云游戏

实践教程

腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

实践教程
云 APK 直播与录制

最近更新时间：2024-09-25 10:58:01

概念说明

云 APK 直播与录制服务基于 云游戏-无客户端模式 。针对“将 App 内容作为直播/录播内容源”的客户场景，可将用户终端的“直播/录播”功能模块迁移至云端 Android 容器，降低了客户端 App 的改造成本与维护成本；也可以定制云 APK 实现不适合在用户终端 App 上实现的复杂功能，扩展整体 App 的能力降低用户终端操作复杂度。

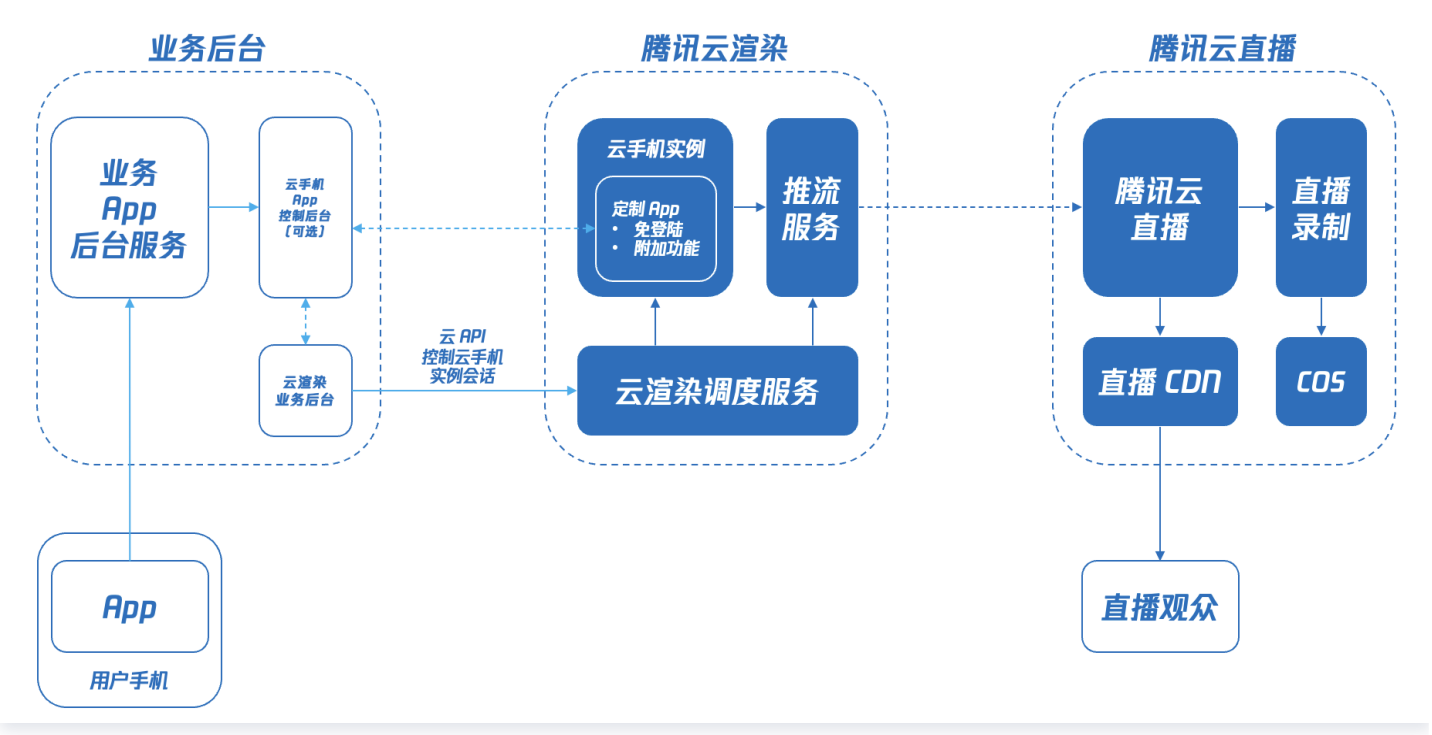
功能优势

- 使用场景广：适用于直播、录制等音视频相关业务场景。
- 接入成本低：无需接入客户端 SDK，只需业务后台调用云 API。
- 改造成本低：云端运行的 APK 改造成本低，原始 APK 可直接部署。
- 功能扩展强：可以定制云端运行的 APK 实现终端 App 不适合实现的复杂功能。

架构流程

实现步骤如下：

1. 在 控制台部署 Android APK，并购买绑定 Android 容器并发 。
2. 参考 云游戏-无客户端模式 控制调度 Android App 运行。
3. 调用 开始云端推流到云直播 API 将 Android 容器中 App 运行的完整画面推流至 腾讯云直播 。
4. 在腾讯云直播可以将此直播流用于播放分发，也可以在腾讯云 直播控制台开启录制 将 Android 容器中 App 运行的完整画面保存成录制文件。



客户场景与实践

适用于直播、录制等音视频相关业务场景。
客户使用场景可参考下列内容。

场景1：游戏直播

- **需求背景**：某国民级别游戏，其线下赛需使用游戏直播能力。
- **原实现方法**：选手手机安装直播插件，本地终端录屏推流直播。
- **场景痛点**：
  - 手机需安装直播插件，对于普通玩家而言，操作成本高。
  - 开播后手机发热发烫、本地推流占用带宽，均影响了选手游戏体验。
  - 实现导播能力，需配合使用 PC 端 OBS 能力或线下导播台。
- **实现方法**：
  - 采用腾讯云“云 APK 直播与录制”能力。
  - 云端运行客户定制化的游戏包，可实时同步选手的游戏画面。
  - 定制化游戏包内置“广告位叠加”、“视角切换”等导播逻辑，可在云端实现导播能力。
  - 将云端 App 画面推流并通过腾讯云直播分发。

场景2：直播录制

- **需求背景**：某国民级别直播 App，需沉淀“主播对战 PK”互动玩法下的视频内容，作为二次分发的 UGC 素材。
- **原实现方法**：运营人员使用手机，在主播开播期间进行手动录屏。
- **场景痛点**：
  - 常规的视频录制方法，所获取的视频流均为单一主播视角的原始画面。
  - 即使通过混流录制，也缺少弹幕的动画效果。
  - 只能通过运营人员使用手机录屏，无法实现规模化、自动化的素材积累。
- **实现方法**：
  - 采用腾讯云“云 APK 直播与录制”能力
  - 云端运行客户定制化的免登录 App
  - 主播开播时，客户业务后台自动发送消息给云端 App，云端 App 收到信令后进入指定房间并播放视频。
  - 配合腾讯云直播录制能力，沉淀视频内容素材。

场景3：app实时操作-分享

- **需求背景**：某教育类 app，需将其 App 内直播间的视频画面，叠加至其“直播间”的实时视频流上。
- **原实现方法**：需要将“互动房间”的业务逻辑植入到原有“直播间”业务逻辑中，导致多个业务逻辑相互耦合。
- **场景痛点**：
  - 需向用户索取视频录制权限，成功率低。
  - 业务改造量大且后续模块升级会带来额外的维护成本
- **实现方法**：
  - 采用腾讯云“云 APK 直播与录制”能力。
  - 云端运行客户定制化的免登录 App。
  - 主播开播时，客户业务后台自动发送消息给云端 App，云端 App 收到信令后进入指定“互动房间”并播放视频。
  - 将云端 App 画面推流，直播间则可以直接显示对应画面。

多人互动

最近更新时间：2023-04-10 16:00:44

客户场景与实践

适用于多人互动、直播等云游戏相关业务场景，客户使用场景可参考下列内容。

场景1：多人互动直播

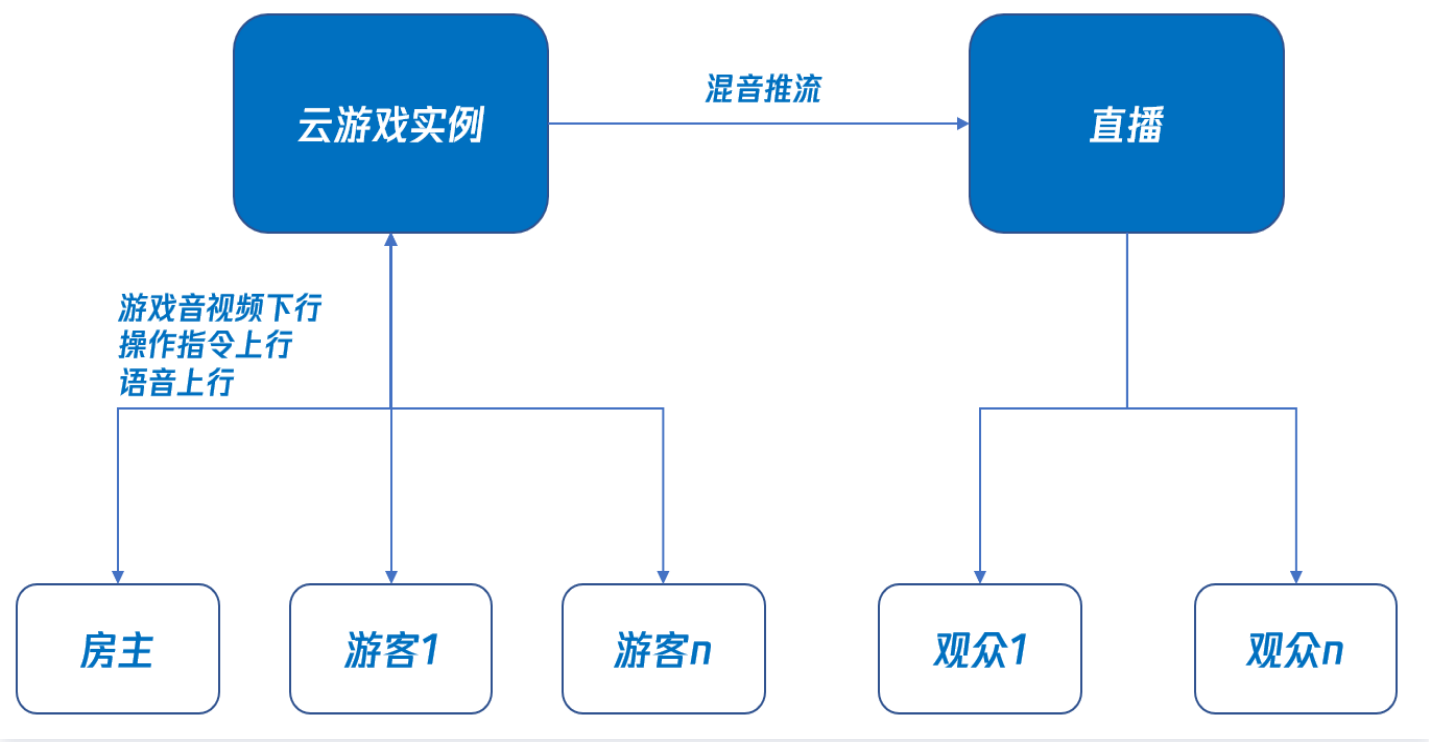
- **业务背景**：某直播厂商，需实现主播直播时与粉丝游戏互动。
- **实现逻辑**：
  - 主播开播时，通过连接云游戏创建房间，粉丝可以向主播发起加入房间申请或者通过主播提供的邀请连接，进入房间与主播进行多人游戏。
  - 其他粉丝可以通过直播观看主播与粉丝的游戏互动直播画面。

场景2：多人擂台游戏

- **业务背景**：某云游戏厂商，需实现多人擂台云游戏。
- **实现逻辑**：
  - 管理员创建云游戏房间。
  - 其他玩家可以排队依次进入进行擂台 PK 游戏，胜利者继续游戏，失败者切换角色或退出。

概念说明

多人互动云游戏可以抽象理解为一个云游戏房间+直播流（可选）的形式。房间由云游戏玩家（房主）创建后，其他玩家（游客）可通过房主的 UserId 加入同一房间，同一房间内的所有玩家通过云游戏连接看到同一个云端画面。



业务用户说明

- **房主：创建房间的玩家**
  - 管理他人是否拥有控制权（点击、键鼠、手柄等操作，针对游戏配置）。
  - 管理房间内所有人的麦克风状态。
- **游客：加入其他人房间的玩家**
  - 需要向房主申请控制权。
  - 开关自己的麦克风。
- **观众：观看直播流，未连接云游戏实例**

  无其他权限，仅能观看画面。

云游戏角色说明

- **Player：拥有游戏控制权**

  人数最多为 7 人。
- **Viewer：仅支持观看**

  人数没有限制。

> **说明：**
> - 房主可以切换角色为 Player 进行游戏控制，也可以切换为 Viewer 仅观看。

- 游客可以是 Player，也可以是 Viewer，房主可以控制游客的角色。

流程说明

房主创建房间

1. 房主客户端向业务后台发起启动云游戏请求，业务后台通过调用 TrylockWorker() 申请锁定云游戏实例。
2. 业务后台通过调用 CreateSession() 创建会话，其中请求参数 HostUserId 需要与 UserId 相同，Role 可选择 Player 或者 Viewer。

> ⚠ **注意：**
> 房主创建房间与单人云游加入游戏流程相同，唯一的区别是调用云 API 创建会话时需填入 HostUserId。

游客加入房间

1. 游客客户端向业务后台发起启动云游戏请求。
2. 业务后台通过调用 CreateSession() 创建会话，其中请求参数 HostUserId 需要与房主的 UserId 相同，Role 可选择 Player 或者 Viewer。

> ⚠ **注意：**
> 仅房主需要调用 TrylockWorker()，游客不需要调用。

推流直播

1. 调用 开始云端推流 API 将云游实例运行的完整画面推流至 腾讯云直播 。
2. 腾讯云直播可以将此直播流用于播放分发，观众通过直播地址观看直播。

切换角色

1. 游客客户端调用 TCGSDK.submitSeatChange() 接口申请切换角色，可以申请切换为 Player 或者 Viewer。

> ⚠ **注意：**
> - Viewer 切换成 Player，需要带上切换的操控席位信息（该席位必须是空的）。
> - Viewer 切换成 Viewer，无需申请，可直接切换。
> - Player 切换成 Viewer，无需申请，可直接切换。
> - Player 切换成 Player，需要带上切换的操控席位信息，Player 之间席位切换会导致游戏内所操控角色切换。
>
> 综上所述，Player 需要严格对号入座，Viewer 不分配坐席。

2. 房主客户端通过回调 onMultiPlayerChange() 接口获取其他玩家申请切换角色的请求，通过调用 TCGSDK.seatChange() 接口切换游客角色。
3. 游客客户端通过回调 onMultiPlayerChange() 接口获取申请切换角色的结果。

切换麦克风状态

1. 房主客户端调用 TCGSDK.changeMicStatus() 接口可以切换其他玩家或者自己的麦克风状态。
2. 游客客户端调用 TCGSDK.changeMicStatus() 只能切换自己的麦克风状态。

排队功能

最近更新时间：2022-04-22 16:01:49

使用场景

当云游戏的用户量大于机器并发数量时，需要引入用户排队系统，来提升用户体验。

关键点

队列不能堵塞。

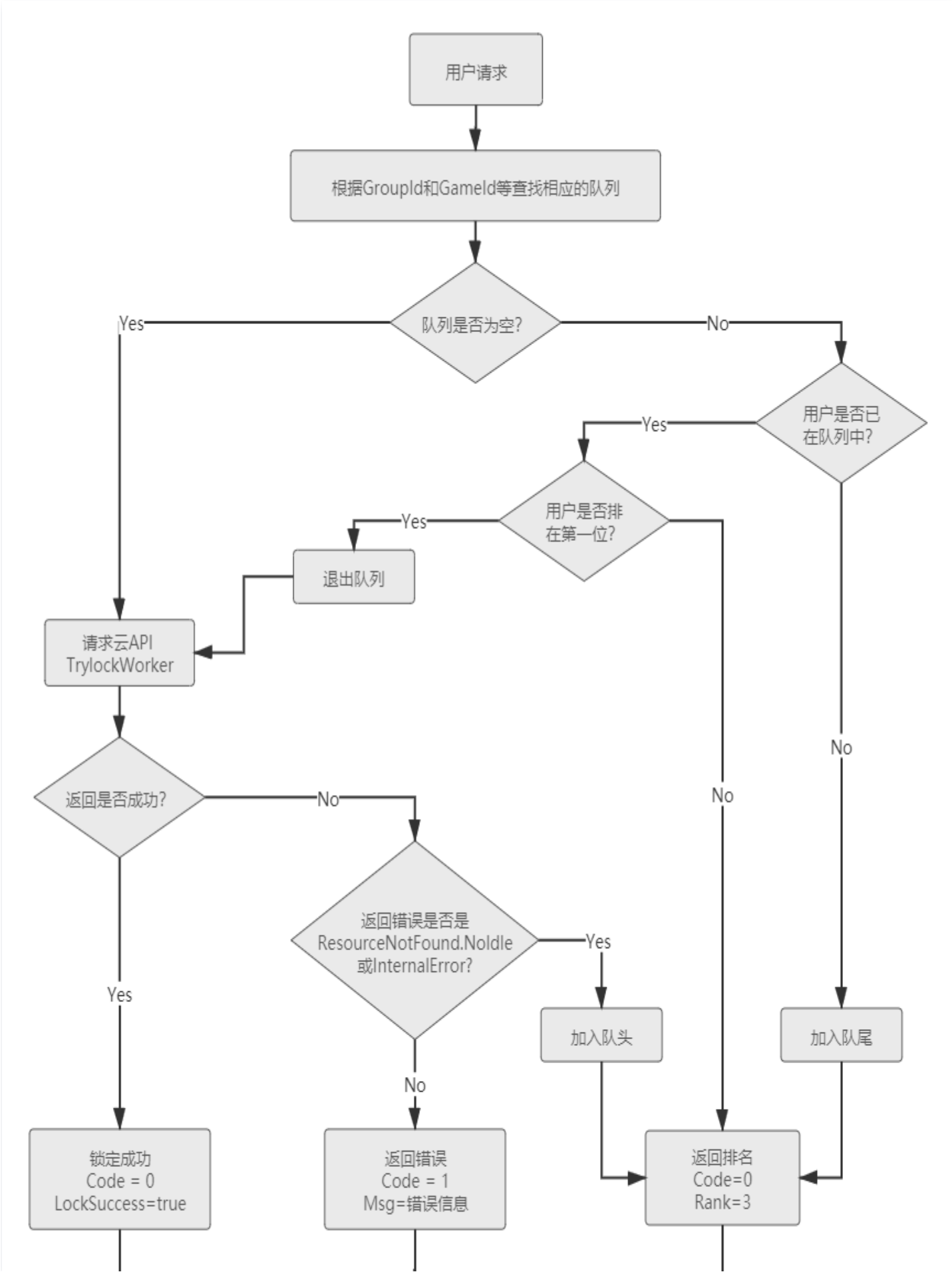
参考方案

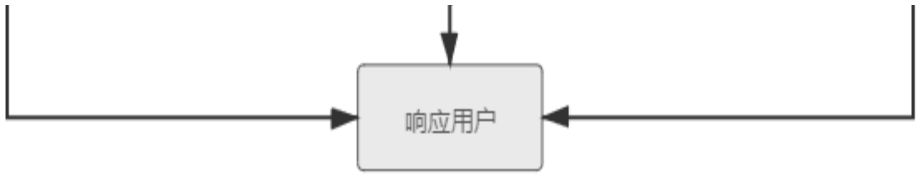
- 业务后台需要建立多个队列，相同 GroupId、GameId 的单独一个队列，如果有 VIP 区分的，也要单独一个队列。用户请求时，需要找到相应的队列进行处理。
- 队列需要支持插入队尾、插入队头和退出队列等操作。
- 用户进入排队后，需要每隔一段时间请求业务后台，来获取当前排名或锁定机器成功进入到下一步。
- 业务后台需要定时或每次用户请求时检测队头，如果队头的上次请求时间距离现在已超过一定时间，就认为超时，需要踢出队列，避免堵塞。
- 提供用户主动退出排队的接口。

为了便于下面流程描述，假设用户响应包括以下字段（以下仅为举例，业务方可自行定义这些字段）：

参数名称	类型	描述
Code	integer	• 0：表示成功 • -1：表示处理错误
Msg	string	错误信息
LockSuccess	boolean	• true：表示锁定成功 • false：表示锁定失败
Rank	integer	当前排名

流程图


用户请求
根据GroupId和GameId等查找相应的队列
队列是否为空?
Yes
No
用户是否已
在队列中?
Yes
用户是否排
在第一位?
Yes
退出队列
请求云API
TrylockWorker
返回是否成功?
No
Yes
返回错误是否是
ResourceNotFound.NoIdle
或InternalError?
Yes
No
No
No
加入队头
加入队尾
锁定成功
Code = 0
LockSuccess=true
返回错误
Code = 1
Msg=错误信息
返回排名
Code=0
Rank=3


响应用户

云游戏存档

最近更新时间：2024-08-22 16:29:01

> **注意：**
> 目前云游戏存档能力，仅支持端游方案。

基本使用

1. 在 对象存储控制台 新建一个 bucket，详情请参见 创建存储桶 。
2. 使用主账号把此 bucket 给腾讯云存档托管账号（大账号 ID：`100011897175` ，AppId：`1300543852` ）授权读写。详细操作说明请参见 设置访问权限 。
3. 通知腾讯云云游戏运营人员，为您的游戏内容配置存档。
4. 游戏配置好存档后，不再需要任何开发工作或人工干预，每次启动前都会下载存档，关闭游戏后也会自动上传存档，实现云存档功能。

获取存档记录

云游戏支持两种方式获取用户存档记录。

通过腾讯云云函数获取用户存档记录

云游戏存档上传到腾讯云对象存储后，都会触发一次云函数事件，客户可以实现自定义云函数来响应事件，以获取存档记录。
此方式需要另外购买腾讯云云函数产品，只能获取成功上传的通知。

- 上传和下载时存档在COS保存路径为：`userData/${GameId}/${UserId}` 。
- 每次上传完成后，会在保存一个的备份存档，路径为：`userData/${GameId}/${UserId}-${TimeStamp}-tx` 。

参数	说明
UserId	CreateSession 时传的用户 ID
GameId	云游戏的游戏 ID
TimeStamp	Unix 时间戳，单位秒

> **说明：**
> 更多具体详情，请参见 COS触发器 。

通过腾讯云 CKafka 获取用户存档记录

云游戏服务下载或上传存档时，都会通过 CKafka 发送通知，客户可以从 CKafka 中读取存档实例。
此方法需要另外购买 CKafka 实例，能够获取下载/上传成功或者失败的通知。

1. **购买 CKafka 实例**，详细操作请参见 创建实例 。
2. **分配公网域名：**
   在 CKafka 实例详情里，选择**接入方式**模块中的**添加路由策略**，新增一条公网路由。完成后就能得到一个公网访问域名。详细操作请参见 添加公网路由 。
3. **创建 Topic：**
   在实例详情页，单击页面顶部的 **Topic 管理**，单击**新建**，新建一个用来收发消息的 Topic。更多详情请参见 创建 Topic 。
4. **创建用户，并给用户授权读写 Topic：**
   在实例详情页，选择**用户管理**，单击**新建**，添加一个用户，设置好用户名和密码。并且在 **ACL 策略管理** 页面，为用户添加 Topic 的读写权限。详细操作请参见 配置 ACL 策略 。
   结束上述步骤后，可将 CKafka 的实例名、域名、Topic、用户名、密码给云游戏运营人员做好配置。
5. **接收消息：**
   - 存档下载通知格式：

```
{
  "Type":"DownloadArchive",
  "AppId":123456,
  "UserId":"user123",
  "GameId":"game-cloudgame",
  "RequestId":"95d11eb6-24df-47f4-a3dc-31c7007c83c1",
  "Data":{
   "ArchiveName":"userData/game-cloudgame/user123",
   "Result":"Success",  // 结果：Success、Fail
   "Size":36335,
   "Timestamp":1622100110
  }
}
```

   - 存档上传通知格式：

```
{
  "Type":"UploadArchive",
  "AppId":123456,
  "UserId":"user123",
  "GameId":"game-cloudgame",
  "RequestId":"95d11eb6-24df-47f4-a3dc-31c7007c83c1",
  "Data":{
   "ArchiveName":"userData/game-cloudgame/user123",
```

```
    "BackupName":"userData/game-cloudgame/user123-1622114014-tx
    "Result":"Success",  // 结果：Success、Fail、Skip
    "Size":36335,
    "Timestamp":1622114022
  }
}
```

字段	说明
RequestId	触发存档操作的云 API RequestId
ArchiveName	存档在 COS 保存的名字
BackupName	存档上传时在 COS 中生成的备份
Size	存档大小
GameId	云游戏的游戏 ID
Result	上传结果，返回 Success、Fail 或 Skip。 Skip 只有在上传事件出现，表示本地没有存档文件可上传。
TimeStamp	Unix 时间戳，单位秒

通过 SDK 获取存档下载/上传的进度和结果

腾讯云云游戏SDK提供了存档相关事件通知，可以用来获取存档下载/上传进度和结果，具体见 JS SDK 文档 和 Android SDK 文档 。

进阶用法

除默认的自动存档功能，为了让客户能更加灵活控制存档，腾讯云云游戏平台还提供了运行时切换存档，运行中主动保存存档等进阶功能。

运行时切换存档

通过 GameContext的ArchiveUrl 传递新存档下载链接即可。
示例：

```
"GameContext":"
{\"ArchiveUrl\":\"https://gamearchive.cos.myqcloud.com/archive.zip\"}"
```

详细说明请参见 切换游戏存档 。

运行时保存存档

详细说明请参见 保存游戏存档 。

常见问题

什么是可重连状态?

“可重连”状态表示在这台机器上，用户已断开连接，但游戏还在运行；如果该用户在一定时间内（默认120秒）请求相同的游戏，可以重新连接上这台机器，继续体验。

虚拟按键

最近更新时间：2022-04-22 16:02:38

当玩家在移动设备上玩端游时，需要通过键盘映射或手柄映射操作云端游戏。为提高玩家的云游戏操控体验，云游戏提供动态可配置的虚拟按键布局的能力，同时也减少您在接入过程中的开发成本和后期维护成本。

> **注意**
> 目前云游戏虚拟按键仅支持 Android 端。

支持说明

类型	说明
库文件	• `tcgui-gamepad.aar` 虚拟按键拓展库。
• 按键类型	• 键盘按键（78个单击按键）。 • 鼠标按键（左、中、右、上滚、下滚）。 • 十字摇杆键（WDSA、上右下左）。 • Xbox 手柄 A、B、X、Y、Select、Start、LB、RB、L3、R3 普通点击键。 • Xbox 手柄延时板机键：LT、RT。（点击时在短时间内连续发多次消息，按下时消息的力度随时间递增，松开后随时间递减。） • Xbox 手柄左、右摇杆键。 • Xbox手柄十字方向键。
编辑功能	• 动态添加/删除按键。 • 动态调整按键位置、大小。 • 增加辅助线方便定位按键。 • 点击类按键支持修改名称，最多显示6个字符。（UI 切图等资源暂不支持动态设置。）

使用说明

关键类介绍

GamepadManager

虚拟按键管理类（继承 RelativeLayout）。

- 注册 OnEditListener 监听虚拟按键编辑事件的回调。
- 注册 IInstructionListener 监听触发按钮事件的回调。

关键流程说明

1. 初始化。

```
mCustomGamePad = new GamepadManager(this);
mCustomGamePad.setEditListener();
mCustomGamePad.setInstructionListener();
// 按键视图应放在游戏视频视图之上，否则可能导致按键无法正常使用
addView(mCustomGamePad);
```

2. 读取配置文件。

```
String mCustomGamePadCfg;
mCustomGamePadCfg = readConfigFile("default_gamepad.cfg");
```

> **注意**
> 此处代码仅演示过程，具体文件的读写由业务侧来实现。

3. 显示虚拟按键或编辑虚拟按键。

```
// 显示虚拟按键
mCustomGamePad.showGamepad(mCustomGamePadCfg);
// 进入编辑模式
mCustomGamePad.editGamepad(mCustomGamePadCfg);
```

4. 启用手柄类按键布局。

关键类	说明
gamepad.needConnected	使用布局前判断是否为手柄按键
SDK.sendGamePadConnected	使用虚拟手柄前需调用接口通知云端启用手柄
SDK.sendGamePadDisconnected	结束虚拟手柄后主动卸载云端手柄

```
if (enable) {
 mCustomGamePad.setVisibility(View.VISIBLE);
 mCustomGamePad.showGamepad(mCustomGamePadCfg);
 // 判断是否为手柄按键
 if (mCustomGamePad.needConnected()) {
```

```
    mSDK.sendGamePadConnected();
   }
  } else {
   mCustomGamePad.setVisibility(View.GONE);
   if (mCustomGamePad.needConnected()) {
    mSDK.sendGamePadDisconnected();
   }
  }
```

5. 监听编辑事件的回调。

```
mCustomGamePad.setEditListener((isChanged, jsonCfg) -> {
    if (isChanged) {
    // 保存编辑后的配置文件
      mCustomGamePadCfg = jsonCfg;
    }
});
```

6. 监听按键点击触发的事件。

```
mCustomGamePad.setInstructionListener(instruction -> {
    if (mSDK != null) {
     // 通过云游戏SDK发送操作指令
      mSDK.sendKmEvents(instruction);
    }
});
```

7. 适配与虚拟鼠标共存。

```
// 设置点击事件的代理，将事件传递给鼠标视图
public void setGamePadTouchDelegate(IGamepadTouchDelegate delegate) {
    mCustomGamePad.setGamePadTouchDelegate(delegate);

}

mControlView.setGamePadTouchDelegate((v, event) -> {
    mGameView.handleMotion(event);
    return true;
});
```

```
```

配置文件

虚拟按键的配置信息以 JSON 格式保存为配置文件，由业务侧负责管理文件的保存与读取，以此来为不同的用户提供个性化虚拟按键配置。

忽略混淆

打包时请设置混淆规则：

- 对 Java 代码不做混淆。

```
-keep class com.tencent.tcggamepad.**{*;}
```

- 对资源文件不做混淆。

```
tools:keep="@drawable/tcg_*"
```

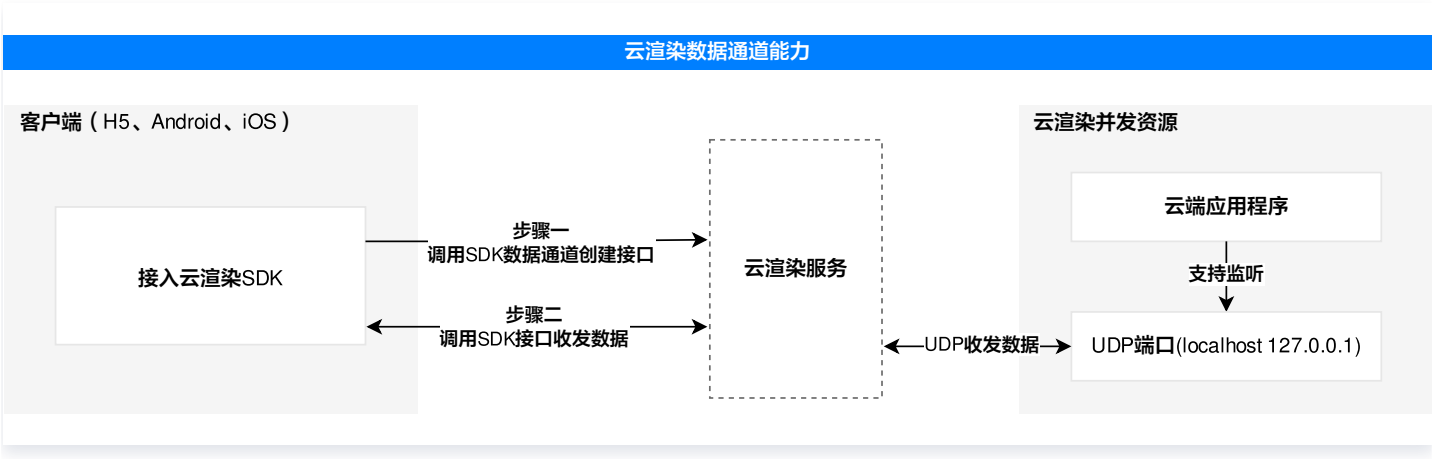
数据通道

最近更新时间：2024-12-04 13:00:12

云游戏环境下，您无法与云端运行的应用直接进行通信，当您需要建立客户端与云端应用的通信时，可以使用我们提供的数据通道能力。

概念说明

腾讯云提供的数据通道能力使用 UDP 协议，数据通道调用逻辑如下图：



使用说明

前提条件

在使用云游戏提供的数据通道前，您需要保证您已按照以下步骤进行准备：

1. 云端应用创建 UDP 服务器，监听一个 UDP 端口（ `localhost 127.0.0.1` 范围建议 10000 – 20000），并开始等待接收 UDP 包。
2. 云渲染的客户端调用云游戏 SDK 接口创建透传通道，SDK 接口里的目标端口参数应为 步骤1 中云端监听的端口。
3. 云渲染的客户端首先发送一个自定义数据包，云端应用 UDP 会收到请求，解析出本地代理端口。
4. 云端应用向 步骤3 拿到的**本地代理端口**发送自定义数据包，数据包将通过创建好的数据通道返回给客户端应用。

接口说明

请根据您的需要参考不同端 SDK 接口说明进行集成：

- 前端 JS SDK
- Android 端 SDK

示例代码

前端（以 JS SDK 为例）

```
(async _ => {
    // 接收云端数据的回调
    const onMessage = msg => {
        console.log("收到云端应用回传数据:", msg);
    };
    // 定时重复创建直到成功
    const result = await new Promise((resolve, reject) => {
        const timer = setInterval(async _ => {
            // 创建数据通道
            const ret = await TCGSDK.createCustomDataChannel({
                destPort: xxxx, onMessage  //destPort: xxxx ，xxxx端口范围
为10000~20000
            });
            if (ret.code == 0) {
                resolve(ret);
                clearInterval(timer);
            }
        }, 2000);// 2秒间隔
    });
    /*
    * 判断是否成功
    * result的结构{code: number, msg: string, sendMessage: Function }
    */
    if (result.code == 0) {
        // 随便发送一个绑定包，使云端应用的UDP服务能获得代理端口
        result.sendMessage('test');
    }
    // 正常收发数据
    result.sendMessage(`${custom_data}`);
})();
```

云端应用 UDP（以 C/C++ 为例）

```
int main() {
    int udp_socket_fd = socket(AF_INET, SOCK_DGRAM, 0);
    if (udp_socket_fd == -1) {
        printf("socket failed!\n");
        return -1;
    }
```

```
    //设置目的IP地址
    struct sockaddr_in bind_addr = { 0 };
    bind_addr.sin_family = AF_INET;
    bind_addr.sin_port = htons(xxxx);// htons(xxxx)中的端口范围为 10000-
20000
    bind_addr.sin_addr.s_addr = inet_addr("0.0.0.0"); // 绑定IP

    // 绑定端口
    int ret = bind(udp_socket_fd, (struct sockaddr *)&bind_addr,
sizeof(bind_addr));
    if (ret < 0) {
        perror("bind fail:");
        close(udp_socket_fd);
        return -1;
    }

    // 开始等待客户端消息
    struct sockaddr_in  upstream_addr = { 0 };  // 用来存放云渲染代理的地址
    int len = sizeof(upstream_addr);
    char buf[1024] = { 0 };// 接收消息缓冲区
    while (true) {
        ret = recvfrom(udp_socket_fd, buf, sizeof(buf), 0, (struct
sockaddr *)& upstream_addr, &len);
        if (ret == -1) {
            break;
        }
        // buf 为前端发来的消息"test"
        // 后续可以用upstream_addr回传消息给前端
        const char* response = "response";
        sendto(udp_socket_fd, response, strlen(response), 0, (struct
sockaddr *) & upstream_addr, sizeof(upstream_addr));
    }
    return 0;
}
```

应用预加载

最近更新时间：2022-01-13 10:33:42

使用场景

如果您的应用启动速度较慢，玩家按照正常的云游戏启动流程打开云游画面后需要等待应用启动，这会损害玩家的使用体验。
此时您可以使用云游戏的预启动能力，通过简单的配置后，云端即可预启动大型应用，玩家连接云游戏实例后可立刻获得应用或游戏画面。

使用说明

具体操作请参见 游戏预热 。

> **注意**
> 手游方案默认开启预加载能力。

云游戏直播推流

最近更新时间：2024-12-11 16:25:23

使用场景

云游戏直播推流，是云端游和云手游并发基础上的附加功能。将云端游戏运行的实时画面，上行推流至腾讯云直播。

场景类型	说明
云游戏场景使用	可将云端游戏画面进行推流直播，避免繁琐的软件配置，轻松成为游戏主播
数字孪生实时渲染	可将云应用与线上直播打通。在地产销售、软件教学、智慧文旅场景实现一对多的画面共享
虚拟直播场景下	主播端的动作和语音被采集，传递给云端算力驱动虚拟角色，并可将云端渲染出的画面进行推流直播

使用说明

在调用云游戏直播推流功能前，请确保：

1. 您的账户保有可用的云端游或者云手游并发。
2. 您已经开通 腾讯云直播 产品。
3. 在云游戏产品 云端推流 页面，您已经绑定了推流域名。

可参见 计费规则 中云游戏直播推流相关描述。

使用中的云游戏并发，可以通过调用云 API，实现将游戏画面推流至腾讯云直播。当前 API 包含开始推流和停止推流功能。

相关接口

接口名称	接口功能
StartPublishStreamToCSS	开始云端推流到云直播
StopPublishStream	停止云端推流

游戏微端

最近更新时间：2023-05-26 14:32:46

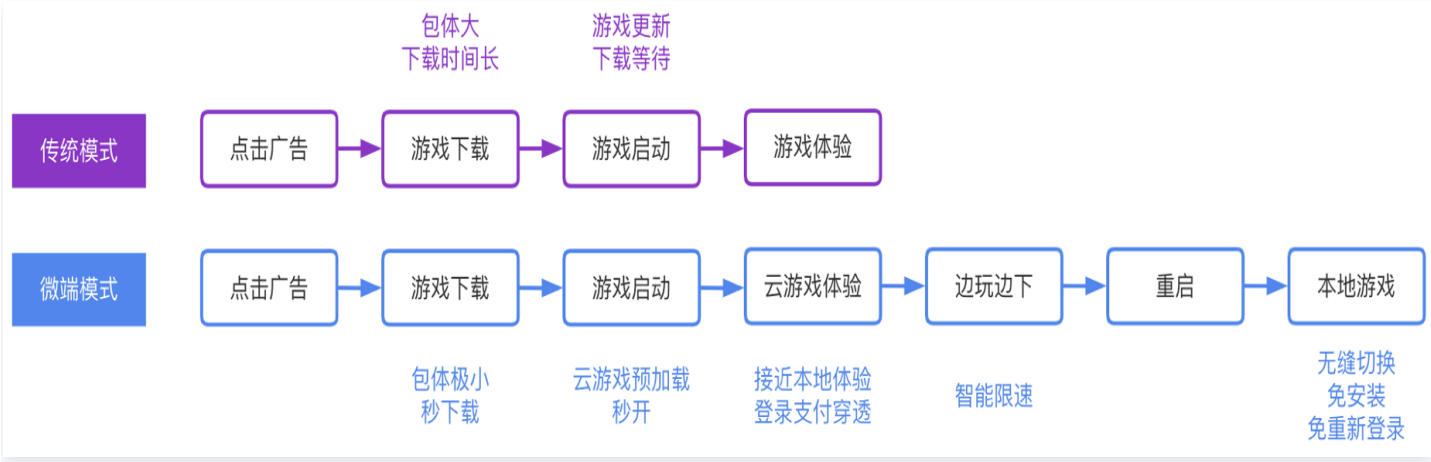
> **注意**
> 目前游戏微端能力，仅支持手游方案。

微端概述

微端介绍

针对手游买量场景下“游戏包体大，获客成本高”的行业痛点，腾讯云渲染依托行业领先的实时云渲染能力，帮助游戏开发者一站快速搭建“游戏微端”，将包体体积降低至数 M 大小，有效降低获客成本。
开发者只需按本文档说明进行开发，便可快速上线“游戏微端”，提升投放效率。

用户体验流程



> **注意**
> 微端即某款游戏的专属云游戏客户端，提供了云试玩、静默下载、登录/支付穿透功能，体验无限接近游戏原生客户端。

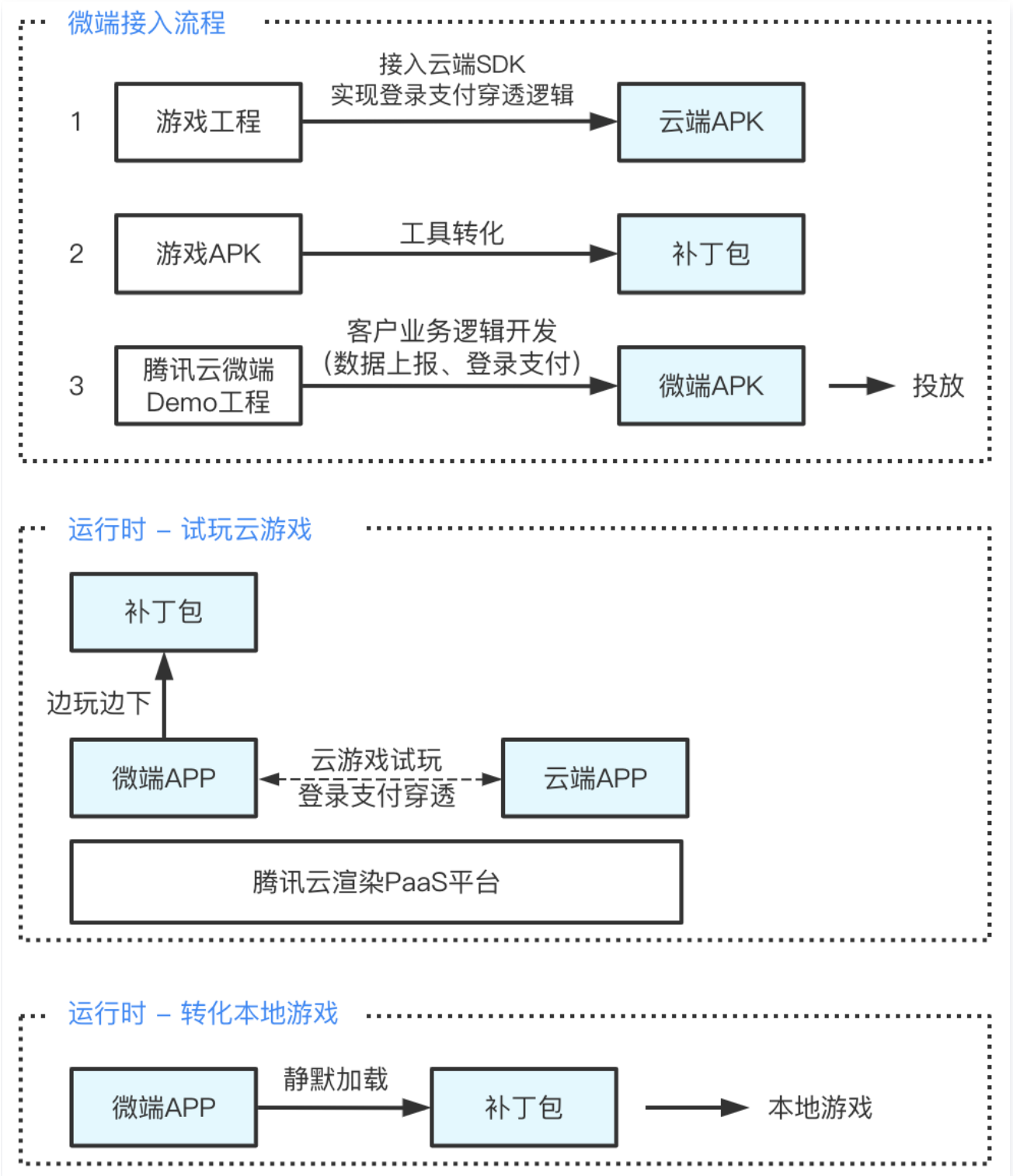
核心能力

模块	能力解释
包体缩小	通过云游戏能力，降低游戏包体体积，可低至10M
边玩边下	玩家使用微端进行游戏过程中，静默下载游戏更新包

更新包免安装	云游戏微端静默升级成更新包，无需安装流程，用户无感知
登录/支付穿透	支持唤起本地 App 进行云游登录/支付

微端业务模块

游戏微端整体业务流程可以拆解为以下模块和关系（蓝色方块为客户需生成的包体）。



模块	说明
微端 App	微端App，集成了微端 SDK，是用于广告投放的包体，供 C 端用户玩云游戏的载体
补丁包	补丁包，集成了热更新 SDK，用于和微端包进行差分生成补丁包，是微端包热更新升级后的目标 App
云端包	云端包，集成了云端 SDK，运行于云端，是 C 端用户在微端上玩的云游戏
云端 SDK	云端 SDK，提供云端 App 登录和支付穿透的能力

我们提供的资源：

- **微端APP示例工程：**该工程中已经集成了微端 SDK（包括云游戏 SDK、热更新 SDK），您可以在此工程的基础上快速进行二次开发，生成您自己的微端包。
- **热更新工具：**集成之后生成更新包，为您的游戏原始包提供热更新能力。
- **云端 SDK：**集成之后生成云端包，为您的游戏原始包提供和微端 App 的通信能力，用于实现登录/支付功能。
- **补丁包管理能力：**补丁包生成之后，我们将您的补丁包上传到服务器中以便微端 App 正常下载补丁包。

> **说明**
> 目前通过线下方式提供**补丁包管理能力**，后续将提供线上的热更新管理后台。

接入流程

详细接入步骤请参见 微端接入流程 。

计费说明

游戏微端解决方案基于云游戏产品构建，故费用包含以下两部分。

云游戏并发费用

- 云游戏并发最多只能同时支持一个玩家运行游戏，您可以根据您业务中用户并发数的峰值、平均值以及用户和潜在用户的地区分布，来决定您购买云游戏并发的数量和地区分布。
- 详细步骤请参见 计费说明 。

CDN 费用

- CDN 用于存放补丁包（快速接入版不需要该能力）。
- 若您使用腾讯云 CDN，则按 CDN 产品计费规则进行收费。