**SDK for Androidインテグレーション仕様書**

|  |  |  |
| --- | --- | --- |
| ファイル状態：  [ ] ドラフト  [ ] 討議資料  [√] 正式発表 | ファイルID： | SDK for Androidインテグレーション仕様書 |
| バージョン： | 2.0 |
| 完成日付： | 2017-09-14 |

|  |  |  |
| --- | --- | --- |
| バージョン | 日付 | バージョン説明 |
| 1.0.0 | 2017-07-31 | ファーストバージョン |
| 2.0.0 | 2017-09-14 | セカンドバージン |

**目次**

一 SDK紹介 3

1. 概要 3

二 SDKマニフェストファイル 3

1 ファイル説明 3

2 開発環境構築 4

2.1 jarファイルを追加 4

2.2 resファイルを追加 5  
2.3 AndroidMainfest.xmlファイルを編集 5

2.3.1 必要なユーザー権限を付与 5

2.3.2 appidを 配置 6

三 メソッド呼び出し 7

3.1 必ず呼び出すメソッド 7

3.1.1 初期化 7

3.1.2 ログインログアウト イベントリスナー 7

3.1.3 ゲームスタート 8

3.1.4 ゲームを終了 9

3.2 SDKに組み込まれてるインタフェースを使う 9

3.2.1 ログイン 9

3.2.2 ホストがQRコードダイアログボックスリッスンを閉じる 9

3.2.3 ゲーム結果を提出 10

3.3 ゲームにはインタフェースが持ってる 10

3.3.1 ログイン 10

3.3.2 ゲーム支払い金額を取得 10

3.2.3 ルーム作成 11

3.2.4 QRコードをスキャン 11

3.2.5 ルームに加入した人の情報を取得 12

3.2.6 ルームへ加入 12

3.2.7 ゲーム結果を提出 12

3.2.8 ログアウト 13

# 一、SDK紹介

## 1、概要

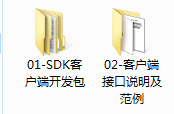
ゲームSDKは私達が開発した統計データサービスを提供するゲームプラグインです。

端末によっての統計と違って、SDKはゲーム内のユーザーアカウントに基づいて統計を行われます。1つの端末で複数のゲームアカウントが含まれてる可能性があるため、アカウントに基づいて統計するほうがリアルなユーザーデータが厳密に反応されます。

二、SDKマニフェストファイル

**1.ファイル説明**

**圧縮フォルダーを開くと：**



* **01-SDK****クライアント開発パッケージLibsフォルダー**

androidバージョンのSDKファイルはゲームに必要とされるjarパッケージでjarファイルが４つ含まれています。

* **02-クライアントインタフェース説明と例示**

SDKクライアントプログラムの例、ファイルで書かれているサンプルコードも含まれています。

* **アップデート説明**

**新しいバージョンのSDKアップデート説明**

**3、組み込み手順説明**

1. 責任者と連絡取って、プロダクトIDと登録認証KEYと課金KEYを取得します。一つのゲームは唯一のIDとKEYを持っています。
2. ローカル開発環境の構築について、SDKによってローカル開発環境を調整します。
3. 导入SDK的功能，开发者可参照文档中的示例代码进行功能的对接。SDKの機能を導入について、開発者はフォルダーの中のサンプルコードを参考して機能の組み込みを行われてください。

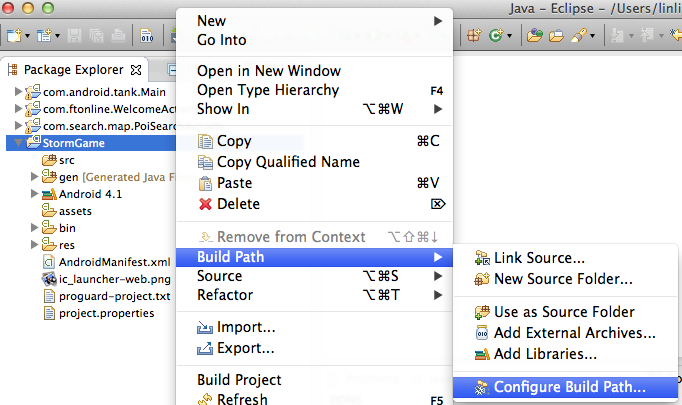
**2.開発環境構築**

## 2.1 jarファイルを追加

解凍して取得したSDKクライアント開発パッケージの中のLibsフォルダーをコーピーし、プロジェクトディレクトリにペーストして、プロジェクトをリフレッシュした後にjarファイルを追加します。

具体的手順：

Eclipseでプロジェクトを右クリックして、Build Path->Config Build Path->Add JARsボタン->さっきLibsフォルダーに入れたjarファイルを全部選択->OK。



## 2.2 resリソースファイルを追加

解凍して取得したSDKクライアント開発パッケージの中のresフォルダーをコーピーし、プロジェクトでのresフォルダーを差し替えます。

## 2.3AndroidMainfest.xmlファイルを編集

ゲームプロジェクト内に配置したAndroidMainfest.xmlファイルにSDKプロジェクトで必要となる権限とservice声明等を追加します。

### 2.3.1 必要なユーザー権限を付与

**必要なユーザー権限：**

<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
<uses-permission android:name="android.permission.Man" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
<uses-permission android:name="android.permission.SYSTEM\_ALERT\_WINDOW" />  
<uses-permission android:name="android.permission.GET\_TASKS" />  
  
<uses-permission android:name="android.permission.VIBRATE" />  
<uses-permission android:name="android.permission.CAMERA" />  
  
<uses-feature android:name="android.hardware.camera.autofocus" />

**必要なactivityを追加：**

<activity  
 android:name="com.yyjia.sdk.ScanQRCodeActivity"  
 android:configChanges="orientation|keyboardHidden|navigation"  
 android:theme="@android:style/Theme.Light.NoTitleBar" />  
<activity  
 android:name="com.yyjia.sdk.CaptureActivity"  
 android:configChanges="orientation|keyboardHidden|navigation"  
 android:theme="@android:style/Theme.Light.NoTitleBar" />

**必要なserviceを追加：**

<service android:name="com.yyjia.sdk.util.FloatViewService" />  
<service  
 android:name="com.yyjia.sdk.util.CxAccessService"  
 android:enabled="true"  
 android:exported="true"  
 android:permission="android.permission.BIND\_ACCESSIBILITY\_SERVICE">  
 <intent-filter>  
 <action android:name="android.accessibilityservice.AccessibilityService" />  
 </intent-filter>  
  
 <meta-data  
 android:name="android.accessibilityservice"  
 android:resource="@xml/game\_sdk\_accessible\_service\_config" />  
</service>  
<service android:name="com.yyjia.sdk.util.GameResultService" />

### 2.3.2 チャネルナンバーとappidを 配置

<meta-data

android:name="TOKYOGAME\_APPID"

android:value="20002" />

**パラメーター説明：**

|  |  |  |
| --- | --- | --- |
| **パラメーター** | **説明** | **デフォルト** |
| TOKYOGAME\_APPID | APPIDは唯一の識別子としてゲームプロダクトに配られます。（複数のゲームが含まれてる場合はその中の一つのゲームにAPPIDを配置すればいいです。） | N/A |

# 三 メソッド呼び出し

# 3.1 必ず呼び出すメソッド

## 3.1.1 初期化

**方法：**

mCenter = GMcenter.*getInstance*(mActivity);

mCenter.onCreate(mActivity);

**ゲームの起動クラスで、ゲームを初期化する時こうするとSDKがアクティブできます。**

**3.1.2ログイン、ログアウト、ログインウィンドウを閉じる、SDK初期化 イベントリスナー**

// 登录登出监听器  
mCenter.setLoginListener(new LoginListener() {  
 // 登录监听方法  
 @Override  
 public void loginSuccessed(String code) {  
 if (code == Information.*LOGIN\_SUSECCEDS*) {  
 // SDK登录成功 游戏(服务端)请求 登录验证  
 ToastUtil.*showShortToast*(mActivity, "LOGIN SUCCESS");  
 } else {  
 ToastUtil.*showShortToast*(mActivity, "LOGIN FAIL");  
 }  
 }  
  
 // 登出监听方法  
 public void logoutSuccessed(String code) {  
 if (code == Information.*LOGOUT\_SUSECCED*) {  
 // 账号 退出 游戏需要重启到 登录界面  
 ToastUtil.*showShortToast*(mActivity, "EXIT SUCCESS");  
 } else {  
 ToastUtil.*showShortToast*(mActivity, "EXIT FAIL");  
 }  
 }  
  
 @Override  
 public void logcancelSuccessed(String code) {  
 if (code == Information.*LOGCANCEL\_SUSECCED*) {  
 ToastUtil.*showShortToast*(mActivity, "CANCEL LOGIN");  
 }  
 }  
});

// 初始化监听方法

mCenter.setInitListener(new InitListener() {  
  
 @Override  
 public void initSuccessed(String code) {  
 // 监听SDK初始化 如果初始化成功 可以选择适合时间登陆  
 if (code == Information.*INITSUCCESSEDS*) {  
 mCenter.checkLogin();  
 } else {  
 ToastUtil.*showShortToast*(mActivity, "INIT FAIL");  
 }  
 }  
});

## 3.1.3 ゲームスタート(ゲーム開始の時に呼び出す)

mCenter.startGame();

## 3.1.4 ゲームを終了

mCenter.exitGame();

// 退出游戏监听方法

mCenter.setExitGameListener(new ExitGameListener() {  
 @Override  
 public void onSuccess() {  
 mCenter = null;  
 mCenter.exitGame();  
 System.*exit*(0);  
 }  
  
 @Override  
 public void onFailure() {  
 ToastUtil.*showShortToast*(mActivity, "Exit Failure");  
 }  
});

**3.2 SDKに組み込まれてるインタフェースを使う（ゲームは自身の状況によって組み込む）**

**3.2.1ログイン**

mCenter.checkLogin();

**3.2.2 ホストがQRコードダイアログボックスリッスンを閉じる**

mCenter.setCloseQrCodeListener(new CloseQrCodeListener() {  
 @Override  
 public void close() {  
 ToastUtil.*showShortToast*(mActivity, "Start Game");  
 mCenter.startGame();  
 }  
});

**3.2.3 ゲーム結果を提出**

*/\*\*  
 \* 提交游戏结果  
 \*  
 \** ***@param*** *score 游戏结果（分数）  
 \** ***@param*** *isCustomSubmitResultInterface 是否使用SDK自带的界面 true 不使用SDK自带的界面(包括显示游戏分数界面和显示游戏最终结果的界面)  
 \*/*

mCenter.submitGameResult(int Score, false);

# 3.3 ゲームにはインタフェースが持ってる（ゲームは自身の状況によって組み込む）

**3.3.1 ログイン**

// 是否自定义登录界面(有则传入true，否则传入false)  
mCenter.setCustomLoginInterface(true);

*/\*\*  
 \* 登录（使用游戏自己的登录界面）  
 \*  
 \** ***@param*** *userName 账号  
 \** ***@param*** *password 密码  
 \*/*

mCenter.login(String userName, String password);

**3.3.2 ゲーム支払い金額を取得**

*/\*\*  
 \* 获取游戏支付金额  
 \*  
 \** ***@param*** *context  
 \** ***@param*** *appId 游戏ID  
 \*/*

mCenter.getGameCost(Context context, String appId);

// 获取游戏支付金额监听

mCenter.setGetGameCostListener(new GetGameCostListener() {  
 @Override  
 public void onSuccess(String money) {  
 Utils.*E*("money = " + money);  
 }  
  
 @Override  
 public void onFailure(String errMsg) {  
  
 }  
});

**3.3.3 ルーム作成**

mCenter.createRoom(mActivity);

// 创建房间监听  
mCenter.setCreateRoomListener(new CreateRoomListener() {  
 @Override  
 public void onSuccess(String result) {  
 Utils.*E*("setCreateRoomListener onSuccess = " + result);  
 }  
  
 @Override  
 public void onFailure(String errMsg) {  
 ToastUtil.*showShortToast*(mActivity, errMsg);  
 }  
});

**3.3.4 QRコードをスキャン**

mCenter.scanQrCode(mActivity);

// 扫描二维码监听

mCenter.setScanQrCodeListener(new ScanQrCodeListener() {  
 @Override  
 public void showResult(String result) {  
 Utils.*E*("result = " + result);  
 }  
});

**3.3.5 ルームに加入した人の情報を取得（ルームに加入した人の情報が要らないとしてもこのメソッドを組み込む）**

mCenter.setJoinRoomUserListListener(new JoinRoomUserList() {  
 @Override  
 public void result(String result) {  
 Utils.*E*("setJoinRoomUserListListener result =" + result);  
 }  
});

**3.3.6 ルームへ加入**

mCenter.joinRoom(mActivity, result);

// 加入房间监听

mCenter.setJoinRoomListener(new JoinRoomListener() {  
 @Override  
 public void onSuccess(String result) {  
 Utils.*E*("JoinRoomListener onSuccess = " + result);  
 ToastUtil.*showShortToast*(mActivity, "Join Room Success");  
 }  
  
 @Override  
 public void onFailure(String errMsg) {  
 ToastUtil.*showShortToast*(mActivity, errMsg);  
 }  
});

**3.3.7 ゲーム結果を提出**

*/\*\*  
 \* 提交游戏结果  
 \*  
 \** ***@param*** *score 游戏结果（分数）  
 \** ***@param*** *isCustomSubmitResultInterface 是否使用SDK自带的界面 true 不使用SDK自带的界面(包括显示游戏分数界面和显示游戏最终结果的界面)  
 \*/*

mCenter.submitGameResult(int Score, true);

mCenter.setSubmitGameResultListener(new SubmitGameResultListener() {  
 @Override  
 public void showGameResult(String result) {  
 Utils.*E*("showGameResult = " + result);  
 }  
});

**3.3.8 ログアウト**

mCenter.logout();