Anti-Cheat Expert

Integration Guidelines

Product Documentation

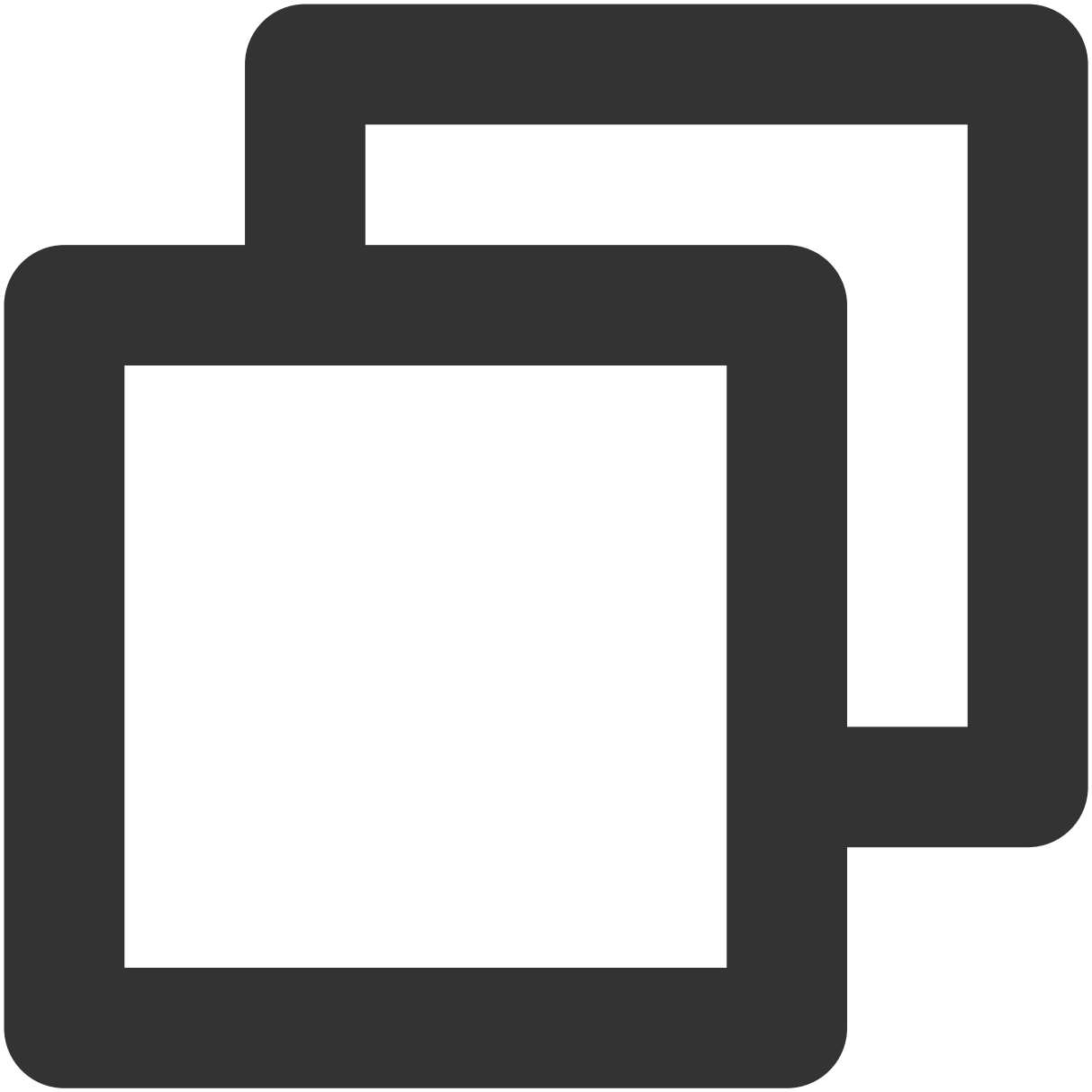
Contents

Integration Guidelines
Integration Guidelines for C

Last updated：2023-12-08 17:48:02

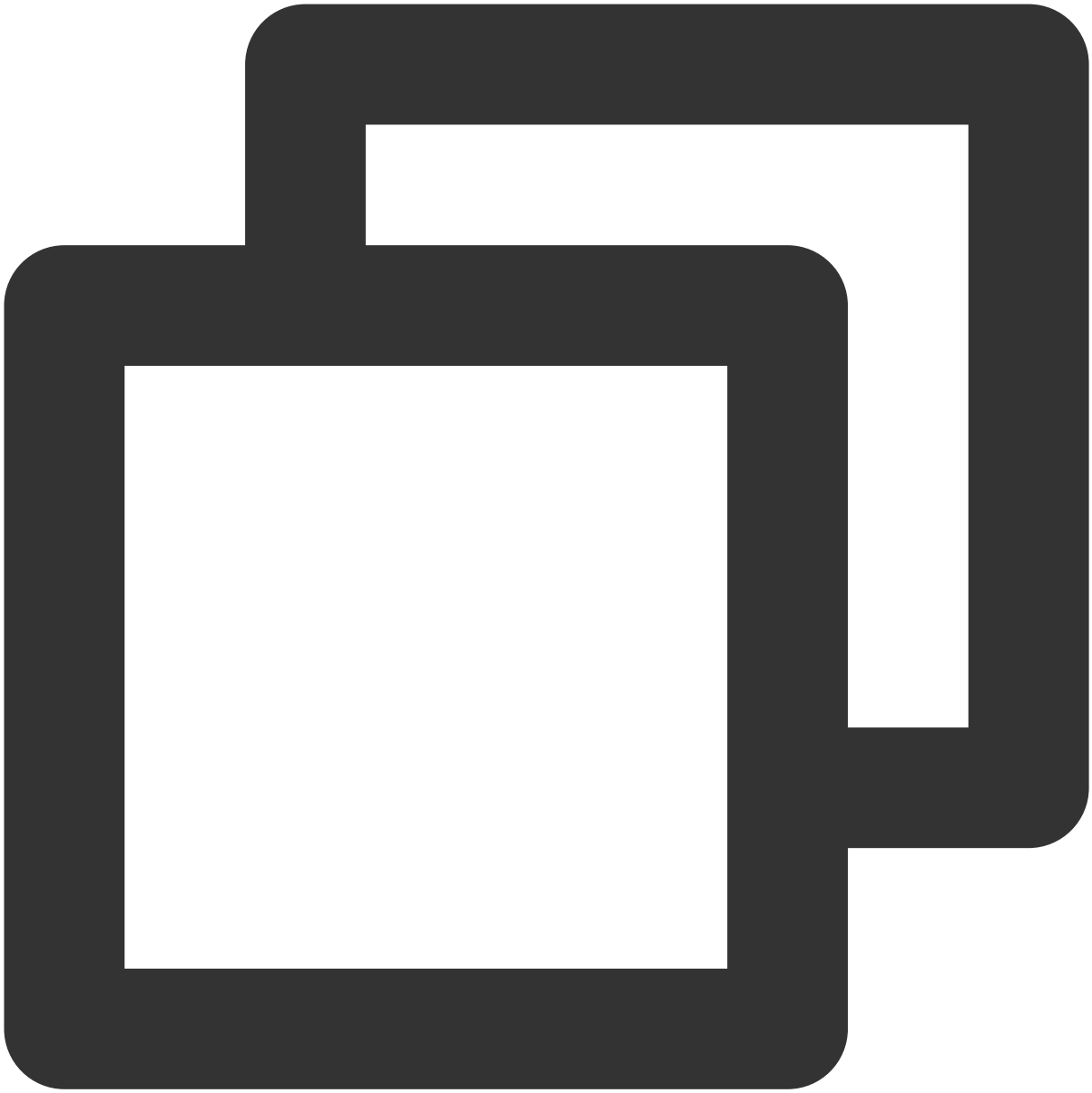
Preparations

Developers need to complete the following steps when integrating the security SDK:
1.1 Copy the SDK dynamic library to the specified project directory associated with the game platform and the CPU architecture.
1.2 Call the SDK API function based on the user's login information.
1.3 Verify whether the SDK is integrated correctly.
The following files are required for the integration of the security SDK to Android OS written in C/C++:



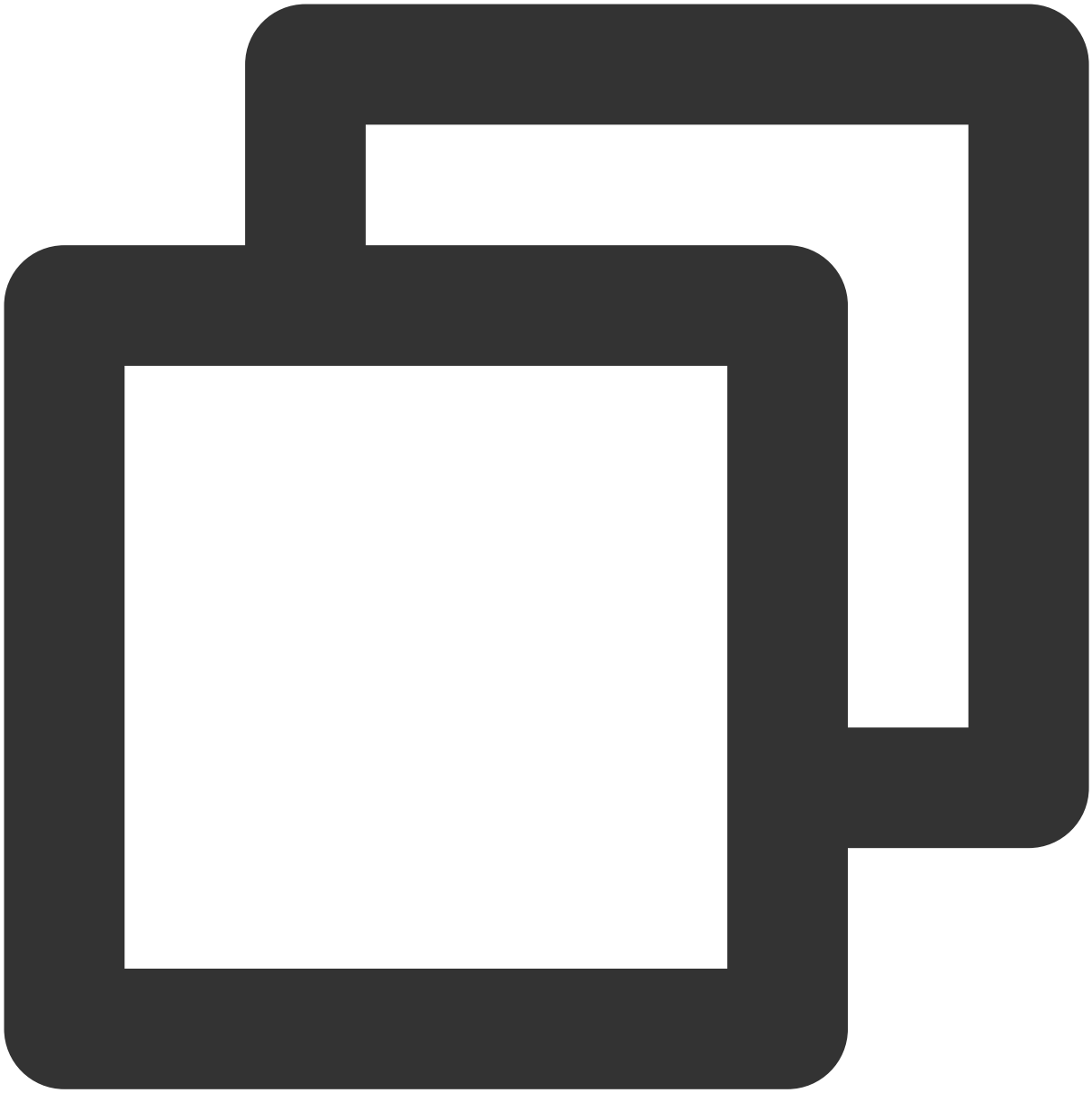
```
tp2.jar
tp2_sdk.h
tss_sdt.h,tss_sdt_ex.h (Security data type is optional. For more information, see "
libtersafe2.so
```

Permissions required:



```
<uses-permission android:name="android.permission.ACCESS_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.GET_TASKS" />
<uses-permission android:name="android.permission.INTERNET" />
```

SDK API functions:

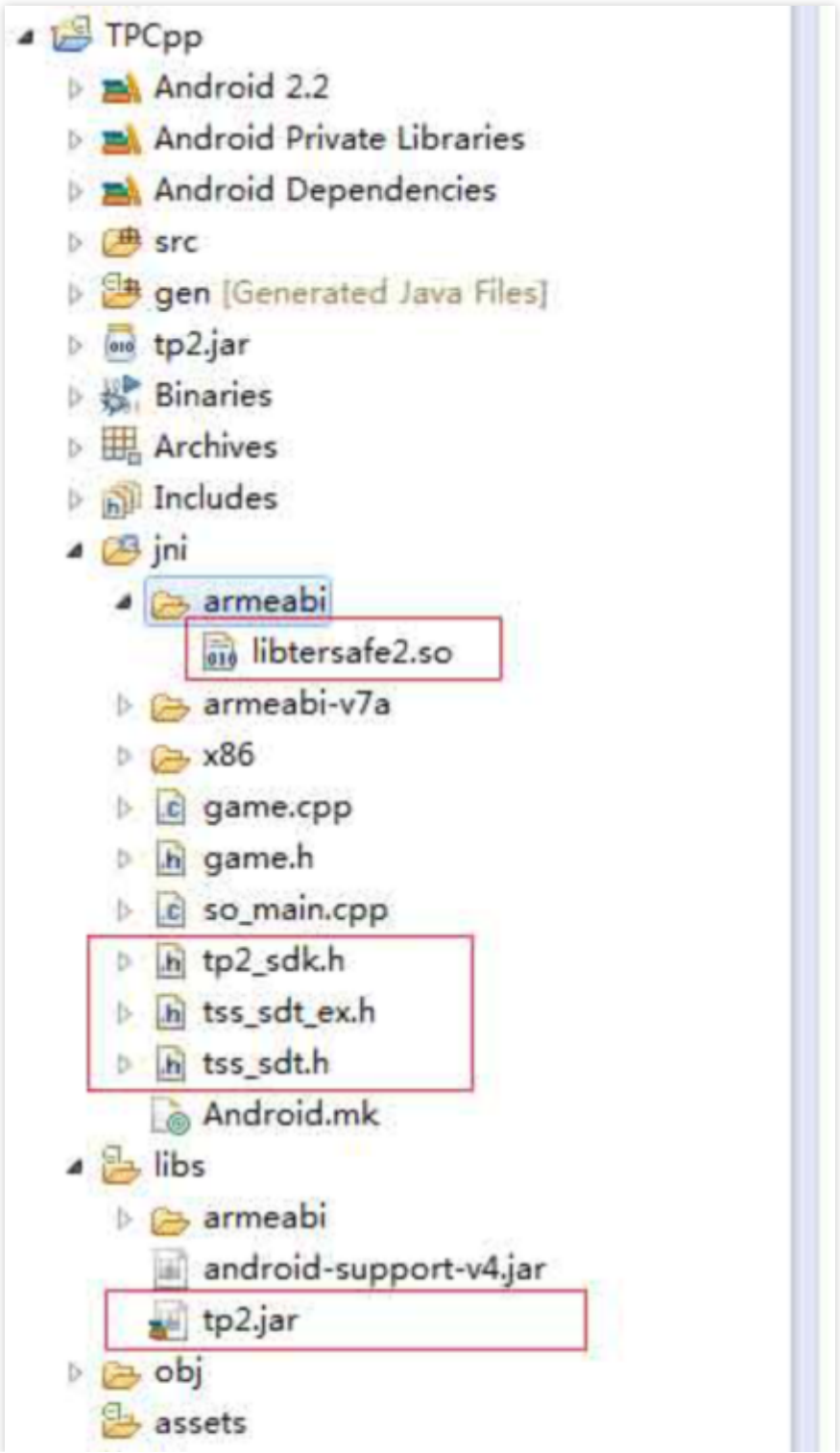


```
Initialization API: tp2_sdk_init_ex
User login API: tp2_setuserinfo
API for switching between foreground and background: tp2_setgamestatus
```

Adding SDK Files to the Project

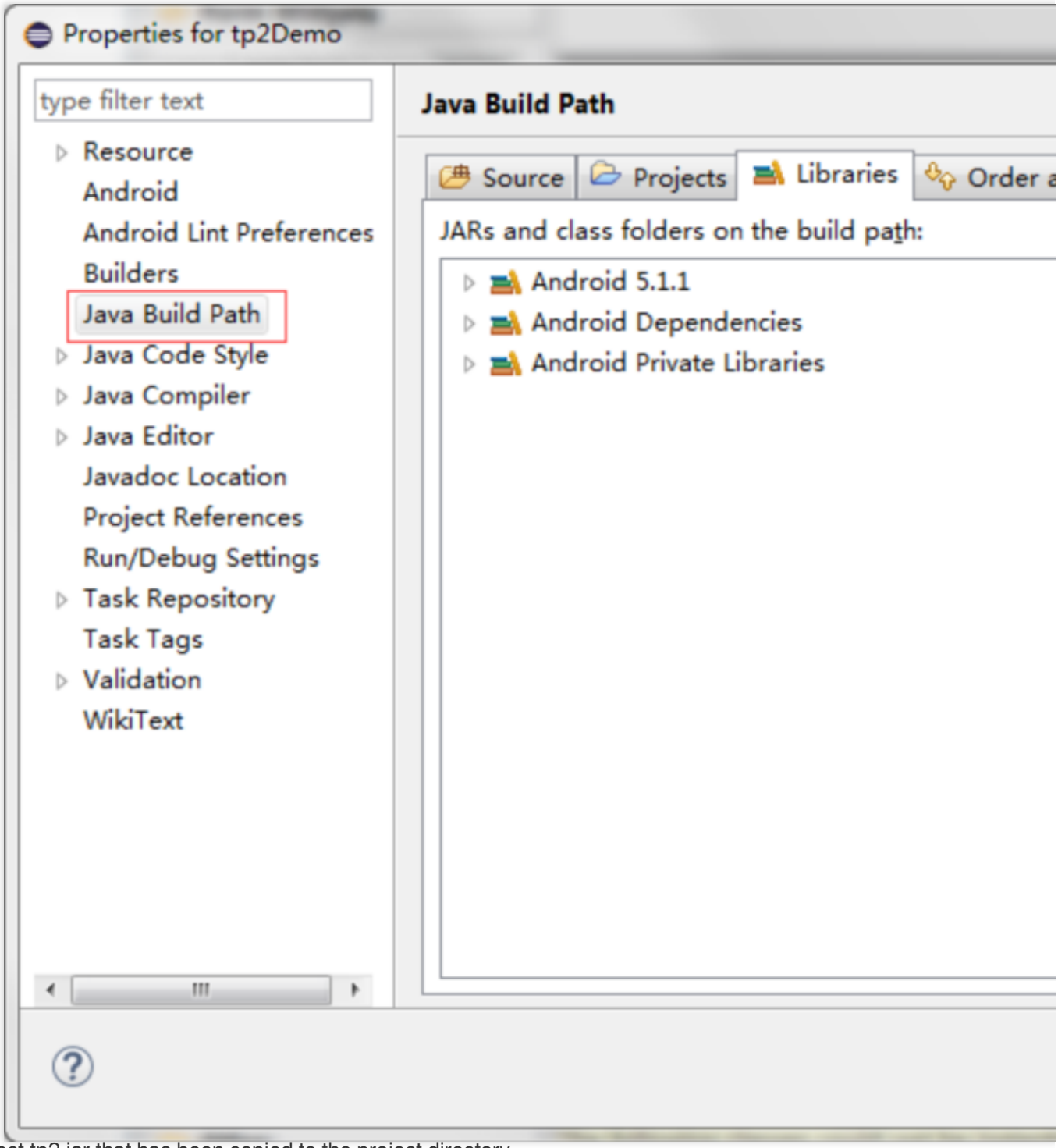
Add files

1. Copy the tp2.jar file from the sdk/android/c directory to the libs directory in the android project directory.
2. Copy the tp2_sdk.h file from the sdk/android/c directory to the jni directory in the android project directory.
3. Copy the tss_sdt.h and tss_sdt_ex.h files from the sdt/c++ directory to the jni directory in the android project directory (Optional. For more information, see "Guide to Integrating C++ Security Data Types.doc".).
4. Copy the folder (containing the libtersafe2.so file) named after the CPU architecture from the sdk/android/c/lib directory to the directory with the corresponding .so file in the jni directory of the android project directory, eg. jni/armeabi and jni/x86. Do not copy unsupported CPU architectures.

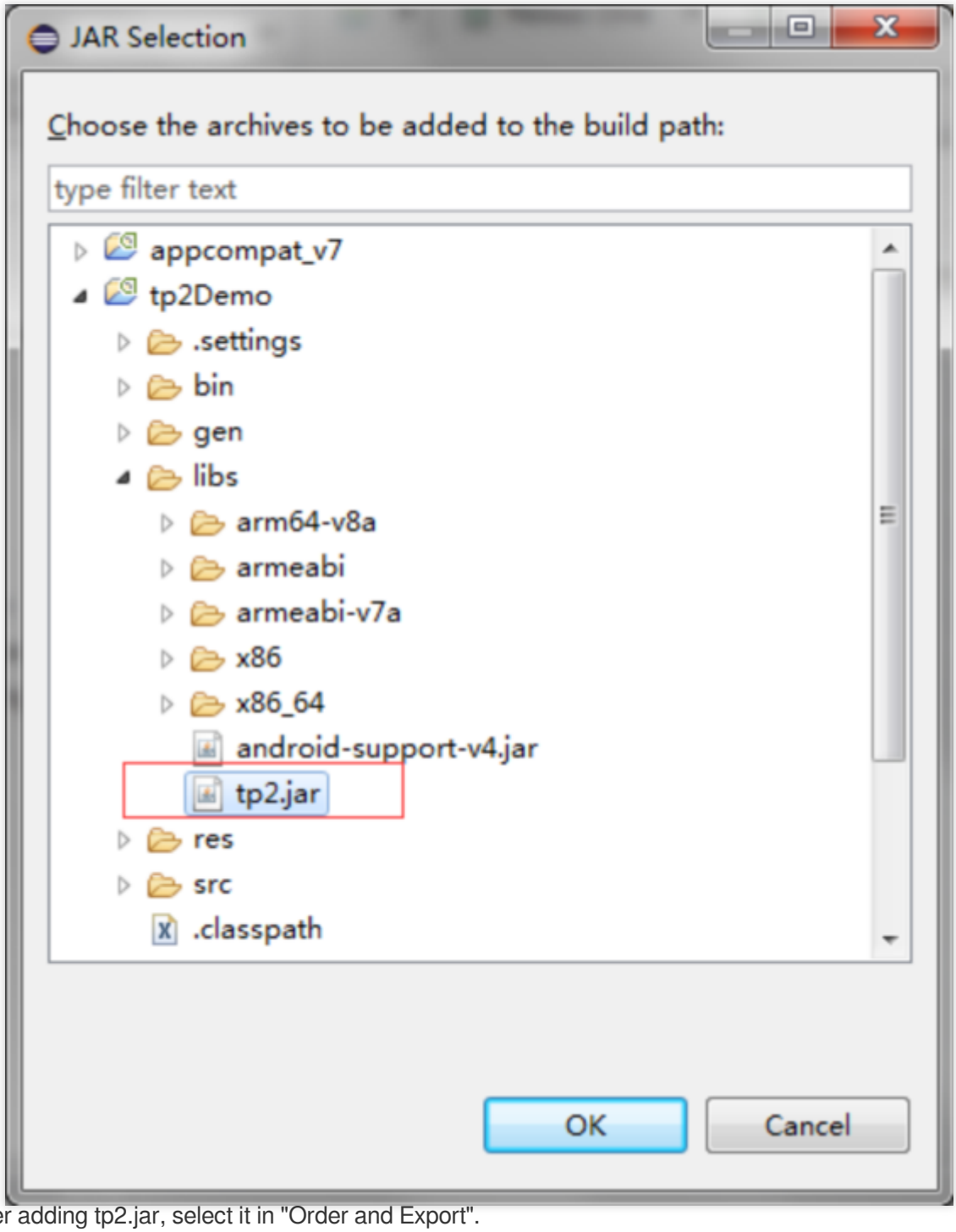


Setting of project attributes

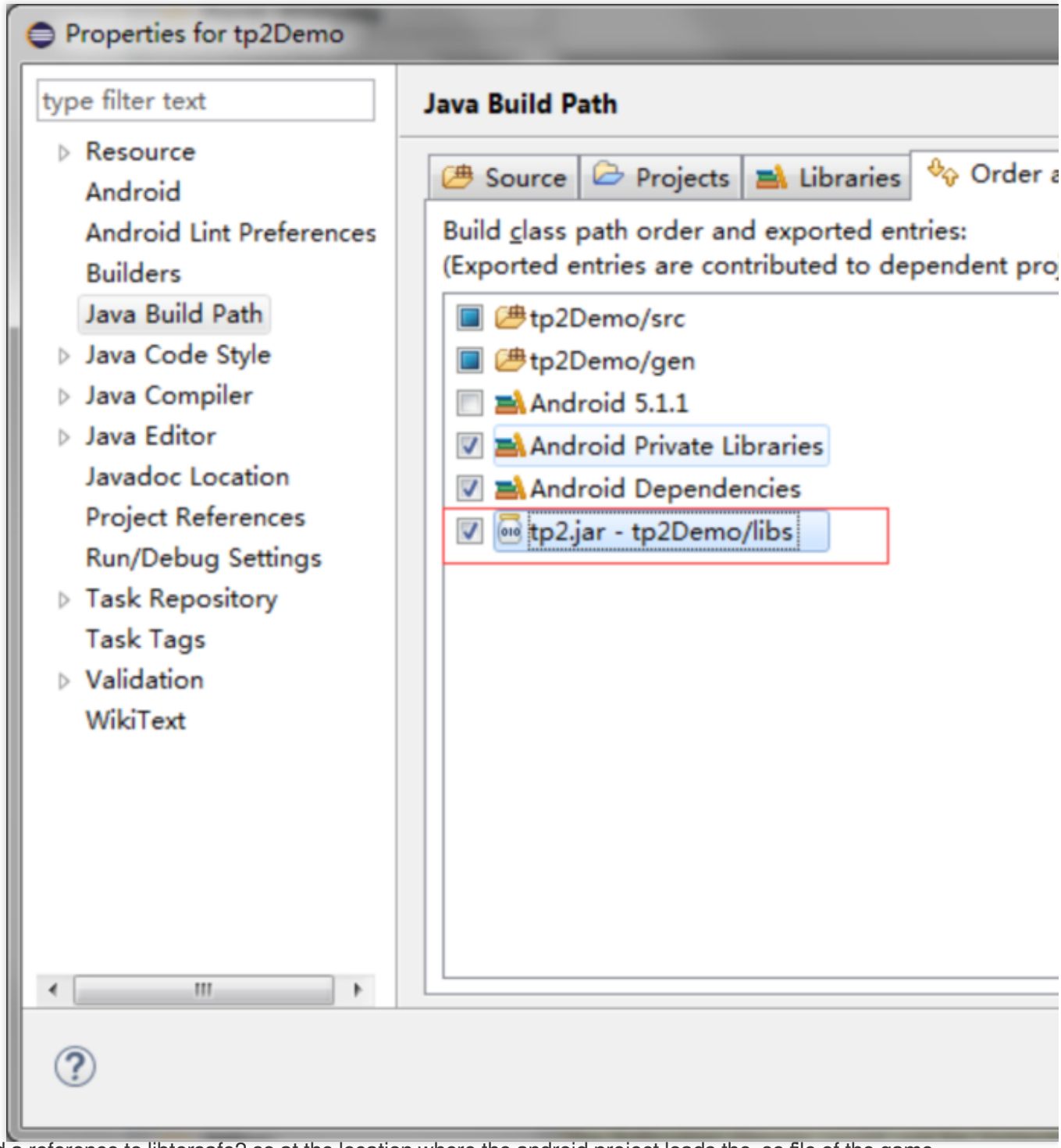
1. Select the game project on the left "Project Explorer" pane in Eclipse, right-click and select "Properties" in the pop-up menu, then select "Java Build Path" on the left of the "Properties" window, and click "Add JARs" in "Libraries" to add tp2.jar.



2. Select tp2.jar that has been copied to the project directory.



3. After adding tp2.jar, select it in "Order and Export".



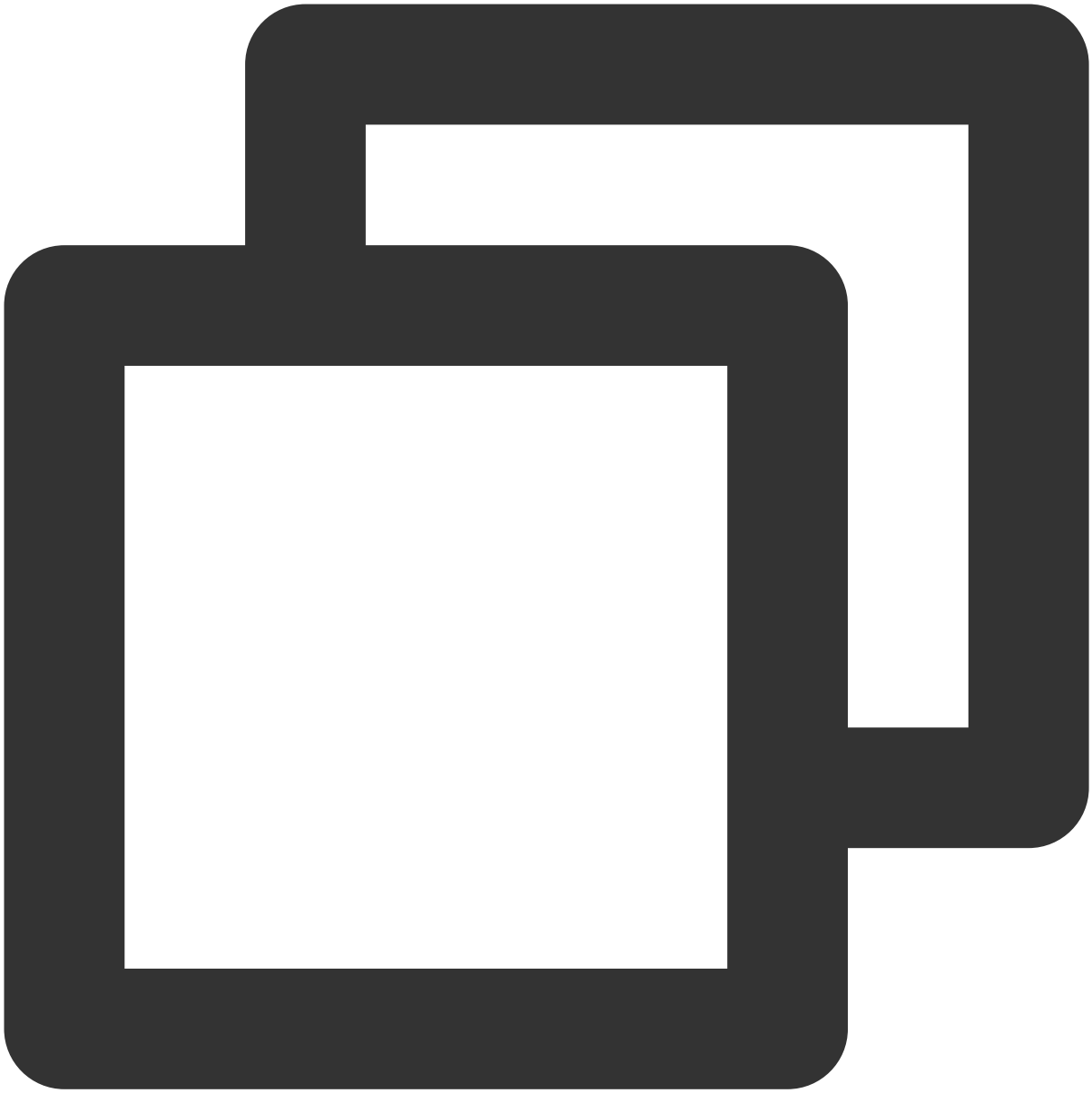
4. Add a reference to libtersafe2.so at the location where the android project loads the .so file of the game.

NO：

libtersafe2.so should be loaded prior to the loading of the .so file of the game.

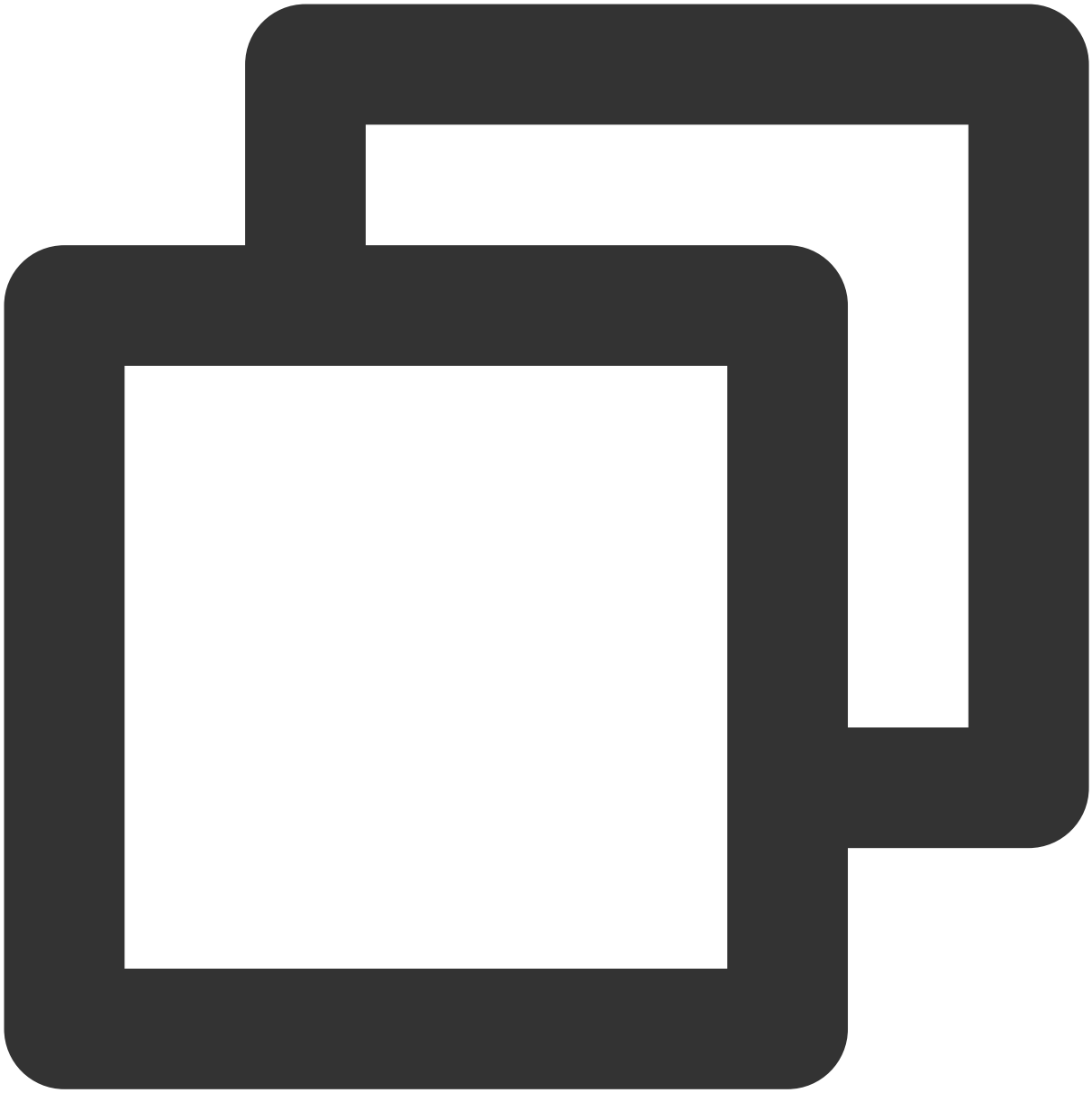
Modify Android.mk

1. Add the following code in jin/android.mk to load libtersafe2.so.



```
include $(CLEAR_VARS)
LOCAL_MODULE:=libtp2
LOCAL_SRC_FILES:=$(TARGET_ARCH_ABI)/libtersafe2.so
include $(PREBUILT_SHARED_LIBRARY)
```

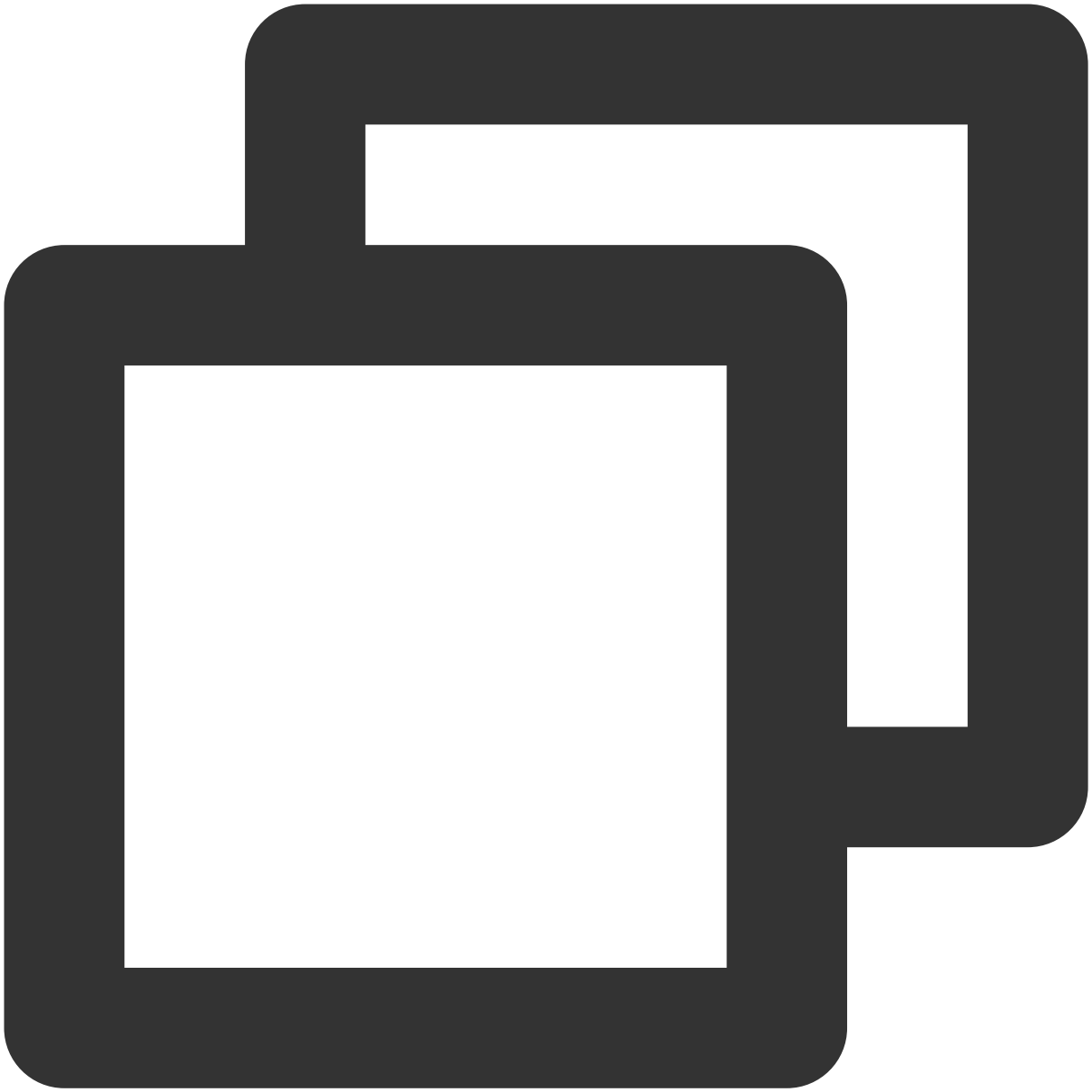
2. In jni/Android.mk, add the following code to the so section of the game to indicate the reference to libtp2.



```
LOCAL_SHARED_LIBRARIES:=libtp2
```

Calling SDK API

Required header file



```
#include "tp2_sdk.h"
```

Initialization API

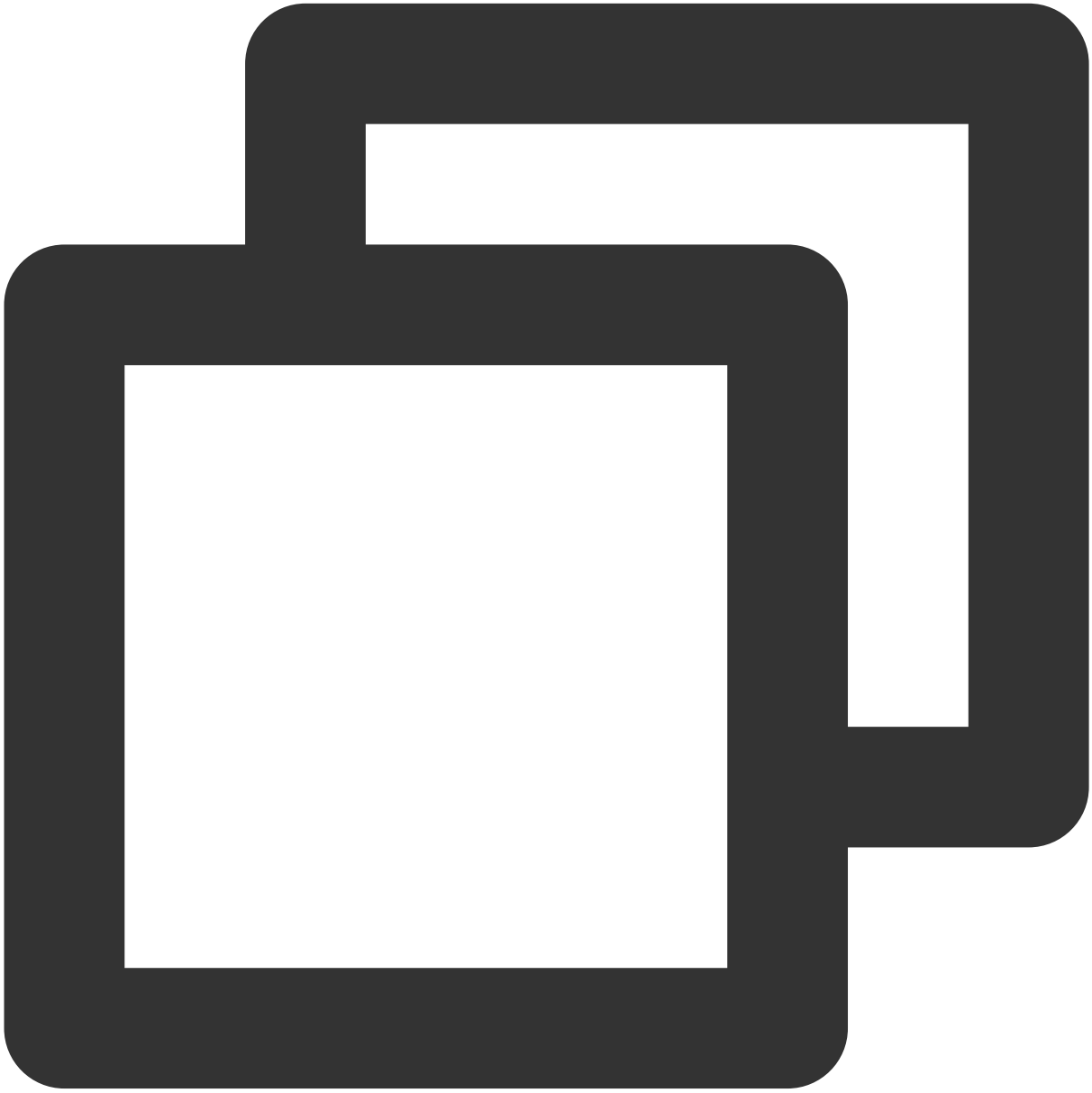
Parameter	Required	Description
game_id	Yes	The game_id assigned by Tencent Cloud
app_key	Yes	The game_key assigned by Tencent Cloud, which corresponds to the game_id.

Both gameID and appKey are automatically generated after a new game has been registered on the Tencent Cloud official website (xxxxxxxxxxxx).

Return value: 0 indicates a successful call.

User login API

Function signature

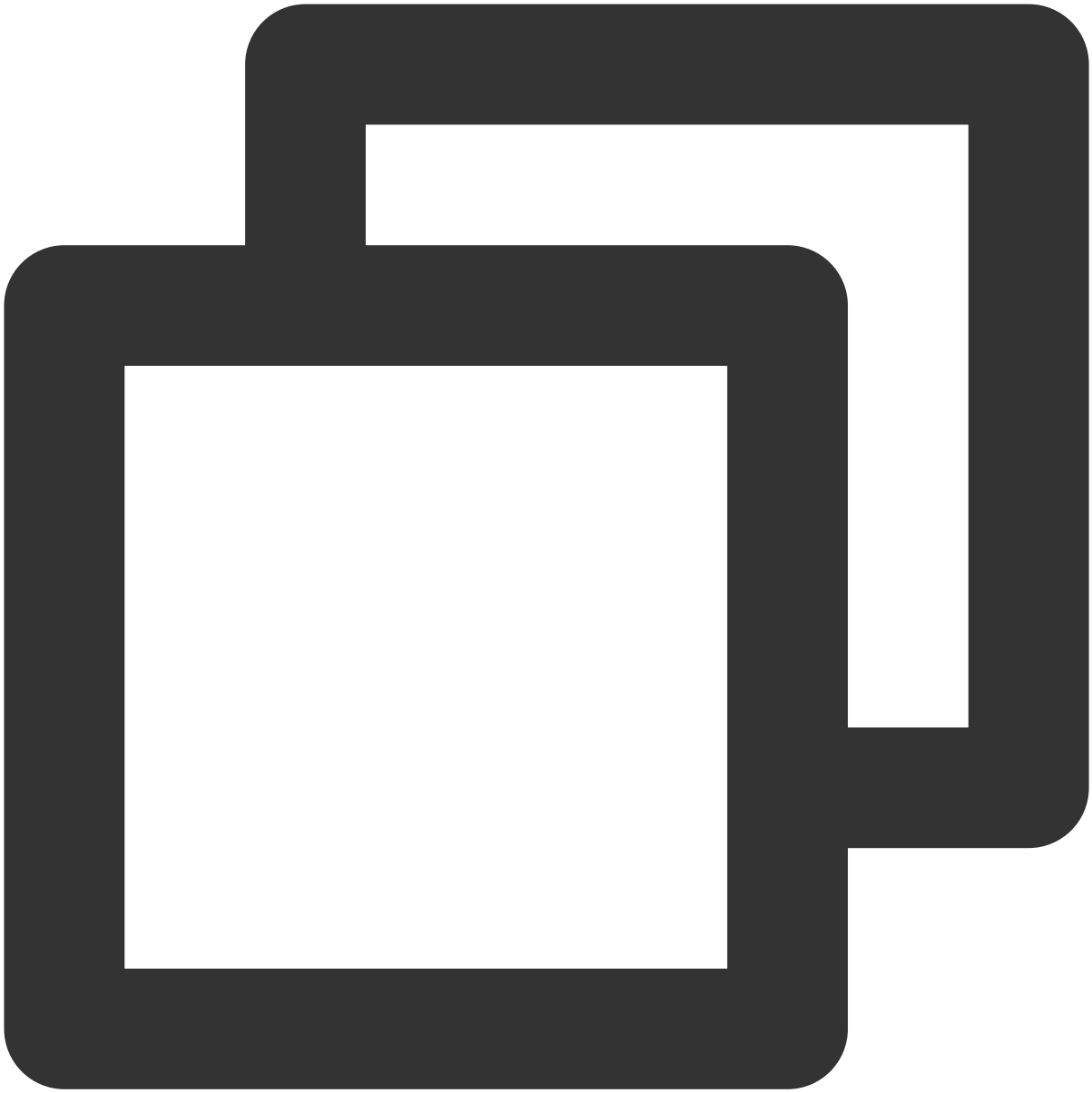


```
int tp2_setuserinfo(int account_type, int world_id, string open_id, string role_id)
```

Parameter description

Parameter	Description
account_type	Account type associated to the operating platform. Refer to TssSdkEntryId below.
world_id	Information on the server where user's game role is created
open_id	User's unique ID, which can be a custom string. This is required for penalties purposes.
string role_id	Identifies the varying roles created by a user

For the account_type, 1 indicates QQ (default), 2 indicates WeChat, and 99 indicates other platforms. For Chinese and international mainstream login platforms, please refer to the following values.

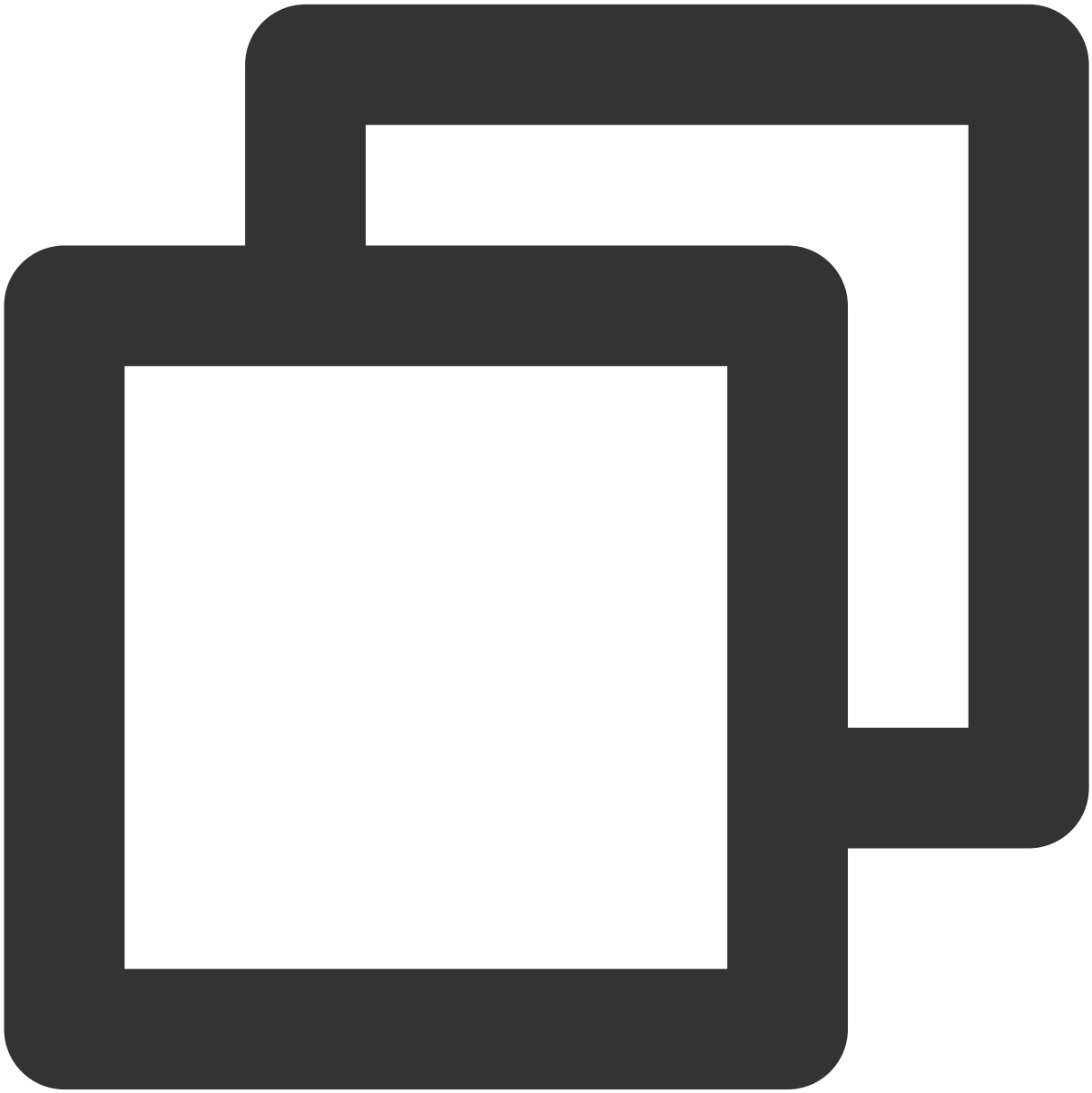


```
enum TssSdkEntryId
{
ENTRY_ID_QZONE = 1, // QQ
ENTRY_ID_MM = 2, // WeChat
ENTRT_ID_FACEBOOK = 3, // facebook
ENTRY_ID_TWITTER = 4, // twitter
ENTRY_ID_LINE = 5, // line
ENTRY_ID_WHATSAPP = 6, // whatsapp
ENTRY_ID_OTHERS = 99, // Other platforms
};
```

world_id is defined by the game. Enter 0 if the game has only one server.
role_id is used to identify different roles of an account under one server. Enter "" if there is only one role.
open_id is assigned by the specific operating platform to uniquely identify users.
Return value: 0 indicates a successful call.

API for switching between foreground and background

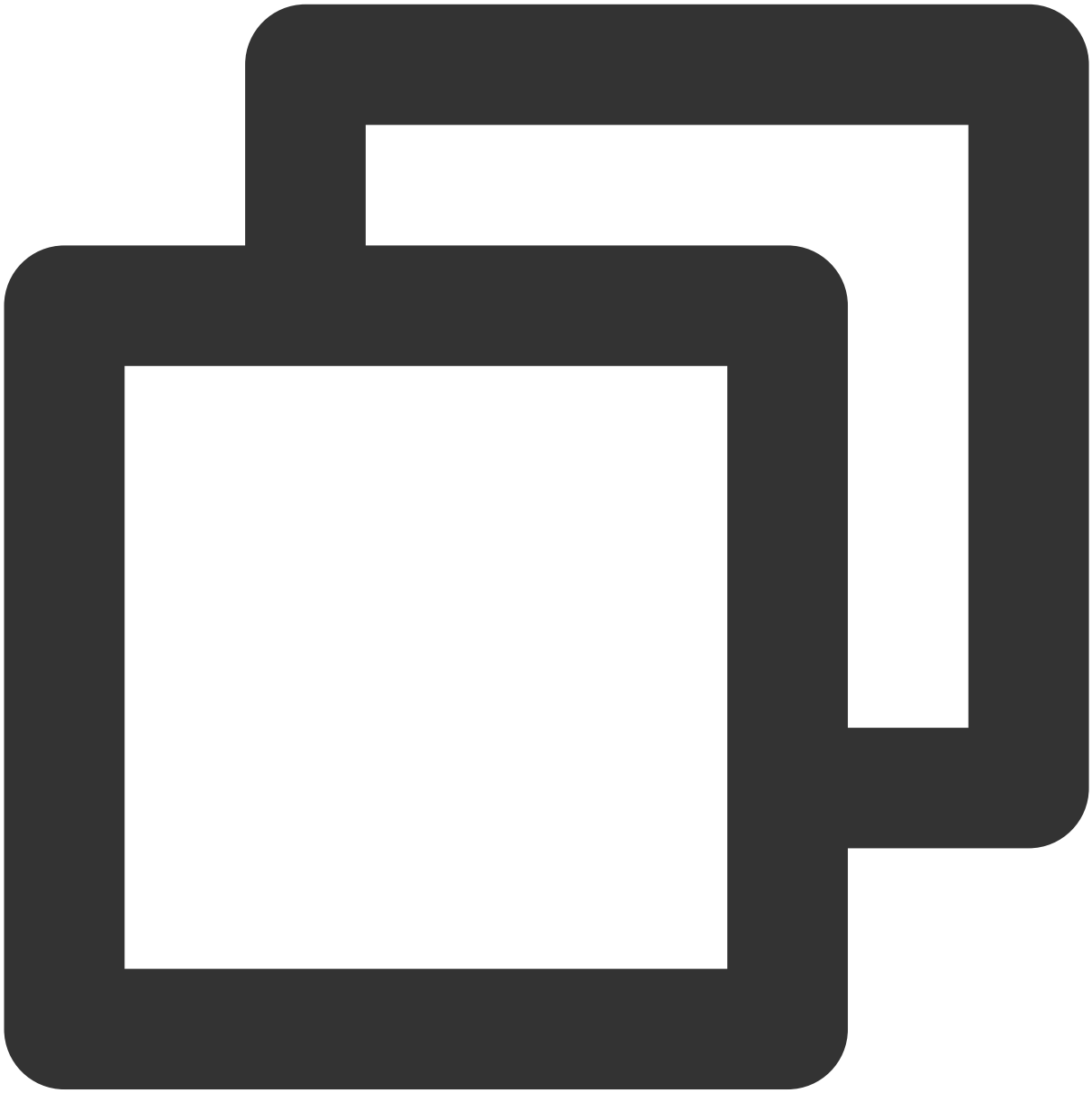
Function signature



```
int tp2_setgamestatus (int status);
```

Parameter	Description
status	foreground TP2_GAME_STATUS_FRONTEND background TP2_GAME_STATUS_BACKEND

Enumeration types



```
enum TP2GameStatus
{
TP2_GAME_STATUS_FRONTEND = 1, // Foreground
TP2_GAME_STATUS_BACKEND = 2 // Background
```

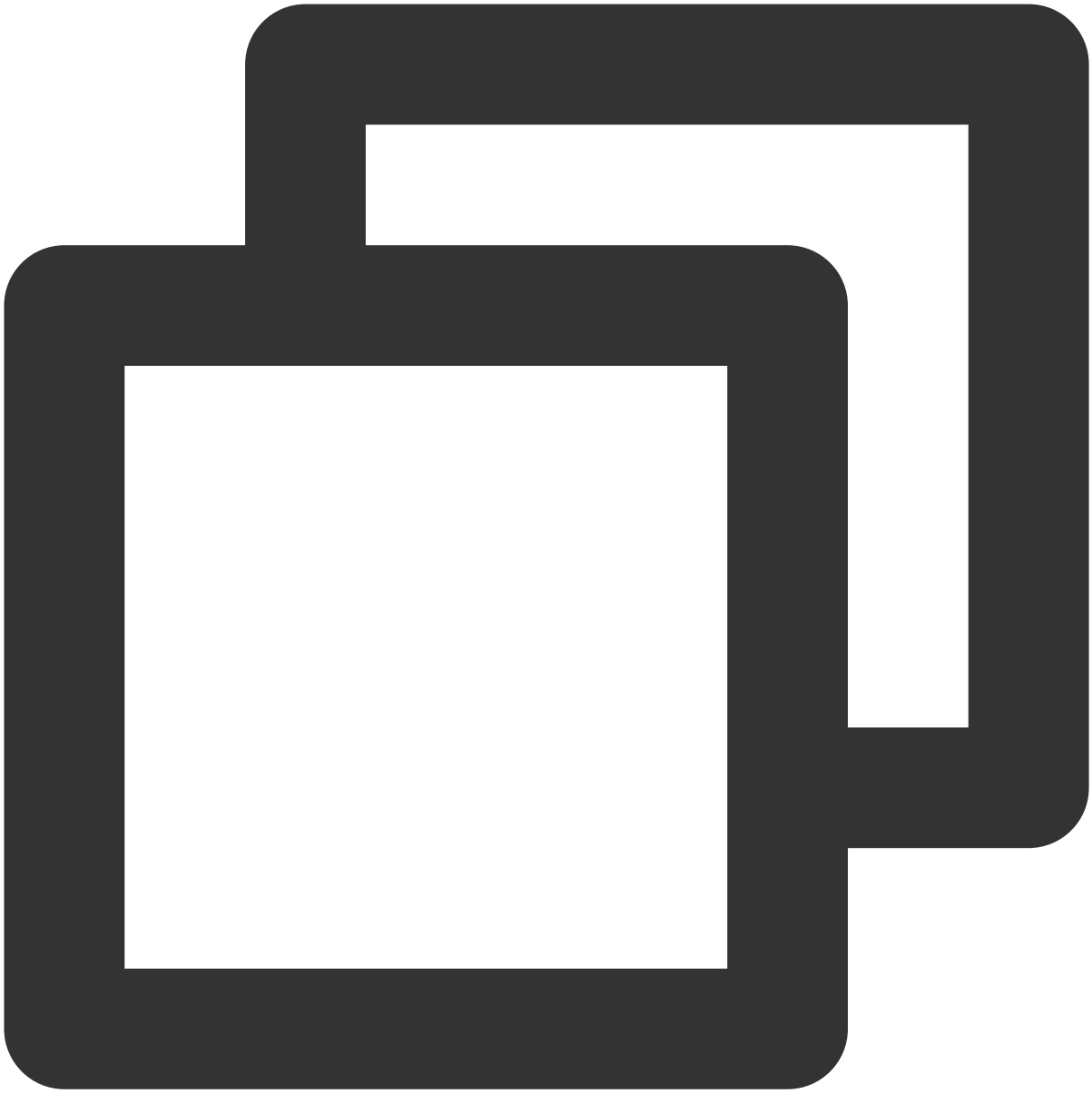
```
}
```

Return value: 0 indicates a successful call.

When to call the function

1. Call tp2_sdk_init_ex immediately after the game is launched. Parameters are game_id and app-key. Calling the security API function earlier can better protect the game process.
2. tp2_setuserinfo is called after the game is authorized by the user to access its login information. If the game has set world_id and role_id, then call the tp2_setuserinfo function after obtaining both world_id and role_id. During gameplay, if you need to retrieve the user's login information in situations like when the network is disconnected or the user logged out and needs to re-login, you will need to call the function again. The parameter to be passed is the user's account information, which can be customized.
3. tp2_setgamestatus is called when the game switches between foreground and background. When the game switches from background to foreground, the parameter is set to Tp2Status. FRONTEND, and when the game switches from foreground to background, the parameter is set to Tp2Status. BACKEND. Some of the SDK functions stop running when the game switches to background, so the API may affect the normal running of SDK functions.

Sample Code

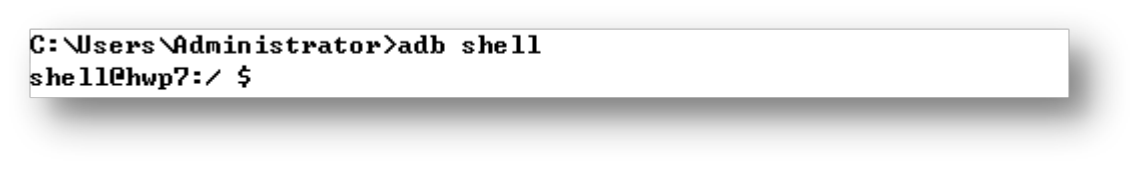


```
void Start ()
{
//Called immediately after the game is launched.
tp2_sdk_init_ex (8888, "a5ab8dc7ef67ca92e41d730982c5c602");
// Called when the user logs in
int account_type = ENTRY_ID_QZONE; /* Account type */
int world_id = 101; /* Server id*/
string open_id = "B73B36366565F9E02C752"; /* Platform-specific user ID */
string role_id = "paladin"; /* Role id*/
tp2_setuserinfo(account_type, world_id, open_id, role_id);
}
```

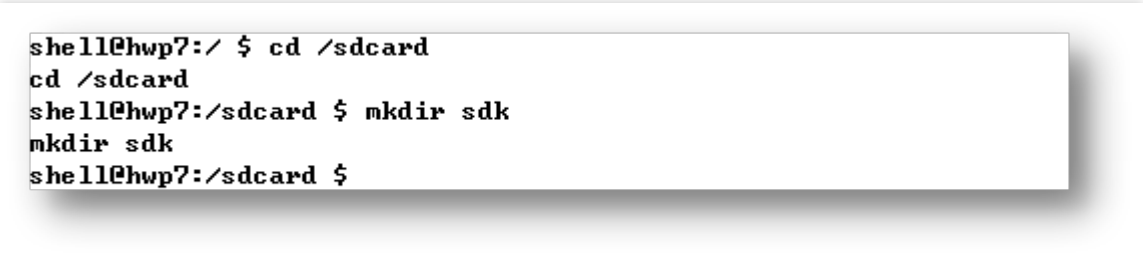
```
// Game switches from background to foreground
void onResume ()
{
tp2_setgamestatus(TP2_GAME_STATUS_FRONTEND);
}

// Game switches from foreground to background
void onPause ()
{
tp2_setgamestatus(TP2_GAME_STATUS_BACKEND);
}
```

Verifying Whether the SDK is Integrated Correctly

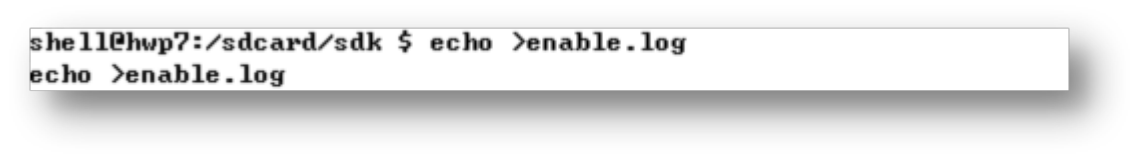
1. Connect your Android phone to a Windows PC via a USB cable. After the connection is successful, log in to the Android ADB console using Windows CMD, as shown below:



2. Type cd /sdcard, press enter, then type mkdir sdk, and press enter, to create the /sdcard/sdk directory. If the directory already exists, a prompt indicating "mkdir failed for /sdcard/sdk. File exists" will appear, and you can proceed to the next step.



3. Type cd /sdcard/sdk to enter the directory, and type echo>enable.log to create an empty file enable.log:



Files under the directory created by the shell may not be accessed on some models. In this case, change the /sdcard/sdk directory with root user or use another mobile phone.

```
shell@hwp7:/sdcard $ su
su
root@hwp7:/mnt/shell/emulated/0 # chmod -R 777 /sdcard/sdk
chmod -R 777 /sdcard/sdk
root@hwp7:/mnt/shell/emulated/0 #
```

4. Start and log in to the game, check whether tp2.log and tlog.log are generated in the /data/data/log directory, as shown below:

```
shell@hwp7:/sdcard/sdk $ ls -l
ls -l
-rwxrwx--- root     sdcard_r          1 2016-04-20 22:37 enable.log
-rwxrwx--- root     sdcard_r       3324 2016-04-20 22:33 tlog.log
-rwxrwx--- root     sdcard_r       4151 2016-04-20 22:33 tp2.log
```

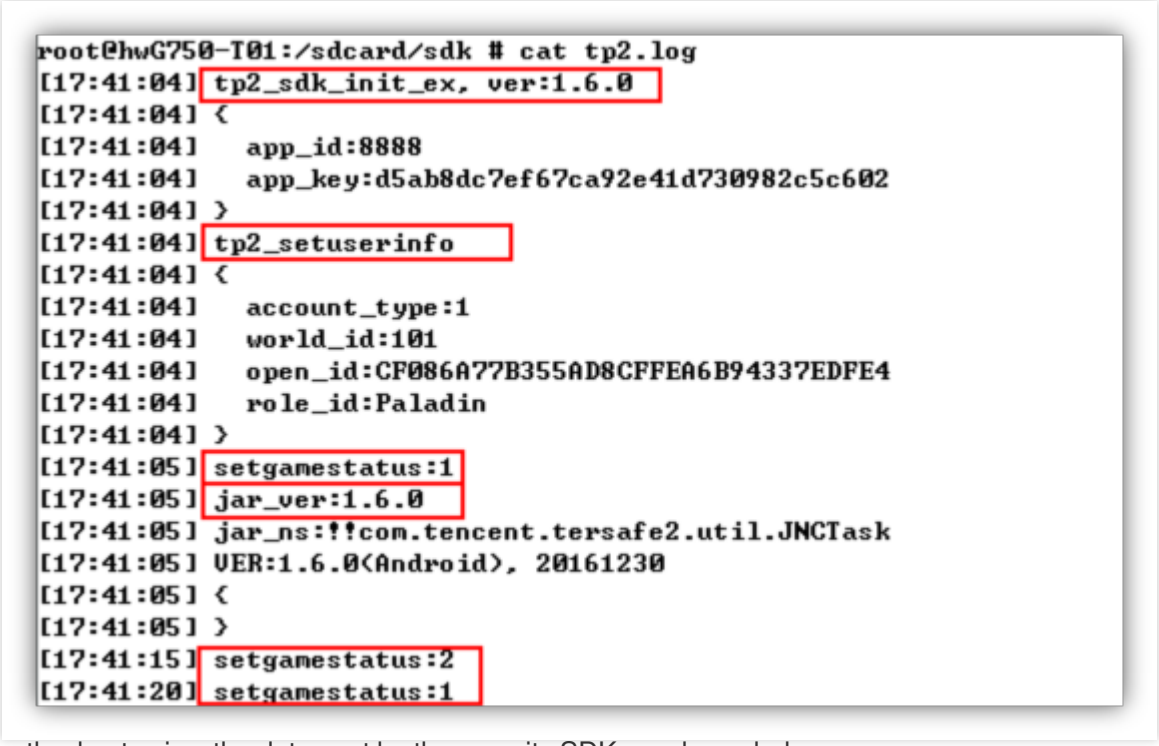
If no log is generated, check whether you have the read/write permission to /sdcard/sdk and enable.log. This directory cannot be read/written on a small number of models. Use another model for testing or change /sdcard/sdk to /data/data/log with root user.

Note:

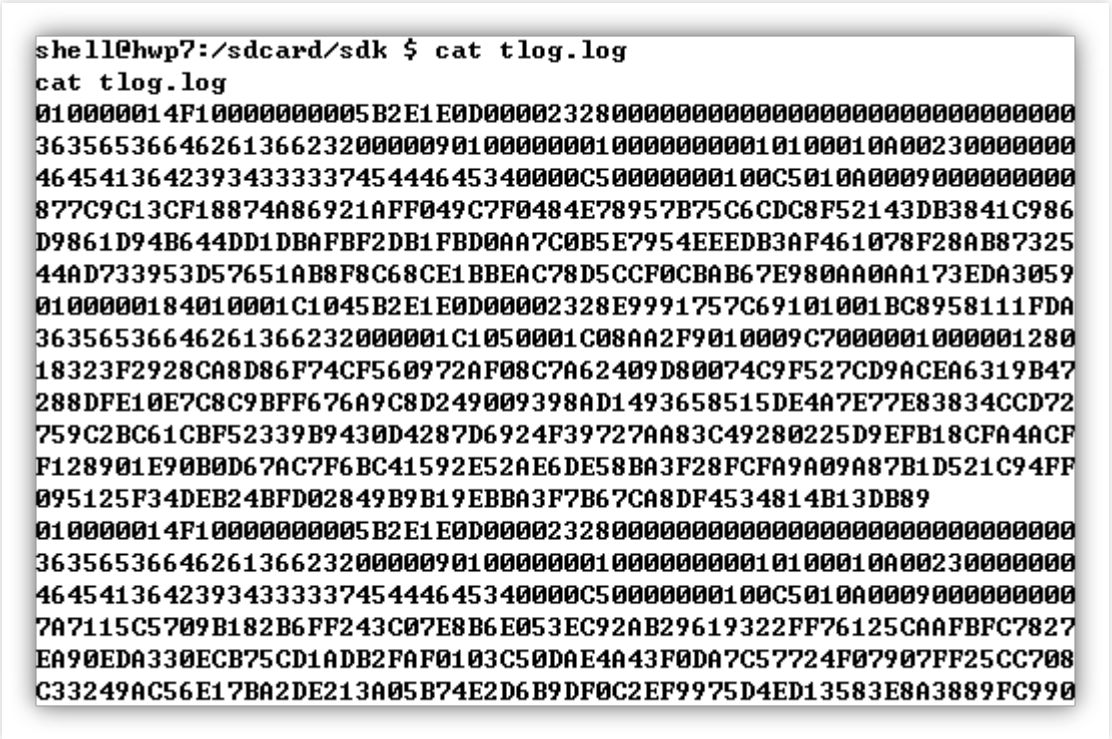
enable.log is only used for testing purposes.

5. Open the tp2.log file, and check whether it contains the information of three native APIs tp2_sdk_init_ex, tp2_setuserinfo and setgamestatus as well as the jar packet's version number jar_ver. Only when all the above conditions are met, can the security SDK run properly. setgamestatus:1 indicates that the current process is running in the foreground, and setgamestatus:2 indicates that the current process is running in the background.

Verify whether the API is correctly called by switching the App between foreground and background, and also check whether the userinfo is entered correctly.



6. Open tlog.log to view the data sent by the security SDK, as shown below:



In addition to reporting some basic process information during initialization, the security SDK also sends data according to the results of periodic security scanning, such as the incorrect signature of the App certificate, modification of memory data, a running add-on process, etc. The tlog.log records the data (only generated during testing) sent by the SDK. Generally, the data size per hour is about 20 KB. You can check the size of `` `tlog.log` `` to calculate the volume of data sent by the security SDK.

Integration Guidelines for Java

Last updated：2023-12-08 17:48:18

Preparations

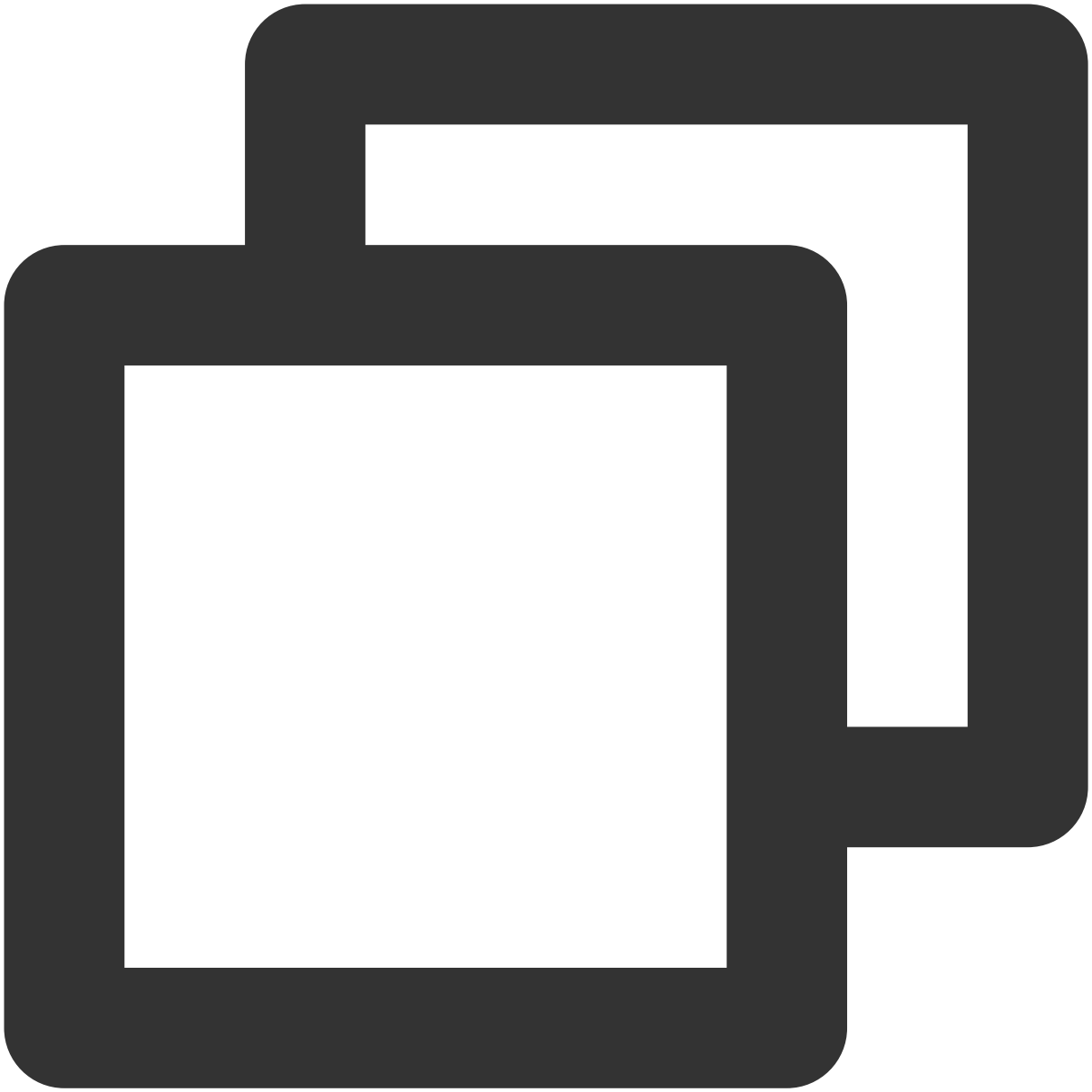
Developers need to complete the following steps when integrating the security SDK:

1.1 Copy the SDK dynamic library to the specified project directory associated with the game platform and the CPU architecture.

1.2 Call the SDK API based on the game_id and user's login information.

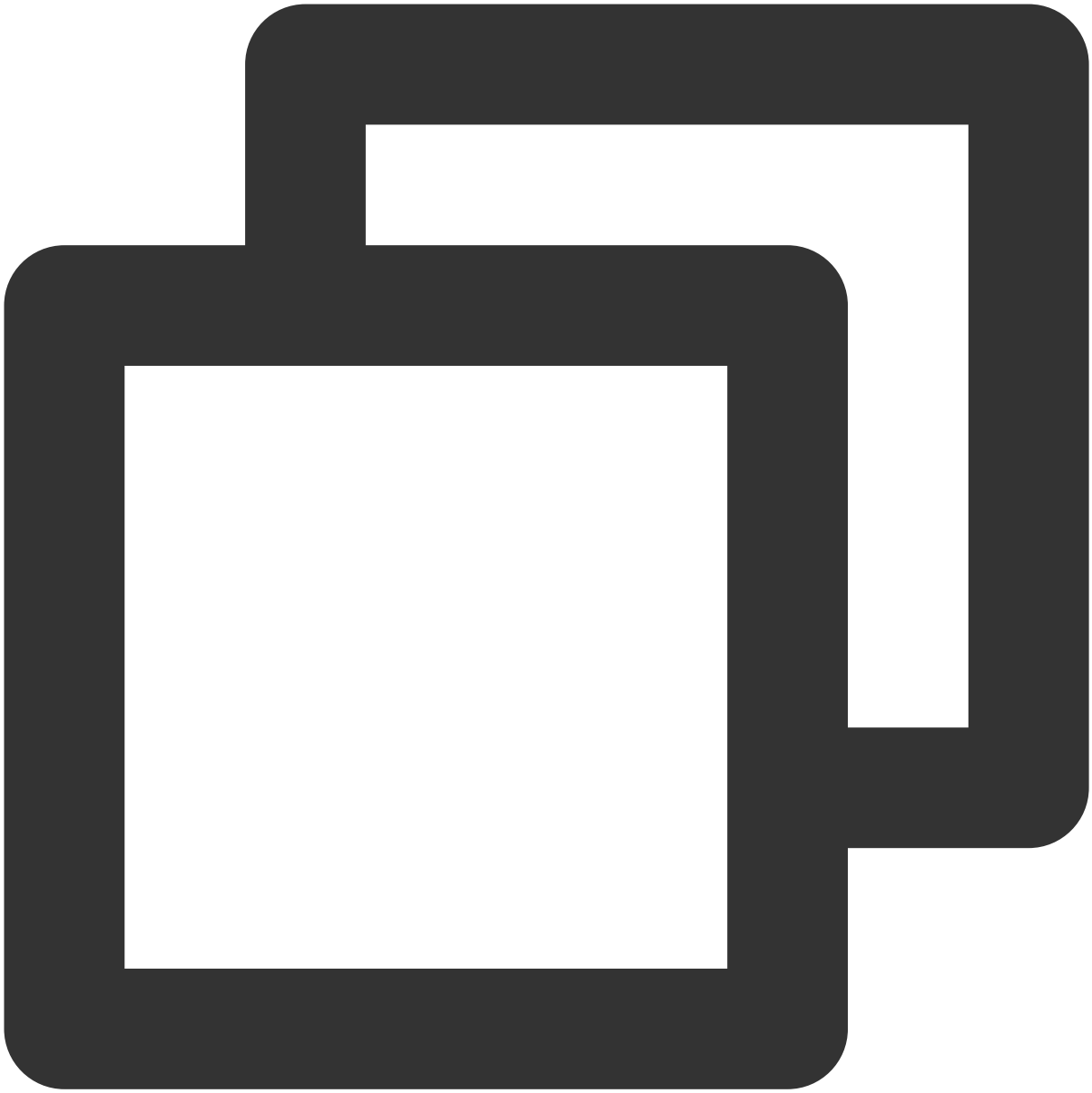
1.3 Verify whether the SDK is integrated correctly.

The following files are required for the integration of the security SDK to Android OS written in C/C++:



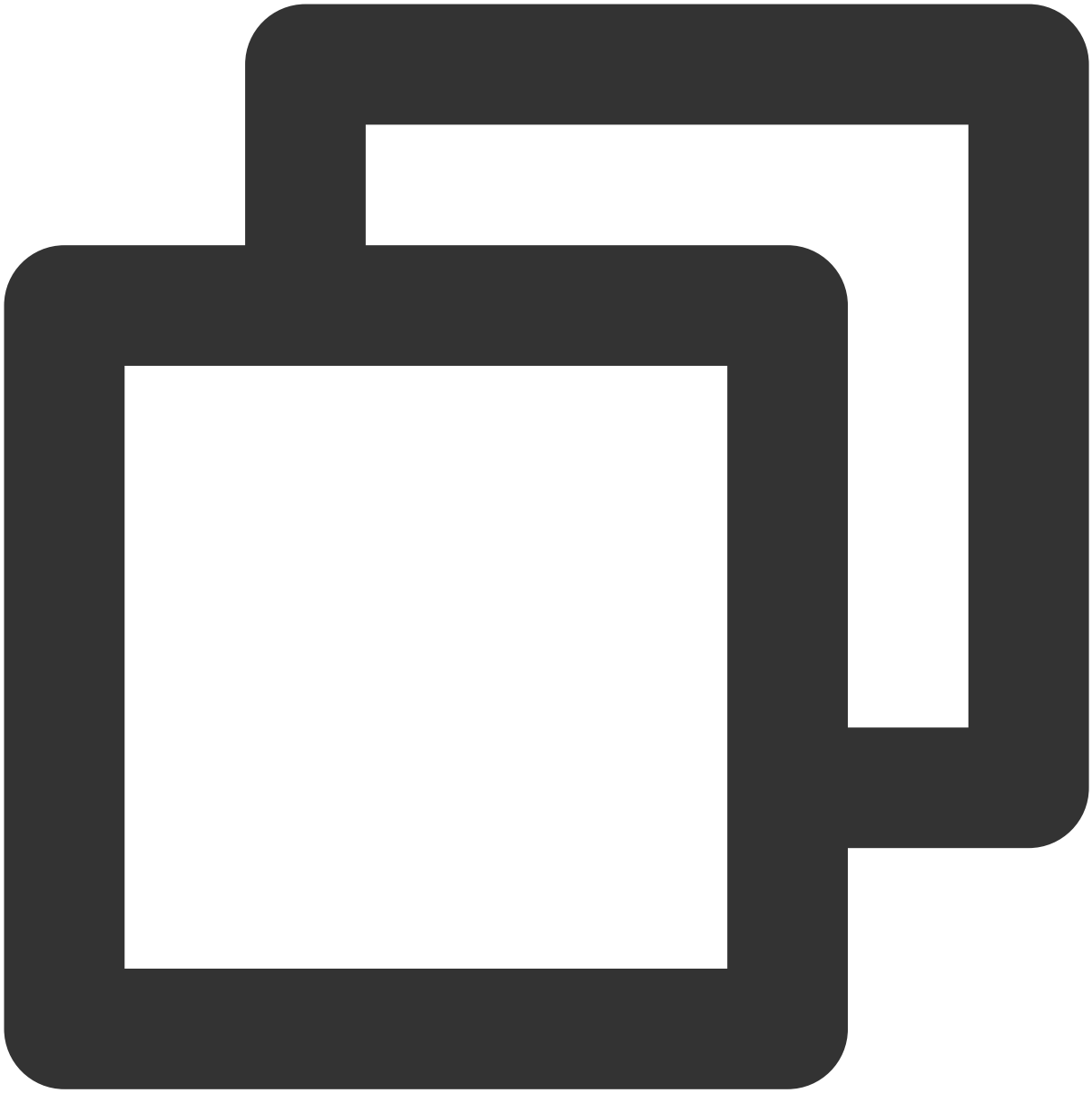
```
tp2.jar
libtersafe2.so
```

Permissions required:



```
<uses-permission android:name="android.permission.ACCESS_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.GET_TASKS" />
<uses-permission android:name="android.permission.INTERNET" />
```

SDK API functions:

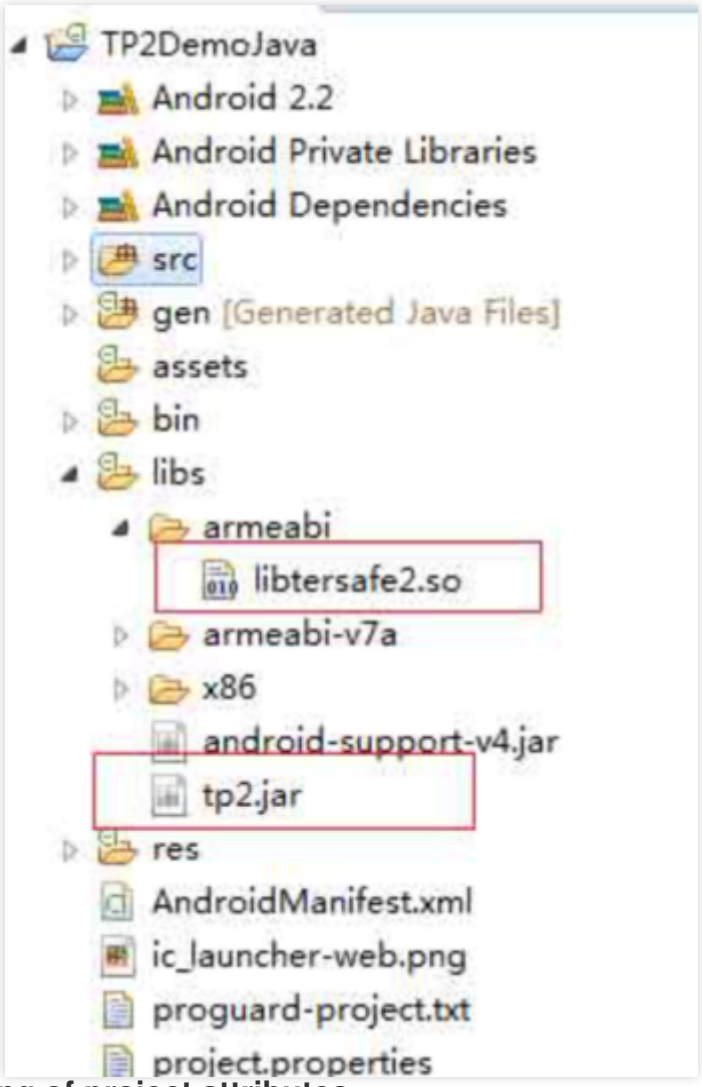


```
Initialization API: initEx
User login API: onUserLogin
API for switching from foreground to background: onAppPause
API for switching from background to foreground: onAppPause
```

Adding SDK Files to the Project

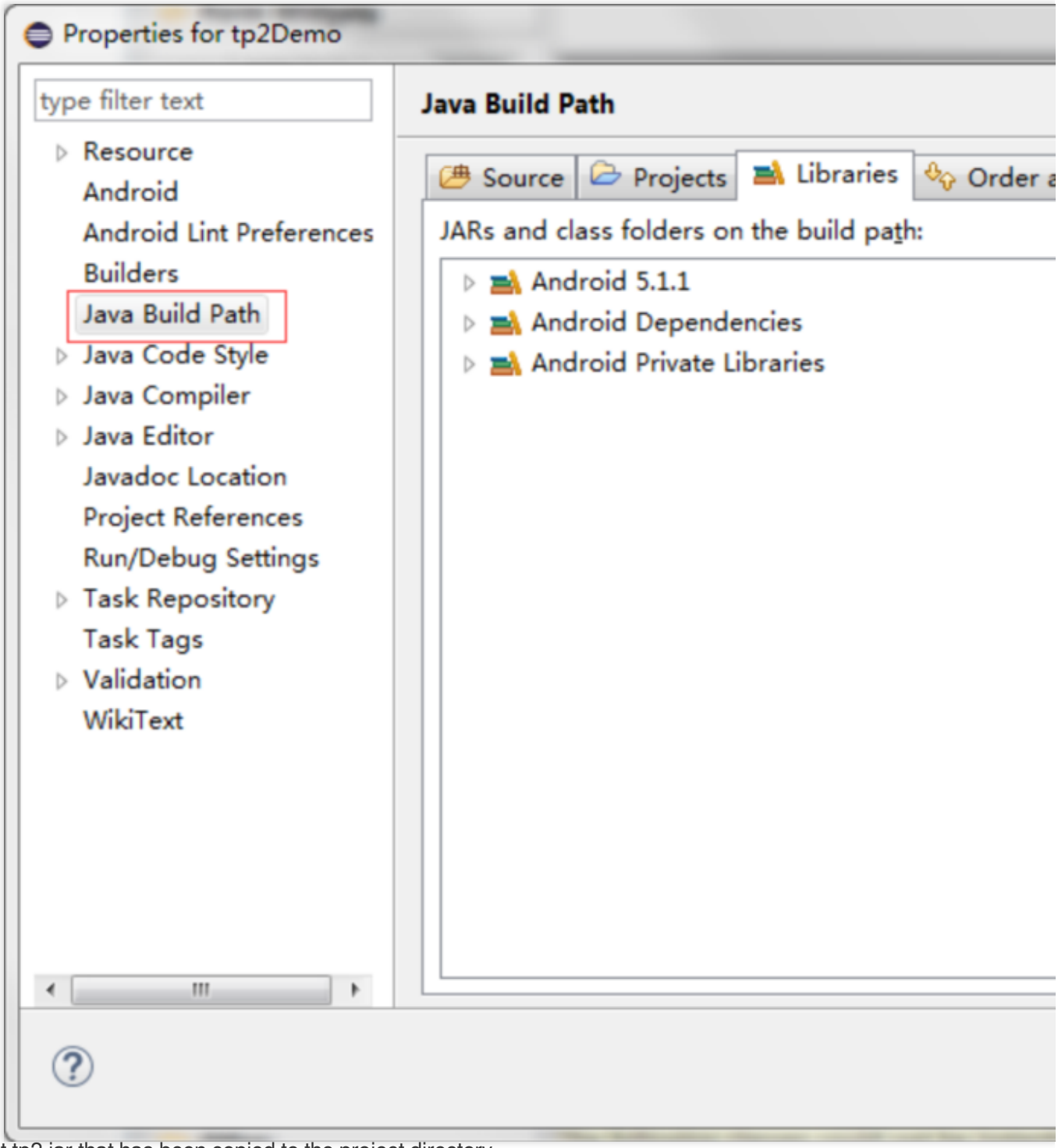
Add files

1. Copy the tp2.jar file from the sdk/android/c directory to the libs directory in the android project directory.
2. Copy the folder (containing the libtersafe2.so file) named after the CPU architecture from the sdk/android/java/lib directory to the libs directory of the android project directory. Do not copy unsupported CPU architectures.

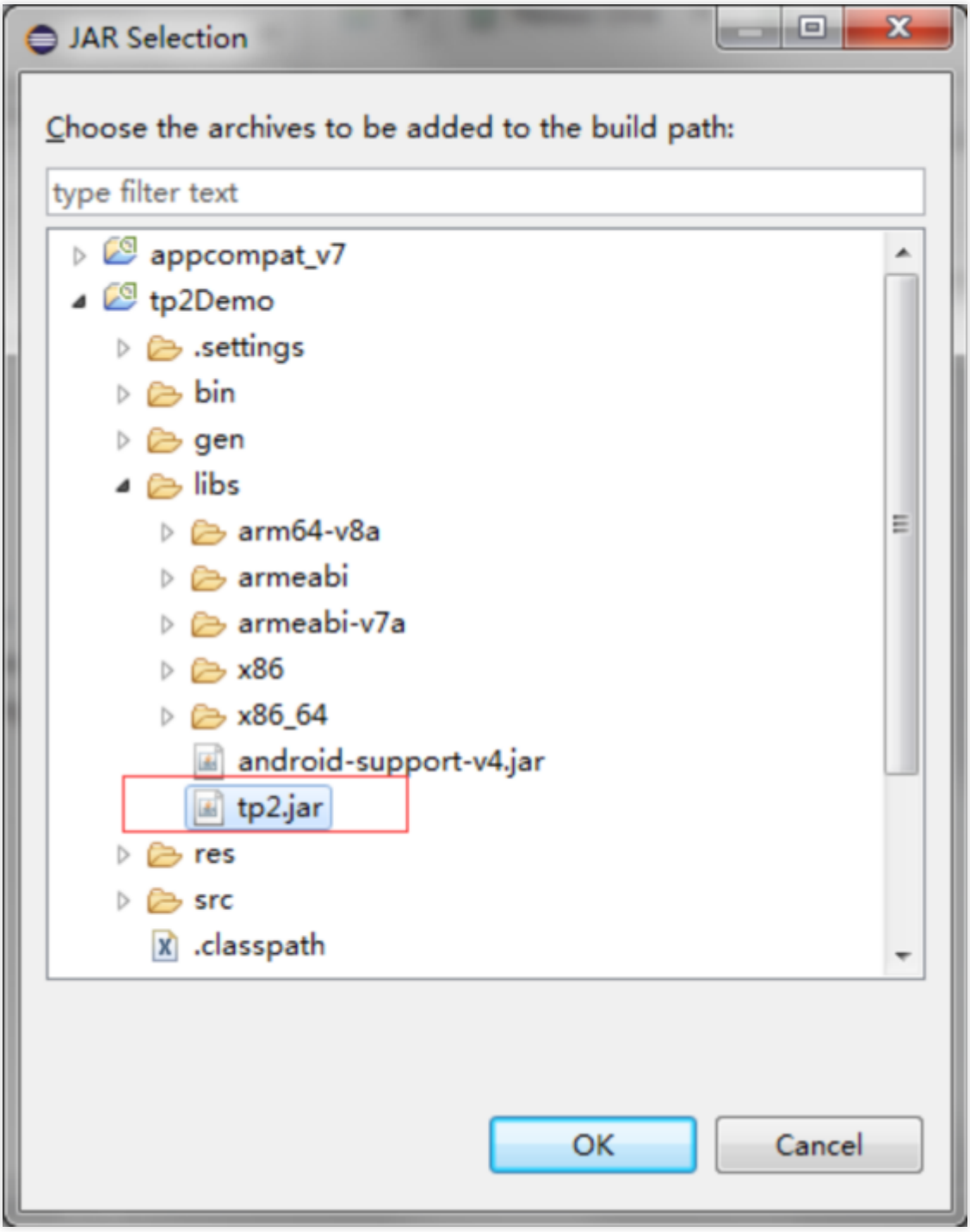


Setting of project attributes

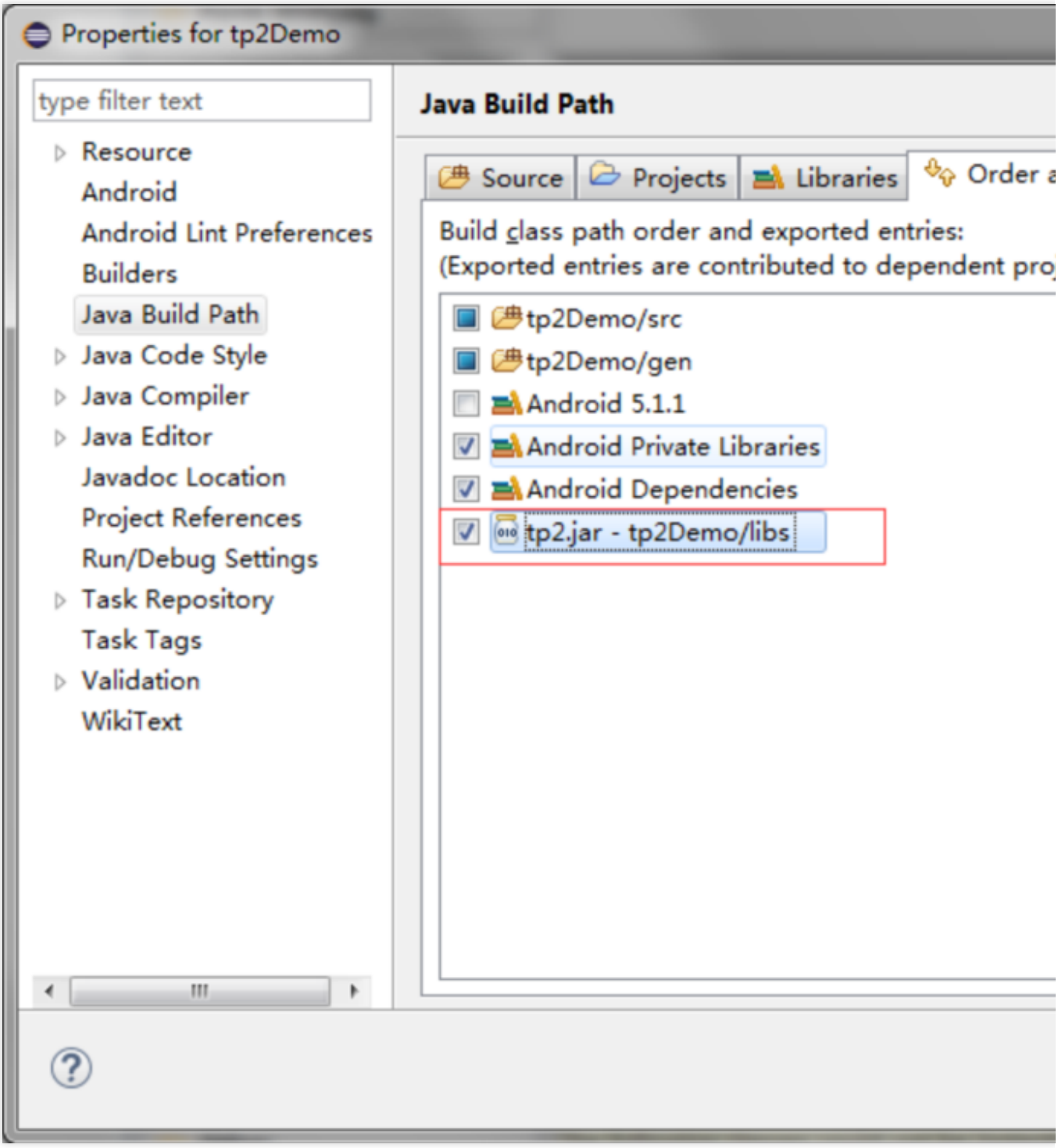
Select the game project on the left "Project Explorer" pane in Eclipse, right-click and select "Properties" in the pop-up menu, then select "Java Build Path" on the left of the "Properties" window, and click "Add JARs" in "Libraries" to add tp2.jar.



Select tp2.jar that has been copied to the project directory.



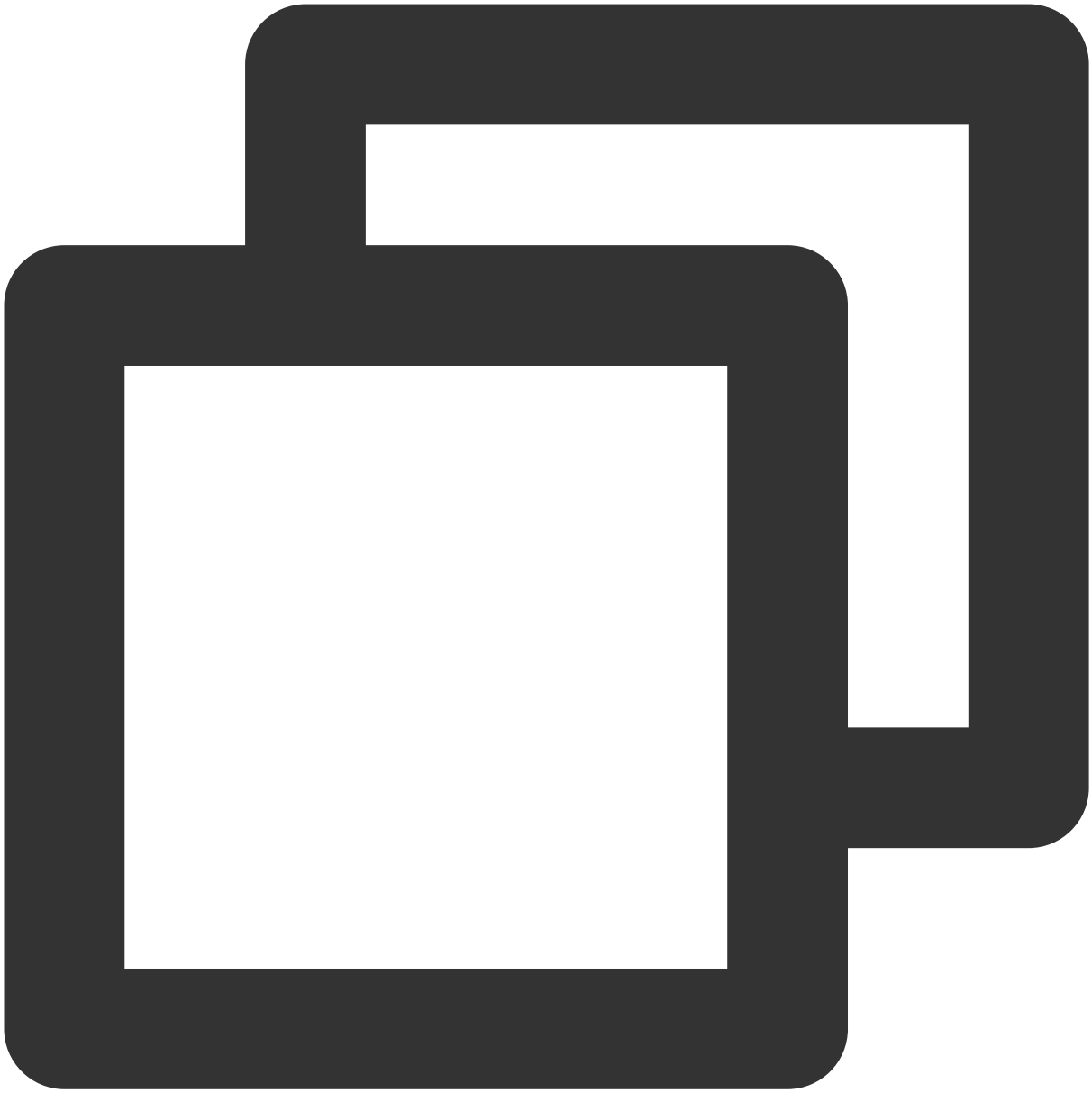
After adding tp2.jar, select it in "Order and Export".



Clean and rebuild the project.

Calling SDK API

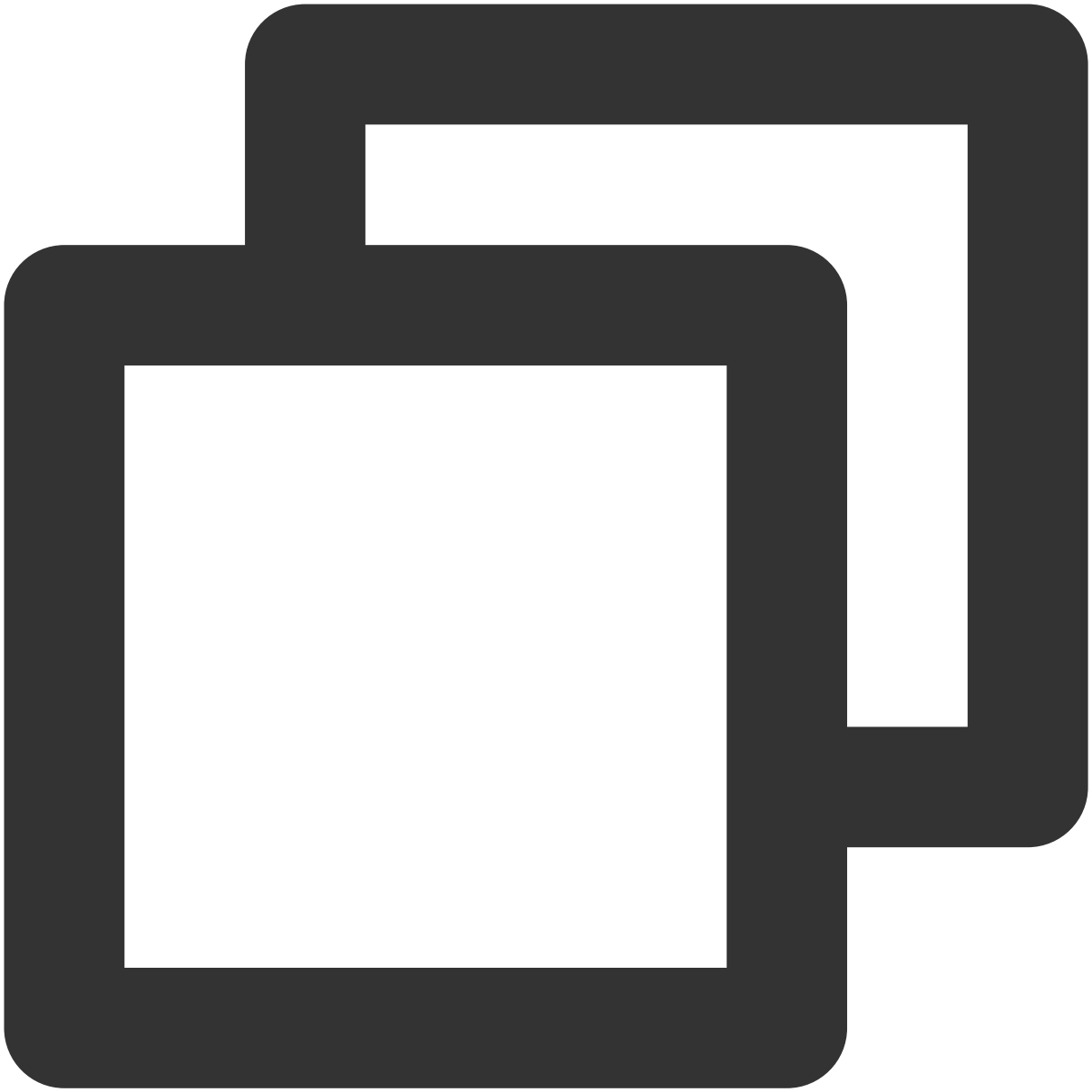
Import the packet.



```
import com.tencent.tersafe2.TP2Sdk;
```

Initialization function

Function signature



```
public static int initEx(int gameId, String appKey);
```

Parameter description

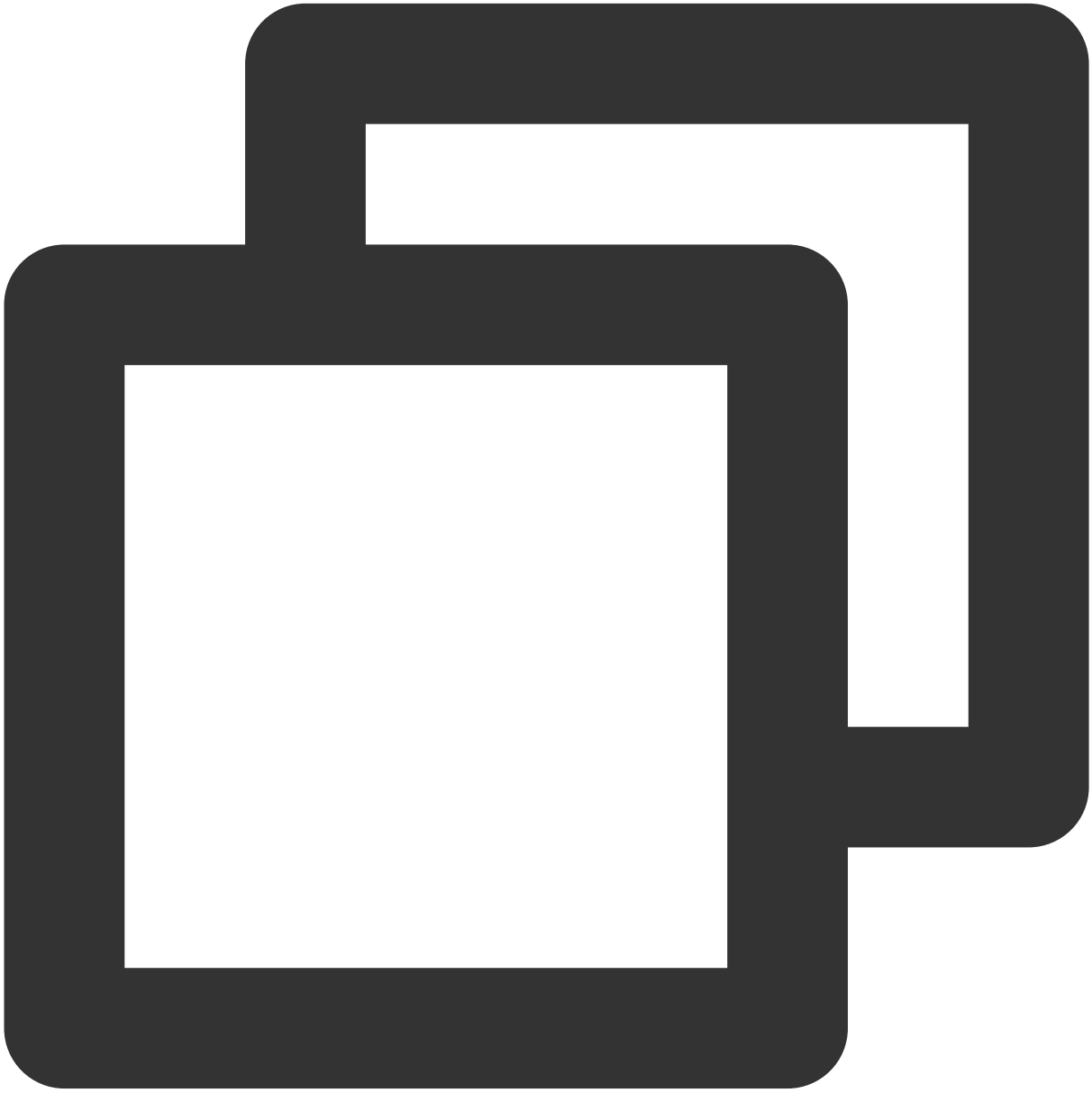
Parameter	Required	Description
gameId	Yes	The game_id assigned by Tencent Cloud
appKey	Yes	The game_key assigned by Tencent Cloud, which corresponds to the game_id.

Both gameID and appKey are automatically generated after a new game has been registered on the Tencent Cloud official website (xxxxxxxxxxx).

Return value: 0 indicates a successful call.

User login API

Function signature

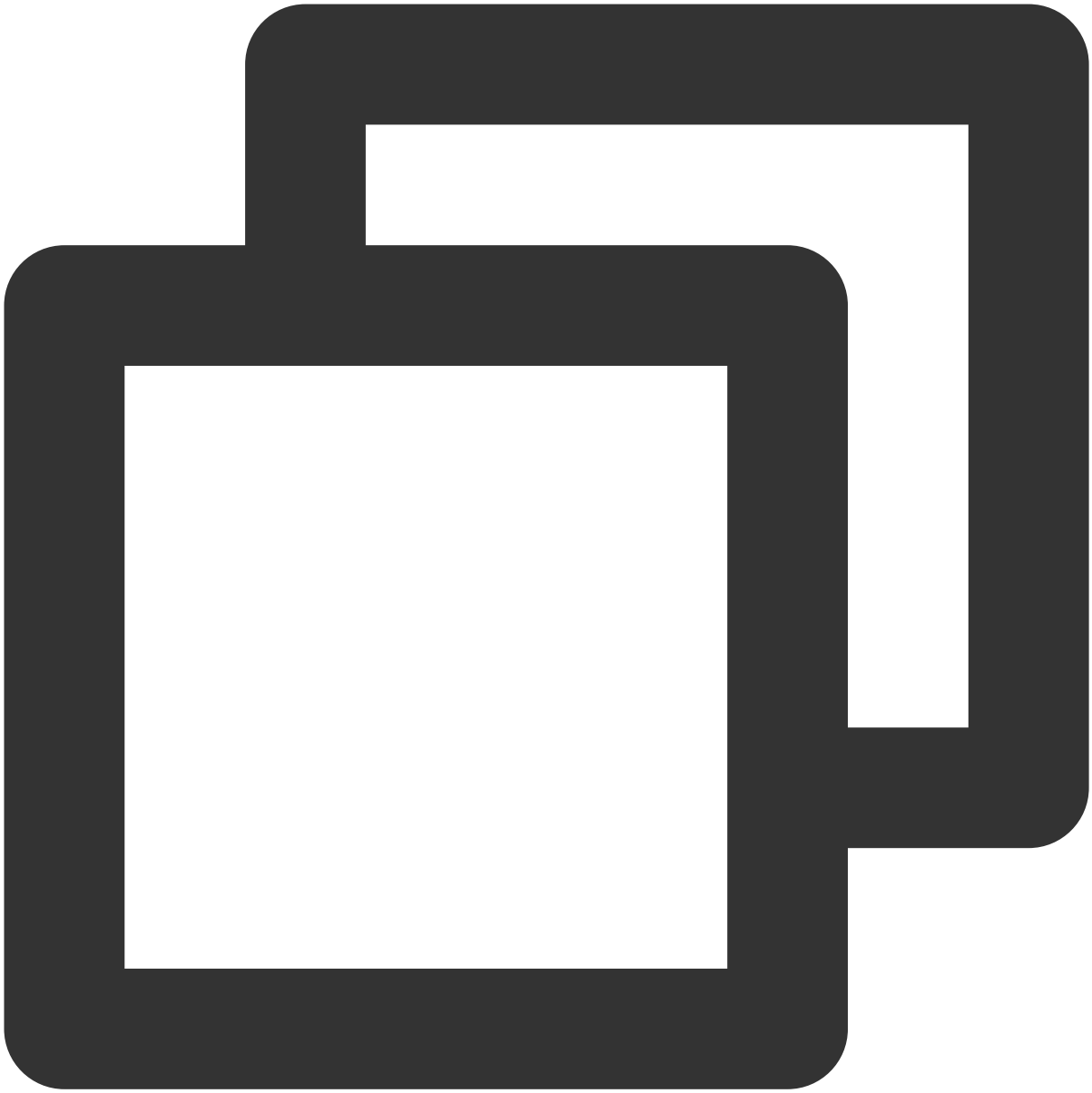


```
public static native int onUserLogin(int accountType, int worldId, String openId, S
```

Parameter description

Parameter	Title 2
account_type	Account type associated to the operating platform. Refer to TssSdkEntryId below.
worldId	Information on the server where user's game role is created
openId	User's unique ID, which can be a custom string. This is required for penalties purposes.
roleId	Identifies the varying roles created by a user

For the account_type, 1 indicates QQ (default), 2 indicates WeChat, and 99 indicates other platforms. Chinese and international mainstream platforms, please refer to the following values.



```
enum TssSdkEntryId
{
ENTRY_ID_QZONE = 1, // QQ
ENTRY_ID_MM = 2, // WeChat
ENTRT_ID_FACEBOOK = 3, // facebook
ENTRY_ID_TWITTER = 4, // twitter
ENTRY_ID_LINE = 5, // line
ENTRY_ID_WHATSAPP = 6, // whatsapp
ENTRY_ID_OTHERS = 99, // Other platforms
};
```

world_id is defined by the game. Enter 0 if the game has only one server.

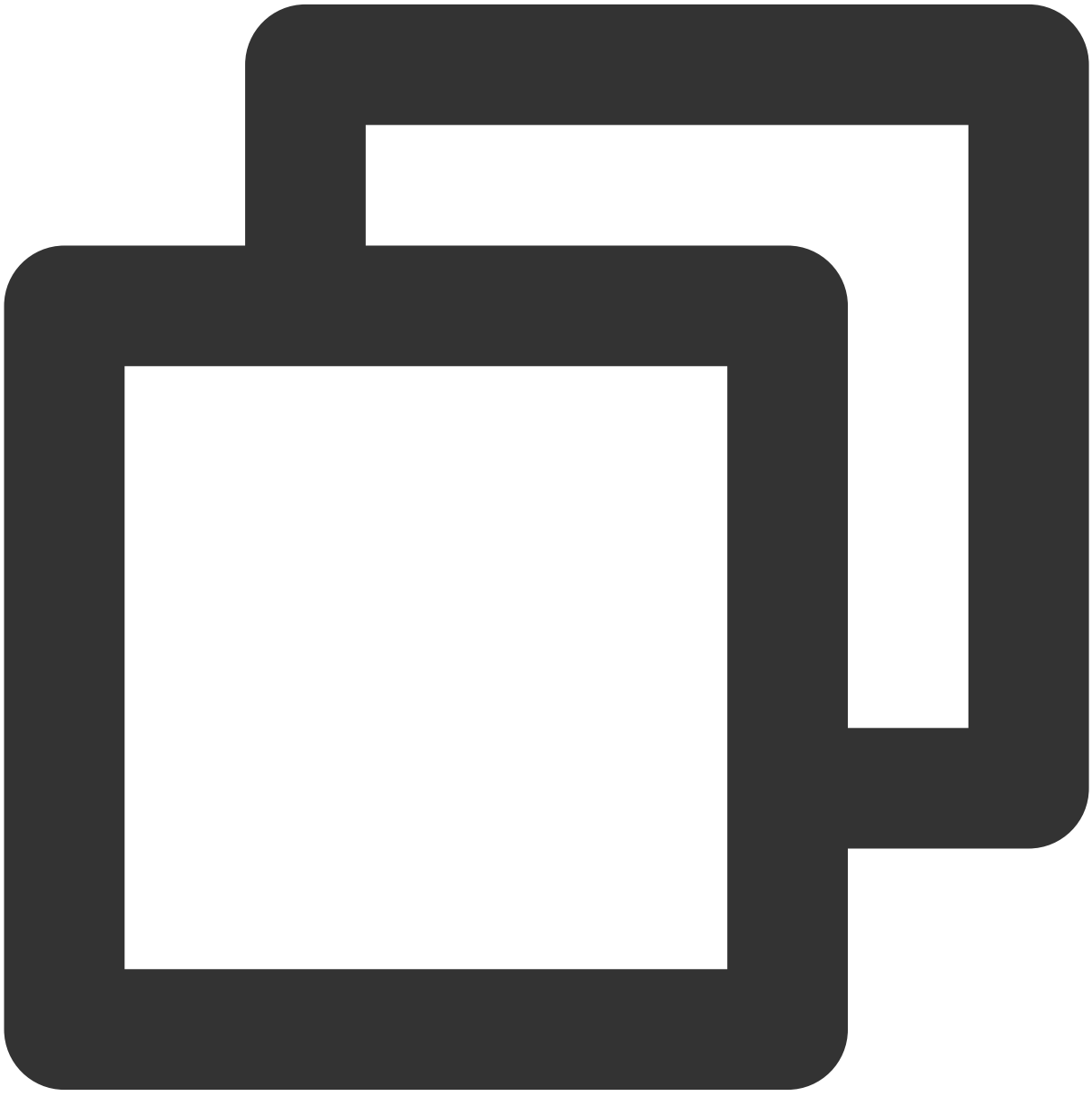
role_id is used to identify different roles of an account under one server. Enter "" if there is only one role.

open_id is assigned by the specific operating platform to uniquely identify users.

Return value: 0 indicates a successful call.

API for switching from foreground to background

Function signature



```
int onAppPause ();
```

If the App switches from foreground to background, the game is inactive.
Return value: 0 indicates a successful call.

API for switching from background to foreground

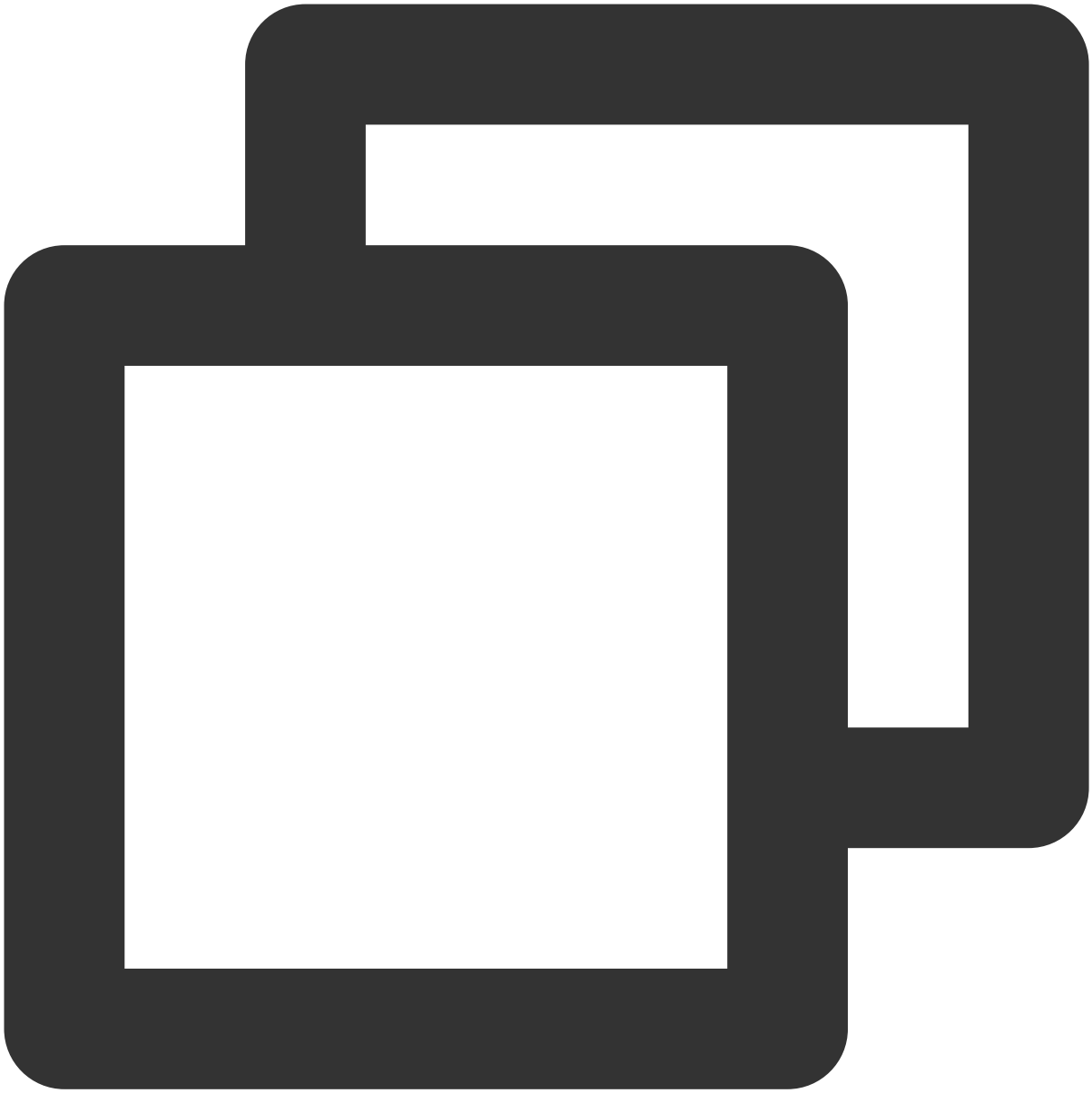
Function signature

If the App switches from background to foreground, the game is active.
Return value: 0 indicates a successful call.

When to call the function

1. Call TP2Sdk.initEx immediately after the game is launched. Parameters are gameID and appKey. Calling the security API function earlier can better protect the game process.
2. TP2Sdk.onUserLogin is called after the game is authorized by the user to access its login information. If the game has set world_id and role_id, then call the TP2Sdk.onUserLogin function after obtaining both world_id and role_id. During gameplay, if you need to retrieve the user's login information in situations like when the network is disconnected or the user logged out and needs to re-login, you will need to call the function again. The parameter to be passed is the user's account information, which can be customized.
3. TP2Sdk.onAppPause is called when the game switches from foreground to background.
4. TP2Sdk.onAppResume is called when the game switches from background to foreground.

Sample Code

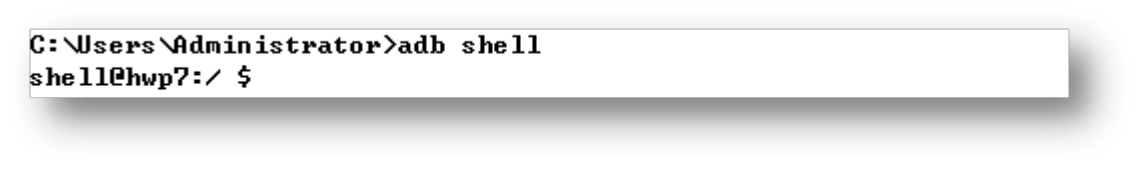


```
public void onCreate()
 {
//Called immediately after the game is launched.
TP2Sdk. initEx(9000, "d5ab8dc7ef67ca92e41d730982c5c602");
int accountType = ENTRYID.ENTRY_ID_QZONE; /* Account type */
int worldId = 1; /* Server id*/
String openId = "B73B36366565F9E02C752"; /* Platform-specific user ID */
String roleId = "paladin"; /* Role id*/
// Called when the user logs in the game
TP2Sdk.onUserLogin(accountType, worldId, openId, roleId);
}
```

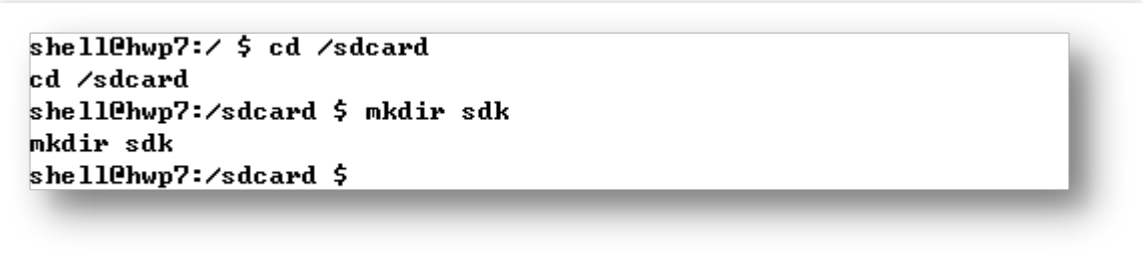
```
// Game switches from foreground to background
public void onPause()
 {
super.onResume();
TP2Sdk.onPause();
}
// Game switches from background to foreground
public void onResume()
{
super.onResume();
TP2Sdk.onResume();
}
```

Verifying Whether the SDK is Integrated Correctly

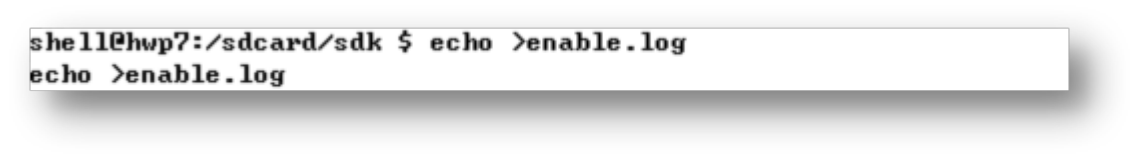
1. Connect your Android phone to a Windows PC via a USB cable. After the connection is successful, log in to the Android ADB console using Windows CMD, as shown below:



2. Type cd /sdcard, press enter, then type mkdir sdk, and press enter, to create the /sdcard/sdk directory. If the directory already exists, a prompt indicating "mkdir failed for /sdcard/sdk. File exists" will appear, and you can proceed to the next step:



3. Type cd /sdcard/sdk to enter the directory, and type echo>enable.log to create an empty file enable.log.



Files under the directory created by the shell may not be accessed on some models. In this case, change the /sdcard/sdk directory with root user or use another mobile phone.

```
shell@hwp7:/sdcard $ su
su
root@hwp7:/mnt/shell/emulated/0 # chmod -R 777 /sdcard/sdk
chmod -R 777 /sdcard/sdk
root@hwp7:/mnt/shell/emulated/0 #
```

4. Start and log in to the game, check whether tp2.log and tlog.log are generated in the /data/data/log directory, as shown below:

```
shell@hwp7:/sdcard/sdk $ ls -l
ls -l
-rwxrwx--- root     sdcard_r          1 2016-04-20 22:37 enable.log
-rwxrwx--- root     sdcard_r       3324 2016-04-20 22:33 tlog.log
-rwxrwx--- root     sdcard_r       4151 2016-04-20 22:33 tp2.log
```

If no log is generated, check whether you have the read/write permission to /sdcard/sdk and enable.log. This directory cannot be read/written on a small number of models. Use another model for testing or change /sdcard/sdk to /data/data/log with root user.

Note:

enable.log is only used for testing purposes.

5. Open the tp2.log file, and check whether it contains the information of three native APIs **tp2_sdk_init_ex, tp2_setuserinfo and setgamestatus** as well as the jar packet's version number **jar_ver**. Only when all the above conditions are met, can the security SDK run properly. setgamestatus:1 indicates that the current process is running in the foreground, and setgamestatus:2 indicates that the current process is running in the background.

Verify whether the API is correctly called by switching the App between foreground and background, and also check whether the userinfo is entered correctly.

```
root@hwG750-T01:/sdcard/sdk # cat tp2.log
[17:41:04] tp2_sdk_init_ex, ver:1.6.0
[17:41:04] {
[17:41:04]   app_id:8888
[17:41:04]   app_key:d5ab8dc7ef67ca92e41d730982c5c602
[17:41:04] }
[17:41:04] tp2_setuserinfo
[17:41:04] {
[17:41:04]   account_type:1
[17:41:04]   world_id:101
[17:41:04]   open_id:CF086A77B355AD8CFFEA6B94337EDFE4
[17:41:04]   role_id:Paladin
[17:41:04] }
[17:41:05] setgamestatus:1
[17:41:05] jar_ver:1.6.0
[17:41:05] jar_ns:!!com.tencent.tersafe2.util.JNCTask
[17:41:05] VER:1.6.0(Android), 20161230
[17:41:05] {
[17:41:05] }
[17:41:15] setgamestatus:2
[17:41:20] setgamestatus:1
```

6. Open tlog.log to view the data sent by the security SDK, as shown below:

```
shell@hwp7:/sdcard/sdk $ cat tlog.log
cat tlog.log
010000014F10000000005B2E1E0D0000232800000000000000000000000000000000
3635653664626136623200000901000000010000000000101000010A0023000000000
4645413642393433333745444645340000C50000000100C5010A00090000000000
877C9C13CF18874A86921AFF049C7F0484E78957B75C6CDC8F52143DB3841C986
D9861D94B644DD1DBAFBF2DB1FBD0AA7C0B5E7954EEEDB3AF461078F28AB87325
44AD733953D57651AB8F8C68CE1BBEAC78D5CCF0CBAB67E980AA0AA173EDA3059
0100000184010001C1045B2E1E0D00002328E9991757C69101001BC8958111FDA
3635653664626136623200001C1050001C08AA2F9010009C7000001000001280
18323F2928CA8D86F74CF560972AF08C7A62409D80074C9F527CD9ACEA6319B47
288DFE10E7C8C9BFF676A9C8D249009398AD1493658515DE4A7E77E83834CCD72
759C2BC61CBF52339B9430D4287D6924F39727AA83C49280225D9EFB18CFA4ACF
F128901E90B0D67AC7F6BC41592E52AE6DE58BA3F28FCFA9A09A87B1D521C94FF
095125F34DEB24BFD02849B9B19EBBA3F7B67CA8DF4534814B13DB89
010000014F10000000005B2E1E0D0000232800000000000000000000000000000000
3635653664626136623200000901000000010000000000101000010A0023000000000
4645413642393433333745444645340000C50000000100C5010A00090000000000
7A7115C5709B182B6FF243C07E8B6E053EC92AB29619322FF76125CAAFBFC7827
EA90EDA330ECB75CD1ADB2FAF0103C50DAE4A43F0DA7C57724F07907FF25CC708
C33249AC56E17BA2DE213A05B74E2D6B9DF0C2EF9975D4ED13583E8A3889FC990
```

In addition to reporting some basic process information during initialization, the security SDK also sends data according to the results of periodic security scanning, such as the incorrect signature of the App certificate, modification of memory data, a running add-on process, etc. The tlog.log records the data (only generated during testing) sent by the SDK. Generally, the data size per hour is about 20 KB. You can check the size of tlog.log to calculate the volume of data sent by the security SDK.

Integration Guidelines for C#

Last updated：2023-12-08 17:48:43

Preparations

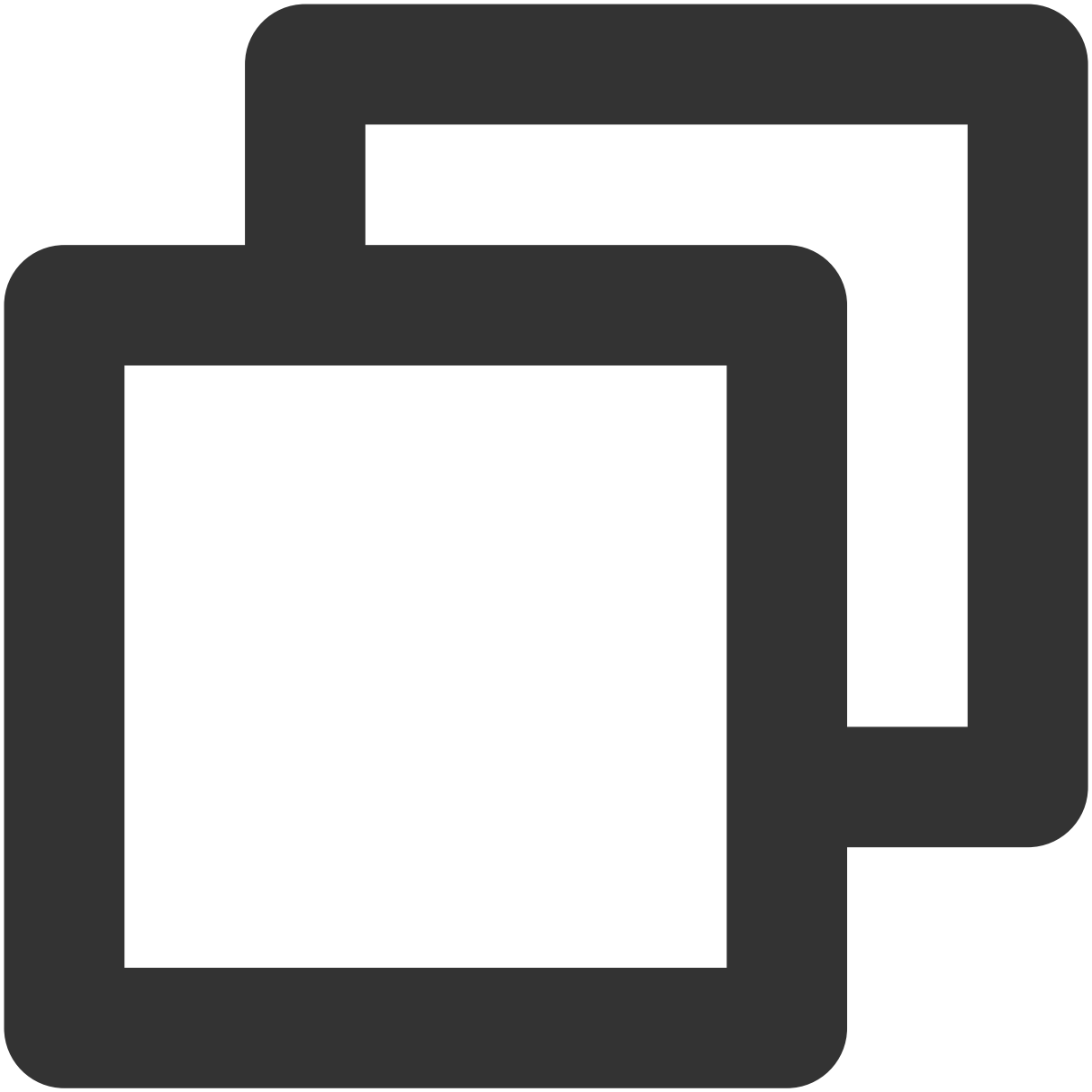
Developers need to complete the following steps when integrating the security SDK:

1.1 Copy the SDK dynamic library to the specified project directory associated with the game platform and the CPU architecture.

1.2 Call the SDK API based on the game_id and user's login information.

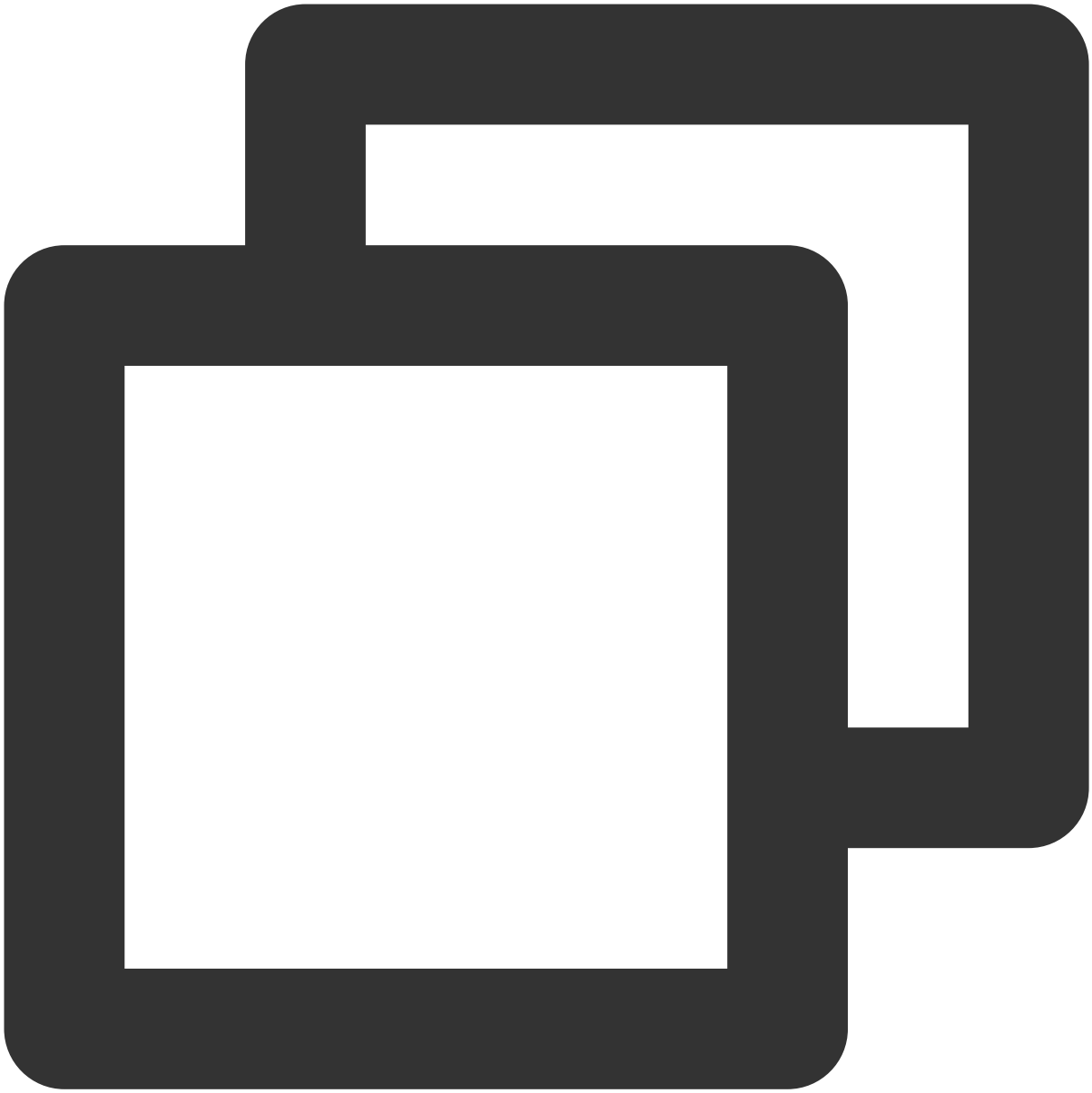
1.3 Verify whether the SDK is integrated correctly.

The following files are required for the integration of the security SDK to Android OS written in C/C++:



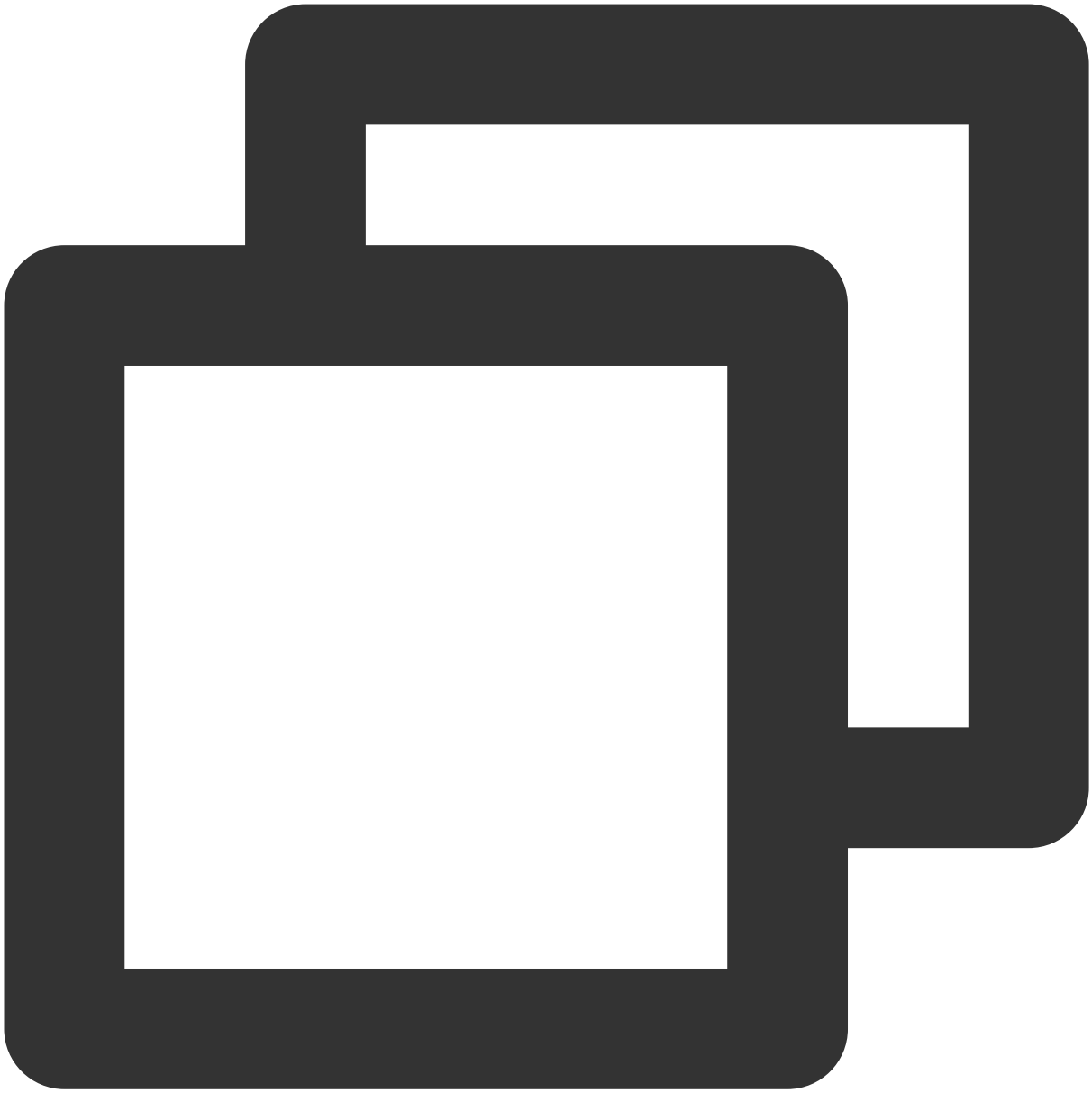
```
tp2.cs
tp2.jar (Android)
libtersafe2.so (Android)
```

Permissions required:



```
<uses-permission android:name="android.permission.ACCESS_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.GET_TASKS" />
<uses-permission android:name="android.permission.INTERNET" />
```

SDK API functions:



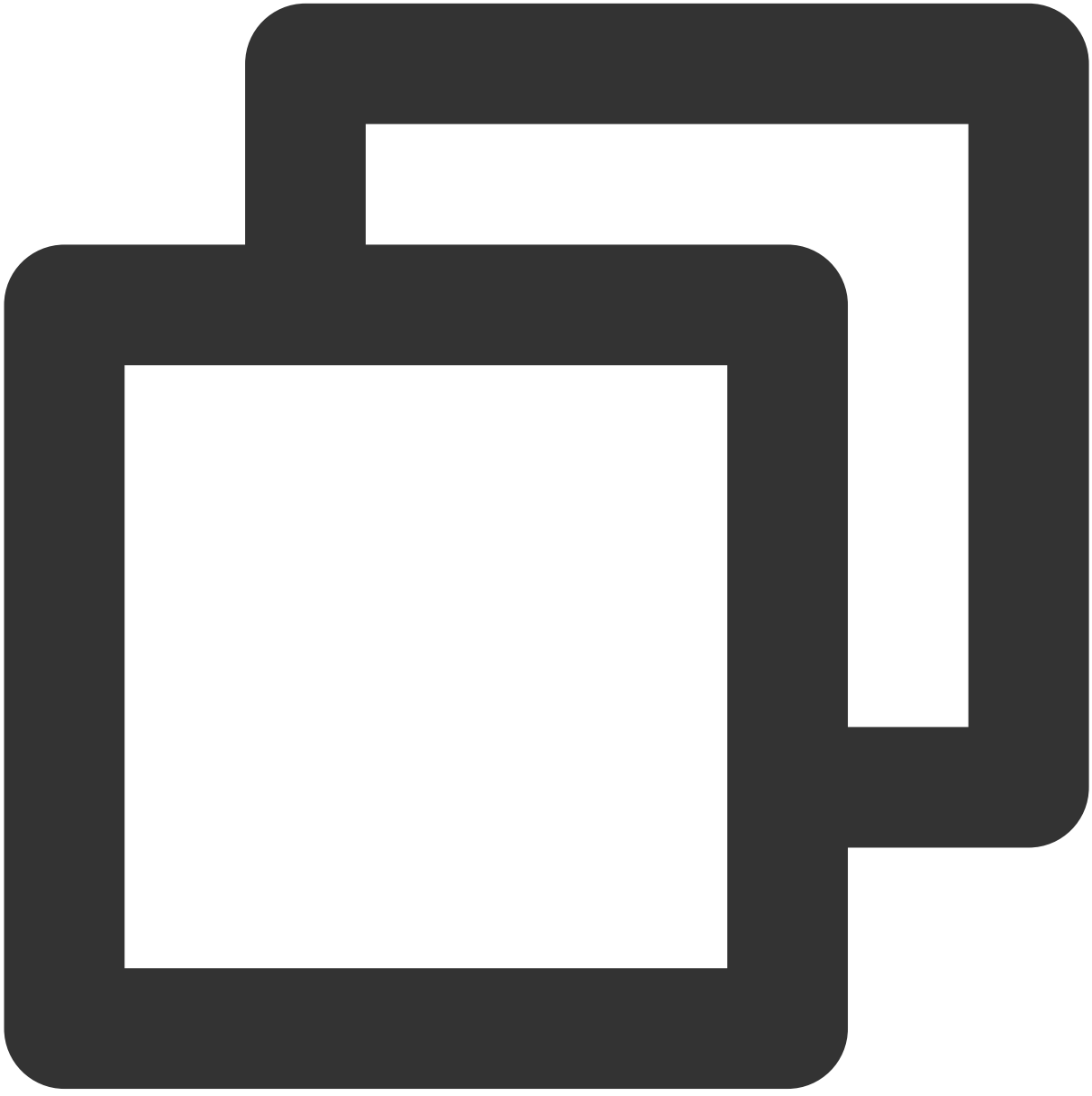
```
Initialization API: Tp2SdkInitEx
User login API: Tp2UserLogin
API for switching between foreground and background: Tp2SetGamestatus
```

Adding SDK Files to the Project

Add files

1. Move tp2.cs from the sdk\\android\\c# directory into the project's Assets directory.
2. Move tp2.jar from the sdk\\android\\c# directory into the project's Assets\\Plugins\\Android directory.
3. Multi-CPU:

When we use Unity5.0, for example, if the game supports multi-CPU architecture (currently only arm-v7a and x86 are supported) on Android, copy libtersafe2.so in both armeabi and x86 folders under the sdk\\android\\c#\\lib directory to the following directories respectively:



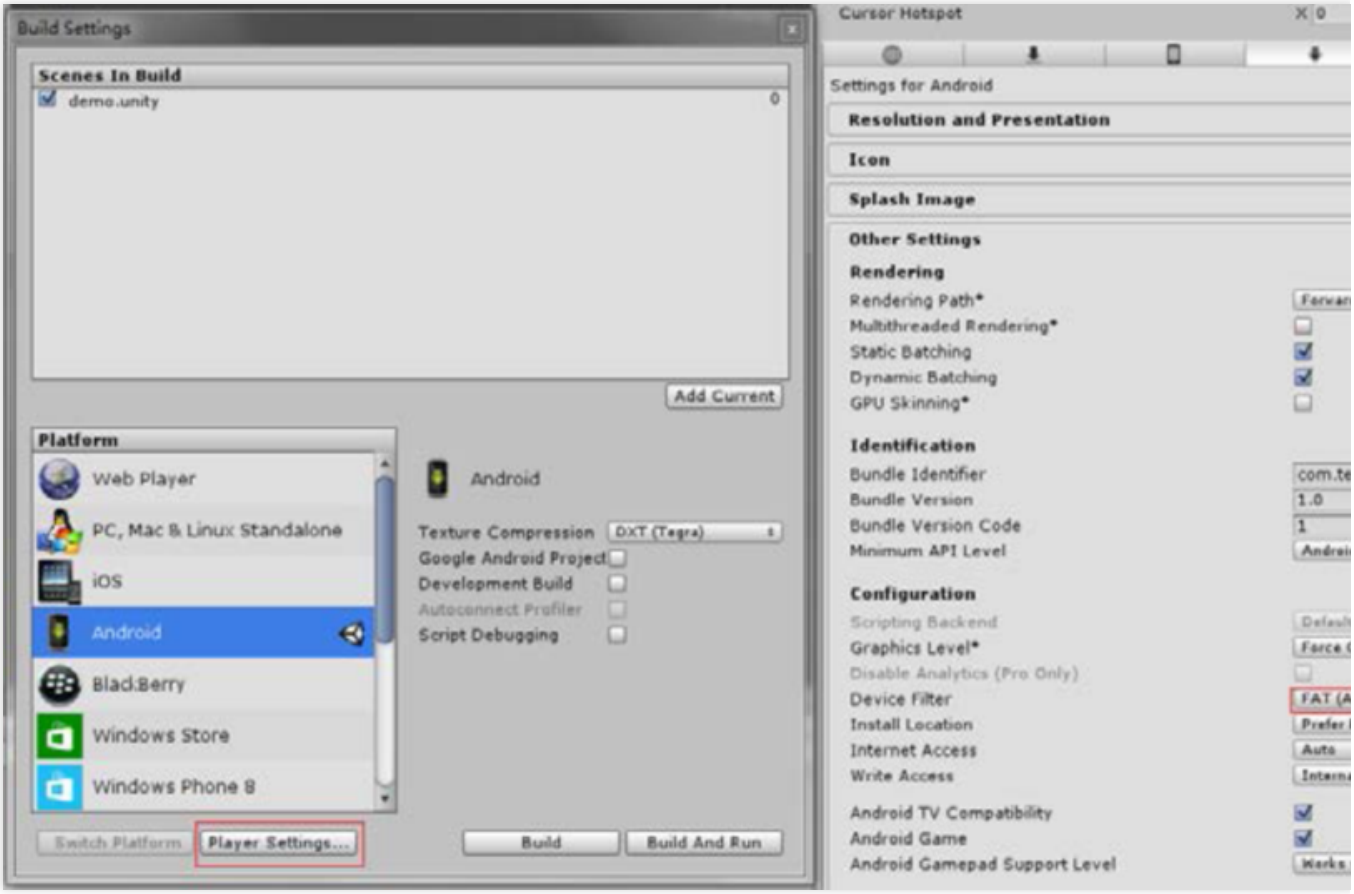
```
Assets/Plugins/Android/libs/armeabi-v7a/
Assets/Plugins/Android/libs/x86/
```

4. Single CPU architecture:

When we use Unity4.5, for example, if the game only supports arm-v7, move tp2.jar provided by SDK and libtersafe2.so from the armeabi-v7a directory to the /Assets/Plugins/Android/ directory.

Setting of project attributes

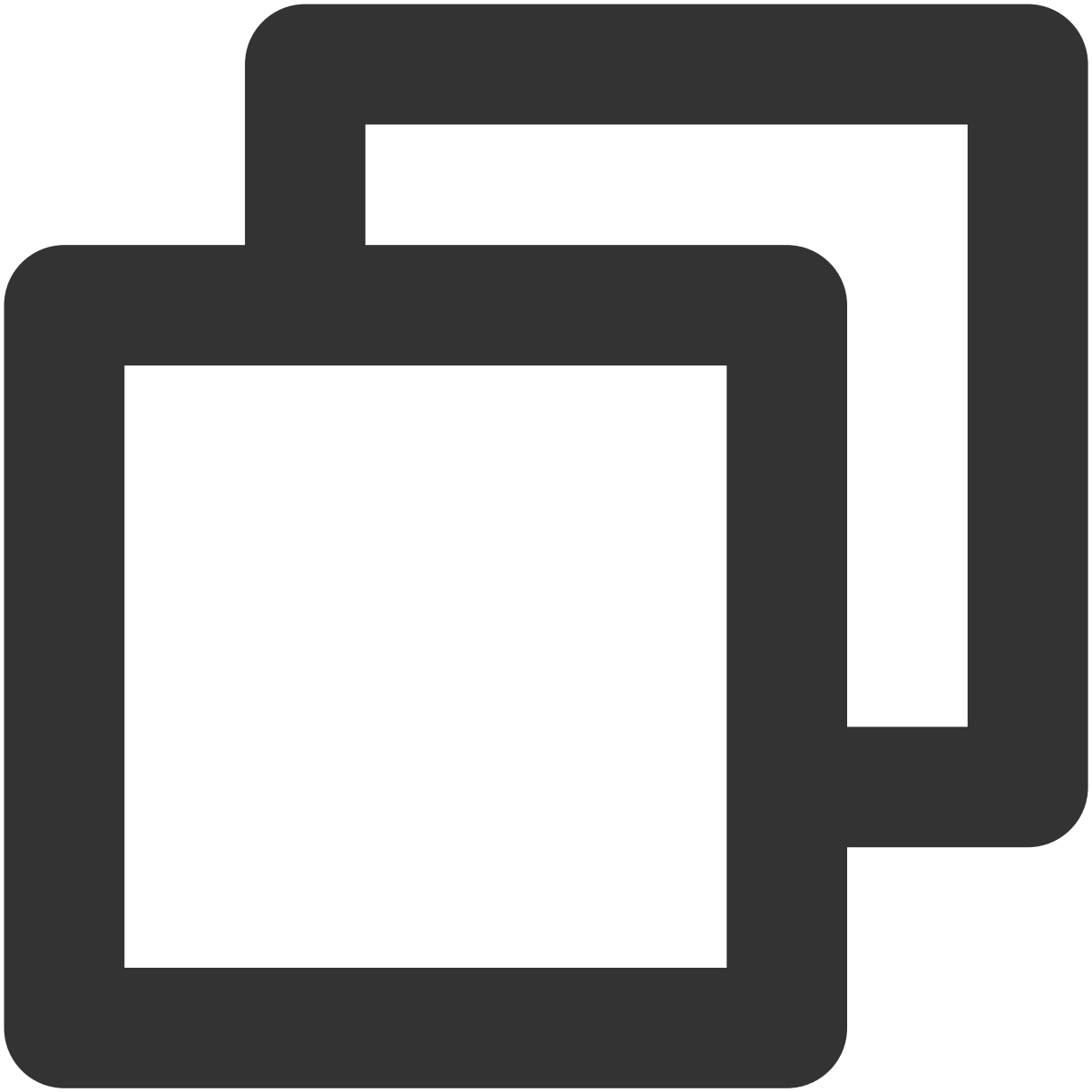
If multi-CPU architecture is supported, select "File" -> "Build Settings" -> "Player Settings" -> "Other Settings" -> "Device Filter" -> "FAT(ARMv7+x86)".



Calling SDK API

Initialization function

Function signature



```
void Tp2SdkInitEx (int gameId, string appKey);
```

Parameter description

Parameter	Required	Description
gameId	Yes	The game_id assigned by Tencent Cloud
appKey	Yes	The game_key assigned by Tencent Cloud, which corresponds

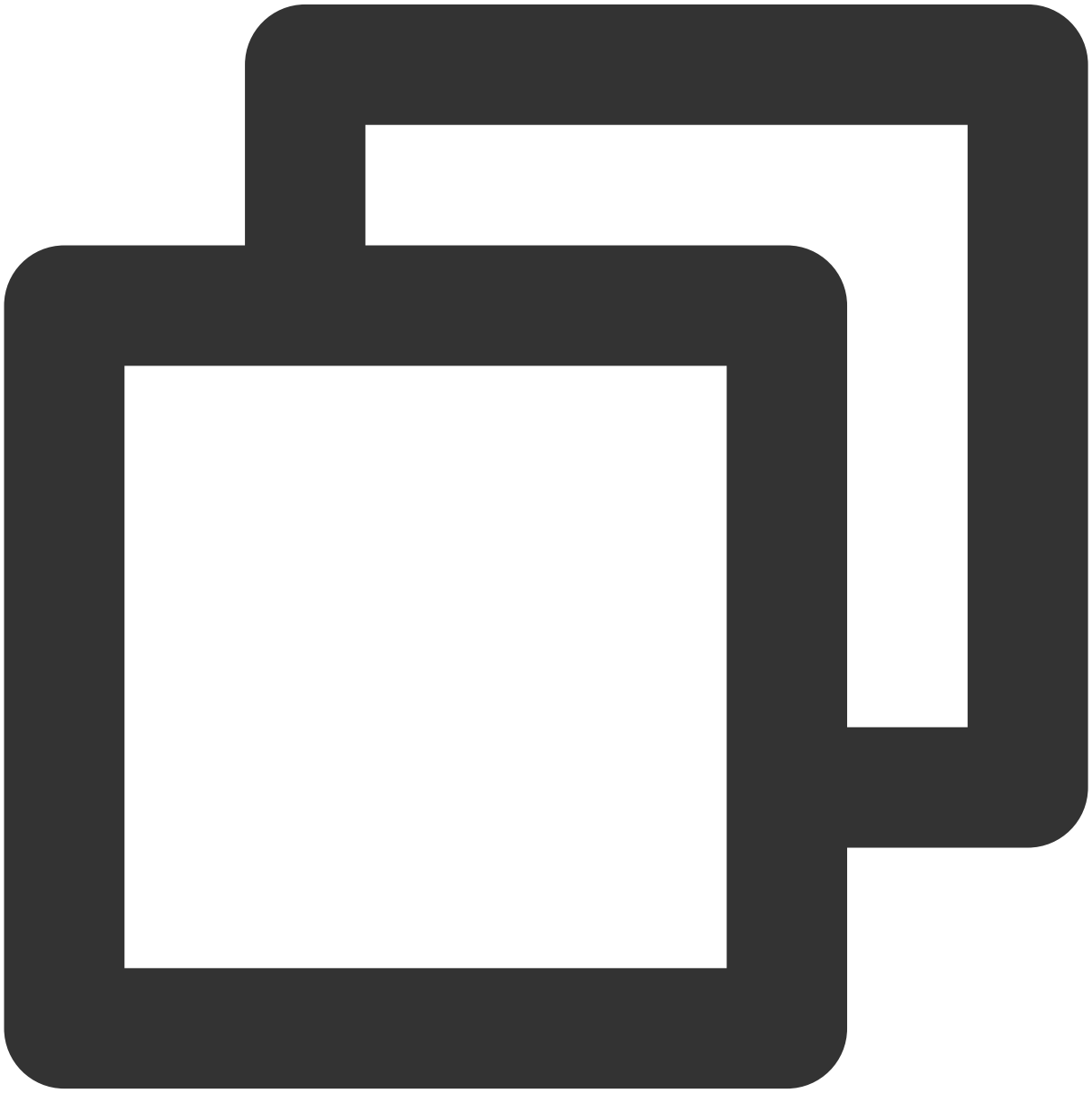
		to the game_id.

Both gameID and appKey are automatically generated after a new game has been registered on the Tencent Cloud official website (xxxxxxxxxxx).

Return value: 0 indicates a successful call.

Set the user information

Function signature

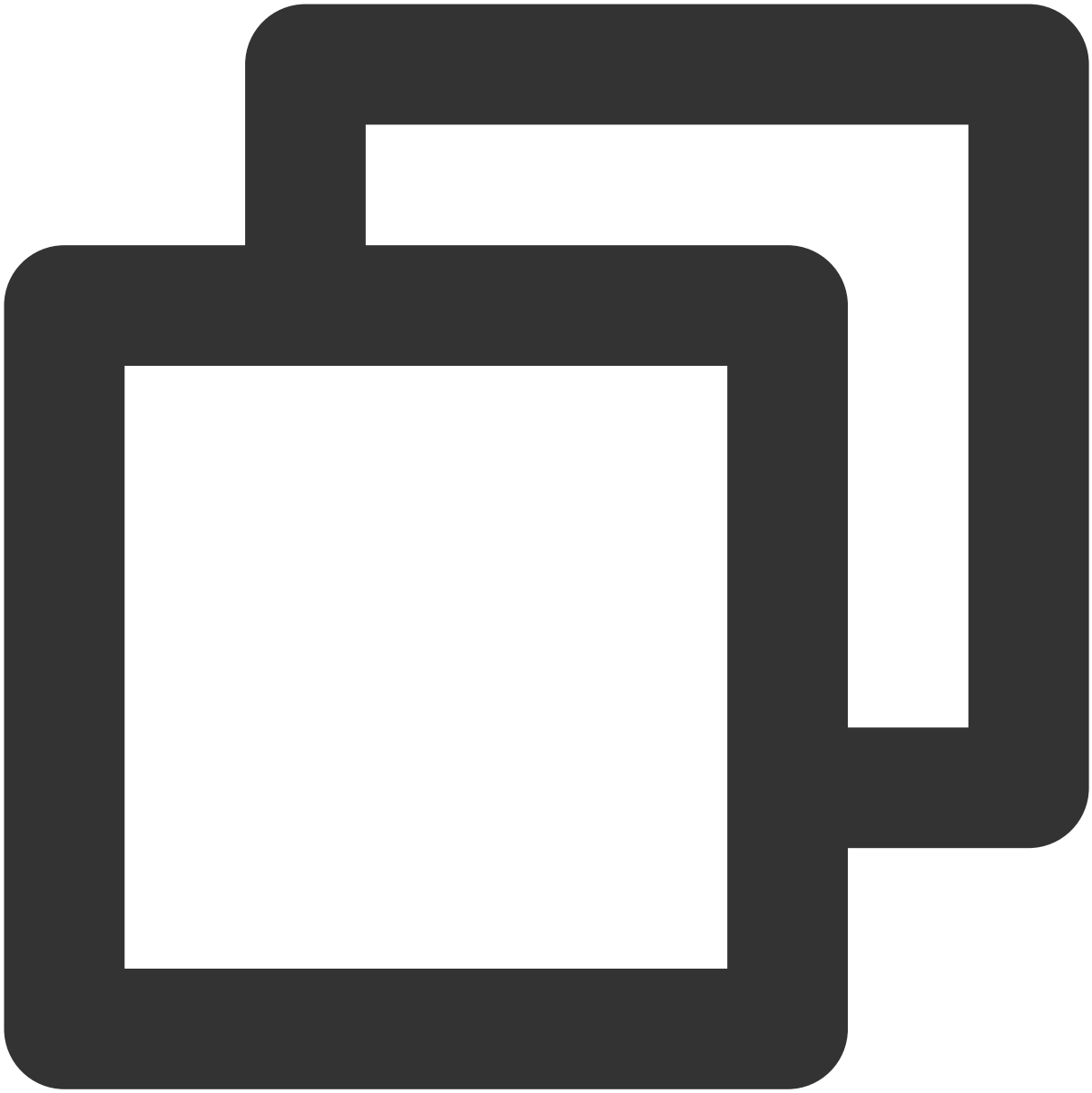


```
void Tp2UserLogin (int accountType, int worldId, String openId, String roleId);
```

Parameter description

Parameter	Title 2
account_type	Account type associated to the operating platform. Refer to TssSdkEntryId below.
world_id	Information on the server where user's game role is created
open_id	User's unique ID, which can be a custom string. This is required for penalties purposes.
role_id	Identifies the roles created by a user

For the account_type, 1 indicates QQ (default), 2 indicates WeChat, and 99 indicates other platforms. Chinese and international mainstream platforms, please refer to the following values.



```
enum TssSdkEntryId
{
ENTRY_ID_QZONE = 1, // QQ
ENTRY_ID_MM = 2, // WeChat
ENTRT_ID_FACEBOOK = 3, // facebook
ENTRY_ID_TWITTER = 4, // twitter
ENTRY_ID_LINE = 5, // line
ENTRY_ID_WHATSAPP = 6, // whatsapp
ENTRY_ID_OTHERS = 99, // Other platforms
};
```

world_id is defined by the game. Enter 0 if the game has only one server.

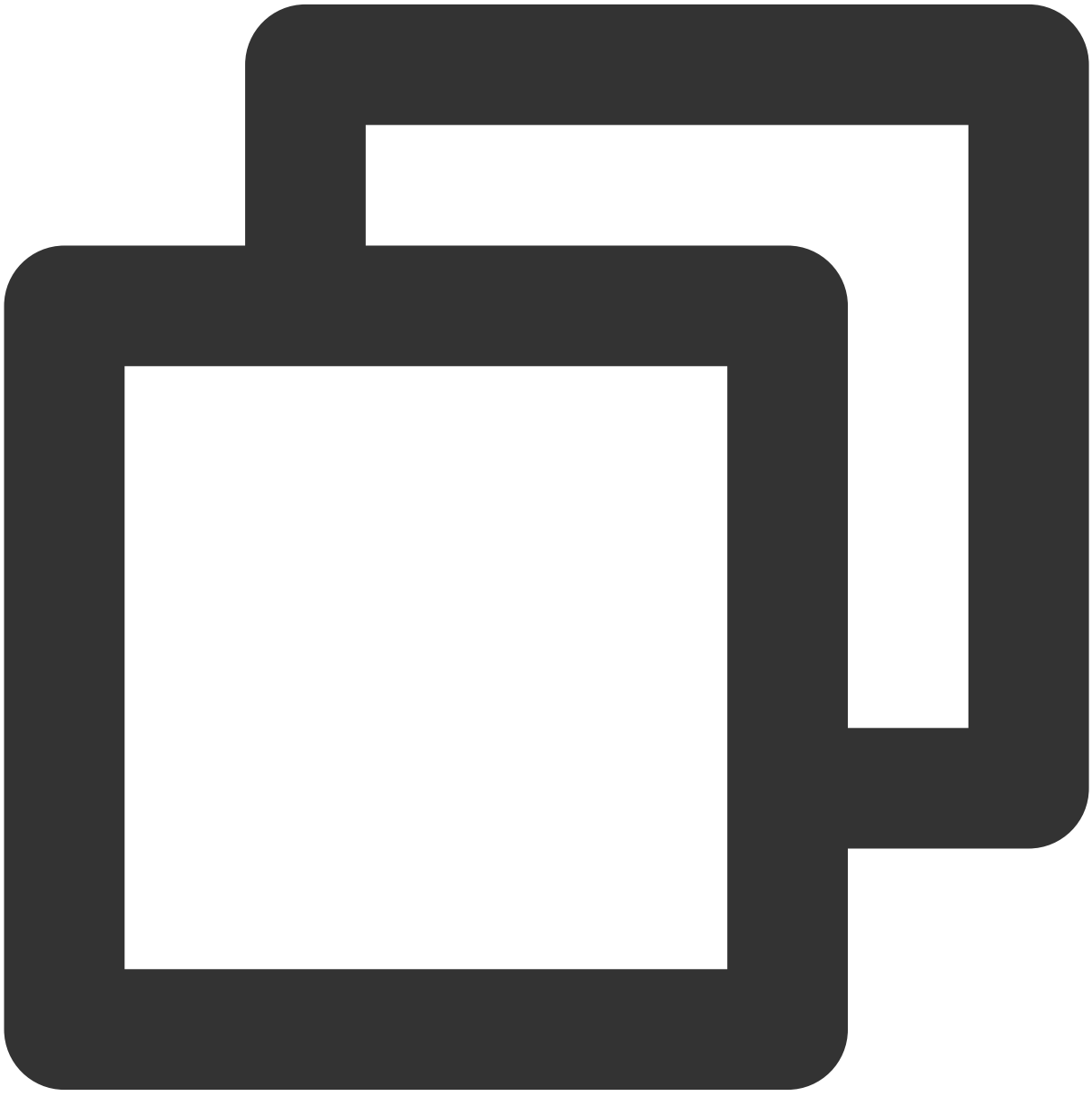
role_id is used to identify different roles of an account under one server. Enter "" if there is only one role.

open_id is assigned by the specific operating platform to uniquely identify users.

Return value: 0 indicates a successful call.

Set the game status

Function signature

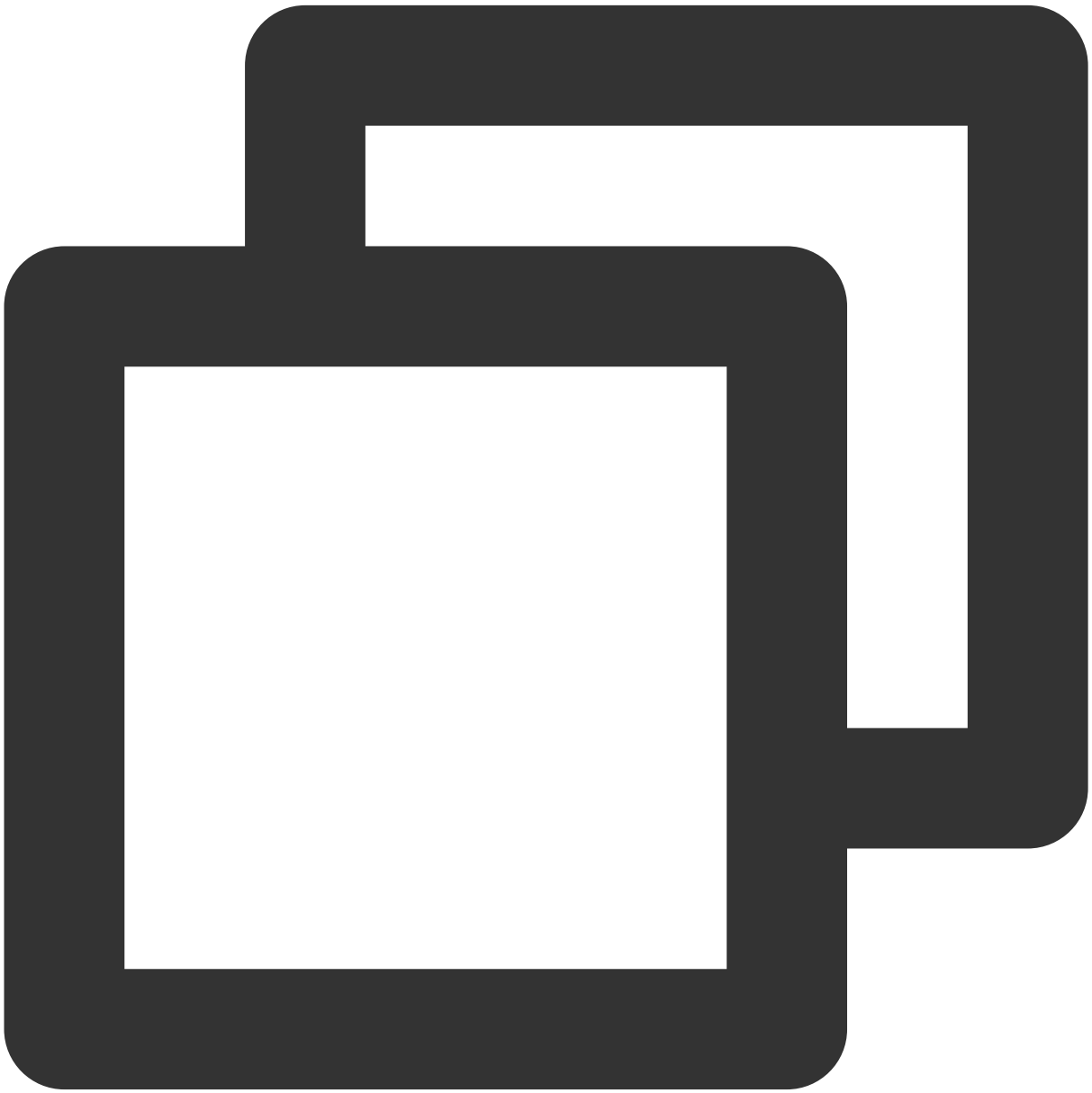


```
void Tp2SetGamestatus (Tp2Status status);
```

Parameter description

Parameter	Description
status	Foreground Tp2Status. FRONTEND Background Tp2Status. BACKEND

Enumeration types



```
public enum Tp2Status
{
```

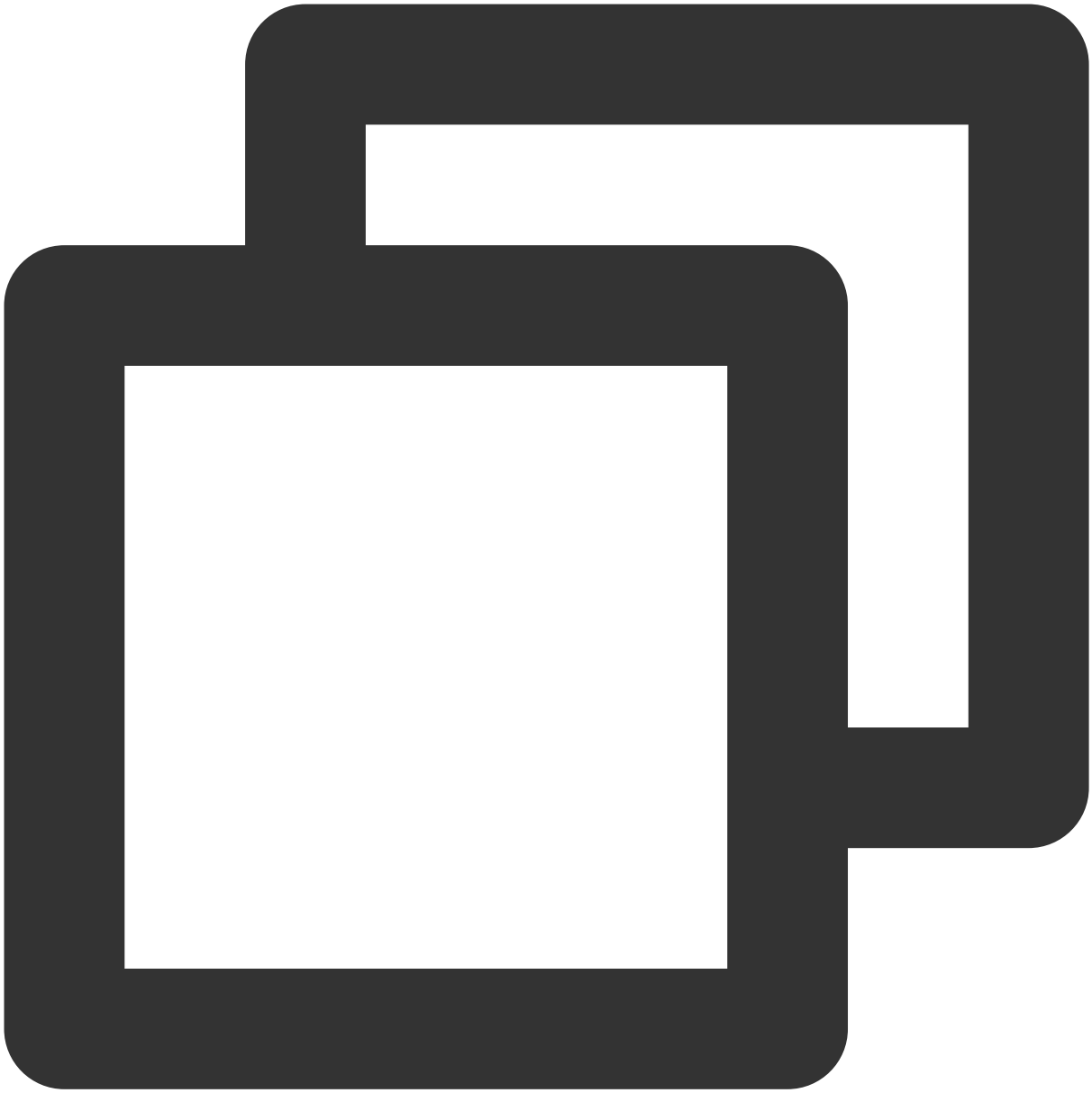
```
FRONTEND = 1, // Foreground
BACKEND = 2 // Background
}
```

Return value: 0 indicates a successful call.

When to call the function

1. Call Tp2SdkInitEx immediately after the game is launched. Parameters are gameID and appKey. Calling the security API function earlier can better protect the game process.
2. Tp2UserLogin is called after the game is authorized by the user to access its login information. If the game has set world_id and role_id, then call the Tp2UserLogin function after obtaining both world_id and role_id. During gameplay, if you need to retrieve the user's login information in situations like when the network is disconnected or the user logged out and needs to re-login, you will need to call the function again. The parameter to be passed is the user's account information, which can be customized.
3. Tp2SetGamestatus is called when the game switches between foreground to background. When the game switches from background to foreground, the parameter is set to Tp2Status. FRONTEND, and when the game switches from foreground to background, the parameter is set to Tp2Status. BACKEND. Some of the SDK functions stop running when the game switches to background, so the API may affect the normal running of SDK functions.

Sample Code



```
void Awake ()
{
Tp2Sdk.Tp2SdkInitEx(8888, "d5ab8dc7ef67ca92e41d730982c5c602");
}
// Called after the user logs in
void Start ()
{
int accountType = (int)Tp2Entry.ENTRY_ID_QZONE ; /* Account type */
int worldId = 100; /* Server id*/
string openId = "B73B36366565F9E02C752"; /* User id*/
string roleId = "paladin"; /* Role id*/
```

```
Tp2Sdk.Tp2UerLogin(accountType, worldId, openId, roleId);
}
// Called when the game switches between foreground and background
void OnApplicationPause (bool pause)
 {
if (pause)
 {
Tp2Sdk.Tp2SetGamestatus(Tp2Status. BACKEND); // Switching to background
}
else
{
Tp2Sdk.Tp2SetGamestatus(Tp2Status. FRONTEND); // Switching to foreground
}
}
```

Verifying Whether the SDK is Integrated Correctly

1. Connect your Android phone to a Windows PC via a USB cable. After the connection is successful, log in to the Android ADB console using Windows CMD, as shown below:

2. Type cd /sdcard, press enter, then type mkdir sdk, and press enter, to create the /sdcard/sdk directory. If the directory already exists, a prompt indicating "mkdir failed for /sdcard/sdk. File exists" will appear, and you can proceed to the next step:

3. Type cd /sdcard/sdk to enter the directory, and type echo>enable.log to create an empty file enable.log

```
shell@hwp7:/sdcard/sdk $ echo >enable.log
echo >enable.log
```

Files under the directory created by the shell may not be accessed on some models. In this case, change the /sdcard/sdk directory with root user or use another mobile phone.

```
shell@hwp7:/sdcard $ su
su
root@hwp7:/mnt/shell/emulated/0 # chmod -R 777 /sdcard/sdk
chmod -R 777 /sdcard/sdk
root@hwp7:/mnt/shell/emulated/0 #
```

4. Start and log in to the game, check whether tp2.log and tlog.log are generated in the /data/data/log directory, as shown below:

```
shell@hwp7:/sdcard/sdk $ ls -l
ls -l
-rwxrwx--- root     sdcard_r        1 2016-04-20 22:37 enable.log
-rwxrwx--- root     sdcard_r     3324 2016-04-20 22:33 tlog.log
-rwxrwx--- root     sdcard_r     4151 2016-04-20 22:33 tp2.log
```

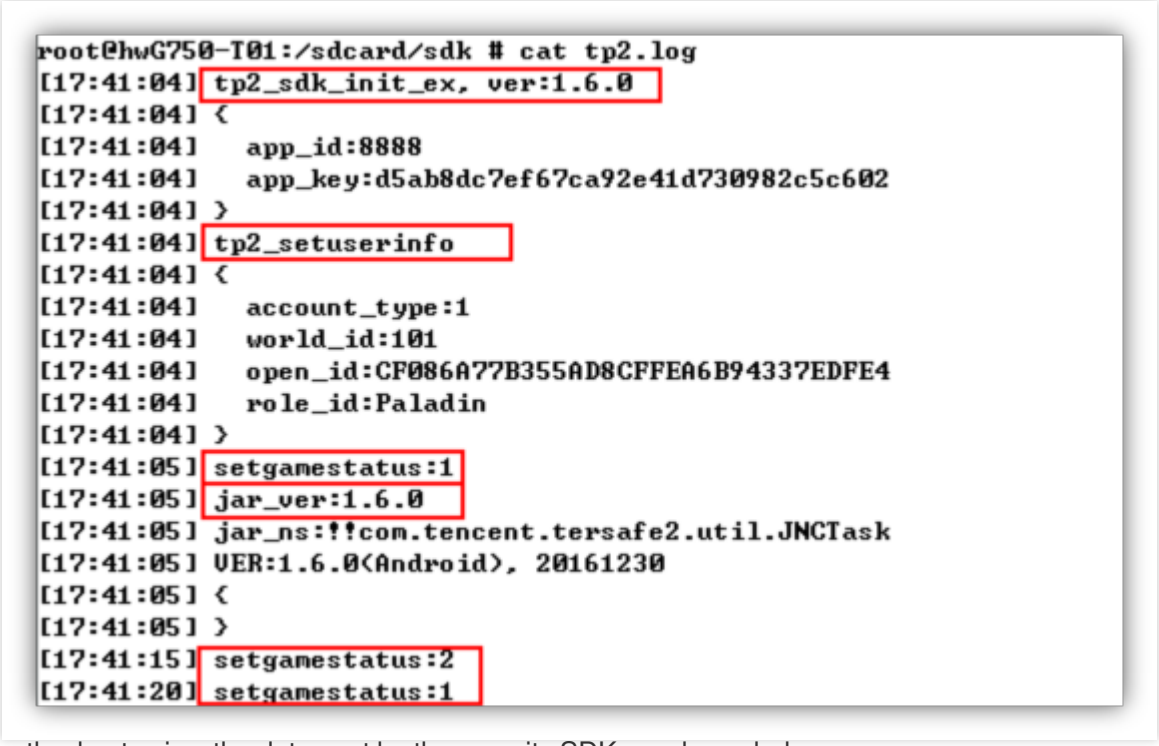
If no log is generated, check whether you have the read/write permission to /sdcard/sdk and enable.log. This directory cannot be read/written on a small number of models. Use another model for testing or change /sdcard/sdk to /data/data/log with root user.

Note:

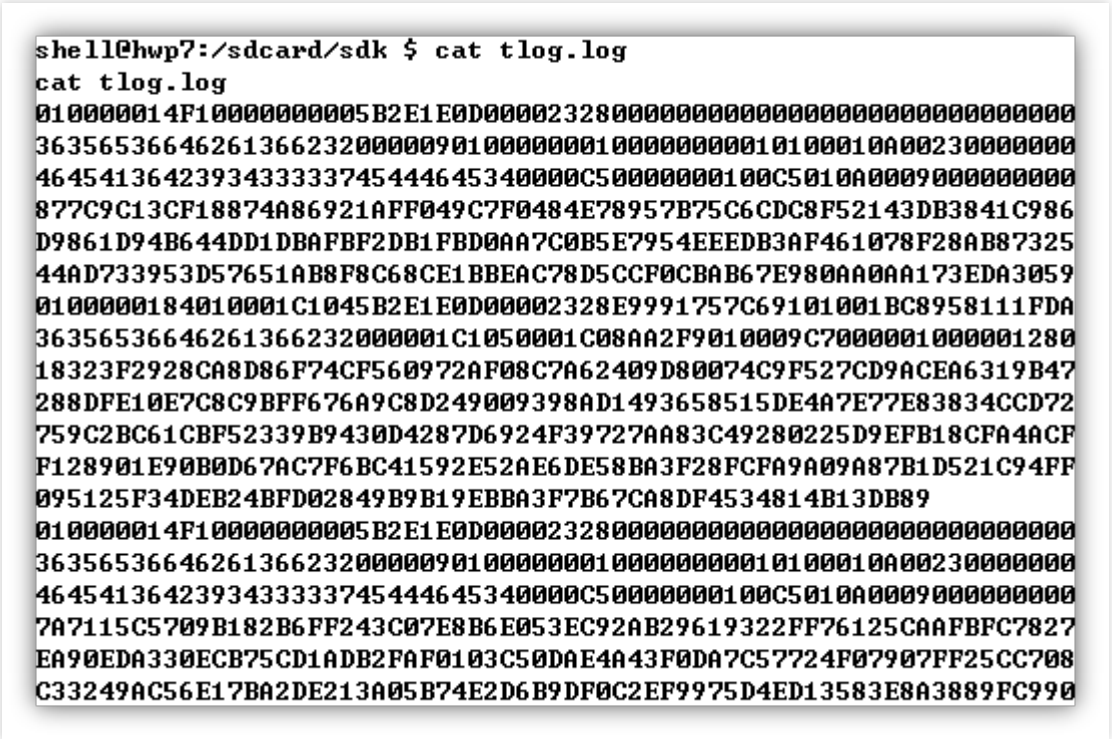
enable.log is only used for testing purposes.

5. Open the tp2.log file, and check whether it contains the information of three native APIs **tp2_sdk_init_ex, tp2_setuserinfo and setgamestatus** as well as the jar packet's version number **jar_ver**. Only when all the above conditions are met, can the security SDK run properly. setgamestatus:1 indicates that the current process is running in the foreground, and setgamestatus:2 indicates that the current process is running in the background.

Verify whether the API is correctly called by switching the App between foreground and background, and also check whether the userinfo is entered correctly.



6. Open tlog.log to view the data sent by the security SDK, as shown below:



In addition to reporting some basic process information during initialization, the security SDK also sends data according to the results of periodic security scanning, such as the incorrect signature of the App certificate, modification of memory data, a running add-on process, etc. The tlog.log records the data (only generated during testing) sent by the SDK. Generally, the data size per hour is about 20 KB. You can check the size of tlog.log to calculate the volume of data sent by the security SDK.