互动游戏paas模块使用说明0406

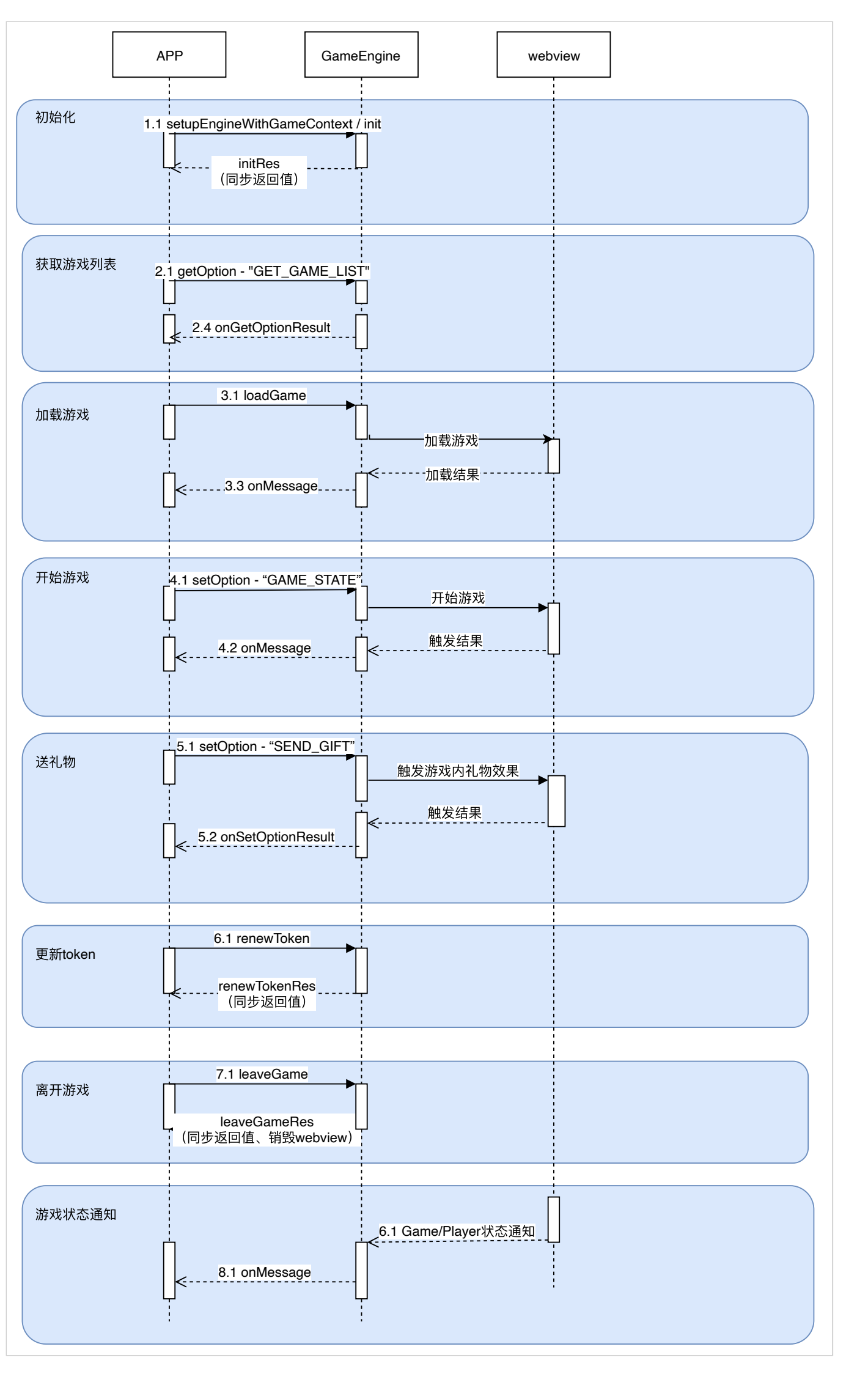
由 guanhaosen@agora.io创建, 最后修改于大约1分钟以前

一、如何引入

paas模块文件格式为：aar（安卓）、xcframework（IOS）

- 安卓类名为 **io.agora.gamesdk.GameEngine**
- IOS类名为 **AgoraGameEngine**

二、接口调用流程



三、接口详细文档

1. 初始化，注册回调（ios使用 setupEngineWithGameContext，安卓使用init）
   - 初始化 AgoraGameEngine 配置参数说明：

参数名	类型	获取方式	用途
appId	string	项目的APPID	分辨客户身份
uid	string	自己定义，用于生成token	配合token实现鉴权
rtmToken	string	通过上一个uid生成的RTM token	对调用进行鉴权

注意：当rtmToken过期时需要调用renewToken更新进行更新。

2. 获取游戏列表（使用 getOption）
   - getOption 支持参数及说明

说明	key/类型	示例json	回调
获取游戏列表	GET_GAME_LIST	{ "page":1, "limit":10, "language":"zh-CN" }	onGetOptionResult

3. 加载游戏（使用 loadGame ）
   - loadGame支持参数及说明

参数名	类型	获取方式	用途	备注
gameId	string	通过游戏列表返回	用于确定需要加载的游戏	从声网获取到游戏列表后，从列表中选择游戏
roomId	string	自己定义，区分不同房间	用来划分用户，同一个roomId的用户会在同一局游戏	参与游戏及观看游戏都需要在同一个roomId下
userId	string	自己定义，区分不同用户	区分同一局游戏内的不同用户	客户需确保在同一局游戏中不重复，否则会影响到断线重新加入等逻辑
role	string	自己定义，区分不同角色	游戏角色。	房主： 1 同玩玩家： 2 观众： 3
language	string	固定枚举	游戏内不同语言	中文："zh-CN" 英文："en"
options	json字符串	游戏选项	定义游戏的独特参数	格式为**json字符串**

options 参数：

参数名	是否必选	类型	说明
name	是	string	用户昵称
avatar	是	string	用户头像
avatar_type	否	number	观众在游戏中形象类型, 0-无头像昵称(默认), 1-有昵称无头像, 2-有昵称有头像, 3-无昵称有头像
to_user	是	string	王国激战观众加入对应阵营的主播id （只有观众需要填写）
left	否	number	单位像素, 游戏内容左侧距离
right	否	number	单位像素, 游戏内容右侧距离
top	否	number	单位像素, 游戏内容顶部距离
bottom	否	number	单位像素, 游戏底部距离
level	否	number	用户等级（保留字段）
time_limit	否	number	一局游戏的时间限制, 单位秒, 默认300
show_join	否	number	0-不显示游戏加入按钮(默认) ,1-显示游戏加入按钮
show_ready	否	number	0-不显示游戏准备按钮(默认) ,1-显示游戏准备按钮
show_start	否	number	0-不显示游戏开始按钮(默认) ,1-显示游戏开始按钮
show_kickout	否	number	0-不显示踢人按钮(默认) ,1-显示踢人按钮

- **注意：**
  1、**roomId、userId需要全局唯一**
  2、**一个游戏房间里只有一个房主**

4. 加入/准备/结束 本局游戏（使用 setOption，key为GAME_STATE）
5. 在游戏进行中的互动操作（触发礼物效果使用setOption接口，key为SEND_GIFT）
   - setOption支持参数及说明

<table>
<tr><th>说明</th><th>权限</th><th>key/类型</th><th>示例json</th><th>回调</th></tr>
<tr><td>用户（本人）加入游戏/退出游戏
<table><tr><th>参数</th><th>类型</th><th>描述</th></tr><tr><td>isIn</td><td>bool</td><td>正确流程：（房主不需要准备操作）
1.加入游戏: isIn=true → 准备游戏 → 开始游戏;
2.离开游戏: 结束游戏 → 取消准备 → isIn=false;</td></tr></table></td><td>观众调用</td><td>GAME_STATE</td><td>{ "state":"app_common_self_in", "data":{ "isIn":true } }</td><td>onMessage
如果调用有问题, 游戏方返回错误码
{ "state":"game_common_error", "data":{ "code":51500, "msg":"调用失败" } }</td></tr>
<tr><td>用户（本人）准备/取消准备
<table><tr><th>参数</th><th>类型</th><th>描述</th></tr><tr><td>isReady</td><td>bool</td><td>准备</td></tr></table></td><td>观众调用</td><td>GAME_STATE</td><td>{ "state":"app_common_self_ready", "data":{ "isReady":true } }</td><td>onMessage</td></tr>
<tr><td>开始游戏
<table><tr><th>参数</th><th>类型</th><th>描述</th></tr><tr><td>isPlaying</td><td>bool（只能传true）</td><td>房主开始游戏</td></tr></table></td><td>房主调用</td><td>GAME_STATE</td><td>{ "state":"app_common_self_playing", "data":{ "isPlaying":true } }</td><td>onMessage</td></tr>
<tr><td>结束（本局）游戏
只用房主才能结束游戏
data为预留字段，传空对象就可以</td><td>房主调用</td><td>GAME_STATE</td><td>{ "state":"app_common_self_end", "data":{} }</td><td>onMessage</td></tr>
<tr><td>向指定用户发送礼物
目前仅允许加入游戏后向自己主播（to_user）送礼物
观众可以再游戏进行中加入游戏内
<table><tr><th>参数</th><th>类型</th><th>描述</th></tr><tr><td>to</td><td>string</td><td>需要送给的主播id</td></tr><tr><td>giftCost</td><td>number</td><td>加入房间后 gift_info 返回的key，详见onMessage</td></tr><tr><td>count</td><td>number</td><td>触发数量</td></tr></table></td><td>加入游戏的观众</td><td>SEND_GIFT</td><td>{ "to":"123", "payload":{ "giftCost":1, "count":1 } }</td><td>onSetOptionResult</td></tr>
</table>

6. 更新token（使用renewToken）
7. 退出游戏并销毁webview（使用leaveGame）
8. 接收Game/Player状态回调（使用onMessage）
   - onMessage支持参数及说明

<table>
<tr><th>说明</th><th>示例json</th></tr>
<tr><td>游戏加载完毕</td><td>{ "state":"game_common_hello", "data":{ "msg":"hello from guess and draw" } }</td></tr>
<tr><td>游戏房间信息
加入房间后会返回
<table><tr><th>字段</th><th>类型</th><th>描述</th></tr><tr><td>limit_time</td><td>number</td><td>单局游戏时长（单位秒）</td></tr><tr><td>gift_info</td><td>object</td><td>礼物效果列表</td></tr></table>gift_info 明细
<table><tr><th></th><th>类型</th><th>描述</th></tr><tr><td>key</td><td>string</td><td>值为发送礼物时的giftCost</td></tr><tr><td>value</td><td>string</td><td>礼物效果明细</td></tr></table></td><td>{ "key":"GAME_STATE", "jsonObject":{ "state":"game_common_room_info", "data":{ "gift_info":{ "1":"生成一个高级士兵", "2":"生成一个英雄" }, "limit_time":300 } } }</td></tr>
<tr><td>用户加入/离开房间事件回调
用户加入或离开游戏时，自己会收到此回调。
<table><tr><th>字段</th><th>类型</th><th>描述</th></tr><tr><td>userId</td><td>string</td><td>用户id</td></tr><tr><td>isIn</td><td>bool</td><td>true为进入房间，false为离开房间</td></tr><tr><td>teamId</td><td>number</td><td>加入哪支队伍</td></tr><tr><td>reason</td><td>number</td><td>-1为进入房间，0 主动退出，1 被踢</td></tr><tr><td>kickUID</td><td>number</td><td>触发踢人的用户id（保留字段，目前无意义）</td></tr></table></td><td>{ "state":"game_common_player_in", "data":{ "userId":"123", "isIn":true, "teamId":1, "reason":0, "kickUID":1 } }</td></tr>
<tr><td>用户准备/取消准备事件回调 用户准备或取消准备时，自己会收到此回调。
<table><tr><th>字段</th><th>类型</th><th>描述</th></tr><tr><td>userId</td><td>string</td><td>用户id</td></tr><tr><td>isReady</td><td>bool</td><td>true为准备，false为取消准备</td></tr></table></td><td>{ "state":"game_common_player_ready", "data":{ "userId":"1001", "isReady":true } }</td></tr>
<tr><td>用户开始/结束事件回调
用户开始或结束每局游戏，或用户离开游戏时，自己会收到此回调。
<table><tr><th>字段</th><th>类型</th><th>描述</th></tr><tr><td>userId</td><td>string</td><td>用户id</td></tr><tr><td>isPlaying</td><td>bool</td><td>游戏开始为true，结束为false</td></tr><tr><td>gameRoundId</td><td>number</td><td>本局游戏ID(游戏厂商生成，在此游戏内全局唯一）</td></tr><tr><td>reason</td><td>number</td><td>isPlaying为true时不需要关注reason，isPlaying为false时reason参数见下方说明</td></tr></table>reason参数明细
<table><tr><th>参数</th><th>说明</th><th>角色</th><th>描述</th></tr><tr><td>0</td><td>本局游戏正常结束</td><td>玩家</td><td>正常结束</td></tr><tr><td>1</td><td>提前结束本局游戏（玩家自己不玩了）</td><td>玩家</td><td>异常结束</td></tr><tr><td>2</td><td>提前结束本局游戏（无真人，只有机器人）</td><td>玩家</td><td>异常结束</td></tr><tr><td>3</td><td>提前结束本局游戏（所有人都不玩了）</td><td>玩家 / 房主</td><td>正常结束</td></tr></table></td><td>{ "state":"game_common_player_playing", "data":{ "userId":"123", "isPlaying":true, "gameRoundId":2, "reason":0 } }</td></tr>
</table>

IOS详细接口文档

IOS接口详情

```
typedef NS_ENUM(NSUInteger, AGELogLevel) {
    AGELogLevelNone = 0,
    AGELogLevelDebug = 1,
    AGELogLevelInfo = 2,
    AGELogLevelWarn = 4,
    AGELogLevelError = 8,
    AGELogLevelFatal = 16,
};
/*
 函数返回值的错误类型
 */
typedef NS_ENUM(NSInteger, AGEErrorCode) {
    /**
     * 0: No error occurs.
     */
    AGEErrorCodeOK = 0,
    /**
     * -1: A general error occurs (no specified reason).
     */
    AGEErrorCodeFAILED = -1,
    /**
     * -2: The argument is invalid.
     */
    AGEErrorCodeInvalidArgument = -2,
    /**
     * -3: The SDK module is not ready.
     */
    AGEErrorCodeNotReady = -3,
    /**
     * -4: The SDK does not support this function.
     */
    AGEErrorCodeNotSupport = -4,
    /**
     * -5: The request is rejected.
     */
    AGEErrorCodeRefused = -5,
    /**
     * -6: The buffer size is not big enough to store the returned data.
     */
    AGEErrorCodeBufferTooSmall = -6,
    /**
     * -7: The SDK is not initialized before calling this method.
     */
    AGEErrorCodeNotInitialized = -7,
    /**
     * -8: The state is invalid.
     */
    AGEErrorCodeInvalidState = -8,
    /**
     * -9: No permission. This is for internal use only, and does
     * not return to the app through any method or callback.
     */
    AGEErrorCodeNoPermission = -9,
};

@interface AGELogConfig : NSObject
/// log文件路径, 你需要保证路径的合法性, 默认沙盒Documents/AgoraGame.log
@property (nonatomic, copy) NSString *filePath;
/// log文件大小单位KB, 默认1024KB
@property (nonatomic, assign) NSInteger fileSizeInKB;
/// log level, 默认是AGELogLevelInfo
@property (nonatomic, assign) AGELogLevel level;
- (instancetype)initWithFilePath:(NSString *)filePath fileSizeInKB:(NSInteger)fileSizeInKB level:(AGELogLevel)level;
- (instancetype)initWithFilePath:(NSString *)filePath fileSizeInKB:(NSInteger)fileSizeInKB;
- (instancetype)initWithFilePath:(NSString *)filePath;
- (instancetype)init;
@end

typedef NSString *AGEOptionID NS_STRING_ENUM;
FOUNDATION_EXPORT AGEOptionID const AGEOptionIDGetGameList;
FOUNDATION_EXPORT AGEOptionID const AGEOptionIDGameState;
FOUNDATION_EXPORT AGEOptionID const AGEOptionIDSendGift;

@protocol AGEGameEngineEventProtocol <NSObject>
- (void)onSetOptionResult:(NSInteger)operationId optionId:(NSString *)optionId result:(BOOL)result reason:(NSString * _Nullable)reason;
- (void)onGetOptionResult:(NSInteger)operationId optionId:(NSString *)optionId result:(BOOL)result outOption:(NSString * _Nullable)outOption;
- (void)onMessage:(NSString *)messageId message:(NSString *)message;
@end

@interface AGEGameContext : NSObject
@property (nonatomic, copy) NSString *appId;                              //appid
@property (nonatomic, copy) NSString *uid;                                //uid
@property (nonatomic, copy) NSString *rtmToken;                           //rtmToken
@property (nonatomic, weak) id<AGEGameEngineEventProtocol> delegate;     //事件回调代理
@property (nonatomic, strong) AGELogConfig *logconfig;                    //日志配置
@end

@interface AGEGameOptions : NSObject
@property (nonatomic, copy) NSString *gameId;      //游戏id
@property (nonatomic, copy) NSString *roomId;      //房间id
@property (nonatomic, copy) NSString *userId;      //用户id, 注意用户id需要全局唯一
@property (nonatomic, copy) NSString *role;        //用户角色
@property (nonatomic, copy) NSString *language;    //游戏语言
@property (nonatomic, copy) NSString *options;     //自定义参数, json字符串, 保留字段, 目前为空字符串就可以
@end

@interface AgoraGameEngine : NSObject

/// AgoraGameEngine单例
+ (instancetype)sharedInstance;

/// 获取版本号
+ (NSString *)getVersion;

/// 设置engineContext
/// @param gameContext engine的必要参数
- (AGEErrorCode)setupEngineWithGameContext:(AGEGameContext *)gameContext NS_SWIFT_NAME(setup(gameContext:));

/// 更新token
/// @param token 游戏相关信息, 注意gameContext.rtmtoken的周期性更新
- (AGEErrorCode)renewToken:(NSString *)token;

/// 加载游戏, 如果已经加载了游戏, 会销毁上一个游戏的webview, 然后新建一个webview放到传的view容器中
/// @param options 游戏相关信息
/// @param view 游戏载入的容器
- (AGEErrorCode)loadGameWithOptions:(AGEGameOptions *)options view:(UIView *)view NS_SWIFT_NAME(loadGame(options:view:));

/// 设置数据
/// @param operationId 操作id, 在代理回调中用来标识某个操作
/// @param optionId 操作类别 支持的optionId: AGEOptionIDGameState, AGEOptionIDSendGift, 但不仅限于这些, 后期可根据业务增加
/// @param args 参数json格式
- (AGEErrorCode)setOptionWithOperationId:(NSInteger)operationId optionId:(NSString *)optionId args:(NSString *)args NS_SWIFT_NAME(setOption(operationId:optionId:args:));

/// 获取数据
/// @param operationId 操作id, 在代理回调中用来标识某个操作
/// @param optionId 操作类别 支持的optionId: AGEOptionIDGetGameList, 但不仅限于这些, 后期可根据业务增加
/// @param args 参数json格式
- (AGEErrorCode)getOptionWithOperationId:(NSInteger)operationId optionId:(NSString *)optionId args:(NSString *)args NS_SWIFT_NAME(getOption(operationId:optionId:args:));

/// 离开游戏, 会销毁webview
- (AGEErrorCode)leaveGame;

/// 销毁资源
- (AGEErrorCode)destroy;

@end
```

安卓详细接口文档

安卓接口详情

```
/**
 * Agora SDK 所需的 字段配置
 */
public class GameContext {
    @NonNull
    private Context context;                                    // Android 上下文
    @NonNull
    private final String  appId;                                //Game appId
    @NonNull
    private final String uid;                                   //uid
    @NonNull
    private final String rtmToken;                              //rtmToken
    @NonNull
    private final LogConfig logConfig;                          // 日志打印所需配置
    @Nullable
    public final IGameEngineEventHandler  eventHandler;         //事件回调
}
/**
 * 加载游戏所需参数
 */
public class GameOptions {
    @NonNull
    private final String gameId;                                //游戏id

    @NonNull
    private final String roomId;                                //房间id

    @NonNull
    private final String userId;                                //用户id 注意用户id需要全局唯一

    @NonNull
    private final String role;                                  //角色

    @NonNull
    private final String language;                              //游戏语言

    @Nullable
    private final String options;                               //自定义参数, json字符串, 保留字段, 可以为空
}
/**
 * 游戏模块
 */
public class GameEngine {
    /**
     * @param config 客户来维护 config 内容字段的有效性,  需定期更新 config 内的内容 (保持rtmToken是有效的) 。
     */
    public static int init(@NonNull GameContext config);
    /**
     * 加载游戏
     * <p>
     * 多次调用, 如果 container 一致, 则只是更新参数, 重新加载游戏
     * 如果 container 不一致, 则会销毁之前的游戏, 再重新加载
     *
     * @param options      加载游戏必要参数
     * @param container 游戏容器
     */
    @MainThread
    public synchronized int loadGame(@NonNull GameOptions options, @NonNull ViewGroup container);
    /**
     * 设置游戏配置数据
     *
     * @param operationId    操作id, 自己定义, 在代理回调中用来标识某个操作
     * @param optionId
     * @param args          json 字符串格式
     */
    public int setOption(int operationId, String optionId, String args);
    /**
     * 获取游戏配置数据
     *
     * @param operationId    操作id, 自己定义, 在代理回调中用来标识某个操作
     * @param optionId
     * @param args          json 字符串格式
     */
    public int getOption(int operationId, String optionId, String args);


    /**
     * 刷新 token
     * @param token 新的 rtmToken
     */
    public int renewToken(@NonNull String token);

    /**
     * 销毁游戏
     */
    public int leaveGame();


    /**
     * 销毁引擎
     */
    public static int destroy();

    /**
     * 获取当前版本
     */
    public static String getVersion();
}


/**
 * 游戏模块回调
 */
public interface IGameEngineEventHandler {
    /**
     * 设置自定义参数的结果回调
     */
    void onSetOptionResult(int operationId, String optionId, boolean result, String reason);
    /**
     * 获取自定义参数的结果回调
     */
    void onGetOptionResult(int operationId, String optionId, boolean result, String outOption);

    /**
     * 游戏事件通知回调
     * @param messageId
     * @param message    json字符串格式的事件
     */
    void onMessage(String messageId, String message);
}
```

回调模块错误信息格式：

1. onSetOptionResult的reason字段
2. onGetOptionResult的outOption字段
3. onMessage的message字段

错误信息格式：例如：{"state": "game_platform_error","data": {"code": 60400,"msg": ""}}

state字段：

game_network_error：网络错误

game_platform_error：其他错误

code字段：

60400：数据解析失败

60408：网络错误（请求失败/超时）

msg字段：

错误描述

四、示例代码

IOS

```
//IOS示例代码
//初始化
    let context = AGEGameContext()
    context.uid = "123"
    context.rtmtoken = "123"
    context.appId = "123"
    context.delegate = self

    var documentsPath = NSSearchPathForDirectoriesInDomains(.documentDirectory, .userDomainMask, true)[0]
    documentsPath += "/AgoraGame.log"
    let conf = AGELogConfig.init(filePath: documentsPath, fileSizeInKB: 10000, level: .debug)
    context.logconfig = conf

    AgoraGameEngine.sharedInstance().setup(gameContext:context)


//获取游戏列表, 在回调里拿返回
    AgoraGameEngine.sharedInstance().getOption(operationId: 1, optionId: AGEOptionID.getGameList.rawValue, args: "{\"page\":1,\"limit\":10,\"language\":\"zh-CN\"}")


//加载游戏
    guard let userId = userIdTF.text else { return }
    let item = dataSourceArray[selectIndexPath!.row]
    let options = AGEGameOptions()
    options.gameId = item["gameId"] as! String;

    options.roomId = "123";
    options.userId = userId;
    options.role = "1";
    options.language = "4";
    options.options = "";
    let string = ["avatar" : "https://terrigen-cdn-dev.marvel.com/content/prod/1x/012scw_ons_crd_02.jpg",
                  "name" : "大牛",
                  "time_limit" : 360].toJsonString()
    options.options = string!

    AgoraGameEngine.sharedInstance().loadGame(options: options, view: gameView)


//游戏过程中的互动, 使用setOption
//送礼物
    AgoraGameEngine.sharedInstance().setOption(operationId: 2, optionId: AGEOptionID.sendGift.rawValue, args: "{\"to\":\"22\",\"payload\":{\"giftCost\":1,\"count\":3}}")
//游戏操作: 准备
    AgoraGameEngine.sharedInstance().setOption(operationId: 3, optionId: AGEOptionID.GameState.rawValue, args: "{\"state\":\"app_common_self_ready\",\"data\":{\"isReady\":true}}"
//游戏操作: 开始游戏
    AgoraGameEngine.sharedInstance().setOption(operationId: 3, optionId: AGEOptionID.GameState.rawValue, args: "{\"state\":\"app_common_self_playing\",\"data\":{\"isPlaying\":tru
//其他游戏操作参考此游戏允许的自定义操作列表即可


//退出游戏
    AgoraGameEngine.sharedInstance().leaveGame()


//回调, 接收游戏内通知以及获取游戏配置信息
extension ViewController:AGEGameEngineEventProtocol {

//设置自定义参数的结果, 例如准备、加入游戏等的结果
    func onSetOptionResult(_ operationId: Int, optionId: String, result: Bool, reason: String?) {
        print("AGE: operationId:\(operationId) result:\(result) reason:\(String(describing: reason))")
    }

//回调即为游戏自定义通知事件, 例如游戏加载完毕、游戏房间信息回调等。
    func onMessage(_ messageId: String, message: String) {
        print("AGE: messageId:\(messageId) message:\(message)")
    }

//示例为获取游戏列表的回调, 根据不同的operationId区分不同的业务
    func onGetOptionResult(_ operationId: Int, optionId: String, result: Bool, outOption: String?) {
```

```
        print("AGE: operationId:\(operationId) optionId:\(optionId) outOption:\(String(describing: outOption))")
        guard let outOptions = outOptions else { return }
        let data = convertStringToDictionary(text: outOptions)

        if let data = data {
            dataSourceArray = data
            DispatchQueue.main.async {
                self.tableView.reloadData()
                self.selectIndexPath = nil
            }
        }
    }
    func convertStringToDictionary(text: String) -> [[String:AnyObject]]? {
       if let data = text.data(using: .utf8) {
           do {
               let json = try JSONSerialization.jsonObject(with: data, options: .mutableContainers) as? [String:AnyObject]
               let td = json?["items"] as? [[String:AnyObject]]
               return td
           } catch {
               print("Something went wrong")
           }
       }
       return nil
   }
}
```

Android 示例代码

游戏列表接口

Demo 参考

```
FetchGameListRequiredBean bean = new FetchGameListRequiredBean(10, 1);
GameEngine.getInstance().getOption(0, GameGetOptions.GET_GAME_LIST, new Gson().toJson(bean));


public class FetchGameListRequiredBean {
    private final int limit;
    private final int page;
    private final String language;

    public FetchGameListRequiredBean(int limit, int page) {
        this.limit = limit;
        this.page = page;
        this.language = Locale.getDefault().getLanguage().equalsIgnoreCase("zh") ? "zh-CN" : "en";
    }
}
```

开始游戏接口

Demo 参考

```
// 获取当前语言
String language = Locale.getDefault().getLanguage().equalsIgnoreCase("zh") ? "zh-CN" : "en";

JoinGameRequiredBean joinGameRequiredBean = new JoinGameRequiredBean(safePadding, localUser.getName(), localUser.getAvatar(), amHost ? null :currentRoom.getUserId());

GameOptions gameOptions = new GameOptions(roomGame.getGameId(), roomId, localUser.getUserId(), getIdentification(roomId), language, new Gson().toJson(joinGameRequiredBean));

GameEngine.getInstance().loadGame(gameOptions, gameContainerFgRoom);
```

```
public class JoinGameRequiredBean {
    private final int left;
    private final int top;
    private final int right;
    private final int bottom;
    @NonNull
    private final String name;
    @NonNull
    private final String avatar;
    @Nullable
    private final String to_user;
    private final int avatar_type = 2;
    private final int time_limit = 360;
    private final int show_join;
    private final int show_ready;
    private final int show_start;
    private final int show_kickout;

    public JoinGameRequiredBean(@NonNull Rect safePadding, @NonNull String name, @NonNull String avatar, @Nullable String targetUser) {
        this.left = safePadding.left;
        this.top = safePadding.top;
        this.right = safePadding.right;
        this.bottom = safePadding.bottom;
        this.name = name;
        this.avatar = avatar;
        this.to_user = targetUser;
        this.show_join = 1;
        this.show_ready = 1;
        this.show_start = 1;
        this.show_kickout = 1;
    }
}
```

赠送礼物接口

Demo 参考

```
// 参数2为 giftCost，王国激战支持值1/2
SendGiftRequiredBean requiredBean = new SendGiftRequiredBean(new SendGiftRequiredBody(currentRoom.getUserId(), gift.getGiftType() % 2 + 1, 1));
GameEngine.getInstance().setOption(2, GameSetOptions.SEND_GIFT, new Gson().toJson(requiredBean));



public class SendGiftRequiredBean {
    private final String to;
    private final SendGiftPayLoad payload;

    public SendGiftRequiredBean(String to, int giftCost, int count) {
        this.to = to;
        this.payload = new SendGiftPayLoad(giftCost, count);
    }
}

public class SendGiftRequiredBody {
    private final String to;
    private final SendGiftPayLoad payload;

    public SendGiftRequiredBody(String to, int giftCost, int count) {
        this.to = to;
        this.payload = new SendGiftPayLoad(giftCost, count);
    }
}


public class SendGiftPayLoad {
    private final int giftCost;
    private final int count;

    public SendGiftPayLoad(int giftCost, int count) {
        this.giftCost = giftCost;
        this.count = count;
    }
}
```