下载

注：当您使用百度云手机账号互通SDK即您表示默认同意相关服务条款及隐私政策；若您不同意相关服务协议及隐私政策，应立即停止接入并停止使用百度云手机互通SDK。

版本号	更新时间	下载地址	备注
V 1.1.11	2022.10.26	Android SDK下载	增加日志排查

SDK配置

导入aar包 SDK包含operationsdk-cloud-xxx.aar，请将aar文件复制到项目libs目录，并根据如下代码配置build.gradle

```
 android {
  //android标签内添加aar目录
   repositories {
      flatDir {
         dirs 'libs'
      }
   }
}

dependencies {
 //添加依赖
 compile fileTree(dir: 'libs', include: ['*.jar'])
   compile(name: 'operationsdk-cloud-xxx', ext: 'aar')
}
```

API

检查云手机环境 通过该接口可以判断当前环境是否为云手机环境，游戏可以根据此接口做出相应的策略。

```
boolean result = BDGameSDK.getInstance(this).isCloudPhoneEnvironment()
```

注册客户端数据监听 在App自定义Application#onCreate方法中注册监听，用于获取客户端发送的指令以及数据。游戏端需处理CUSTOM_DATA类型的Action事件，根据附录中Action参数说明从data数据集中取出对应数据 *必须在程序启动初始化处注册，否则收不到消息*

```
BDGameSDK.getInstance(this).listenerClient(new BDGameActionListener() {
        @Override
        public void processAction(OperationAction action, Map<String, String> data) {
            switch (action){
                case CUSTOM_DATA:
                    String uid = data.get("uid");
                    // do login
                    break;
            }
        }
    });
```

往客户端发送消息 在App自定义Application#onCreate方法中注册监听，用于获取客户端发送的指令以及数据。游戏端需处理CUSTOM_DATA类型的Action事件，根据附录中Action参数说明从data数据集中取出对应数据 *必须在程序启动初始化处注册，否则收不到消息*

```
 /**
 * 发送自定义数据接口,使用的是CUSTOM_DATA action事件
 * @param data  发送得数据map
 * @param  isEncode  是否需要加密发送,默认是加密
 */
BDGameSDK.getInstance(this).sendCustomDataToClient(Map<String, String> data, boolean isEncode);
/**
 * 发送指定action数据
 * @param action  发送得action,客户端需要选择对应action接收
 * @param data  发送得数据json
 * @param  isEncode  是否需要加密发送,默认是加密
 */
BDGameSDK.getInstance(this).sendDataToClient(OperationAction action, JSONObject data, boolean isEncode);
```

查看接收消息日志开关

```
BDGameSDK.getInstance(this).setDebugLog(true);
```

附录

Proguard说明

```
-keep enum * {
  *;
}
```

云手机Web SDK

云手机Web SDK 主要功能是为Web赋予直连云手机的能力，用户在运营平台配置好云手机之后，Web端通过SDK进行直连操作。

Web SDK 发行版包

> 云手机Web SDK 发行版包括云手机Web SDK文档、JSSDK 文档、完整的Demo示例。以下使用 <SDK_PATH> 表示SDK解压根目录。
>
> - **对接文档**：<SDK_PATH>/sdk/云手机Web SDK文档.md，原文档
> - **JSSDK 文档**：<SDK_PATH>/sdk/JSSDK 文档.docx，JSSDK文档

- **示例项目源码**：<SDK_PATH>/sdk/index.html，Demo文件
- **Web SDK**：<SDK_PATH>/sdk/websdk/*，Web SDK文件，封装了请求sdk接口的相关方法
- **JSSDK**：<SDK_PATH>/sdk/jssdk/*，JSSDK文件，JSSDK封装对直连实例进行操作的相关方法和回调

云手机 Web SDK 使用

版本号	更新时间	下载地址	备注
latest	-	Web SDK下载	最新版本
V 1.0.5	2021.12.27	Web SDK下载	性能优化，提升稳定性
V 1.0.4	2021.08.30	Web SDK下载	JSSDK支持按需开启语音权限、摄像头上下旋转
V 1.0.3	2021.07.09	Web SDK下载	JSSDK支持支持开启扫码和人脸识别功能、更新Demo
V 1.0.2	2021.07.02	Web SDK下载	新增透传云端方法
V 1.0.1	2021.06.28	Web SDK下载	更新优化index.html文件
V 1.0.0	2021.06.11	Web SDK下载	包含Web SDK对接相关文件

运行环境

需要服务器运行环境，添加域名白名单，支持现代浏览器PC、H5端。

SDK引入及配置

申请密钥对(AK和SK)

登录百度智能云控制台，进入“云手机>安全组> 秘钥对”页面

ID	AK	SK	类型
n	u7fkUQ3mjGaqZAHr08**********	HiIkc0fp8rGVKo3T7a2QeEtDmR5vX4jUWO******	SDK

引入SDK

WebSDK 依赖 JSSDK，需要按顺序引入。latest代表最新版，也可以替换为具体版本，如redfinger.min.210602.js。

```
 <!--JSSDK-->
<script src="//bj.bcebos.com/v1/yunapp-static/bac/sdk/jssdk/redfinger.min.latest.js"></script>
<!--Web SDK-->
<script src="//bj.bcebos.com/v1/yunapp-static/bac/sdk/websdk/yap-utils.umd.latest.js"></script>
```

SDK权限配置

需要提供开发者域名并联系相关人员添加域名白名单，未添加白名单访问会出现跨域问题。

快速入门

HTML:

```
  <button onclick="play()">直连</button>
<!--实例标签-->
<div id="play-box" style="height: 500px"></div>
<!--JSSDK-->
<script src="//bj.bcebos.com/v1/yunapp-static/bac/sdk/jssdk/redfinger.min.latest.js"></script>
<!--Web SDK-->
<script src="//bj.bcebos.com/v1/yunapp-static/bac/sdk/websdk/yap-utils.umd.latest.js"></script>
```

JavaScript:

```
  // 替换百度智能云-云手机 AK、SK
 const ak = 'PVr5RaC0p58Bo***';
 const sk = 'VBy9VWdaUP1q4W8xLXrr2MEZqMDKe***';
 /*
 * 获取实例编号地址 https://console.bce.baidu.com/bac#/example/list 实例编号
 * 设备编码：非必填，padCode与appId不能同时为空，同时存在padCode优先，存在padCode则直连云手机
 * 获取设备编码地址 https://console.bce.baidu.com/bac#/app/list 应用ID
 * 应用id：非必填，应用id不为空表示应用直连，否则为云手机直连。直连应用还需要填写pkgName用来使SDK启动云手机应用，appId通知服务端直连应用。
 * userId：必填，内容自定义
 * */
 const padCode = 'VM010151089016';
 const appId = '100705';
 const userId = 'baidu';
 const pkgName = 'com.xxx.xxx';

 const Sign = yapUtils.Sign; // 签名方法
 const Tools = yapUtils.tools; // 接口封装
 const axios = yapUtils.axios; // 封装axios

 function play() {
   // 申请直连实例接口签名
   let sign = new Sign({
     ak: ak,
     sk: sk,
     host: 'https://yunapp-api.baidu.com/api/armcmapi',
     path: '/sdk/v1/device/connect/apply',
     params: {
       padCode: padCode,
       appId: appId,
       userId: userId,
     }
   });
   // 申请直连实例
   axios({
     method: 'post',
     url: sign.getUrl(), // 获取签名后的请求地址
     data: sign.getBody(), // 获取签名后的请求体
     headers: {
       appKey: ak
     }
   }).then(res => {
     // 获取直连设备信息
     if (res.data.data && res.data.data.connectInfo) {
   const deviceToken = JSON.stringify(res.data.data.connectInfo.deviceToken);
       // 发起直连，Tools.play 封装了 redfinger.play，参考JSSDK 文档1.直连实例
   Tools.play({
         app: {pkgName: pkgName}, // 直连应用包名
         deviceToken
       }, redfinger, {})
     }
   });
 }
```

JSSDK

JSSDK 封装对直连实例进行操作的相关方法和回调，JSSDK 的核心对象为 window.redfinger。

redfinger 属性

```
  redfinger.config // 返回实例配置信息
redfinger.version // 返回JSSDK版本
```

redfinger 方法

```
  redfinger.sendCommand(redfinger.KEY_TYPE. KEY_HOMEPAGE) // 指令控制方法 - 返回桌面
redfinger.destory(); // 退出直连
```

redfinger 回调

```
redfinger.onRunInformation = function(type, info) { // 运行信息回调
  console.log(type, info)
}
```

redfinger H5&云端消息透传相关方法回调

```
// 云端透传失败回调
redfinger.onTransparentMsgFailed = function (type, msg, service) {
  console.log('onTransparentMsgFailed', type, msg, service)
}

// 云端透传回调
redfinger.onTransparentMsg = function (type, msg, service) {
  console.log('onTransparentMsg', type, msg, service)
}

// 透传云端消息
redfinger.sendTransparentMsg(type, otpions, packageName);

// type 类型H5端默认为 1
// packageName ${应用包名}/com.baidu.operationsdk.BDGameService
// otpions 透传信息，数据格式需要转化为json格式
var options = {
  'pkg': 'com.yunserver.serverapp', // 应用包名
  'action': 'CUSTOM_DATA', // 透传动作
  'extraData': {'uid': '设备号:VM010151094070 时间戳:1625140088930'} // 透传自定义信息
};

// options 数据格式转换JSON
options = JSON.stringify(options);

// 发送示例：
redfinger.sendTransparentMsg(1, options, 'com.yunserver.serverapp/com.baidu.operationsdk.BDGameService');

// 实际发送数据格式：
redfinger.sendTransparentMsg(1,'{\\"pkg\\":\\"com.yunserver.serverapp\\",\\"action\\":\\"CUSTOM_DATA\\",\\"extraData\\":{\\"uid\\":\\"设备
号:VM010151094070 时间戳:1625140088930\\"}}', 'com.yunserver.serverapp/com.baidu.operationsdk.BDGameService');
```

redfinger 支持扫码和人脸识别功能

```
redfinger.config.isCameraLive = true // 开启扫码功能
```

redfinger 开启语音功能

```
redfinger.config.isMicrophoneLive = true; // 开启语音权限
```

redfinger 摄像头上下反转

```
// 解决部分云手机设备摄像头上下倒置问题
redfinger.config.cameraTransform = 180; // 值为摄像头上下旋转角度，可选项 0、180
```

更多API请参考：JSSDK 文档

Web SDK

Web SDK 封装了Web端请求SDK接口的相关方法，Web SDK的核心对象为 window.yapUtils。

yapUtils.Sign

yapUtils封装了请求SDK接口的签名方法。

```
// 实例化Sign给SDK申请直连实例接口签名
const sign = new Sign({
  ak: ak,
  sk: sk,
  host: 'https://yunapp-api.baidu.com/api/armcmapi',
  path: '/sdk/v1/device/connect/apply',
  params: {
    padCode: padCode,
    appId: appId,
    userId: userId,
    onlineTime: 60 * 10
  }
});
```

yapUtils.axios

yapUtils对axios进行了封装。

```
  // 使用签名后的地址和请求提发起 axios 请求
axios({
  method: 'post',
  url: sign.getUrl(), // 获得签名后请求地址
  data: sign.getBody(), // 获得签名后的请求体
  headers: {
    appKey: ak
  }
}).then(res => {
  console.log('申请直连实例返回：', res);
});
```

yapUtils.tools

yapUtils.tools 封装了redfinger的部分方法和回调。

1. Tools.play() 接口参数说明

参数：类型	必填	默认值	释义
response: PlayResponse	是	-	Tools.getDevice()接口返回的数据，PlayResponse 格式见下表1.1
redfinger: any	是	-	百度红手指提供的全局对象 redfinger
opt: InitConfig	是	-	参数配置，回调接口，InitConfig 格式见下表1.2

1.1 response: PlayResponse 数据结构说明

参数：类型	必填	默认值	释义	备注
app: object	是	{pkgName:""}	包名	包名传空，会直接拉起云手机传包名，会拉起对应app
`deviceToken: string` | 是 | - | token | 这个token，是百度智能云手机开放接口返回的一个字符串				

1.2 opt: InitConfig 格式说明

参数：类型	必填	默认值	释义	备注
isWebRTC: boolean	否	falses	是否开启webRTC	
divName: string	否	'play-box'	投屏区域，div 的id	
onError: Function	否	noop	报错的回调	
isWss: boolean	否	false	是否使用wss	
keyboard：boolean	否	false	是否开启键盘输入	
onTimeout: Function	否	noop	设备闲置到了设定的时间的回调	
onStartPlay: Function	否	noop	开始直连前的回调	
onWebrtcRunInfo: Function	否	noop	webRtc 投屏信息的回调	
onOpen: Function	否	noop	出现画面的回调	
timeout: number	否	3	无操作时间 ，单位分，前台闲置超时时长后台闲置超时时间需要联系接口人更改相应控制服务	
adapt:Adapt	否	-	直连云手机的画质信息，格式见下表2.2.1	

1.2.1 adapt:Adapt 格式说明 注意：如果运营平台适配的应用的参数，将会采用运营平台的设置

参数：类型	必填	默认值	释义	备注
compressionType: number	否	3	压码方式3: 硬压4: 软压	
sound: number	否	1	1: 开启声音!1: 关闭声音	
fps: number	否	20	帧率	
bitrate: number	否	3200	码率	
resolution: number	否	3	分辨率1: 低；2: 标清 480p；3: 高清 720p；4: 超清 1080p	非webRTC 的情况下，超清是720p, 建议手动设置为4

附录 Web SDK & JSSDK

SDK版本记录

JSSDK：

- redfinger.min.210602.js：支持百度智能云云手机直连播放
- redfinger.min.210701.js：新增H5&云端消息透传相关方法回调

- redfinger.min.210709.js：支持开启扫码和人脸识别功能
- redfinger.min.210726.js：JSSDK 默认支持语音功能
- redfinger.min.210804.js：JSSDK 优化
- redfinger.min.210810.js：修复bug
- redfinger.min.210825.js：支持摄像头上下反转
- redfinger.min.210826.js：按需开启语音权限
- redfinger.min.1.5.6.js：优化性能，提升稳定性
- redfinger.min.latest.js：redfinger.min.1.5.6.js

Web SDK:

- yap-utils.umd.210611.js：支持百度智能云云手机直连播放
- yap-utils.umd.latest.js：yap-utils.umd.210611.js

DEMO示例

测试地址：https://bj.bcebos.com/v1/yunapp-static/bac/sdk/index.html

DEMO源码：bac/sdk/index.html

二维码：

Q & A

1. 部署DEMO需要替换AK、SK、padCode等。运行DEMO需要服务器运行环境，如部署服务器需要添加域名白名单才能访问，否则会出现跨域问题：Access to XMLHttpRequest at 'https://yunapp-api.baidu.com/api/armcmapi/sdk/v1/device/connect/apply?padCode=VM010151089016&appId=100705&onlineTime=600&auth_ver=2&appkey=PVr5RaC0p58BoXQ2&time=1623728218793&sign=a90d3aa4f2db4bfa5910529526f0ea20' from origin 'http://localhost:63342' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource.。
2. 直连提示message: 没有可直连该应用的实例，status: 5800003。设备被占用，需要释放上一次直连设备，或者直连其他设备。
3. 释放设备周期，经历哪些阶段，需要多久？

设备释放是异步的，根据实例配置，释放直连设备后可能需要经过清理缓存、重启设备、初始化等，时间无法准确评估。

4. 直连云手机模式。

直连云手机分为两种模式，js模式和Webrtc模式，两种模式都基于WebSocket。

js模式会在浏览器中插入canvas进行渲染，Webrtc模式会在浏览器中插入video进行播放云手机画面，性能更优。

直连时使用Webrtc模式需要修改isWebRTC和升级设备支持Webrtc，火狐浏览器暂不支持Webrtc模式。

Tools.play({ app: {pkgName: ''}, deviceToken: deviceToken }, redfinger, { isWebRTC: true, // 是否开启Webrtc。（开启Webrtc以后性能会有所提升，需要升级设备支持Webrtc） isWss: true, // 是否支持https })

5. 是否支持https？

直连云手机支持https，直连时isWss设置为true即可，默认为false。如果isWss为false时，通过https访问则会报错：Mixed Content: The page at 'blob:https://bj.bcebos.com/448d8622-b8ab-456d-b426-c5e4ba1b7256' was loaded over HTTPS, but attempted to connect to the insecure WebSocket endpoint 'ws://gznx.cloud-control.top:9701/'. This request has been blocked; this endpoint must be available over WSS.

6. 如何调用云手机home键？

参考JSSDK 文档第12.指令控制方法 使用场景：当用户按下返回、home键、任务键（Menu）时 使用方法： redfinger.sendCommand(command) 参数说明：包含返回、Home、和Menu redfinger.KEY_TYPE = { KEY_MENU：139, KEY_BACK：158, KEY_HOMEPAGE：172 } 返回桌面：举例 redfinger.sendCommand(redfinger.KEY_TYPE.KEY_HOMEPAGE) 调试技巧： 直连云手机后，直接在浏览器控制台输入此方法即可执行。

相关协议&政策

服务等级协议SLA

服务水平协议SLA

协议生效时间：2024年10月24日

本服务等级协议（Service Level Agreement，以下简称 "SLA"）规定了百度智能云向客户提供的ARM云手机（Arm Cloud Mobilephone，简称"ARMCM"）服务可用性等级指标及赔偿方案。

1. 服务可用性定义

1.1 服务周期：一个服务周期为一个自然月

1.2 服务周期总分钟数：服务周期内的总天数×24（小时）×60（分钟）

1.3 服务不可用分钟数：当某一分钟内，甲方所有尝试与指定的云手机实例建立连接的连续请求均失败，则视为该分钟内该云手机实例服务不可用。在一个服务周期内，单个云手机实例的不可用分钟数为其所有不可用分钟数的总和。

1.4 服务可用性计算公式：以单台云手机实例为维度，服务可用性=【（服务周期总分钟数-服务不可用分钟数）/服务周期总分钟数】*100%

2. 服务可用性

2.2 服务可用性承诺 2.1 对于单个云手机实例维度，乙方承诺一个服务周期内云手机实例的服务可用性不低于99%；

2.2 如云手机实例服务未达到上述服务可用性承诺，甲方可以根据约定获得赔偿。赔偿范围不包括以下原因所导致的不可用：

1）乙方预先通知甲方后进行系统维护所引起的，包括割接、维修、升级和模拟故障演练；

2）任何乙方所属设备以外的网络、设备故障或配置调整引起的；

3）甲方的应用程序或数据信息受到黑客攻击而引起的；

4）甲方维护不当或保密不当致使数据、口令、密码等丢失或泄漏所引起的；

5）甲方的疏忽或由甲方授权的操作所引起的；

6）不可抗力以及意外事件引起的；

7）未达到服务可用性承诺的云手机实例只赔付不高于同等数量的云手机实例服务自身，不赔付云手机服务所关联的其他云服务；

8）其他非乙方原因所造成的云手机实例服务无法正常使用。

3. 服务赔偿条款

3.1. 赔偿标准 1）对于单个云手机实例，可按照下表中的标准获得赔偿，赔偿方式仅限于抵扣结算费用且赔偿总额不超过未达到服务可用性承诺当月客户甲方就该云手机实例支付的当月单个云手机实例月度服务费。

服务可用性（%）	**赔付时长（小时）**
等于或高于99%	不赔付
低于99%但等于或高于95%	(99%-服务可用性)*月度自然日天数*24小时*2
低于95%	(99%-服务可用性)*月度自然日天数*24小时*3

3.2. 赔偿申请时限 甲方可以在每月收到账单后3个工作日内提出。超出申请时限的赔偿申请将不被受理，甲方有客观合理事由延期申请赔偿的除外。

百度云手机软件开发工具包隐私政策

百度云手机软件开发工具包隐私政策

欢迎使用百度云手机软件开发工具包（SDK）（简称“百度云手机SDK”）服务！

百度云手机SDK 是一款为移动应用开发者（以下简称“开发者”）提供百度云手机仿真务的软件开发工具包（SDK）。开发者在其移动应用内集成百度云手机SDK后，可通过百度云手机SDK平台向其移动应用（以下简称“开发者应用”）的最终用户（以下简称“最终用户”）提供本隐私政策所说明的功能及服务。 此隐私政策旨在帮助开发者及最终用户了解我们收集最终用户个人信息的类型及我们如何利用和保护最终用户的个人信息。为了便于开发者及最终用户阅读及理解，我们将专门术语进行了定义，请参见本隐私政策“附录1：名词解释”来了解这些定义的具体内容。

特别说明：

1、如果开发者在其移动应用中集成并使用百度云手机SDK服务，则开发者应承诺：

（1）开发者应遵守收集、使用最终用户个人信息有关的所有可适用法律、政策和法规，保护用户个人信息安全。

（2）开发者应将在其移动应用中集成并使用百度云手机SDK服务的情况，以及百度云手机SDK对最终用户必要个人信息的收集、使用和保护规则（即本隐私政策），在其移动应用的显著位置或以其他可触达最终用户的方式告知最终用户（包括但不限于：在其移动应用隐私政策显眼处提供最终用户可浏览本隐私政策的链接），并获得最终用户对于百度云手机SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度云手机SDK服务期间持续有效的授权同意。如果开发者的移动应用最终用户是未满14周岁的未成年人，请开发者务必确保获得最终用户的父母或其他监护人对于百度云手机SDK收集、使用最终用户相关个人信息的完整、合法、在使用百度云手机SDK服务期间持续有效的授权同意。

（3）开发者应向最终用户提供易于操作的查阅、更正、补充、删除、复制或转移其个人信息，撤回或更改其授权同意，注销其个人账号，要求开发者就个人信息处理规则作出解释说明等用户权利实现机制。

2、我们希望集成并使用百度云手机SDK服务的开发者应用以合法合规的方式收集、使用最终用户的个人信息，但我们并不了解且无法控制任何开发者以及他们的移动应用如何使用他们所控制的最终用户个人信息，因此也不应为其行为负责。我们建议最终用户在认真阅读开发者应用相关隐私政策，在确认充分了解并同意他们如何收集、使用最终用户的个人信息后再使用开发者应用。

3、本隐私政策不适用于展示在、链接到或再封装我们的服务的那些适用第三方隐私政策、并由第三方提供的服务。虽然第三方展示在、链接到或再封装我们的服务，但我们并不了解或控制其行为，因此也不为其行为负责。请开发者及最终用户在已查看并接受其隐私政策之前，谨慎访问或使用其服务。

4. 最终用户具体获得的百度云手机SDK服务内容由开发者根据其移动应用需要进行选择，可能因为最终用户所使用的开发者应用不同而有所差异，百度云手机SDK可能获得的个人信息取决于最终用户所使用的开发者应用的具体类型/版本以及最终用户所使用的功能。如果在部分开发者应用版本中不涵盖某些服务内容或未提供特定功能，则本隐私政策中涉及到上述服务/功能及相关个人信息的内容将不适用。

请开发者及最终用户务必认真阅读本隐私政策，在确认充分了解并同意后再集成并使用百度云手机SDK服务。

本隐私政策将帮助开发者及最终用户了解以下内容：

1. 我们如何收集和使用最终用户的个人信息
2. 我们如何使用 Cookie 和同类技术
3. 我们如何共享、转让、公开披露最终用户的个人信息
4. 我们如何保存及保护最终用户的个人信息
5. 我们如何保障最终用户的个人信息相关权利行使
6. 我们如何处理未成年人的个人信息
7. 隐私政策的修订
8. 如何联系我们

我们珍视最终用户在向我们提供最终用户个人信息时对我们的信任，我们将按照本隐私政策处理最终用户的个人信息并保障最终用户信息的安全。

一、我们如何收集和使用最终用户的个人信息

（一）为帮助开发者向最终用户提供相应功能及服务

为了帮助开发者向最终用户提供相应功能及服务，当最终用户使用相应功能及服务时，我们会通过开发者应用向系统申请最终用户设备的相应权限。开发者应确保最终用户可以随时通过取消系统授权开发者应用获取相应设备权限或其他开发者应用提供的授权设置，停止我们对最终用户个人信息的收集，之后最终用户可能将无法使用基于相应个人信息而提供的相关服务或功能，或者无法达到基于相应个人信息提供的相关服务拟达到的效果，但不会影响最终用户正常使用百度云手机SDK的其他不基于相应个人信息即可实现的业务功能。

1.各项功能及服务涉及的个人信息

序号	功能及服务	个人信息类型	收集方式	适用系统版本
1	为帮助开发者向最终用户提供应用服务功能，以便准确分析存在问题	设备信息、位置信息、传感器信息	SDK直接采集	iOS及Android
2	为帮助开发者向最终用户提供登录功能	剪切板中的口令、链接等内容	SDK直接采集	iOS及Android

位置为敏感个人信息，最终用户可以随时通过取消系统授权开发者应用获取相应设备权限或其他开发者应用提供的授权设置停止我们对上述敏感个人信息的收集，之后最终用户可能将无法使用基于上述敏感个人信息而提供的相关服务或功能，或者无法达到基于上述敏感个人信息提供的相关服务拟达到的效果，但不会影响最终用户正常使用百度云手机SDK的其他不基于上述敏感个人信息即可实现的业务功能。

2. 设备权限调用

为了保证最终用户能正常使用百度云手机SDK相应功能及服务，我们会通过开发者应用向系统申请最终用户设备的以下系统设备权限，申请前我们会征询最终用户的同意，最终用户可以选择“允许”或“禁止”权限申请。经过最终用户的授权后我们会开启相关权限，最终用户可以随时在系统中取消授权，最终用户取消授权会导致最终用户无法使用相关的业务功能，但不会导致最终用户无法使用其他业务功能。各项功能及功能对设备权限的调用情况如下：

Android & IOS系统版本

设备权限	功能及服务	权限授权方式
读取/写入外部存储卡	升级程序下载、图片保存、错误日志保存	**授权方式由设备系统开发方以及开发者应用决定；当最终用户同意向开发者应用授予该权限时开启**
读取/写入外部存储卡	升级程序下载、图片保存、错误日志保存	
网络	升云端功能应用及服务	
获取位置	错误日志分析	
访问电话状态（部分机型也称为"设备信息"权限）	错误日志分析	
摄像头	使用百度云手机进行拍摄	
录音	使用百度云手机进行录音	
安装应用	检查到有更新包时下载安装	
震动	操作百度云手机时提示用户	
读取日志	百度云手机收集异常信息上报	
检索正在运行的应用	百度云手机客服判断当前页是否是在最上层	

在不同设备中，权限显示方式及关闭方式可能有所不同，具体请最终用户参考设备及系统开发方说明或指引。

（二）保证服务安全、优化和改善服务目的

为了帮助开发者向最终用户提供上述功能及服务，同时为了更准确定位并解决开发者以及最终用户在使用百度云手机SDK产品和服务时遇到的问题，改进及优化百度云手机SDK产品和服务在开发者侧以及最终用户侧的双重体验，更准确定位并解决最终用户在使用百度云手机SDK服务时遇到的问题，改进及优化百度云手机SDK的服务体验，提高百度云手机SDK服务的安全性，预防、发现、调查欺诈、危害安全、非法或违反与我们的协议、政策或规则的行为，以保护开发者、最终用户、我们或我们的关联公司、合作伙伴及社会公众的合法权益，我们会收集最终用户的**设备信息、位置信息、日志信息及其他与登录环境相关的信息。**

（三）个人信息的匿名化处理

在不公开披露、对外提供最终用户个人信息的前提下，百度公司有权对匿名化处理后的用户数据库进行挖掘、分析和利用（包括商业性使用），有权对产品/服务使用情况进行统计并与公众/第三方共享匿名化处理后的统计信息。

（四）事先征得授权同意的例外

请注意，在以下情形中，收集、使用个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；
2. 为订立、履行个人作为一方当事人的合同所必需；
3. 为履行法定职责或者法定义务所必需；
4. 为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；
5. 与犯罪侦查、起诉、审判和判决执行等直接有关的；
6. 出于维护最终用户或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；
7. 依照法律法规的规定在合理的范围内收集最终用户自行向社会公众公开或其他已经合法公开的个人信息；
8. 依照法律法规的规定在合理的范围内从合法公开披露的信息中收集最终用户的个人信息，如合法的新闻报道、政府信息公开等渠道；
9. 为公共利益实施新闻报道、舆论监督等行为，在合理的范围内处理个人信息；
10. 学术研究机构基于公共利益开展统计或学术研究所必要，且对外提供学术研究或描述的结果时，对结果中所包含的个人信息进行去标识化处理的；
11. 法律法规规定的其他情形。

提示最终用户注意，当我们要将信息用于本隐私政策未载明的其它用途时，会事先征求最终用户的同意。

二、我们如何使用 Cookie 和同类技术

Cookie是支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制。当最终用户使用百度云手机SDK产品或服务时，我们会向最终用户的设备发送一个或多个Cookie或匿名标识符。当最终用户与百度云手机SDK服务进行交互时，我们允许Cookie或者匿名标识符发送给百度公司服务器。Cookie 通常包含标识符、站点名称以及一些号码和字符。运用Cookie技术，百度公司能够了解最终用户的使用习惯，记住最终用户的偏好，省去最终用户输入重复信息的步骤，为最终用户提供更加周到的个性化服务，或帮最终用户判断最终用户账户的安全性。Cookie还可以帮助我们统计流量信息，分析页面设计和广告的有效性。

我们不会将 Cookie 用于本政策所述目的之外的任何用途。最终用户可根据自己的偏好管理或删除 Cookie。有关详情，请参见 AboutCookies.org。最终用户可以清除计算机上保存的所有 Cookie，大部分网络浏览器都设有阻止 Cookie 的功能。但如果最终用户这么做，则需要在每一次访问我们的网站时亲自更改用户设置，但最终用户可能因为该等修改，无法登录或使用依赖于Cookie的百度公司提供的服务或功能。

三、我们如何共享、转让、公开披露最终用户的个人信息

（一）共享

除非经过您本人事先单独同意或符合其他法律法规规定的情形，我们不会向除百度公司以外的第三方共享您的个人信息，但经过处理无法识别特定个人且不能复原的除外。

对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照依法采取保密和安全措施来处理个人信息。

1. 在下列情况下，经过最终用户的授权同意，我们可能会共享的个人信息： 仅为实现本隐私政策中声明的目的，我们的某些服务将由授权合作伙伴提供。我们可能会与合作伙伴共享最终用户的某些个人信息，以提供更好的客户服务和用户体验。我们仅会出于合法、正当、必要、特定、明确的目的共享最终用户的个人信息，并且只会共享与提供服务相关的个人信息。我们的合作伙伴无权将共享的个人信息用于任何其他用途。

目前，我们的授权合作伙伴包括以下类型：

(1) 服务平台或服务提供商。百度各产品接入了丰富的第三方服务。当最终用户选择使用该第三方服务时，最终用户授权我们将该信息提供给第三方服务平台或服务提供商，以便其基于相关信息为最终用户提供服务。

(2) 软硬件/系统服务提供商。当第三方软硬件/系统产品或服务与百度的产品或服务结合为最终用户提供服务时，经最终用户授权，我们会向第三方软硬件/系统服务提供商提供最终用户必要的个人信息，以便最终用户使用服务，或用于我们分析产品和服务使用情况，来提升最终用户的使用体验。

(3) 广告、咨询类服务商/广告主。未经最终用户授权，我们不会将最终用户的个人信息与提供广告、咨询类服务商共享。但我们可能会将经处理无法识别最终用户的身份且接收方无法复原的信息，例如经匿名化处理的用户画像，与广告或咨询类服务商或广告主共享，以帮助其在不识别最终用户个人的前提下，提升广告有效触达率，以及分析我们的产品和服务使用情况等。

2. 对我们与之共享个人信息的公司、组织和个人，我们会对其数据安全环境进行调查，与其签署严格的保密协定，要求他们按照我们的说明、本隐私政策以及其他任何相关的保密和安全措施来处理个人信息。

（二）转让

我们不会将最终用户的个人信息转让给除关联公司外的任何公司、组织和个人，但以下情况除外：

1. 事先获得最终用户的明确授权或同意；

2. 满足法律法规、法律程序的要求或强制性的政府要求或司法裁定；

3. 如果我们或我们的**关联公司**涉及合并、分立、清算、资产或业务的收购或出售等交易，最终用户的个人信息有可能作为此类交易的一部分而被转移，我们将确保该等信息在转移时的机密性，并尽最大可能确保新的持有最终用户个人信息的公司、组织继续受此隐私政策的约束，否则我们将要求该公司、组织重新向最终用户征求授权同意。

（三）公开披露

我们仅会在以下情况下，公开披露最终用户的个人信息：

1. 获得最终用户的单独同意；

2. 基于法律法规、法律程序、诉讼或政府主管部门强制性要求下。

（四）共享、转让、公开披露个人信息时事先征得授权同意的例外

在以下情形中，共享、转让、公开披露个人信息无需事先征得最终用户的授权同意：

1. 与国家安全、国防安全直接相关的；

2. 为订立、履行个人作为一方当事人的合同所必需；

3. 为履行法定职责或者法定义务所必需；

4. 与公共安全、公共卫生、重大公共利益直接相关的。例如：为应对突发公共卫生事件，或者紧急情况下为保护自然人的生命健康和财产安全所必需；

5. 与犯罪侦查、起诉、审判和判决执行等直接相关的；

6. 出于维护个人信息主体或其他个人的生命、财产等重大合法权益但又很难得到本人同意的；

7. 依照法律法规的规定在合理的范围内处理个人自行公开或者其他已经合法公开的个人信息，例如：合法的新闻报道、政府信息公开等渠道等；

8. 法律、行政法规另有规定的情形。

四、我们如何保存及保护最终用户的个人信息

（一）保存期限

我们将在百度云手机SDK自身提供服务以及开发者应用提供服务所需的期限内保存您的个人信息，但法律法规对保存期限另有规定、您同意留存更长的期限、保证服务的安全与质量、实现争议解决目的、技术上难以实现等情况下，在前述保存期限到期后，我们将依法、依约或在合理范围内延长保存期限。

在超出保存期限后，我们将根据法律规定删除您的个人信息或进行匿名化处理。

（二）保存地域

原则上，我们在中华人民共和国境内收集和产生的个人信息，将存储在中华人民共和国境内。如您的个人信息可能会被转移到您使用产品或服务所在国家/地区的境外管辖区，或者受到来自这些管辖区的访问，我们会严格履行法律法规规定的义务并按照法律规定事先获得您的单独同意。此类管辖区可能设有不同的数据保护法，甚至未设立相关法律。在此类情况下，我们会按照中国现行法律的规定传输您的个人信息，并会确保您的个人信息得到在中华人民共和国境内足够同等的保护。例如，我们会请求您对跨境转移个人信息的同意，或者在跨境数据转移之前实施数据去标识化等安全举措。

（三）安全措施

1. 我们会以“最小化”原则收集、使用、存储和传输用户信息，并通过用户协议和隐私政策告知您相关信息的使用目的和范围。

2. 我们非常重视信息安全。我们成立了专责团队负责研发和应用多种安全技术和程序等，我们会对安全管理负责人和关键安全岗位的人员进行安全背景审查，我们建立了完

善的信息安全管理制度和内部安全事件处置机制等。我们会采取适当的符合业界标准的安全措施和技术手段存储和保护您的个人信息，以防止您的信息丢失、遭到被未经授权的访问、公开披露、使用、毁损、丢失或泄漏。我们会采取一切合理可行的措施，保护您的个人信息。我们会使用加密技术确保数据的保密性；我们会使用受信赖的保护机制防止数据遭到恶意攻击。

3. 我们会对员工进行数据安全的意识培养和安全能力的培训和考核，加强员工对于保护个人信息重要性的认识。我们会对处理个人信息的员工进行身份认证及权限控制，并会与接触您个人信息的员工、合作伙伴签署保密协议，明确岗位职责及行为准则，确保只有授权人员才可访问个人信息。**若有违反保密协议的行为，会被立即终止与百度公司的合作关系**，并会被追究相关法律责任，对接触个人信息人员在离岗时也提出了保密要求。

4. 我们提醒您注意，互联网并非绝对安全的环境，**当您通过百度云手机SDK中嵌入的第三方社交软件、电子邮件、短信等与其他用户交互您的地理位置或行踪轨迹信息时，不确定第三方软件对信息的传递是否完全加密，注意确保您个人信息的安全。**

5. 我们也请您理解，在互联网行业由于技术的限制和飞速发展以及可能存在的各种恶意攻击手段，即便我们竭尽所能加强安全措施，也不可能始终保证信息的百分之百安全。**请您了解，您使用我们的产品和/或服务时所用的系统和通讯网络，有可能在我们控制之外的其他环节而出现安全问题。**

6. **根据我们的安全管理制度，个人信息泄露、毁损或丢失事件被列为最特大安全事件，一经发生将启动公司最高级别的紧急预案**，由安全部、公共事务部、法务部等多个部门组成联合应急响应小组处理。

（四）安全事件通知

1. 我们会制定网络安全事件应急预案，及时处置系统漏洞、计算机病毒、网络攻击、网络侵入等安全风险，在发生危害网络安全的事件时，我们会立即启动应急预案，采取相应的补救措施，并按照规定向有关主管部门报告。

2. 个人信息泄露、毁损、丢失属于公司级特大安全事件，我们会负责定期组织工作组成员进行安全预案演练，防止此类安全事件发生。若一旦不幸发生，我们将按照最高优先级启动应急预案，组成紧急应急小组，在最短时间内追溯原因并减少损失。

3. 在不幸发生个人信息安全事件后，我们将按照法律法规的要求，及时向您告知安全事件的基本情况和可能的影响、我们已采取或将要采取的处理措施、您可自主防范和降低的风险的建议、对您的补救措施等。我们将及时将事件相关情况以站内通知、短信通知、电话、邮件等您预留的联系方式告知您，难以逐一告知时我们会采取合理、有效的方式发布公告。同时，我们还将按照监管部门要求，主动上报个人信息安全事件的处置情况。请您理解，根据法律法规的规定，如果我们采取的措施能够有效避免信息泄露、篡改、丢失造成危害的，除非监管部门要求向您通知，我们可以选择不向您通知该个人信息安全事件。

五、我们如何处理未成年人的个人信息

百度公司非常重视对未成年人信息的保护。

百度云手机SDK的产品和服务主要面向成年人。如果最终用户是未满14周岁的未成年人，请务必在使用已集成百度云手机SDK的开发者应用前，在父母或其他监护人监护、指导下共同仔细阅读开发者应用隐私政策、本隐私政策以及《百度儿童个人信息保护声明》，并在征得监护人同意的前提下使用开发者应用或提供个人信息。

如果我们发现在未事先获得可证实的父母同意的情况下收集了未成年人的个人信息，将会采取措施尽快删除相关信息。

如果任何时候监护人有理由相信我们在未获监护人同意的情况下收集了未成年人的个人信息，请通过官方渠道联系我们，我们会采取措施尽快删除相关数据。

六、我们如何保障最终用户的个人信息相关权利行使

按照中国相关的法律、法规、标准，以及其他国家、地区的通行做法，我们将尽力协调、支持并保障最终用户对自己的个人信息行使以下权利：

1. 查阅权、更正及补充权、复制权、帐号注销权

鉴于最终用户通过开发者应用使用百度云手机SDK服务，并且使用服务时并不会注册/登录百度帐号，为保障最终用户查阅、更正、补充、复制其个人信息以及注销其开发者应用帐号的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要查阅、更正、补充、复制其个人信息或注销其开发者应用帐号，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可通过八、如何联系我们中的方式与我们取得联系，我们将尽力协调、支持并保障最终用户的上述权利实现。

2. 删除权

为保障最终用户删除其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要删除其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，在以下情形中，最终用户可以通过八、如何联系我们中的方式与我们取得联系，向我们提出删除个人信息的请求：

1. 如果我们违反法律法规或与最终用户的约定处理其个人信息；

2. 如果我们的处理目的已实现、无法实现或者为实现处理目的不再必要；

3. 如果我们停止提供产品或者服务，或者保存期限已届满；

4. 如果最终用户撤回同意；

5. 法律、行政法规规定的其他情形。 当最终用户从我们的服务中删除信息后，我们可能不会立即在备份系统中删除相应的信息，但会在备份更新时删除这些信息。请最终用户知晓并理解，如果法律、行政法规规定的或本隐私政策说明的保存期限未届满，或者删除个人信息从技术上难以实现的，我们会停止除存储和采取必要的安全保护措施之外的处理。

3. 撤回同意

每个业务功能需要一些基本的个人信息才能得以完成。对于额外收集的个人信息的收集和使用，最终用户可以随时给予或收回其授权同意。

最终用户可以在设备系统中直接关闭本隐私政策说明的我们可能调用的设备系统权限，或开发者应用提供的其他授权设置（如适用），改变同意范围或撤回其授权。

当最终用户撤回同意后，我们无法继续为最终用户提供撤回同意所对应的服务，也将不再使用最终用户相应的个人信息。但最终用户撤回同意的决定，不会影响此前基于其同意而开展的个人信息处理。

4. 可携带权

为保障最终用户转移其个人信息的权利实现，我们在本隐私政策以及其他与开发者的约定中，要求开发者承诺提供便于操作的实现方式。如最终用户要转移其个人信息，应通过开发者提供的上述权利实现方式，如开发者未按照承诺进行提供，最终用户可以通过八、如何联系我们中的方式与我们取得联系，我们将尽力协调、支持并保障其上述权利实现。

在法律法规规定的条件下，同时符合国家网信部门规定指令和条件的，如果技术可行，最终用户也可以要求我们将其个人信息转移至最终用户指定的其他主体。

5. 提前获知产品和服务停止运营权。

百度云手机SDK愿一直陪伴开发者和最终用户，若因特殊原因导致百度云手机SDK产品停止运营，我们将在合理期间内在产品或服务的主页面或站内信或向开发者和最终用户发送电子邮件或其他合适的能触达其方式通知开发者和最终用户，并将停止对最终用户个人信息的收集，同时会按照法律规定对已收集的最终用户的个人信息进行删除或匿名化处理。

6. 获得解释的权利

最终用户有权要求我们就个人信息处理规则作出解释说明。最终用户可以通过八、如何联系我们中的方式与我们取得联系。

七、隐私政策的修订与通知

我们的隐私政策可能变更。

未经最终用户明确同意，我们不会削减最终用户按照本隐私政策所应享有的权利。我们会在本页面上发布对本隐私政策所做的任何变更。

对于重大变更，**我们会在百度云手机SDK官方网站的主要曝光页面向开发者以及最终用户公示，并以任何可触达的方式通知开发者。若开发者不同意该等变更可以停止集成并使用百度云手机SDK产品和服务，若开发者继续集成并使用百度云手机SDK产品和/或服务，即表示开发者同意受修订后的本隐私政策的约束，并将此变更通知最终用户，获取最终用户对此变更的完整、合法、在使用百度云手机SDK服务期间持续有效的授权同意**。若最终用户不同意该等变更，可以停止使用百度云手机SDK产品和服务，开发者应向用户提供相应实现机制；若最终用户继续使用百度云手机SDK产品和/或服务，即表示最终用户同意受修订后的本隐私政策的约束。

本政策所指的重大变更包括但不限于：

1. 我们的服务模式发生重大变化。如处理个人信息的目的、处理的个人信息类型、个人信息的使用方式等；

2. 我们在所有权结构、组织架构等方面发生重大变化。如业务调整、破产并购等引起的所有者变更等；

3. 个人信息共享、转让或公开披露的主要对象发生变化；

4. 最终用户参与个人信息处理方面的权利及其行使方式发生重大变化；

5. 我们负责处理个人信息安全的责任部门、联络方式及投诉渠道发生变化时；

6. 个人信息安全影响评估报告表明存在高风险时。

本政策更新后，我们会将本政策的旧版本存档，供最终用户查阅。

如有本隐私政策未尽事宜，以《百度隐私权保护声明》为准。

八、如何联系我们

百度云手机SDK的成长离不开各方开发者及最终用户的共同努力，我们非常感谢开发者及最终用户对百度云手机SDK数据更新、使用反馈方面的贡献。

开发者及最终用户可以通过官方渠道反馈开发者及最终用户对百度云手机SDK产品和服务的建议以及在使用过程中遇到的问题，以帮助我们优化产品功能及服务，使更多用户更加便捷的使用我们的产品和服务。

开发者及最终用户可以通过个人信息保护问题反馈平台（http://help.baidu.com/personalinformation） 同我们联系。

开发者及最终用户也可以通过如下联络方式同我们联系：

中国北京市海淀区上地十街10号

北京百度网讯科技有限公司 法务部

邮政编码：100085

为保障我们高效处理开发者/最终用户的问题并及时向开发者/最终用户反馈，需要开发者/最终用户提交身份证明、有效联系方式和书面请求及相关证据，我们会在验证开发者/最终用户的身份后处理开发者/最终用户的请求。

如果开发者/最终用户对我们的回复不满意，特别是最终用户认为我们的个人信息处理行为损害了最终用户的合法权益，开发者/最终用户还可以通过以下外部途径寻求解决方案：**向北京市海淀区人民法院提起诉讼。**

附录1：名词解释

个人信息是指以电子或者其他方式记录的与已识别或者可识别的自然人有关的各种信息，不包括匿名化处理后的信息。个人信息包括姓名、出生日期、身份证件号码、个人生物识别信息、住址、通信通讯联系方式、通信记录和内容、帐号密码、财产信息、征信信息、行踪轨迹、住宿信息、健康生理信息、交易信息等。

敏感个人信息是指一旦泄露或者非法使用，容易导致自然人的人格尊严受到侵害或者人身、财产安全受到危害的个人信息，包括生物识别、宗教信仰、特定身份、医疗健康、金融账户、行踪轨迹等信息，以及不满十四周岁未成年人的个人信息。

设备是指可用于使用百度产品和/或服务的装置，例如桌面设备、平板电脑或智能手机。

设备信息可能包括最终用户用于安装、运行百度云手机SDK的终端设备的设备属性信息（例如最终用户的硬件型号，操作系统版本及系统情况，设备配置，国际移动设备身份码IMEI、国际移动用户识别码IMSI、网络设备硬件地址MAC、广告标识符IDFA、供应商标识符IDFV、移动设备识别码MEID、匿名设备标识符OAID 、集成电路卡识别码ICCID 、Android ID、硬件序列号等设备标识符）、设备连接信息（例如浏览器的类型、电信运营商、使用的语言、WIFI信息）以及设备状态信息（例如设备应用安装列表）。

日志信息是指我们的服务器所自动记录最终用户在访问百度云手机SDK时所发出的请求，例如最终用户的IP 地址、浏览器的类型和使用的语言、硬件设备信息、操作系统的版本、网络运营商的信息、最终用户访问服务的日期、时间、时长等最终用户在使用我们的产品或服务时提供、形成或留存的信息。

Cookie是指支持服务器端（或者脚本）在客户端上存储和检索信息的一种机制，通过增加简单、持续的客户端状态来扩展基于Web的客户端/服务器应用。服务器在向客户端返回HTTP对象的同时发送一条状态信息，并由客户端保存。状态信息中说明了该状态下有效的URL范围。此后，客户端发起的该范围内的HTTP请求都将把该状态信息的当前值从客户端返回给服务器，这个状态信息被称为Cookie。

位置信息是指通过GPS信息，WLAN接入点、蓝牙、基站以及其他传感器信息所获取的精确位置信息，也包括通过IP地址或其他网络信息等获取的粗略地理位置信息。

用户画像是指通过收集、汇聚、分析个人信息，对某特定自然人个人特征，如其职业、经济、健康、教育、个人喜好、信用、行为等方面做出分析或预测，形成其个人特征模型的过程。直接使用特定自然人的个人信息，形成该自然人的特征模型，称为直接用户画像。使用来源于特定自然人以外的个人信息，如其所在群体的数据，形成该自然人的特征模型，称为间接用户画像。

去标识化是指个人信息经过处理，使其在不借助额外信息的情况下无法识别特定自然人的过程。 匿名化是指通过对个人信息的技术处理，使得个人信息主体无法被识别，且处理后的信息不能被复原的过程。个人信息经匿名化处理后所得的信息不属于个人信息。

百度平台是指百度公司旗下各专门频道或平台服务（包括百度搜索、百度APP及衍生版、百度百科、百度知道、百度贴吧、百度智能云、百度手机助手及其他百度系产品https://www.baidu.com/more/） 等网站、程序、服务、工具及客户端。

关联公司是指百度云手机SDK的经营者北京百度网讯科技有限公司及其他与百度公司存在关联关系的公司的单称或合称。“关联关系”是指对于任何主体（包括个人、公司、合伙企业、组织或其他任何实体）而言，即其直接或间接控制的主体，或直接或间接控制其的主体，或直接或间接与其受同一主体控制的主体。前述“控制”指，通过持有表决权、合约或其他方式，直接或间接地拥有对相关主体的管理和决策作出指示或责成他人作出指示的权力或事实上构成实际控制的其他关系。

再次感谢开发者以及最终用户对百度云手机SDK的信任和使用！

北京百度网讯科技有限公司

2022年4月2日更新

2022年4月2日生效

常见问题

使用类

云手机的收费标准是什么？

详情请查看产品定价。 服务器整机售卖价格，欢迎提交工单联系客服，咨询专属福利。

可以在移动端使用云手机吗？

可以，客户需要自行将云手机的Android SDK或Web SDK集成到第三方APP上，即可在第三方APP上使用云手机。

支持连接的云手机实例规格有哪些？

目前有三种规格：
高配旗舰型：旗舰级ARM处理芯片；
中配普通型：主流级ARM处理芯片；
入门基础型：入门级ARM处理芯片。

云手机支持IOS系统吗？

支持，通过WebRtc投屏可以支持

退订管理问题

非产品不可用问题，不支持退订；

如遇到产品不可用问题（如云手机连接闪退、云手机系统不兼容应用等），购买后7天内可退订，费用以系统实际计费天数为准来收取，超过7天不支持退订。

请您在仔细了解并试用云手机后再进行购买。

客户常见问题

为什么不能安装、卸载、更新APP？

云手机处在连接状态上，默认有用户使用设备，不能安装、卸载、更新app，需要断开释放设备才可以进行以上操作

是否可以安装应用？在哪里安装应用？

可以，在控制台安装应用

为什么APP会安装失败？

APP最低兼容版本必须是安卓6.0,如果miniSdkVersion>23就不可以安装

云手机支持远程adb调用吗？

暂不支持adb，如有疑问请联系工单客服。

云手机可否支持绑定SIM卡？云手机能否收到登录短信？

暂不支持

一台电脑同时登录多台云手机，这些云手机的IP是否一样？

这些云手机虚拟IP不一样，每一台云手机都有自己独立的虚拟IP。

是否可以批量安装APP？

可以，控制台支持批量安装APP。

需要海外机房怎么办？

智能云云手机官网暂不支持，如有需要请联系工单客服。

云手机是否支持IP代理？

不支持，根据国家相关政策及有关部门规定，云手机不提供动态IP代理服务。