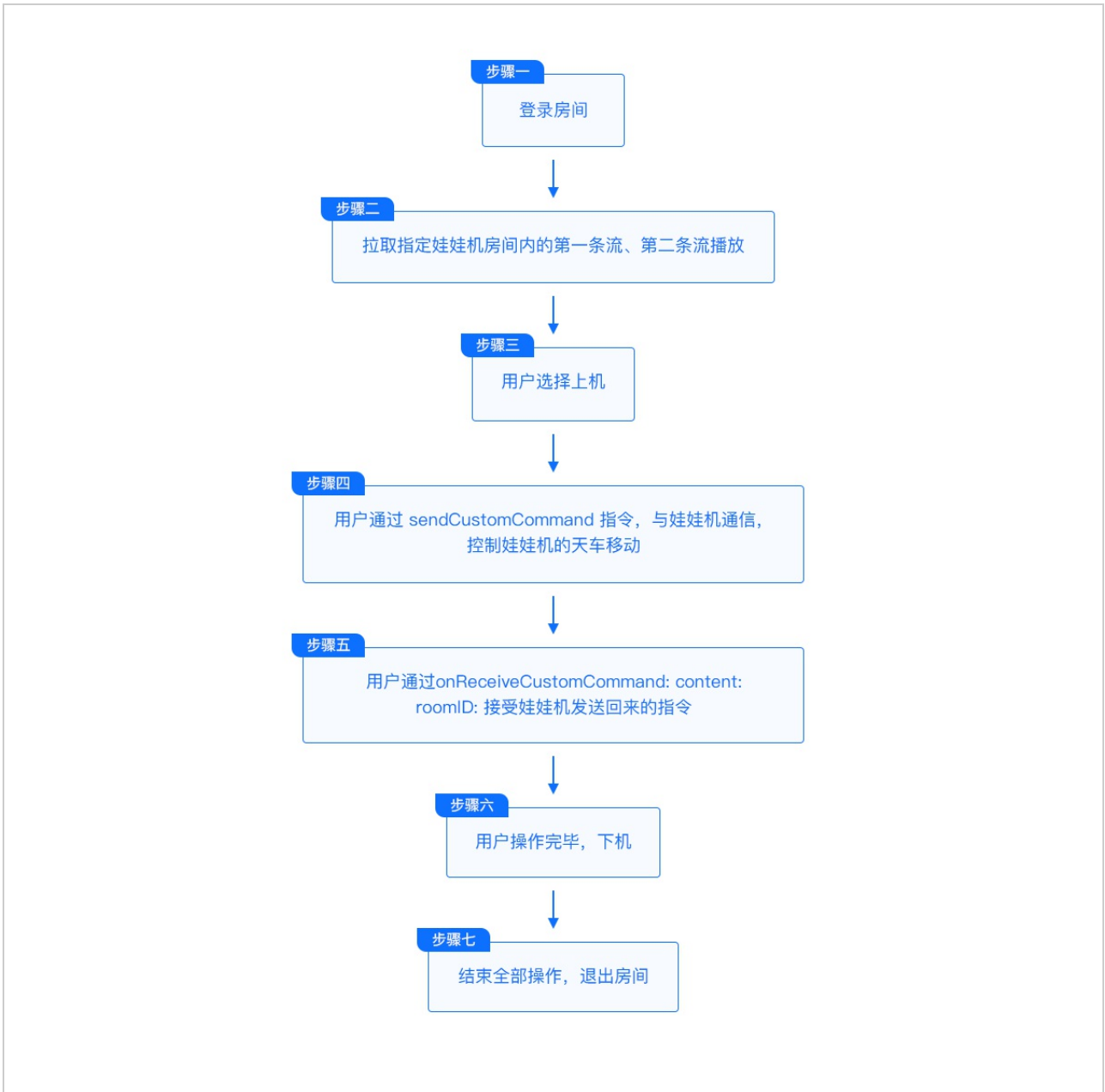


功能实现流程

娃娃机 web 端场景的主要结构及流程如下图所示：



请注意：

1. 上图中的 **用户选择上机** 步骤，可使用 jZego.js SDK 拉流切换实现。
2. 开发者注意区分 **娃娃机 App 端** 和 **娃娃机**。前者指的是，提供给玩家抓娃娃的、安装在 iOS 或 Android 平台的 App。后者指的是，直接与娃娃机通过串口通信的、安装在 Android 平台上的 App，类似于娃娃机 Server。
3. 为了便于开发者更快理解 WaWaji Client 中的逻辑，下述每节会将功能核心源码片段挑出来并加以讲解。开发者亦可直接阅读 WaWaji Client 源码，两者是一致的。

1 娃娃机系统实现流程

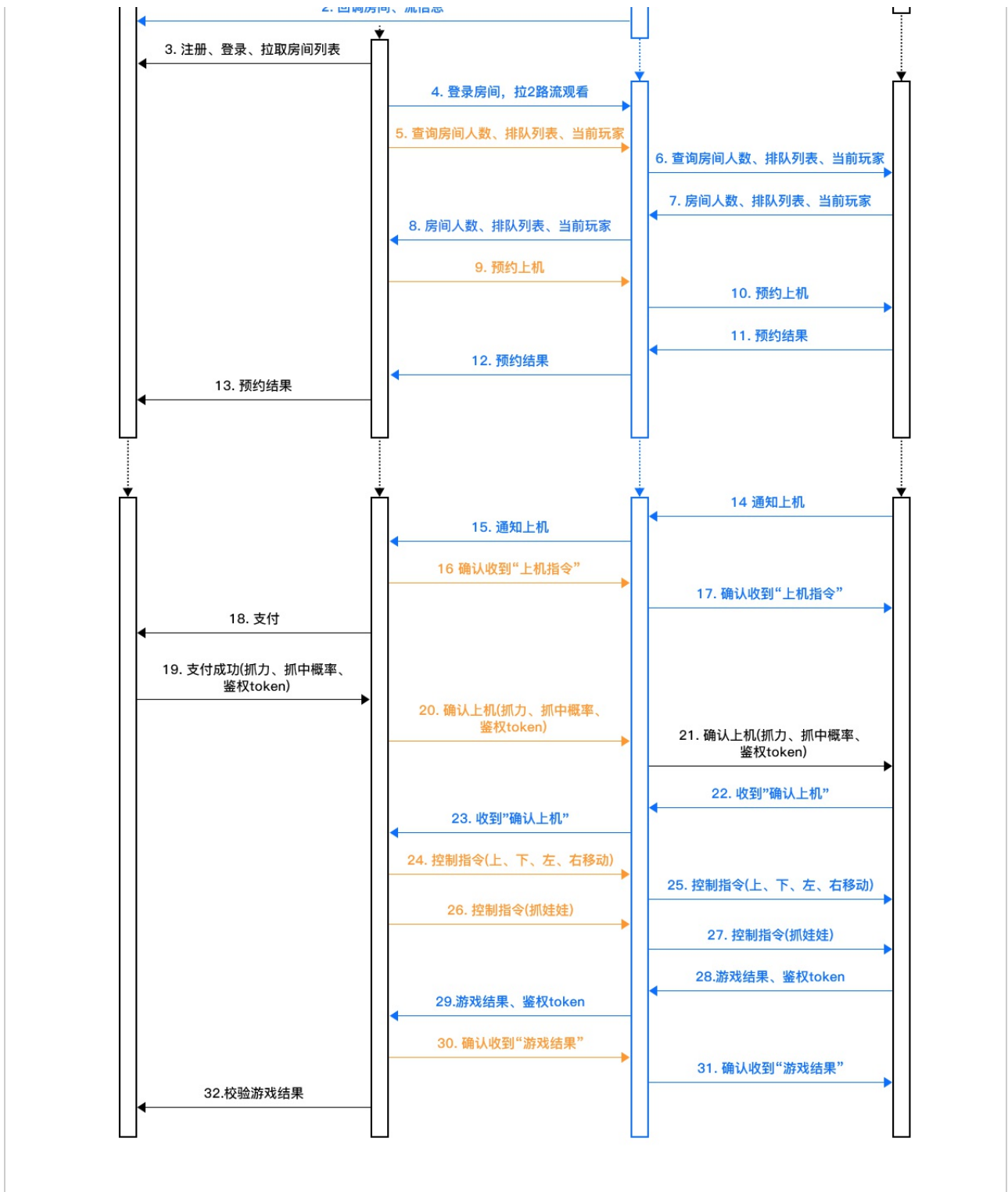
1、安装娃娃机控制端APK到安卓板子上

2、娃娃机启动，推流成功后，Zego后台会给业务后台POST 流创建的相关信息(业务后台提供回调地址)，用于业务侧维护娃娃机列表

3、客户端的开发

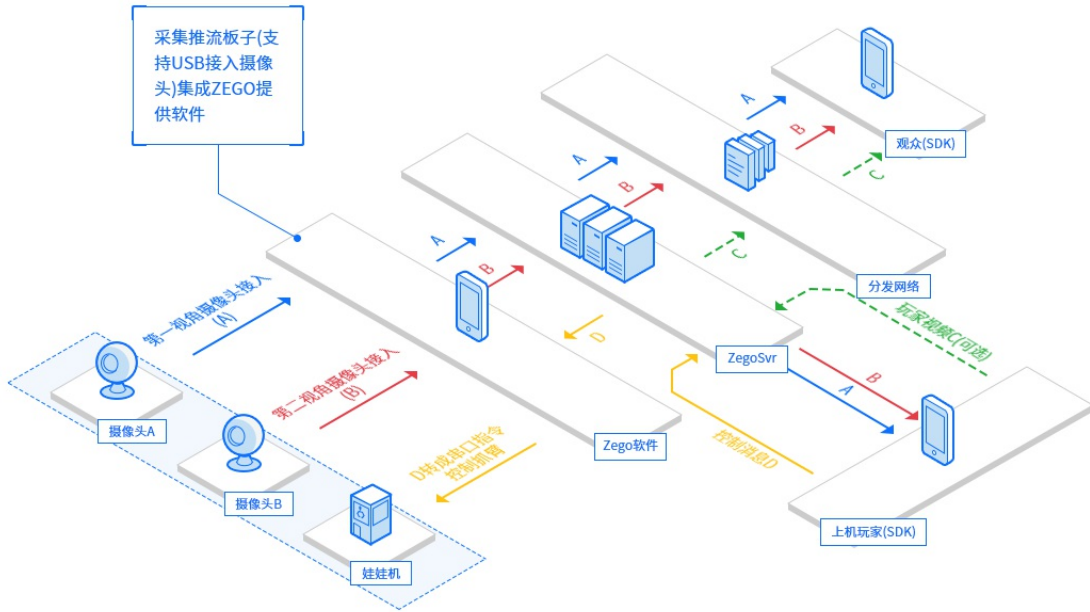
娃娃机系统实现流程如下图所示。该方案中，娃娃机控制端无需与业务后台直接通信。





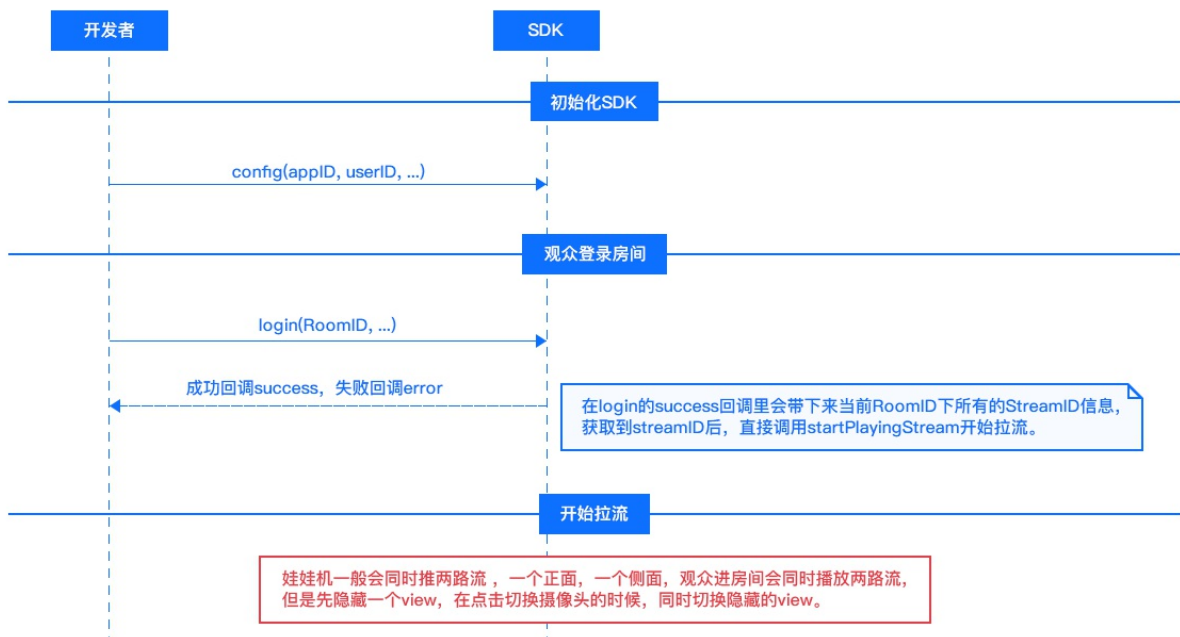
2 系统架构图

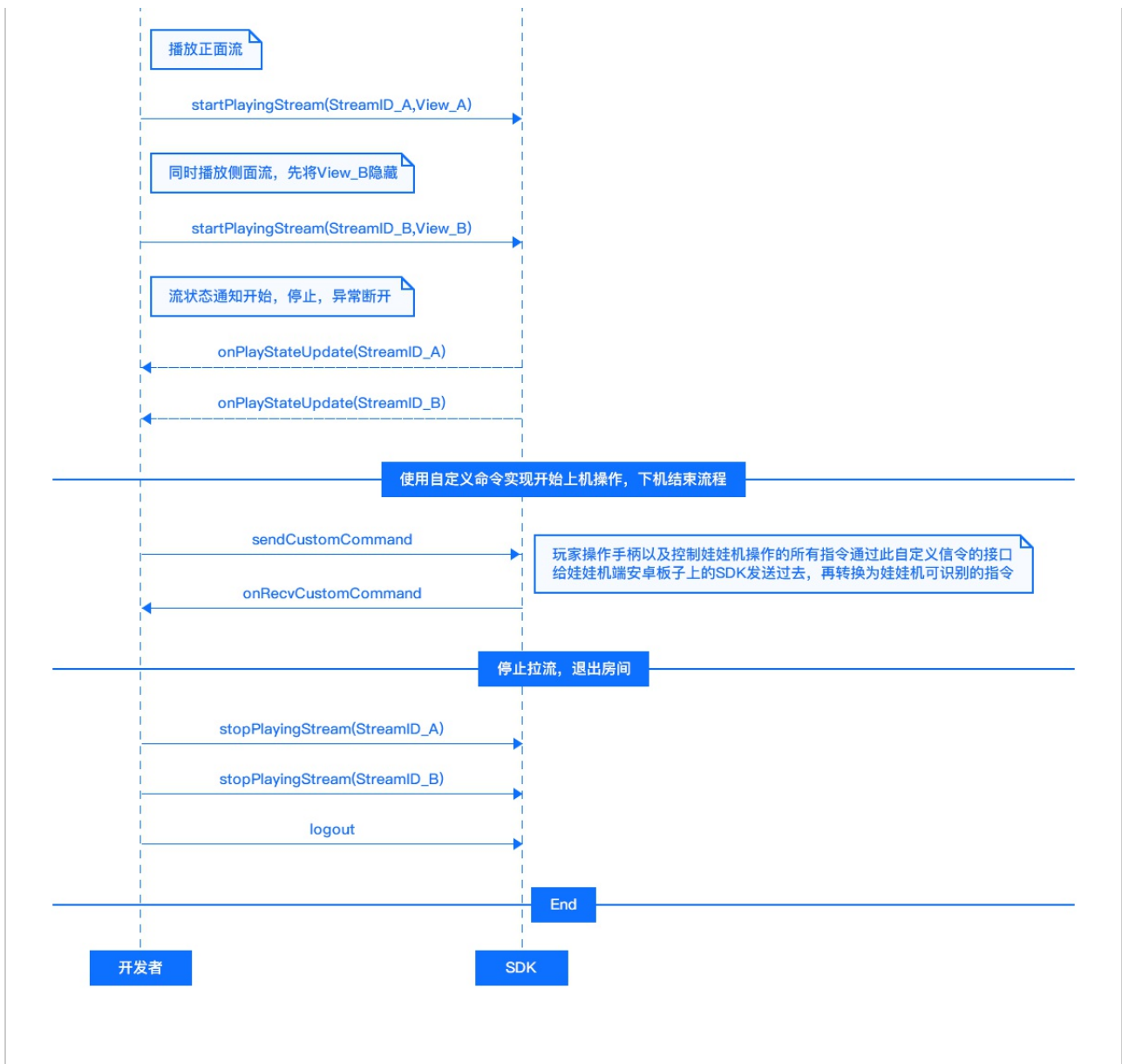
这里只需要两个普通的摄像头USB接入到安卓板子即可



3 API 调用时序图

娃娃机客户端API调用流程图





4 实现流程

4.1 通过config接口注入基本参数配置

```
//new 一个实例
var zg = new ZegoClient();

// 1. 配置参数
zg.config({
    appid: appid,           // 必填, 应用id, 由即构分配
    idName: idName,       // 必填, 用户自定义id
    nickName: nickName,   // 必填, 用户自定义昵称
    server: server        // 必填, 接入服务器地址, 由即构分配
    logUrl: logUrl        // 必填, logServer地址, 由即构分配
});
```

4.2 登录房间

以下所有步骤均 [基于登录房间成功的前提](#)。WaWaji Web 中相关源码片段如下，仅供参考：

```
// 2. 登录
zg.login(roomID, role, token, function(streamList) {
    // 登录成功回调
    // code
}, function(err) {
    // 登录失败回调
    // code
});
```

4.3 拉流

观众想看到娃娃机画面，需先拉流

WaWaji Web 中拉流相关源码片段如下，仅供参考：

```
// 3. 初始化流 流信息可在登录成功回调函数中获得
// 得到流信息后，传入流ID和指定的原生canvas元素，播放与该ID对应的视频流

// 正面
zg.startPlayingStream(useStreamList[0].stream_id, frontView);
// 正面流音量设置为最大
zg.setPlayVolume(useStreamList[0].stream_id, 100);

// 侧面
zg.startPlayingStream(useStreamList[1].stream_id, sideview);
// 侧面流音量设置为静音
zg.setPlayVolume(useStreamList[1].stream_id, 0);
```

1. 调用 `stopPlayingStream` 停止播放指定流
2. 调用 `startPlayingStream` 重新播放指定流

请注意：

1. 目前 WaWaji Web 使用的方案是，进入房间后，创建两个 view 分别播放两条流数据，用户可通过手动切换 view，继而切换当前可见流。
2. 这里需要注意的是，如果某一条流从可见切换为不可见，需要调用 `setPlayVolume` 改变流播放声音为无声，否则可能造成流画面和流声音混乱的情况。反之亦然。

4.4 发送指令

用户需要调用 `sendCustomCommand` 接口发送指令给娃娃机（此处的娃娃机，指的是控制娃娃机硬件的 **Server** 端，后面简称为娃娃机），娃娃机收到指令后，做出对应的响应。

请注意，此处指令是发送给娃娃机控制端，而不是房间里的其他玩家或自己。

WaWaji Web 中发出指令相关源码片段如下，仅供参考：

```
zg.sendCustomCommand(  
  [anchor_id],  
  custom_msg,  
  function(seq, custom_content) {  
    console.log('customCMD 成功', custom_content);  
  },  
  function(err, seq, custom_content) {  
    console.log('customCMD 失败', custom_content);  
  }  
);
```

娃娃机客户端与控制端信令交互流程请参考：[娃娃机-信令交互](#)

4.5 接收指令

WaWaji Web 端可以通过 `onRecvCustomCommand` 接受娃娃机返回的命令：

```
// 接收消息接口  
zg.onRecvCustomCommand = function(from_userid, from_idName, custom_content) {  
  // code  
}
```

娃娃机客户端与控制端信令交互流程请参考：[娃娃机-信令交互](#)

4.6 退出

如果用户不再进行游戏，退出当前的娃娃机房间，注意调用退出房间，确保停止拉流，并清空状态。

WaWaji Client 中退出房间相关源码片段如下，仅供参考：

```
// 登出
zg.stopPlayingStream(useStreamList[1].stream_id);
zg.stopPlayingStream(useStreamList[0].stream_id);
zg.logout()
```

5 安全方案

5.1 娃娃机房间登录安全

5.1.1 基本流程

- App与业务后台建立通讯，获取Token信息
- App调用ZegoClient.Login登陆Zego服务器，传入Token信息，验证通过后，完成登陆。
- 之后ZegoClient保持与Zego服务器的长连接，处理发送或接收的消息
- App调用ZegoClient.Logout登出Zego服务器

5.1.2 login_token信息

- login_token信息为标准json格式，具体为：

```
{
  "ver": 1, //int类型
  "hash": 710a5199398176a316ebcc88bc5b4470, //字符串类型
  "nonce": 随机串, 需要保证同一USER_ID在失效时间内不重复, 建议按guid生成, //
字符串类型
  "expired": 失效时间, unix_timestamp //int64类型, 单位秒
}
```

- login_token信息由业务后台负责, 其中hash的生成算法如下:
hash=MD5(app_id+app_key_32+id_name+nonce+expired)。
app_key_32: 通过app_key运算获得。

算法: 剔除app_key里的"0x", ", "字符后, 获取前面32字节即为app_key_32 (具体算法可参考以下代码)

id_name: 为客户登录时候传入的字符串用户id

nonce: 一次性随机字符串串

expired: 过期时间

- login_token传输过程中, 经过base64加密。
- 每次登陆都要重新获取login_token
- go语言login_token生成示例代码

```

func makeTokenSample(appid uint32, app_key string, idname string, expire
d_add int64) (ret string, err error) {
    nonce := UniqueId()
    expired := time.Now().Unix() + expired_add //单位:秒

    app_key = strings.Replace(app_key, "0x", "", -1)
    app_key = strings.Replace(app_key, ",", "", -1)
    if len(app_key) < 32 {
        return "", fmt.Errorf("app_key wrong")
    }

    app_key_32 := app_key[0:32]
    source := fmt.Sprintf("%d%s%s%s%d", appid, app_key_32, idname, nonce, exp
ired)
    sum := GetMd5String(source)

    token := tokenInfo{}
    token.Ver = 1
    token.Hash = sum
    token.Nonce = nonce
    token.Expired = expired

    buf, err := json.Marshal(token)
    if err != nil {
        return "", err
    }

    encodeString := base64.StdEncoding.EncodeToString(buf)
    return encodeString, nil
}

```

- php语言login_token生成示例代码

```

public function getToken(int $app_id, string $app_key, string $idname, int $expired_add)
{
    $nonce = uniqid();
    $expired = time() + $expired_add; //单位:秒

    $app_key = str_replace("0x", "", $app_key);
    $app_key = str_replace(",", "", $app_key);
    if(strlen($app_key) < 32) {
        return false;
    }
    $app_key_32 = substr($app_key, 0, 32);

    $source = $app_id.$app_key_32.$idname.$nonce.$expired;
    $sum = md5($source);

    $tokenInfo = [
        'ver' => 1,
        'hash' => $sum,
        'nonce' => $nonce,
        'expired' => $expired,
    ];
    $token = base64_encode(json_encode($tokenInfo));
    return $token;
}

```

- 如何保证获取Token信息的过程是安全的？业务APP需要和业务后台建立一种安全通讯和鉴权机制，业务APP使用自有的账户体系 / 第三方认证体系的登陆完成后，业务APP和业务后台交互获取该login_token，AppSecret是存储在业务后台的。

5.2 娃娃机信令安全

5.2.1 问题

- 娃娃机信令是指App通过ZegoClient.sendCustomCommand的接口发送娃娃机的控制命令，包括上机前或者上机过程如何确保控制是被授权过的？比如支付环节过程，娃娃机接受控制请求，需要确保用户已经完成支付。

5.2.2 解决方案

- 对custom_msg进行鉴权，对ZEGO透明
- 对custom_msg进行鉴权的一种方案是在msg中携带token, Zego负责透传，业务方负责生成和验证

5.2.3 基本流程-App

- App成功登陆之后与娃娃机通讯可以建立msg_token进行消息验证。（如支付验证流程）
- App与业务后台建立通讯，实现自己msg_token获取策略, token生成机制和更新策略对ZEGO透明。
- App调用ZegoClient.sendCustomCommand，传入携带token的msg, Zego服务负责发送到娃娃机。
- 娃娃机收到msg对msg_token进行鉴权，通过后执行请求命令，具体哪些命令需要鉴权，App可以自行决定。

5.2.4 如何保证获取msg_token的过程是安全的

msg_token对Zego是透明的，由业务设计具体方案去实施，对于业务APP很自然的方案是在业务APP使用自有的账户体系 / 第三方认证体系的登陆完成后，业务APP和业务后台交互获取和更新该Token.

5.3 安全验证

带token登录验证流程

蓝色部分：调用ZEGO SDK实现

橙色部分：ZEGO开放源代码给开发者

黑色部分：客户自己开发

