**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： | 5.2 |
| 完成日付： | 2017-10-25 |

|  |  |  |
| --- | --- | --- |
| バージョン | 日付 | バージョン説明 |
| 1.0.1 | 2015-02-25 | ファーストバージョン |
| 2.0 | 2015-07-28 |  |
| 2.1 | 2016-04-11 |  |
| 3.0 | 2016-05-05 |  |
| 3.1 | 2016-06-28 |  |
| 3.2 | 2016-07-20 |  |
| 4.0 | 2017-02-10 |  |
| 4.1 | 2017-03-13 |  |
| 5.0 | 2017-08-24 |  |
| 5.1 | 2017-09-29 |  |
| 5.2 | 2017-10-25 | 最新バージョン |

**目次**

一 SDK紹介 3

1.1 概要 3

1.2 SDKマニフェストファイル 3

1.3 組み込み手順説明 4

二 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 チャネル番号とappidを 配置 7

三 SDK組み込み 8

3.1 SDK初期化 8

3.2 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 8

3.3 ログインインタフェース 10

3.4 アカウントを切り替えやログアウト 10

3.5 課金インタフェース 10

3.6 sessionidを取得 11

3.7 ユーザーキャラクター情報を提出 11

3.8 ゲームを終了 11

3.9 フローティングウィンドウの表示を隠す（ログイン成功する時に自動的にフローティングウィンドウを呼び出す） 12

3.10以下の声明で相応のメソッドを実行 12

# 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。

統計範囲：

* 新規ユーザー

新規ユーザーのアカウントの数を統計出来ます。

* アクティブユーザー

同じ末端でユーザーがネット環境で登録した回数

* チャネルデータ

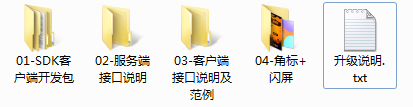
異なるディストリビューションチャネルを基づいてユーザーデータを分析

* バージョンデータ

ゲームの各バージョンでユーザーデータ分析を行います

**1.2 SDKマニフェストファイル**

**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8つ含まれています。

* **02-サーバーサイドインタフェース説明**

サーバーサイドインタフェース説明ドキュメント

* **03-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。

* **04-隅にあるアイコン+スプラッシュスクリーン**

隅にあるアイコン+スプラッシュスクリーンドキュメント

* **アップデート説明**

**新しいバージョンのSDKアップデート内容**

**1.3 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. 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。

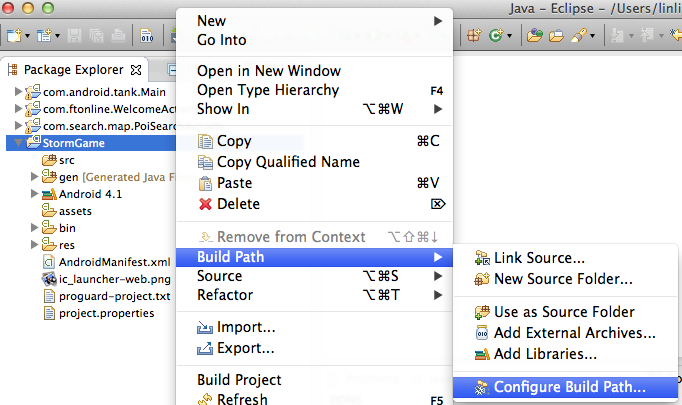
**二、開発環境構築**

## 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。

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。

## 2.3 AndroidMainfest.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.CALL\_PHONE" />  
<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.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.GET\_TASKS" />  
<uses-permission android:name="android.permission.WRITE\_APN\_SETTINGS" />  
<uses-permission android:name="android.permission.INSTALL\_PACKAGES" />  
<uses-permission android:name="android.permission.SYSTEM\_ALERT\_WINDOW" />  
<uses-permission android:name="android.permission.READ\_LOGS" />  
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  
<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

**必要なactivityを追加：**

<activity  
 android:name="com.yyjia.sdk.PayActivity"  
 android:configChanges="orientation|keyboardHidden|navigation"  
 android:screenOrientation="behind"  
 android:theme="@android:style/Theme.Light.NoTitleBar" />  
<activity  
 android:name="com.yyjia.sdk.HeePayActivity"  
 android:configChanges="orientation|keyboardHidden|navigation"  
 android:theme="@android:style/Theme.Light.NoTitleBar" />  
<activity  
 android:name="com.alipay.sdk.app.H5PayActivity"  
 android:configChanges="orientation|keyboardHidden|navigation"  
 android:exported="false"  
 android:screenOrientation="behind" />  
<activity  
 android:name="com.alipay.sdk.auth.AuthActivity"  
 android:configChanges="orientation|keyboardHidden|navigation"  
 android:exported="false"  
 android:screenOrientation="behind" />  
<activity  
 android:name="com.heepay.plugin.activity.WeChatNotityActivity"  
 android:configChanges="orientation|keyboardHidden|screenSize"  
 android:screenOrientation="behind"  
 android:theme="@android:style/Theme.NoDisplay" />

**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>

### 2.3.2 チャネルナンバーとappidを 配置

<meta-data

android:name="TOKYOGAME\_APPID"

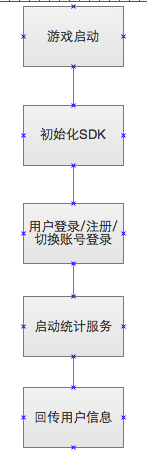
android:value="20002" />

**パラメーター説明：**

|  |  |
| --- | --- |
| **パラメーター** | **説明** |
| android:name | TOKYOGAME\_APPID は固定値 |
| android:value | APPIDは唯一の識別子として提携ゲームプロダクトに配られます |

# 三、SDK組み込み

**SDK作業プロセス**



ゲーム起動-SDK初期化-ユーザーログイン/サインイン/アカウント切り替え-統計サービスを起動-ユーザー状態を転送

**SDK作業プロセス説明：**

* **ゲーム起動すると、SDKを初期化します**
* **ゲーム内のユーザー情報が変わる時に、例えばログイン成功、サインイン成功、アカウントを切り替えてログインした後、ユーザー情報をSDKへ送信します。**

## 3.1 SDK初期化

**方法：**

mCenter = GMcenter.*getInstance*(Context context);

mCenter.onCreate(Activity activity);

**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；**

**3.2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**

Center.setLoginListener(LoginListener loginListener);

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*(MainActivity.this, "登录成功");  
 mCenter.showFloatingView(MainActivity.this);  
 } else {  
 ToastUtil.*showShortToast*(MainActivity.this, "登录失败");  
 }  
 }  
  
 // 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*(MainActivity.this, "退出游戏成功");  
 } else {  
 ToastUtil.*showShortToast*(MainActivity.this, "退出游戏失败);  
 }  
 }  
  
 @Override  
 public void logcancelSuccessed(String code) {  
 if (code == Information.*LOGCANCEL\_SUSECCED*) {  
 ToastUtil.*showShortToast*(MainActivity.this, "取消登录);  
 }  
 }  
});  
  
mCenter.setInitListener(new InitListener() {  
  
 @Override  
 public void onSuccess(int code) {  
 mCenter.checkLogin();  
 }  
  
 @Override  
 public void onFailure(int code, String errMsg) {  
 if (code == Information.*CODE\_GET\_APP\_ID\_ERROR*) {  
 // 获取AppId失败  
 } else if (code == Information.*CODE\_NET\_TIME\_OUT*) {  
 // 网络连接超时  
 } else if (code == Information.*CODE\_SERVER\_RETURN\_DATA\_ERROR*) {  
 // 服务端返回数据错误  
 }  
 ToastUtil.*showShortToast*(MainActivity.this, errMsg);  
 }  
});

// 游戏退出监听方法(当调用exitDialog()方法，必须设置此监听)

mCenter.setExitGameListener(new ExitGameListener() {  
 @Override  
 public void onSuccess() {  
 ToastUtil.*showShortToast*(MainActivity.this, "Exit Game Success");  
 }  
  
 @Override  
 public void onFailure() {  
 ToastUtil.*showShortToast*(MainActivity.this, "Exit Game Failure");  
 }  
  
 @Override  
 public void onCancel() {  
 ToastUtil.*showShortToast*(MainActivity.this, "Exit Game Cancel");  
 }  
});

## 3.3 ログインインタフェース

mCenter.checkLogin();

## 3.4 アカウントの切り替えやログアウト

mCenter.logout();

## 3.5 課金インタフェース

パラメーター説明：

Float money=0.01f;

String productname=“商品名”;

String serverId =”サーバー番号”;

String charId=”キャラクター番号”;

String cporderId=”CPオーダー番号”;

String callbackInfo=”拡張情報 課金成功した後にゲームサーバーへ送信する”;

PayListener支払いイベントリスナー

実例：

Center.pay(Context context, float rmb, String productname,  
 String serverId, String charId, String cporderid,  
 String callbackInfo, PayListener payListener);

mCenter.pay(MainActivity.this, 0.01f, "test", "1", "1", System.*currentTimeMillis*() + "", "123456", new PayListener() {  
 @Override  
 public void paySuccessed(String code, String cporderid) {  
 if (code == Information.*PAY\_SUSECCED*) {  
 ToastUtil.*showShortToast*(MainActivity.this, "PAY SUCCESS");  
 } else {  
 ToastUtil.*showShortToast*(MainActivity.this, "PAY FAIL");  
 }  
}  
  
 @Override  
 public void payGoBack() {  
 Utils.*E*("payGoBack");  
 ToastUtil.*showShortToast*(MainActivity.this, "PAY BACK");  
 }

});

## 3.6 sessionidを受け取る

Center.getSid();

urlencodeした sessionidを返す

## 3.7ユーザーキャラクター情報を提出

パラメーター説明：

String serverId=”サーバーID”;

String serverName=”サーバー名”;

String roleId=”キャラクターID”;

String roleName=”キャラクター名”;

String roleLevel=”キャラクターレベル”;

String roleCTime=”キャラクター作成時間　タイムスタンプ”;

// 当角色信息变动的时候调用这个接口（例如角色创建、升级等情况）

mCenter.submitRoleInfo(String serverId, String serverName,  
 String roleId, String roleName, String roleLevel, String roleCTime);

## 3.8 ゲームを終了

// 如果游戏没有自己的退出对话框

mCenter.exitDialog(MainActivity.this);

// 如果游戏有自己的退出对话框

// 只要游戏能确保正常退出的时候能走onDestroy()方法,就在onDestroy()的方法里的super.onDestrot()之前调用，不能的话再游戏确认退出的时候调用

mCenter.exitGame(MainActivity.this);

## 3.9 フローティングウィンドウの表示を隠す（ログイン成功する時に自動的にフローティングウィンドウを呼び出す）

mCenter.hideFloatingView(MainActivity.this);

mCenter.showFloatingView(MainActivity.this);

// 在onResume里调用

@Override  
protected void onResume() {  
 super.onResume();  
 if (mCenter != null) {  
 mCenter.onResume(this);  
 mCenter.showFloatingView(MainActivity.this);  
 }  
}

// 在onPause里调用  
@Override  
protected void onPause() {  
 super.onPause();  
 if (mCenter != null) {  
 mCenter.hideFloatingView(this);  
 }  
}  
  
// 在onStop里调用  
@Override  
protected void onStop() {  
 super.onStop();  
 if (mCenter != null) {  
 mCenter.onStop(MainActivity.this);  
 }  
}

## 3.10 以下の声明で相応のメソッドを実行

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 mCenter.onCreate(MainActivity.this);  
}

@Override  
protected void onStart() {  
 super.onStart();  
 mCenter.onStart(MainActivity.this);  
}

@Override  
protected void onResume() {  
 super.onResume();  
 if (mCenter != null) {  
 mCenter.onResume(MainActivity.this);  
 }  
}

@Override  
protected void onPause() {  
 super.onPause();  
 if (mCenter != null) {  
 mCenter.onPause(MainActivity.this);  
 mCenter.hideFloatingView(this);  
 }  
}

@Override  
protected void onStop() {  
 super.onStop();  
 Utils.*E*("onStop");  
 if (mCenter != null) {  
 mCenter.onStop(MainActivity.this);  
 }  
}

@Override  
protected void onRestart() {  
 super.onRestart();  
 mCenter.onRestart(MainActivity.this);  
}

@Override  
protected void onDestroy() {  
 if (mCenter != null) { // 必须在super.onDestroy()之前调用  
 mCenter.onDestroy(this);  
 }  
 super.onDestroy();  
}

@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 if (mCenter != null) {  
 mCenter.onActivityResult(this, requestCode, resultCode, data);  
 }  
}

**提案：**

ゲームのtargetSdkVersionのバージョン番号はなるべくAndroid 6.0バージョン(23)以下にしてください，つまりtargetSdkVersion<23,　6.0権限問題のせいで必要以上の権限判断を行うことを避けられます。

ゲームのActivity横縦画面の切り替えの判断をしないなら、AndroidManifestファイルで相応のActivityで以下のコードを追加してください

android:configChanges="orientation|keyboardHidden|navigation|screenSize"