SDK for Android manual

File status	File name :	SDK for Android manual
[]Draft	Version:	4.1
[ ]Discussion		
[√]Formal issuance	Complet date:	2017-03-13

Version	Date	Description
1.0.1	2015-02-25	First Edition
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	

Table of Contents

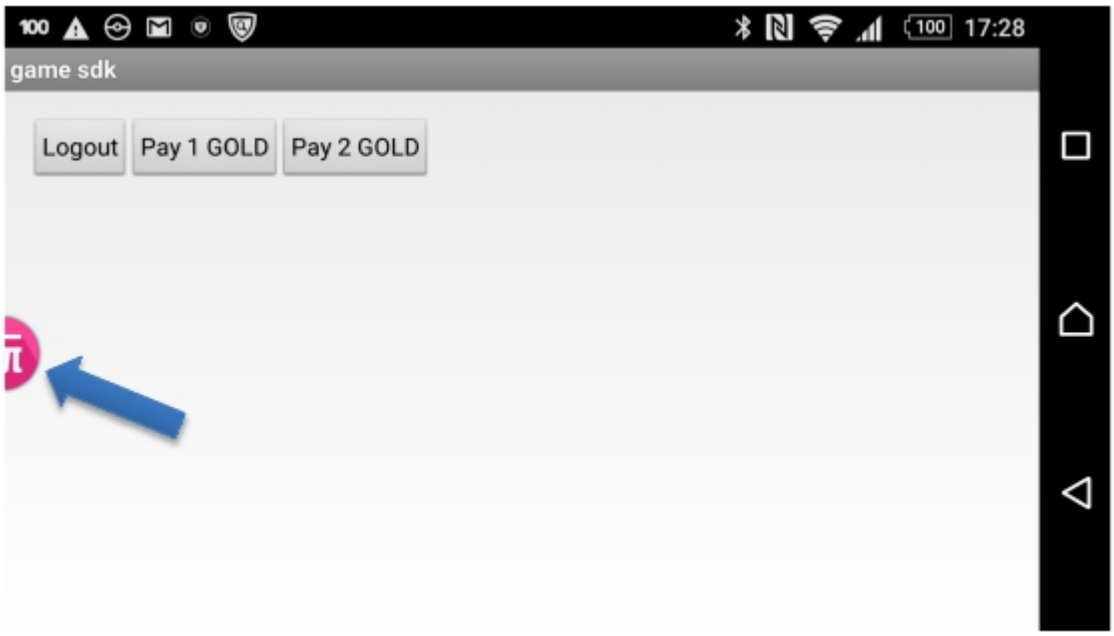
1 SDK Introduction 3
1. Introduction 3
2.1 Adding JAR Files 5
2.2 Adding Resource Files 5
2.3 Modify AndroidMainfest.xml 5
2.3.1 Adding permissions 5
2.3.2 Setting appid 7
3 Add aggregate function of SDK 7
3.1 Initializing aggregate function 8
3.2 Event listener of login/logout 8
3.3 Registrations Interface 9
3.4 Switch User ID or Delete 10
3.5 Interface of point charge 10
3.6 Getting sessionid 11
3.7User Information 11
3.8 Game Over 12
3.9 Hiding Floating Window 12

1) SDK Introduction

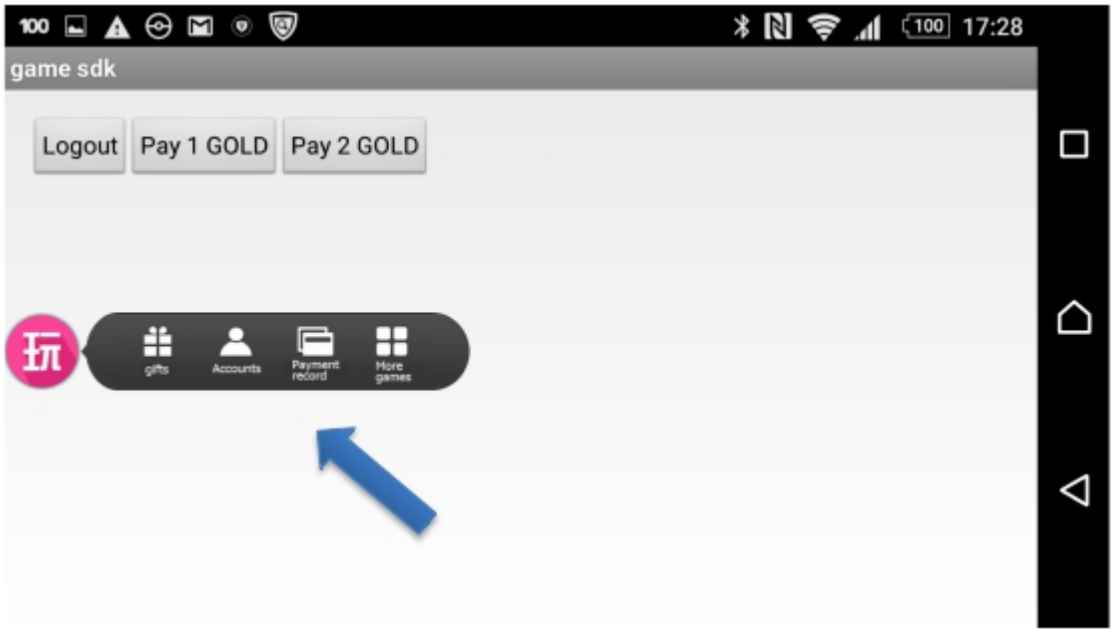
1.1 Outline

Game SDK is a plug-in for ser account management, point consumption, statistics service.

After implements the SDK, the following buttons will appear on the screen of the application.



When the user touches this button, the following menu appears, you can log in and browse history.



You can change the design of buttons and menus by modifying the SDK source.

Unlike general per-device statistics, the SDK is based on statistics for the game ID of the user in the game. There may be multiple game IDs in one facility at the same time. This SDK can more accurately reflect the user situation of the game from the game ID statistics.

Statistics are in the following range:

- Add a new user

  You can get the number of newly added user IDs.

- Active User

  You can get the number of times users have connected to the Internet on the same terminal.

- Sub channel data

  Analyze user data using each sub channel.

- Data for each game version

  Analyze user data for different versions.

2) **SDK** File List

Content included in decompressed file：

l 01- Client SDK

There are seven JAR files in Android version SDK file, all are JAR packages required for games.

l 02- Server API

Server side interface description document

l 03- Client Demo

Sample program for SDK for client and necessary sample code.

l 04- Design Material

Balloon and Splash

3 、Steps to implement SDK

The main steps to implement the SDK are as follows

1） Please contact your account manager and obtain product ID, registration verification key, payment key. These are issued for each game.

2） We will build a development environment locally. A local environment suitable for SDK incorporation is required。

3）

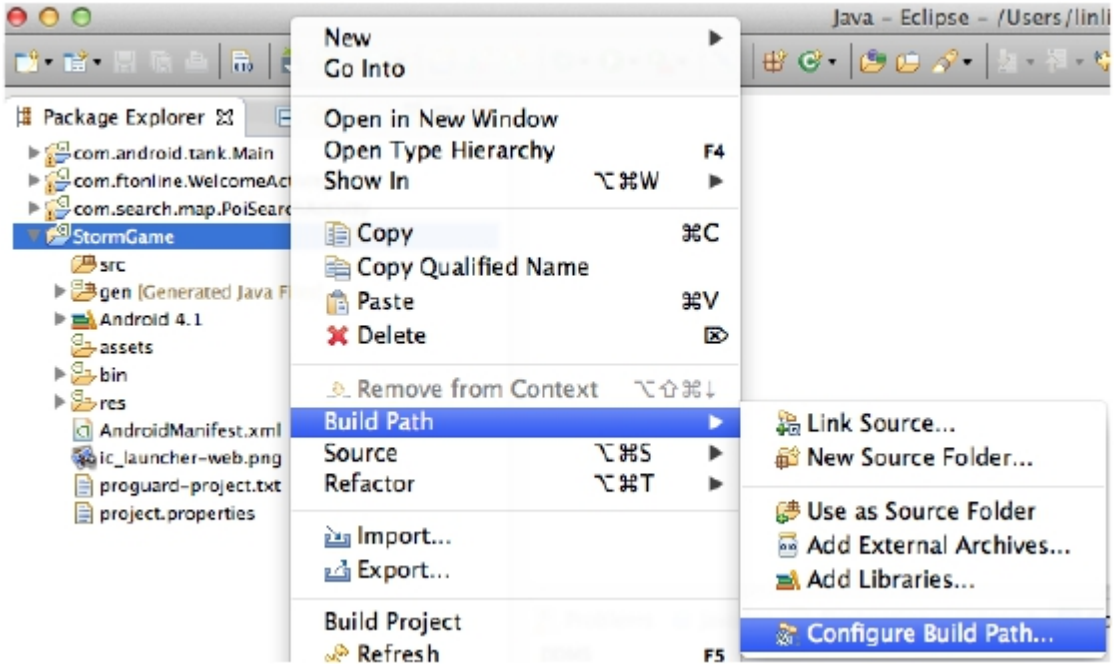
Developers incorporate features that correspond to the code shown in the development documentation.

一 、Construction of development environment

2.1 Adding JAR Files

Copy the Libs folder of the SDK client development kit that appears after decompressing the compressed file into the project of the game to be distributed, reload it, and install the JAR file. The concrete steps are as follows.

Right click on the project in Eclipse, select Add JARS on the screen displayed under Build Path-> Config Build Path, add all the JAR files in the Libs folder to the program, and click OK.



2.2 Adding Resource Files

Copy the Res folder in the SDK client development kit that appears after decompressing the compressed file and replace it with the res folder in the delivered project file.

2.3 Modify AndroidMainfest.xml

Add the necessary authority and service declaration for the SDK program to the AndroidMainfest.xml file in the game project.

2.3.1 Adding permission

Required User Permissions:

<uses-permissionandroid:name="android.permission.INTERNET"/>

<uses-permissionandroid:name="android.permission.ACCESS_WIFI_STATE"/>

<uses-permissionandroid:name="android.permission.CHANGE_WIFI_STATE"/>

<uses-permissionandroid:name="android.permission.ACCESS_NETWORK_STATE"/>

<uses-permissionandroid:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permissionandroid:name="android.permission.READ_PHONE_STATE"/>
<uses-permissionandroid:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<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"/>

Add necessary activity：

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

Add necessary service：

<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:resource="@xml/game_sdk_accessible_service_config" />
</service>

2.3.2 Setting channel number and appid

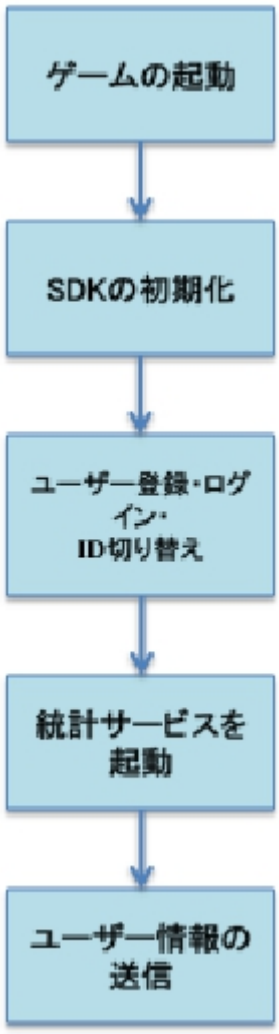
<meta-data
android:name="TOKYOGAME_APPID"
android:value="20002" />

Description of the parameters:

Parameter	Description	Default
TOKYOGAME_APPID	Unique ID for each game	N/A

三 Add aggregate function of SDK

Once the development environment is completed, you can start the SDK statistical function.



Working process of SDK

Explanation of statistical work process of SDK ：

I Initialize SDK at game start

I When user information changes in the game, for example, when log in success, register success, ID change, start SDK statistics service and add user information to SDK I will send.

3.1 Initialization of SDK statistics service

Method：

GMcenter. getInstance1 (Context context);
mCenter.onCreate(Activity)

Please activate the SDK in this way at the timing of the initialization of the game, such as when the game starts.

3.2 Login, logout, login screen closure, SDK initialization, event listener

```
mCenter.setLoginListener(LoginListener listener);
mCenter.setLoginListener(new LoginListener() {
                @Override
                publicvoid loginSuccessed(String code) {
                // TODO Auto-generated method stub
                    if (code==Information.LOGIN_SUSECCEDS){
                        String sid=mCenter.getSid();// getting sessionid

                        Toast.makeText(MainActivity.this , "Login Success",
                    Toast.LENGTH_LONG).show();
                    }else {
                        Toast.makeText(MainActivity.this , "Login Failure",
                    Toast.LENGTH_LONG).show();
                    }
                }

                @Override
                publicvoid logcancelSuccessed(String code) {
```

```
                    if (code==Information.LOGCANCEL_SUSECCED){
                        Toast.makeText(MainActivity.this , "Login
cancel",Toast.LENGTH_LONG).show();
                    }
                }

                @Override
                publicvoid logoutSuccessed(String code) {
                    // TODO Auto-generated method stub
                        if (code==Information.LOGOUT_SUSECCED){
                            String sid=mCenter.getSid(); // get sessionid


                            Toast.makeText(MainActivity.this , "Logout Success",
                    Toast.LENGTH_LONG).show();
                        }else {
                            Toast.makeText(MainActivity.this , "Logout Failure",
                        Toast.LENGTH_LONG).show();
                        }
                    }
                    });

mCenter.setInitListener(new InitListener() {

                    @Override
                    publicvoid initSuccessed(String code) {
                        if (code==Information.INITSUCCESSEDS){
                            // When initialization is successful or registration is required, call this interface
                            //mCenter.checkLogin();
                        }
                    }
                    });
```

3.3 Registration interface

```
GMcenter.checkLogin();
```

3.4 Switch account or log off

GMcenter.logout();

3.5 Point Charge Interface

Parameter Description:

Float money=0.01f;

String productname="Product name";

String serverId ="Server ID";

String charId="Character ID";

String cporderId="CP Order Number";

String callbackInfo=" **Additional information Send to server when charge is successful**";

PayListener **Payment Event listener**

An example:

```
GMCenter.pay(MainActivity.this , money, productname,String serverId,String charId,String
cporderId,String callbackInfo,new PayListener() {

                @Override
                publicvoid paySuccessed(String code,String cporderid) {

                    if (code==Information.PAY_SUSECCED){
                        Toast.makeText(MainActivity.this , cporderid+"Charge success ",
Toast.LENGTH_LONG).show();
                    }else {
                        Toast.makeText(MainActivity.this , "Charge failure ",
    Toast.LENGTH_LONG).show();
                    }
                }

                @Override
                publicvoid payGoBack() {
```

```
                                    Toast.makeText(MainActivity.this , "Return to charge",
Toast.LENGTH_LONG).show();
                                                        }
                        });
```

3.6 Getting sessionid

GMCenter.getSid();
Getting urlencoded sessionid

3.7 Providing user capital information

Parameter Description:

String serverId=”Server ID”;

String serverName=”Server Name”;

String roleId=”Character ID”;

String roleName=”Character Name”;

String roleLevel=”Character Level”;

String roleCTime=” Time when the character was made, timestamp”;

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

■3.8 Game over

```
mCenter.exitGame();
```

■ 3.9 Hiding Floating Window

```
mCenter.hideFloatingView();
mCenter.showFloatingView();
Call from onResume
if (mCenter != null) {
mCenter.showFloatingView();
  }
Call from onStop
if (mCenter != null) {
mCenter.hideFloatingView();
}
```