# 360 Android Application Access Document

<table>
<thead>
<tr>
<th>Confirmed by</th>
<th>Revision No.</th>
<th>Revision Time</th>
<th>Content</th>
<th>Modifier</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2</td>
<td>September 19, 2012</td>
<td>Draft</td>
<td>Dong Wei, Cai Wenhua, Zhang Yuzhi, She Shuai</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>November 06, 2012</td>
<td>Third-time Doc Modification</td>
<td>She Shuai</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>November 07, 2012</td>
<td>Third-time Doc Update</td>
<td>Guo Yizun, He Qian, Gai Xiangsheng</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>November 08, 2012</td>
<td>Third-time Doc Update</td>
<td>She Shuai</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>November 22, 2012</td>
<td>Forth-time Doc Modification</td>
<td>Guo Yizun</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>December 12, 2012</td>
<td>Fifth-time Doc Modification</td>
<td>Sun Zhibin, She Shuai</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>January 09, 2013</td>
<td>User login mode can only be “code” mode. Parameters for payment 360SDK interface screen direction and if the background is transparent are added. Example code for the Demo project is updated.</td>
<td>Sun Zhibin, Zhang Yuzhi, She Shuai</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>March 15, 2013</td>
<td>Seventh-time Doc Modification, including the 360SDK interface form modification. Modified interfaces: login interface, payment interface. Added interfaces: account switch interface, logout interface, bbs interface, real name registration interface, addiction prevention inquiry interface,</td>
<td>Lin Xin, He Qian, Zhang Lingjun</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Description</td>
<td>Author(s)</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------</td>
<td></td>
</tr>
<tr>
<td>0.7.2</td>
<td>April 02, 2013</td>
<td>Add: express game, game trial, game trial registration; Modify: payment interface is modified to the effect that the payment channel can be customized.</td>
<td>Lin Xin</td>
<td></td>
</tr>
<tr>
<td>0.7.3</td>
<td>April 12, 2013</td>
<td>Seventh-time Doc Modification</td>
<td>Lin Xin</td>
<td></td>
</tr>
<tr>
<td>0.7.4</td>
<td>April 19, 2013</td>
<td>Add: GET_TASKS; Add: READ_SMS; Add: WAKE_LOCK authorities; Add: Deposit card (express payment)</td>
<td>Kang Yonggen, Lin Xin</td>
<td></td>
</tr>
<tr>
<td>0.7.5</td>
<td></td>
<td>Add: RECEIVE_SMS authorities Add: READ_EXTERNAL_STORAGE authorities; Add: meta from AppId to AndroidManifest; Add: mobile number binding, suspension window setting interface.</td>
<td>Lin Xin</td>
<td></td>
</tr>
<tr>
<td>0.7.6</td>
<td></td>
<td>Modify: Screenshot processing under non-root; Add: Customer service function;</td>
<td>Zhao Kai</td>
<td></td>
</tr>
<tr>
<td>0.8.2</td>
<td>2013-6-24</td>
<td>Modify: Rename &quot;Express payment&quot; into &quot;Deposit card&quot; Delete: 360 currency card payment Add: UnionPay payment Add: Customizable default payment channel Add: VIBRATE; Add: SMS payment interface; Add: Screenshots posting interface</td>
<td>Kang Yonggen, Zhao Kai, Zhang Lingjun</td>
<td></td>
</tr>
<tr>
<td>0.8.4</td>
<td>2013-7-26</td>
<td>Add: Credit card payment Add: ACCESS_COARSE_LOCATION authority; Add: ACCESS_FINE_LOCATION authority; Modify: The return data formats of Login and Account switch interfaces. Delete: 360SDK game trial, express game and game trial registration interfaces</td>
<td>Zhang Lingjun, Lin Xin</td>
<td></td>
</tr>
<tr>
<td>0.8.6</td>
<td>2013-8-19</td>
<td>Modify: AndroidManifest.xml Add: QcoinActivity Delete: RemoteContainerActivity Modify: The return data format of Logout interface to distinguish keys Modify: ProGuard configuration Update: UnionPay’s all files</td>
<td>Lin Xin</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Changes</td>
<td>Author</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>0.8.6</td>
<td>2013-8-27</td>
<td>Add: android-support-v4.jar Add: Flowchart of Payment interface Add: Specify which interfaces must be accessed and which interfaces are optional for access, and add the function description of optional interfaces Add: The detailed access description of the addiction prevention system Modify: The flowchart of Login interface was originally placed in the payment process; after the modification, it is placed in the login process. Delete: UnionPay’s all files</td>
<td>He Qian</td>
<td></td>
</tr>
</tbody>
</table>

360 Application Open Platform October 8, 2013
# Table of Content

1. Overview ........................................................................................................................................... 3  
   Important Reminder: .......................................................................................................................... 3  
   Which interfaces must be accessed, which interfaces are optional .................................................. 4  
2. Access Preparation ........................................................................................................................... 4  
   2.1 Before Access ............................................................................................................................. 4  
   2.2 Using 360SDK ............................................................................................................................. 5  
3. Login and Authorization Process ...................................................................................................... 12  
   3.1 Process Introduction .................................................................................................................... 12  
   3.2 Interface Introduction .................................................................................................................. 15  
   3.2.1 Login – 360SDK Interface, Application Client Calling ......................................................... 15  
   3.2.2 Account switch interface - 360 SDK Interface, Application Client Calling ............................. 18  
   3.2.3 Obtaining access token–calling the server-end interface and application server .................. 19  
   3.2.4 Obtaining User Information – Server End Interface Called by the Application Server (Recommended) ........................................................................................................................... 20  
   3.2.5 Exchanging for Access Token – Server End Interface Called by the Application Server .......... 21  
   3.2.6 Querying Access Token Information – Server End Interface .............................................. 22  
4. Payment Process ................................................................................................................................ 24  
   4.1 Process Introduction ..................................................................................................................... 24  
   4.1.1 The calling process of payment .............................................................................................. 24  
   4.2 Interface Introduction .................................................................................................................. 25  
   4.1.2 Payment Interface – 360SDK Interface Called by the Application Client .............................. 25  
   4.1.3 Payment Result Notification Interface – Interface provided by the application server, and called back by the 360 server ................................................................. 26  
   4.1.4 Order Confirmation Interface – Server End Interface, Called by the Application Server .......... 31  
   4.3 Deposit card (express payment) access (0.7.4 New) .................................................................. 32  
   4.3.1 Importing of library files ......................................................................................................... 33  
   4.3.2 Importing the resource file (Before compiling the code, please generate R.java and do not obfuscate R.java) ..................................................................................................................... 33  
   4.3.3 Androidmanifest.xml file configuration .............................................................................. 34  
5. 360SDK other functions .................................................................................................................... 35  
   5.1 Other interfaces of 360SDK ........................................................................................................... 35  
   5.1.1 Logout interface– 360 SDK Interface, Application Client Calling .......................................... 35  
   5.1.2 Opening BBS interface – 360 SDK Interface, Application Client Calling .............................. 37  
   5.1.3 Addiction prevention inquiry interface – 360 SDK Interface, Application Client Calling .......... 39  
   5.1.4 Real name registration interface – 360 SDK Interface, Application Client Calling ............... 42  
   5.1.5 Authority self-check interface – 360 SDK Interface, Application Client Calling .................. 45  
   5.1.6 Pro-packet info inquiry interface– 360 SDK Interface, Application Client Calling ................ 46  
   5.1.7 Cellphone number binding interface– 360 SDK Interface, Application Client Calling .......... 49  
   5.1.8 Suspension window setting interface – 360 SDK Interface, Application Client Calling .......... 50
1. Overview

This document describes the technical process about how the Android mobile applications access to the 360 platform via the 360 open platform interface. 360 open platform provides login and payment services to Android applications.

The target audiences of this document are the Android application developers, including client-side and server-side engineers.

Important Reminder:

1.1 The application server URL for DEMO can only be available to DEMO. It is prohibited that the officially launched online games use the DEMO application server as the formal application server. Users please set up their own application servers.

1.2 During debugging, please use the Authority self-check interface of 360SDK (refer to Section 5.2 for details). Check the authority statement in the game’s AndroidManifest.xml to avoid authority losses.

1.3 It is prohibited to save AppSecret in the cellphone. What is stored in AndroidManifest is Private Key, not Secret Key. For detailed description, see Appendix 6.2.

1.4 Do not use @string to quote the meta-data value in AndroidManifest.xml. When uploading APK, the server reads the value from here. Read failure can affect the statistics and billing. Be sure to fill in the value correctly.

1.5 Cooperating game or application parties should be able to apply for a unified login service. If you want to apply, please send the formal signature package to the technical support to extract Public Key of the signature. This Key is used for security verification during login; otherwise, you can’t automatically login via the unified login service of 360SDK but have to enter your username and password to login. In this case, security verification will fail when the user logins for the first time, causing some troubles like being unable to login.

1.6 The version contains a log-opening jar package which can use Logcat to view the log. When using the log-opening jar package, please install it with no covered installation after uninstalling any game or application; otherwise, the newly installed jar package can not work.

1.7 If the cellphone’s local time is not set correctly but is far from the correct time, this may result in failure of login and payment.

1.8 When the cellphone uses the WAP access point, the login and payment may not work normally. In this case, SDK may prompt that WAP cannot work normally.

1.9 Developers please use the document as a reference. Do not directly copy codes in the document. For details, please refer to the demo code.
Which interfaces must be accessed, which interfaces are optional

Interfaces which must be accessed:
All the interfaces of login module (Account Switch Interface is required to be added in the game menu);
All the interfaces of payment module;
Logout interface;
Real name registration interface
Addiction prevention inquiry interface

Optional interfaces:
1. Authority self-check interface and Pro-packet info inquiry interface can help developers to improve development efficiency. It is recommended to use them
2. Opening BBS interface provides users a place to share experience and exchange ideas and can improve the game user’s loyalty.
2.1. Cellphone number binding interface provides users with the functionality to bind their phone numbers and can improve account security.
2.2. Suspension window setting interface integrates 360 BBS, customer service and cellphone number binding functions.
2.3. SMS payment interface is suitable for small-amount, weakly networked games.
2.4. Screenshots posting interface can send screenshots to 360 BBS and can help users share experience with each other.

2. Access Preparation

2.1 Before Access

The Android application access follows the OAuth2 process. The developers can read the following sections to get familiar with the OAuth2 process. [OAuth2 process](http://wiki.dev.app.360.cn/index.php?title=OAuth_2.0%E6%96%87%E6%A1%A3)

Before calling the interface, the developer needs to apply for app_key and app_secret. app_key on the application open platform [http://open.app.360.cn/?from=open_dt](http://open.app.360.cn/?from=open_dt). They are the unique identifier of an application, and app_secret is equivalent to the secret key of the application. app_secret is not allowed to be stored in the client and can not be leaked.

The current download site of mobile game SDK (mobile game payment SDK): [http://openapi.360.cn/page/sdk_cmos](http://openapi.360.cn/page/sdk_cmos)

Here provides:
1. Mobile application submission guidelines;
2. Various versions of SDK & access documentations;
3. The download of server-side SDK;
4. The download of APK’s self-check tools (including Private Key’s calculation tool); (after downloading APK, please use this tool for self check)
5. Video tutorials for the access process (introduce each access step in detail; developers with initial access to SDK are suggested to watch them)
6. FAQ for the access process.
Please be sure to download the latest version of mobile game SDK at the site.

**Note:**

The app_key is bounded with an application, and one app_key can only be assigned to one application. If multiple applications use the same app_key, it will lead to abnormal billing and affect the income; it will also lead to SDK upgrade failure, statistics errors, and other critical issues that will affect the developer’s income and user experiences.

For safety reasons, the app_secret cannot be embedded in the mobile applications, but can be only stored on the application servers to avoid decompiling mobile applications from obtaining this information. The leakage of app_secret might be used to falsify payment success notification message and lead to economic damages to both parties.

### 2.2 Using 360SDK

Payment 360SDK currently supports API5 and higher system versions and is compatible with new cellphones installed with Android 4.0 and above. While compiling it, please use API17 or higher versions. Its development environment configuration process is as follows (please refer to the demo development environment configuration as shown in the following figure; when importing the demo project, please use UTF-8 and import API17):

![Configuration Process Diagram](image-url)
2.2.1 Importing jar package
a. Duplicate 360SDK.jar to the application project directory libs. If the libs directory does not exist, create it and import the jar package. (Note: Don’t let 360SDK.jar participate in code obfuscation.)
b. Import annotations.jar in the Android development SDK directory into the project, while annotations.jar shall be saved in the directory %ANDROID_SDK_HOME%/tools/support. If ADT of new version has been automatically imported, there is no need for further operation.
c. Import /extras/droid/support/v4/android-support-v4.jar in the directory of SDK developed by Android into the project’s Build Path.

2.2.2 Add file under the directory assets
a. Duplicate pro.jar and alipay_plugin.apk to application project directory assets.

Tips:
Deposit card (express payment) needs to add more resources, which are placed in the released zipped paypalm directory. See Section 4.3 for details;
UnionPay payment needs to add more resources, which are placed in the released zipped unionpay directory. See Section 4.4 for details.

b. Unzip the file res.zip manually to the application project directory assets/res/. The directory structure is as the following pictures.

![Figure: Directory res resources](image)

c. Replace the background LOGO in the application project directory assets/res/: a 480*800 game LOGO background picturegame_login_bg.png. When the 360SDK login interface is called in the non-transparent background mode, this LOGO will be displayed as the login interface background. If the 360SDK login interface is called in the transparent background mode, the picture can be deleted.

2.2.3 Configuring AndroidManifest.xml for application project
a. Add authority (the 360SDK authority can be used for interface self-check, and refer to section 5.1 for details):

```xml
<![CDATA[
<manifest ...
<application ...
<activity ...
   android:name ...
   android:meta-data ...
</activity>
</application>
</manifest>]]>
```
b. Add activity. Please note that it needs to be embedded in the <application> element block (0.8.6 has changes compared with previous versions):

```xml
<activity
    android:name="com.qihoopay.insdk.activity.ContainerActivity"
    android:configChanges="fontScale|orientation|keyboardHidden|locale|navigation|screenSize|uiMode"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
</activity>

<activity
    android:name="com.qihoopp.qcoinpay.QcoinActivity"
    android:configChanges="fontScale|orientation|keyboardHidden|locale|navigation|screenSize|uiMode"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" 
    android:windowSoftInputMode="stateAlwaysHidden|adjustPan" />
</activity>
```

c. Add meta-data in AndroidManifest.xml. Please note that it needs to be embedded in the <application> element block:

```
Tips:
```
Don’t use @string to quote the meta-data value. Please directly fill in the original value. When uploading APK, the server reads the value from here and uses it to identify whether or not the application is a payment SDK application and adds it into the statistics and billing. Read failure can affect the statistics and billing. Be sure to fill out the value correctly.

<!—Add the necessary meta-data: QHOPENSDK_APPID for 360SDK. The value here is app_id. Please apply for app_id at 360 Application Open Platform http://open.app.360.cn/?from=open_dt-->
<meta-data
    android:name="QHOPENSDK_APPID"
    android:value="102094835" />
</meta-data>

<!—Add the necessary meta-data: QHOPENSDK_APPID for 360SDK. The value here is app_id. Please apply for app_id at 360 Application Open Platform http://open.app.360.cn/?from=open_dt-->
<meta-data
    android:name="QHOPENSDK_APPID"
    android:value="102094835" />
</meta-data>

<!—Add meta-data necessary for 360SDK: QHOPENSDK_PRIV ATEKEY for 360SDK. The value here is private_key, and its generation algorithm is md5(app_secret + "#" + app_key) in lower cases. Note: private_key here is not app_secret but md5 (app_secret + "#" + app_key). app_secret is not allowed to be saved in the client! -->
<meta-data
    android:name="QHOPENSDK_APPKEY"
    android:value="8689e00460eabb1e66277eb4232fde6f" />
</meta-data>

<!—Add meta-data necessary for 360SDK: QHOPENSDK_CHANNEL. The value here is usually default. If you want to modify it, please contact us. The value begins with a character. It does not support pure numbers and does not support Chinese characters and other double-byte characters, either. It is used to label the sub-channel of the application side. -->
<meta-data
    android:name="QHOPENSDK_CHANNEL"
    android:value="Default" />
</meta-data>

2.2.4. Add code using 360SDK by using Matrix class and the IDispatcherCallback call back interface
a. import the following classes (0.8.4 has changes compared with previous versions)
b. Use the Matrix init method in applying the main Activity

Used for 360SDK initialization. The method must be called at least once in applying the main Activity onCreate() function; otherwise, 360SDK cannot be used.

**Parameter:**

- activity: activity object
  - If isLandscape is displayed in Landscape mode
  - Callback, result is returned in json format.

**Example:**

```java
//introduce vertical screen animation parameter
Matrix.init(this, false, new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        // TODO your job
    }
});
```

c. Use the destroy method of Matrix while applying main Activity

```java
public static void destroy(Context context)

Call it in the main Activity onDestroy() function to release the resources.

**Parameter:**

- Context

**Example:**

```java
@override
protected void onDestroy() {
    super.onDestroy();
    Matrix.destroy(this);
}
```

d. Use the invokeActivity method of Matrix to call the interface

```java
public static void invokeActivity(Context context, Intent intent, IDispatcherCallback callback)
```
This method is used in calling 360SDK interfaces of login calling, account switch, payment, logout, real name registration. For more details, please refer to the interface format of each interface.

Parameter:
Context

Intent—the parameter value is stored in Extra. For the parameter, please refer to the related interface description and example code, including FUNCTION_CODE for interface distinguishing.

callback—onFinished data parameter returns the result in json string format or null.

e. Calling the asynchronous command interface using the execute method of Matrix

public static void invokeActivity(Context context, Intent intent, IDispatcherCallback callback)

This method is used in calling command interfaces of addiction prevention inquiry, running SDK self-check and SDK information inquiry and so on. For more details, please refer to the interface format of each interface.

Parameter:
Context

intent—the parameter value is stored in Extra. For the parameter, please refer to the related interface description, including FUNCTION_CODE for interface distinguishing.

callback—onFinished data parameter returns the result in json string format or null.

f. Obtaining the basic information using the get method of Matrix

public static String getAppId(Context context)

Get meta-data QHOPENSDK_APPID placed in AndroidManifest.xml in 3.3

public static String getAppKey(Context context)

Obtaining meta-data QHOPENSDK_APPKEY in AndroidManifest.xml of 3.3

public static String getPrivateKey(Context context)

Obtaining meta-data QHOPENSDK_PRIVATEKEY in AndroidManifest.xml of 3.3

public static String getChannel()

Obtaining meta-data QHOPENSDK_CHANNEL in AndroidManifest.xml of 3.3

public static String getVersionName(Context context)

Obtaining pro.jar

Parameter:
context

2.2.5 If you want to obfuscate the java code, please do not obfuscate the classes in jars which are to be compiled together with the java code. You can add the following classes into the ProGuard configuration to exclude them from being obfuscated (0.8.6 has changes compared with previous versions):

-keep public class <your package name>. R{ *, }
-keep public class <your package name>. R$* { *, }
3. Login and Authorization Process

Through login and authorization process, the Android application can obtain the access token of the current user to retrieve the login information such as the user ID. After obtaining the user ID, the application can access to the 360 payment system.

3.1 Process Introduction

The 360 open platform login process deploys the oauth2 protocol standard authorization process. The application party should store and use access_token (in various API for obtaining user information and payment) and refresh_token (for refreshing access_token) obtained from the 360 open platform.

Login process – Exchanging access token with authorization code.
1. The game’s login module calls SDK for login. (Section 3.2.1)
2. SDK communicates with 360Auth Server for the user’s login and returns authorization code.
3. SDK returns authorization code to the game’s login module
4. The game’s login module returns authorization code back to the developer's Game Security Server (Section 3.2.2)
5. The game developer’s Game Security Server initiates a request from 360Auth Server to obtain access token
   a. Obtain access token with authorization code
   b. Use access token to request user info
6. Game Security Server finishes logging in the game account, and the application client enters the game. (Section 3.2.3)

The process requires the involvement of the application server to exchange the access token with the authorization code. Because the request from the application server to the 360 server uses HTTPS protocol, its safety is assured, and the storage of app_secret on the server will not be leaked.

The access token obtained by the application has time limit, due to safety consideration. If the access token times out, please get a new token via refresh token. For details, please refer to section 3.2.4.

Note: If the user logs in again, a new token will be obtained, and the previous token will become invalidated, which means if the user logs into the same game with the same account on different machines, only the token obtained during the last login is valid. The previous logins will fail to call the 360 interface (e.g. payment interface).

360SDK supports mobile express user registration. For the user registration process, please see Figure 4.

---

1The current validity period is 10 hours and might be changed in the future. The application needs to read the actual overtime in the data returned from the interface
3.2 Interface Introduction

3.2.1 Login – 360SDK Interface, Application Client Calling

**Function description:**
Security process, the application calls the 360SDK login interface, and the 360SDK displays the login page to guide the user through login and authorization. Every time the login succeeds, 360SDK will return the authorization code, and the application uses the authorization code to request the application server to replace the token information of access token (access token is the necessary parameter for identification during payment operation).

Returned result after successful login. The login page will stay when the login fails, unless the user clicks the return button to abort login.

0.7 and above version of SDK provides the automatic login function. SDK of 0.7 version and above provides the automatic login function. The user only needs to enter the username and password once and can log into the game automatically later. To realize automatic login, the user needs to provide the formal signature package to 360 to guarantee safety by providing the legal signature.

![Figure 5 Automatic Login](image)

**Interface format:**

```java
public static void invokeActivity(Context context, Intent intent, IDispatcherCallback callback)
```

**Interface example:**

```java
/**
 * Using 360SDK login interface
 */
```
protected void doSdkLogin(boolean isLandScape, boolean isBgTransparent) {
    Intent intent = getLoginIntent(isLandScape, isBgTransparent);
    Matrix.invokeActivity(this, intent, mLoginCallback);
}

/**
 * Generating Intent for 360SDK login interface calling
 * @param isLandScape is displayed in Landscape mode or not
 * @param isBgTransparent is background-transparent or not
 */
private Intent getLoginIntent(boolean isLandScape, boolean isBgTransparent) {
    return getLoginIntent(isLandScape, isBgTransparent);
}

// Login response mode: CODE mode.
protected static final String RESPONSE_TYPE_CODE = "code";

/**
 * Generating Intent for 360SDK login interface calling
 */
private Intent getLoginIntent(boolean isLandScape, boolean isBgTransparent) {
    Bundle bundle = new Bundle();
    // interface related parameter, if 360SDK screen is displayed in Landscape mode.
    bundle.putBoolean(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandScape);
interface related parameter, if 360SDK login page background is transparent.

```java
bundle.putBoolean(ProtocolKeys.IS_LOGIN_BG_TRANSPARENT, isBgTransparent);
```

// *** Following parameters not about the screen ***

// Required. Login response mode: CODE mode, i.e. the returned Authorization Code mode.

```java
bundle.putString(ProtocolKeys.RESPONSE_TYPE, RESPONSE_TYPE_CODE);
```

// Required. 360SDK login module being used.

```java
bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_LOGIN);
```

```java
Intent intent = new Intent(this, ContainerActivity.class);
intent.putExtras(bundle);
return intent;
}
```

The json data format of callback (Note: 0.8.4 has changes compared with previous versions):

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>errno</td>
<td>Required returned value;</td>
</tr>
<tr>
<td></td>
<td>Int-type Json value ;</td>
</tr>
<tr>
<td></td>
<td>0 Success Other values mean failure.</td>
</tr>
<tr>
<td>data</td>
<td>When error_code is 0, value returned is required;</td>
</tr>
<tr>
<td></td>
<td>JSONObject-type Json value.</td>
</tr>
<tr>
<td>code</td>
<td>When error_code is 0, value returned is required;</td>
</tr>
<tr>
<td></td>
<td>String-type Json value;</td>
</tr>
<tr>
<td></td>
<td>Authorization code, used for once with short validity, exchanged for access</td>
</tr>
<tr>
<td></td>
<td>token by requesting the application server.</td>
</tr>
<tr>
<td>state</td>
<td>Not open for the moment.</td>
</tr>
</tbody>
</table>

```
{"data":{"state":"test_state111","code":"199062142a4671b71bd1f262a8dea288495c4e90411580319"},"errno":0}
```

<table>
<thead>
<tr>
<th>Returned value aborted</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
</tr>
</tbody>
</table>

callback example:

```java
// callback of login, registration
private IDispatcherCallback mLoginCallback = new IDispatcherCallback() {

    @Override
    public void onFinished(String data) {
        Log.d(TAG, "mLoginCallback, data is " + data);
    }
}
```
String authorizationCode = parseAuthorizationCode(data);
onGotAuthorizationCode(authorizationCode);

private static final String AUTH_CODE = "code";

/**
 * Obtaining the authorization code from Json string
 * @param data Json string
 * @return authorization code
 */
private String parseAuthorizationCode(String data) {
    String authorizationCode = null;
    if (!TextUtils.isEmpty(data)) {
        try {
            JSONObject json = new JSONObject(data);
            int errCode = json.optInt("errno", -1);
            if (errCode == 0) {
                // only code mode login is supported
                JSONObject content = json.optJSONObject("data");
                authorizationCode = content.optString(AUTH_CODE);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
    Log.d(TAG, "parseAuthorizationCode=" + authorizationCode);
    return authorizationCode;
}

3.2.2 Account switch interface - 360 SDK Interface, Application Client Calling
Function description:
The game side needs to add the "Switch account" entry in the game’s menu to facilitate the user to switch accounts. The application calls the 360SDK account switch interface, and the 360SDK displays the login page. The user can change the account for login. The following login process and returned result are the same as those of the login interface.

Interface formate:

```java
public static void invokeActivity(Context context, Intent intent, IDispatcherCallback callback)
```

Example code:

```java
/**
 * Using the 360SDK account switch interface
 */

@ oatslandscape displays the login page in Landscape mode or not
@ param isBgTransparent displays the login page with transparent background or not

protected void doSdkSwitchAccount(boolean isLandScape, boolean isBgTransparent) {
    Intent intent = getSwitchAccountIntent(isLandScape, isBgTransparent);
    Matrix.invokeActivity(this, intent, mAccountSwitchCallback);
}
```

Intent parameter description:
Same as the login interface except FUNCTION_CODE

```java
intent.putExtra(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_SWITCH_ACCOUNT);
```

json format description for callback (Note: 0.8.4 has changes compared with previous versions):
Same as the login interface

callback example code:
Same as the login interface

3.2.3 Obtaining access token—calling the server-end interface and application server

The application obtains the authorization code from calling back the 360SDK login interface, and the authorization code needs to be exchanged for the access token. The application sends the authorization code to the application server, and the application server obtains the access token by sending request to the 360 server-end /oauth2/access_token interface.

Note: please do not send request from the client end, for app_secret may be leaked.

Parameter Illustration:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Required</td>
<td>Parameter Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>access_token</td>
<td>Y</td>
<td>Authorized access token</td>
</tr>
<tr>
<td>expires_in</td>
<td>Y</td>
<td>Validity of the Access Token, in seconds</td>
</tr>
<tr>
<td>refresh_token</td>
<td>Y</td>
<td>Token to refresh the Access Token, the validity of which is 14 days</td>
</tr>
<tr>
<td>scope</td>
<td>Y</td>
<td>The final access scope of the Access Token, i.e. the actual access list granted to the user. Its value can only be basic for now</td>
</tr>
</tbody>
</table>

3.2.4 Obtaining User Information – Server End Interface Called by the Application Server (Recommended)

When the application has obtained the access token, it can call server end interface /user/me of the 360 open platform to obtain user information. The application can call this interface from the server end or the client end, but it is recommended to call from the server end to avoid storing the access token on the client end.

Parameter Description:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>access_token</td>
<td>Y</td>
<td>Authorized access token</td>
</tr>
<tr>
<td>fields</td>
<td>N</td>
<td>The returned fields can be customized. Multiple attributes need to be separated with commas. If this parameter is not</td>
</tr>
</tbody>
</table>
specifying the returned fields will be id,name,avatar by default.

Returned parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Y</td>
<td>360 User ID, returned by default</td>
</tr>
<tr>
<td>name</td>
<td>Y</td>
<td>360 User Name, returned by default</td>
</tr>
<tr>
<td>avatar</td>
<td>Y</td>
<td>360 User Avatar, returned by default</td>
</tr>
<tr>
<td>sex</td>
<td>N</td>
<td>360 User Gender, only returned if included in the fields. Returned value: Male, Female or Unknown</td>
</tr>
<tr>
<td>area</td>
<td>N</td>
<td>360 User Region, only returned if included in the fields</td>
</tr>
<tr>
<td>nick</td>
<td>N</td>
<td>User Nickname, returned null if not specified</td>
</tr>
</tbody>
</table>

Request example:

https://openapi.360.cn/user/me.json?access_token=12345678983b38aabcdef387453ac8133ac3263987654321&fields=id,name,avatar,sex,area

Returned example:

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
{
  "id": "201459001",
  "name": "360U201459001",
  "avatar": "http://u1.qhimg.com/qhimg/quc/48_48/22/02/55/220255dq9816.3eceac.jpg?f=d140ae40ee93e8b08ed6e9c53543903b",
  "sex": "Unknown",
  "area": ""
}

After obtaining the user information, the application needs to store the bonding between the user account and 360 account, and store the user information for future usage, for example, the 360 User ID is needed for payment.

3.2.5 Exchanging for Access Token – Server End Interface Called by the Application Server

The access token obtained by the application has time limit. If the user login time exceeds this time limit, the token expires and calling the open platform interface will fail. The application server can call the /oauth2/access_token interface again, and exchange for a new access token with the refresh token. In the meantime, the refresh token will also be updated.
### Parameter Description:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant_type</td>
<td>Y</td>
<td>Fixed value of refresh_token</td>
</tr>
<tr>
<td>refresh_token</td>
<td>Y</td>
<td>Refresh token used to refresh the access token</td>
</tr>
<tr>
<td>client_id</td>
<td>Y</td>
<td>app key</td>
</tr>
<tr>
<td>client_secret</td>
<td>Y</td>
<td>app secret</td>
</tr>
<tr>
<td>scope</td>
<td>Y</td>
<td>Its value can only be basic for now</td>
</tr>
</tbody>
</table>

**Request example:**

```plaintext
https://openapi.360.cn/oauth2/access_token?grant_type=refresh_token&refresh_token=12065961868762ec8ab911a3089a7ebdf11f8264d5836fd41&client_id=0fb2676d5007f123756d1c1b4b5968bc&client_secret=8d9e3305c1ab18384f56.....&scope=basic
```

### Returned parameters:
The parameters are the same as the returned parameters during obtaining the access token in section 3.2.2, except that the access token and refresh token are both updated.

**Returned example:**

```plaintext
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
    "access_token":"120652e586871bb6bbcd1c7b77818fb9c95d92f9e0b735873",
    "expires_in":"36000",
    "scope":"basic",
    "refresh_token":"12065961868762ec8ab911a3089a7ebdf11f8264d5836fd41 "
}
```

### 3.2.6 Querying Access Token Information – Server End Interface

The application can query the specific access token information via the /oauth2/get_token_info interface. The application can get the information that if the token is issued by this application, the correspondent 360 User ID, and how soon the token will expire.

Please note that if the access token is invalid, it has expired or if the user has logged in again and obtained a new token to replace the current one, the interface will return

```plaintext
{"error_code":"4010202","error":"access token has expired (OAuth2 ) "}
```

### Returned parameter description:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_key</td>
<td>Y</td>
<td>Application key</td>
</tr>
<tr>
<td>user_id</td>
<td>Y</td>
<td>Authorized User ID</td>
</tr>
</tbody>
</table>
### 360 Open Platform Document

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>The time access_token expires, UNIX timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>expires_in</td>
<td>Y</td>
<td>The access_token validity left, in seconds.</td>
</tr>
</tbody>
</table>

#### Request example:

```plaintext
https://openapi.360.cn/oauth2/get_token_info.json?access_token=48318605f40a967b0d0857c6029980aa8406440de382fd15
```

#### Returned example:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
{
  "app_key": "2d49f00dad3c4c70c71ac8da2fe7228c",
  "user_id": "212146652",
  "expires_at": "1355917281",
  "expires_in": "3582"
}
```
4. Payment Process

4.1 Process Introduction

4.1.1 The calling process of payment

1. The application calls the SDK payment interface (section 4.2.1)
2. SDK displays the payment page and guides the user through the payment process
   a. If the amount is specified while calling the interface, the payment page of a fixed amount will display, as shown in Figure 9
   b. If the amount is not specified while calling the interface, the recharge page will display, as shown in Figure 10
3. After successful payment, the 360 server will call back the notification interface on the application server (section 4.2.2) to notify the payment result
4. (Optional) the application server calls the order confirmation interface on the 360 server (section 4.2.3), and the server will return the confirmation result
5. The application server notification interface will return OK to the 360 server and recharge for the user
4.2 Interface Introduction

4.1.2 Payment Interface – 360SDK Interface Called by the Application Client

Function description:
When the application calls the 360SDK payment interface, 360SDK will pop up the payment selection page, and the user needs to finish the payment on the page.

About the application order ID: If the order number app_order_id of the application side is specified, the order number of the application side can’t be repeatedly submitted but is only limited to an application order; in other words, it can only be paid for one time no matter whether or not the payment is successful. This measure is deployed to avoid repeated payment. The payment notification for application party will return the application order ID and the 360 order ID at the same time.

Because the access token is bound with the current login user id, the payment security is enhanced. Please pay attention to the token validity. Payment interface calling will fail after the access token expires. The game can take care of the token expiration problem in two ways. One is user re-login, and the other is calling the refresh token interface. For details, please refer to relevant login process content.

Interface format:
public static void invokeActivity(Context context, Intent intent, IDispatcherCallback callback)

Interface example:

```java
/**
 * Using the 360SDK payment interface
 *
 * @param isLandScape displays the payment page in Landscape mode or not
 * @param isFixed if the payment amount is fixed
 */
private void doSdkPay(boolean isLandScape, boolean isFixed) {
    QihooPayInfo pay = getQihooPayInfo(isFixed);
    Intent intent = getPayIntent(isLandScape, pay);
    Matrix.invokeActivity(this, intent, mPayCallback);
}
```

**Intent parameter and example:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Parameter Illustration</th>
<th>Data format</th>
</tr>
</thead>
<tbody>
<tr>
<td>product name</td>
<td>Y</td>
<td>The purchased product name, specified by the application, recommended in Chinese. Strongly recommend not to use English comma (,) and double quotes (&quot;).</td>
<td>String, 10 Chinese characters at most</td>
</tr>
<tr>
<td>amount</td>
<td>Y</td>
<td>The purchased product value, in fen. If the user recharges at the chosen amount, it can be unspecified</td>
<td>String, Smallest unit “100”</td>
</tr>
<tr>
<td>notifyuri</td>
<td>Y</td>
<td>Payment result notification uri</td>
<td>String, 255 characters at most</td>
</tr>
<tr>
<td>productid</td>
<td>Y</td>
<td>The purchased product ID, specified by the application</td>
<td>String, 16 characters at most</td>
</tr>
<tr>
<td>app name</td>
<td>Y</td>
<td>Game Name</td>
<td>String, 16 Chinese characters at most</td>
</tr>
<tr>
<td>appuser name</td>
<td>Y</td>
<td>User name in the application, game role name, for example. If the application bonds the 360 account and the application account, the recharge does not distinguish the game server, but recharge is added to the unique user account, and roles in different game servers can all use the recharge, the 360 user name can be used.</td>
<td>String, 16 Chinese characters at most</td>
</tr>
<tr>
<td>appuser id</td>
<td>Y</td>
<td>User ID in the application. If the application bonds the 360 account and the application account, the recharge does not distinguish the game server, but recharge is added to the unique user account, and roles in different game servers can all use the recharge, the 360 user name can be used.</td>
<td>String, 32 characters at most</td>
</tr>
<tr>
<td>ext1</td>
<td>N</td>
<td>Application extension information 1, returned as original.</td>
<td>String, 255 characters at most</td>
</tr>
<tr>
<td>ext2</td>
<td>N</td>
<td>Application extension information 2, returned as original.</td>
<td>String, 255 characters at most</td>
</tr>
<tr>
<td>qihoo user id</td>
<td>Y</td>
<td>360 Account Id</td>
<td>String, Integer</td>
</tr>
<tr>
<td>accesstoken</td>
<td>Y</td>
<td>User access token, please note the validity and refreshing necessity if used</td>
<td>String, 64 characters at most</td>
</tr>
<tr>
<td>apporderid</td>
<td>Y</td>
<td>Application Order No., unique in an application</td>
<td>String, 32 characters at most</td>
</tr>
<tr>
<td>rate</td>
<td>Y</td>
<td>The default proportion between RMB and the game coin. For example, 2 means 1 yuan can exchange for 2 game coins</td>
<td>String, Integer</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>----------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>screen orientation</td>
<td>N</td>
<td>Screen direction, transverse screen by default</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

Please note that the character set for all parameters is UTF-8. If Chinese is used, please pay attention to the character set.

Example

```java
/***
 * Generating Intent for 360SDK payment interface calling
 *
 * @param isLandScape
 * @param pay
 * @return Intent
 */
private Intent getPayIntent(boolean isLandScape, QihooPayInfo pay) {
    Bundle bundle = new Bundle();
    // interface related parameter, if 360SDK screen is displayed in Landscape mode.
    bundle.putBoolean(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandScape);
    // *** Following parameters not about the screen ***
    // Setting parameters in QihooPay.
    // Required. User access token of maximum 64 characters. Please pay attention to the validity and refreshing.
    bundle.putString(ProtocolKeys.ACCESS_TOKEN, pay.getAccessToken());
    // Required. 360 account id, integer.
    bundle.putString(ProtocolKeys.QIHOO_USER_ID, pay.getQihooUserId());
    // Required. Value of the purchased goods, in fen. When the value is bigger than or equal to 100 fen, 360SDK runs the fixed amount payment process; when the value is 0, 360SDK runs flexible amount payment process.
    bundle.putString(ProtocolKeys.AMOUNT, pay.getMoneyAmount());
    // Required. The default proportion between RMB and the game coin. For example, 2 means that 1 yuan can be exchanged for 2 game coins. Integer.
    bundle.putString(ProtocolKeys.RATE, pay.getExchangeRate());
    // Required. Purchased product name of maximum 10 Chinese characters, specified by the application. Chinese is recommended.
```
bundle.putString(ProtocolKeys.PRODUCT_NAME, pay.getProductName());

// Required. ID of the purchased product of maximum 16 characters, specified by the application.
bundle.putString(ProtocolKeys.PRODUCT_ID, pay.getProductId());

// Required. Payment result notification uri provided by the application party, 255 characters at maximum. 360 server calls back the payment interface to the uri. For detailed protocol, please check the document, payment result notification interface-application server provided interface.
bundle.putString(ProtocolKeys.NOTIFY_URI, pay.getNotifyUri());

// Required. Game or application name of maximum 16 Chinese characters.
bundle.putString(ProtocolKeys.APP_NAME, pay.getAppName());

// Required. Username inside the application, game role name for example. If the application is bound with 360 account and application account, 360 username can be used, 16 Chinese characters maximally. (the recharge does not distinguish game servers, recharged to the uniform user account, and can be used by roles in all game servers).
bundle.putString(ProtocolKeys.APP_USER_NAME, pay.getAppUserName());

// Required. User id inside the application.
// If the application is bound with 360 account and application account, the recharge does not distinguish game servers, recharged to the uniform user account, and can be used by roles in all game servers. 360 username can be used, 32 characters maximally.
bundle.putString(ProtocolKeys.APP_USER_ID, pay.getAppUserId());

// Optional. Application extension information 1, original value returned, 255 characters at maximum.
bundle.putString(ProtocolKeys.APP_EXT_1, pay.getAppExt1());

// Optional. Application extension information 2, original value returned, 255 characters at maximum.
bundle.putString(ProtocolKeys.APP_EXT_2, pay.getAppExt2());

// Optional. Application order number, unique in the application, 32 characters at maximum.
bundle.putString(ProtocolKeys.APP_ORDER_ID, pay.getAppOrderId());

// Required. Using the 360SDK payment module.
bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_PAY);

Intent intent = new Intent(this, ContainerActivity.class);
intent.putExtras(bundle);
return intent;
}

### Json data format of callback:

<table>
<thead>
<tr>
<th>error_code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Failure</td>
</tr>
<tr>
<td>-1</td>
<td>User is canceled</td>
</tr>
<tr>
<td>-2</td>
<td>Payment is being processed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>error_msg</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error message</td>
</tr>
</tbody>
</table>

Returned upon success

```
{error_code: 0, error_msg: "Payment Success", content: ""}
```

Returned upon failure

```
{error_code: 1, error_msg: "Payment Failure", content: ""}
```

Cancel return

```
{error_code: -1, error_msg: "Payment is cancelled", content: ""}
```

The payment is being processed

```
{error_code: -2, error_msg: "Payment is being processed", content: ""}
```

### callback example:

```java
// Payment callback
private IDispatcherCallback mPayCallback = new IDispatcherCallback() {

    @Override
    public void onFinished(String data) {
        Log.d(TAG, "mPayCallback, data is " + data);
        JSONObject jsonRes;
        try {
            jsonRes = new JSONObject(data);
            // error_code status code 0 Payment Success 1 Payment failed -1 Payment Cancelled -2 Payment Being Processed
            // error_msg status description
            int errorCode = jsonRes.getInt("error_code");
            String errorMsg = jsonRes.getString("error_msg");
        }
    }
}
```
String text = getString(R.string.pay_callback_toast, errorCode, errorMsg);
Toast.makeText(SdkUserBaseActivity.this, text, Toast.LENGTH_SHORT).show();
} catch (JSONException e) {
    e.printStackTrace();

4.1.3 Payment Result Notification Interface – Interface provided by the application server, and called back by the 360 server

When the application calls the payment interface, the returned address notify_uri of payment result notification needs to be specified. After successful payment, the 360 server will send the payment result to this address in GET mode (support for both GET and POST is recommended). After the application receives the validation parameters, the application will perform recharge for the user in the game.

After receiving the notification message, the application will respond with OK, to denote that the notification has been received. If the response is of other value or the application does not respond, the notification is deemed as failure, and the 360 server will resend the notification for several times. This mechanism is deployed to avoid order missing.

The application should be prepared for multiple notifications to avoid multiple charging. Meanwhile, what needs special attention is that the responded “ok” represents that the application has normally received the message and it isn’t needed to continue sending notifications. It does not mean whether the order is successful or not, or whether the application has processed the order or not. For the repeated notifications, the application may find that the order has been successfully processed and there is no need for further processing, but the application still needs to respond with OK. Otherwise, the 360 server will regard the notification as unsuccessful and resend the notification.

Parameters for mobile payment are as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Parameter Description</th>
<th>Included in the signature or not</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_key</td>
<td>Y</td>
<td>Application app key</td>
<td>Y</td>
</tr>
<tr>
<td>product_id</td>
<td>Y</td>
<td>The purchased product ID</td>
<td>Y</td>
</tr>
<tr>
<td>amount</td>
<td>Y</td>
<td>Total price, in fen</td>
<td>Y</td>
</tr>
<tr>
<td>app_uid</td>
<td>Y</td>
<td>User ID in the application</td>
<td>Y</td>
</tr>
<tr>
<td>app_ext1</td>
<td>N</td>
<td>Application expansion information 1, returned as original</td>
<td>Y</td>
</tr>
<tr>
<td>app_ext2</td>
<td>N</td>
<td>Application expansion information 2, returned as original</td>
<td>Y</td>
</tr>
<tr>
<td>user_id</td>
<td>N</td>
<td>360 Account ID</td>
<td>Y</td>
</tr>
<tr>
<td>order_id</td>
<td>Y</td>
<td>360 returned payment order ID</td>
<td>Y</td>
</tr>
<tr>
<td>gateway_flag</td>
<td>N</td>
<td>If the payment is successful, it will return “success” The application will only add</td>
<td>Y</td>
</tr>
</tbody>
</table>

money to the user account if its value is “success”
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_key</td>
<td>Y</td>
<td>Application app key</td>
</tr>
<tr>
<td>product_id</td>
<td>Y</td>
<td>The purchased product ID</td>
</tr>
<tr>
<td>amount</td>
<td>Y</td>
<td>Total price, in fen</td>
</tr>
</tbody>
</table>

4.1.4 Order Confirmation Interface – Server End Interface, Called by the Application Server

To avoid forged payment success notification, the application can use this interface to perform notification data validation. The data can be validated by adding app_key and app_secret to the notification message received on the payment result notification interface (Section 4.2.2), directly calling the interface but needing no signature.

The interface is provided by the 360 server. For the need of app_secret, it must be called by the application server end, not the client end. The interface can be called using POST or GET.

Interface address:
https://openapi.360.cn/pay/verify_mobile_notification.json?Parameter

Parameter Illustration:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>app_key</td>
<td>Y</td>
<td>Application app key</td>
</tr>
<tr>
<td>product_id</td>
<td>Y</td>
<td>The purchased product ID</td>
</tr>
<tr>
<td>amount</td>
<td>Y</td>
<td>Total price, in fen</td>
</tr>
<tr>
<td>Parameter</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>app_uid</td>
<td>Y</td>
<td>User ID in the application</td>
</tr>
<tr>
<td>order_id</td>
<td>Y</td>
<td>360 payment for order ID</td>
</tr>
<tr>
<td>app_order_id</td>
<td>N</td>
<td>Application Order ID: needs to be specified if validation is required during ordering</td>
</tr>
<tr>
<td>sign_type</td>
<td>Y</td>
<td>Only md5 supported for now</td>
</tr>
<tr>
<td>sign_return</td>
<td>Y</td>
<td>The parameters transferred by the application to the order confirmation interface “sign_return”</td>
</tr>
<tr>
<td>client_id</td>
<td>Y</td>
<td>Application app key again</td>
</tr>
<tr>
<td>client_secret</td>
<td>Y</td>
<td>Application app_secret</td>
</tr>
</tbody>
</table>

The parameters in the table all come from the payment notification message received by the application money-adding interface, and can be provided as original, except that client_id and client_secret are added by the application.

If the parameters are provided correctly, the order confirmation will return data in the json format.

Validation success will return

```
{"ret":"verified"}
```

Validation unssuccess will return

```
{"ret":"invalid"}
```

If the called parameters are incomplete or incorrect, the returned value will be the error code consistent for the open platform interface. For example:

```
{"error_code":"4010102","error":"oauth_consumer_key N/A ( OAuth1.0a )"}
```

### 4.3 Deposit card (express payment) access (0.7.4 New)

Deposit card (express payment) gains access using jar package and binding mode, and all the library files and resource files need to be imported to the project manually. Project configuration picture after adding the Deposit card (express payment)
4.3.1 Importing of library files

1. Importing jar package

   → Duplicate zsh_app_360game.jar in SDK/paypalm/libs/ to the project libs directory, and edit the project Java Build Path to import the package to the project.

2. Importing the so file

   → Duplicate the file folder armeabi/ in SDK/paypalm/libs/ to the project libs directory; if CPU of mips type needs to be supported, duplicate mips/ to the project libs directory.

3. Importing the bin file

   → Duplicate bin_app_plugin.bin in SDK/paypalm/assets/ to the project assets directory

4.3.2 Importing the resource file (Before compiling the code, please generate R.java and do not obfuscate R.java)
Duplicate all the directories and files in SDK /paypalm/res/ to the project res/ directory. After the resource is imported, all resource files containing “zsht” string are resource files for Deposit card (Express payment).

It isn’t allowed to obfuscate R.java. Change <your package name> into the apk package name you define in the “package=” item in AndroidManifest.xml.

-keep public class <your package name>.R{*;}
-keep public class <your package name>.R$*{!*;}

4.3.3 Androidmanifest.xml file configuration

1. Add the necessary authority by adding the following code:

```xml
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

2. Configure Deposit card (Express payment) jar package related Activity by adding the following code:

```xml
<activity>
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:name="cn.paypalm.jar.game360.InitialAct"
    android:screenOrientation="behind"
    android:theme="@style/FloatingWindowStyle"
    android:windowSoftInputMode="stateAlwaysHidden" />
</activity>

<activity>
    android:configChanges="orientation|keyboardHidden|screenSize"
    android:name="cn.paypalm.jar.game360.serMessageAct"
    android:screenOrientation="behind"
    android:theme="@style/FloatingWindowStyle"
    android:windowSoftInputMode="stateAlwaysHidden" />
</activity>
```
5. 360SDK other functions

5.1 Other interfaces of 360SDK

5.1.1 Logout interface—360 SDK Interface, Application Client Calling

Function description:
Calling the interface will show an entrance to 360 BBS, providing for users an entrance for interaction and sharing, and improving the game user’s stickiness. "Exit Game" will directly call the logout interface. “Enter BBS” will open the equipment’s built-in browser to enter 360 BBS (the system can’t ensure the user’ login state at this time) and directly call the logout interface at the same time.
Figure 10  Logout dialogue box

Interface formate:

```java
public static void invokeActivity(Context context, Intent intent, IDispatcherCallback callback)
```

Interface example:

```java
/**
 * Using 360SDK logout interface
 *
 * @param isLandScape displays the payment page in Landscape mode or not
 */
protected void doSdkQuit(boolean isLandScape) {
    Intent intent = getQuitIntent(isLandScape);
    Matrix.invokeActivity(this, intent, mQuitCallback);
}
```

Intent parameter and example:

```java
/**
 * Generating Intent for 360SDK logout interface calling
 *
 * @param isLandScape
 * @return Intent
 */
private Intent getQuitIntent(boolean isLandScape) {
```
Bundle bundle = new Bundle();
// Interface related parameter, if 360SDK screen is displayed in Landscape mode.
bundle.putBoolean(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandScape);
// Required. Using the 360SDK logout module.
bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_QUIT);

Intent intent = new Intent(this, ContainerActivity.class);
intent.putExtras(bundle);
return intent;

Json data format of callback (Note: 0.8.6 has changes compared with previous versions):

Enter BBS

{"which": 1,"label": "Enter BBS "}

Exit Game

{"which": 2,"label": "Exit Game "}

Return key / icon X in the upper right corner

{"which": 0,"label": " return key/X is closed "}

callback example:

// Logout callback
private IDispatcherCallback mQuitCallback = new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        // TODO your job
    }
}

5.1.2 Opening BBS interface – 360 SDK Interface, Application Client Calling

Function description:
Opening BBS interface provides for game users a place to share experience and exchange ideas and can improve the game users’ stickiness. When you call 360 BBS interface, you can enter 360 BBS configured for the application (appkey) in the webview way. This interface can ensure that the user’s login state.

Figure 11  360 BBS page

**Interface formate:**

```java
public static void invokeActivity(Context context, Intent intent, IDispatcherCallback callback)
```

**Interface example:**

```java
/**
 * Use 360SDK’s BBS interface
 *
 * @param isLandScape displays the payment page in the landscape mode or not
 */
protected void doSdkBBS(boolean isLandScape) {
    Intent intent = getBBSIntent(isLandScape);
    Matrix.invokeActivity(this, intent, null);
}
```

**Intent parameter and example:**

```java
/**
 * Generate Intent to call 360SDK BBS interface
 */
```
private Intent getRealNameRegisterIntent(boolean isLandscape) {
    Bundle bundle = new Bundle();
    // Interface related parameter, if 360SDK screen is displayed in Landscape mode.
    bundle.putBoolean(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandscape);
    // Required. Use 360SDK BBS module.
    bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_BBS);

    Intent intent = new Intent(this, ContainerActivity.class);
    intent.putExtras(bundle);
    return intent;
}

5.1.3 Addiction prevention inquiry interface – 360 SDK Interface, Application Client Calling

Function description:
The addiction prevention system is to meet the requirements of the Chinese law on online game management, so any game must access the addiction prevention inquiry and real name registration interfaces.

How to use: The game party calls the addiction prevention inquiry interface to query whether or not the user has been at least 18 years old. If the user is at least 18 years old, he will be allowed to play the game normally without any reminder.

If the user is under the age of 18, the user will be subject to limitations of the addiction prevention system:
While the user is playing the game, the system will remind him of his cumulative online time or playtime.
If his cumulative playtime is over 3 hours, his game revenue (experience, money) will be halved. Thereafter, the system will prompt the user once every 30 minutes.
If his cumulative playtime is over 5 hours, his game revenue will be reduced to 0. Thereafter, the system will prompt the user once every 15 minutes.
If a minor’s accumulative offline time reaches 5 hours, his cumulative online time will be reduced to 0. If he goes on-line once again, his cumulative online time will be renewed from zero.
If the user has not registered his ID information and the system can’t query his age, the system will call the real name registration interface (see Section 5.1.4) and require the user to make real-name registration. If the user does not conduct real-name registration, the user will be regarded to be under the age of 18 and will be subject to limitations of the addiction prevention system.
The addiction prevention inquiry interface can query the user’s current anti-addiction state, such as over 18, under 18, or not registering the user’s identity information.

**Interface format:**

```java
public static void execute(Context context, Intent intent, IDispatcherCallback callback)
```

**Interface example:**

```java
/**
 * The realization of callback in this method is only used for test. In practice, the game developers need to take care of this
 *
 * @param qihooUserId
 * @param accessToken
 */
protected void doAntiAddictionQuery(String qihooUserId, String accessToken) {
    Intent intent = getAntiAddictionIntent(qihooUserId, accessToken);
    Matrix.execute(this, intent, new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {
            Log.d("demo,anti-addiction query result = ", data);
        }
    });
}
```

**Intent parameter and example:**

```java
/**
 * Generating the Intent parameter for addiction prevention inquiry interface
 *
 * @param qihooUserId
 * @param accessToken
 *
 * @return Intent
 */
private Intent getAntiAddictionIntent(String qihooUserId, String accessToken) {
    Bundle bundle = new Bundle();
    // Required. User access token of maximum 64 characters. Please pay attention to the validity and refreshing.
    bundle.putString(ProtocolKeys.ACCESS_TOKEN, accessToken);
    ```
// Required. 360 account id, integer.
bundle.putString(ProtocolKeys.QIHOO_USER_ID, qihooUserId);

// Required. Using the 360SDK addiction prevention inquiry module.
bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_ANTI_ADDICTION_QUERY);

Intent intent = new Intent(this, ContainerActivity.class);
intent.putExtras(bundle);

return intent;

Json data format of callback:

Returned result

```
{"content": "\{"ret": [\"199062142","3\"]\}"}
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>error_code</td>
<td>0: Inquiry Success Other values mean inquiry failure.</td>
</tr>
<tr>
<td>error_msg</td>
<td>Error message</td>
</tr>
<tr>
<td>content</td>
<td>json object, ret array included</td>
</tr>
<tr>
<td>ret</td>
<td>json object array</td>
</tr>
<tr>
<td>qid</td>
<td>Qihoo UserId</td>
</tr>
<tr>
<td>status</td>
<td>0: User data not Found; 1: Underage; 2: Adult.</td>
</tr>
</tbody>
</table>

callback example:

```java
new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        if (!TextUtils.isEmpty(data)) {
            try {
                JSONObject resultJson = new JSONObject(data);
                int errorCode = resultJson.getInt("error_code");
                if (errorCode == 0) {
```
JSONObject contentData = resultJson.getJSONObject("content");
JSONArray retData = contentData.getJSONArray("ret");
Log.d(TAG, "ret data = " + retData);
int status = retData.getJSONObject(0).getInt("status");
Log.d(TAG, "status = " + status);
if (status == 0) {
    // TODO You may call real name registration
} else if (status == 1) {
    // TODO You may adopt addiction prevention strategy
} else if (status == 2) {
    // TODO You may remind the users at due time
} else {
    Toast.makeText(SdkUserBaseActivity.this, 
            resultJson.getString("error_msg"), Toast.LENGTH_SHORT).show();
}
}
} catch (JSONException e) {
    Toast.makeText(SdkUserBaseActivity.this, 
            e.getString(R.string.anti_addiction_query_exception), 
            Toast.LENGTH_LONG).show();
e.printStackTrace();
}

5.1.4 Real name registration interface – 360 SDK Interface, Application Client Calling

Function description:
Provide real name registration interface. When the game party calls the addiction prevention inquiry interface and finds out that the user does not fill out the real name registration information, the interface can be called to initiate real name registration.
Figure 11 Real Name Registration

**Interface format:**

```java
public static void invokeActivity(Context context, Intent intent, IDispatcherCallback callback)
```

**Interface example:**

```java
/**
  * Initiate real name registration
  *
  * @param isLandScape Whether to display the login page in Landscape mode
  * @param isBgTransparent Whether to display the login page with transparent background
  * @param clientId
  */
  
protected void doSdkRealNameRegister(boolean isLandScape, boolean isBgTransparent, String qihooUserId) {
    Intent intent = getRealNameRegisterIntent(isLandScape, isBgTransparent, qihooUserId);
    Matrix.invokeActivity(this, intent, mRealNameRegisterCallback);
}
```

**Intent parameter and example:**

```java
/**
  * Generating Intent for real name registration login interface
  *
  * @param isLandScape Displayed in Landscape mode or not
  */
```
private Intent getRealNameRegisterIntent(boolean isLandScape, boolean isBgTransparent, String qihooUserId) {

    Bundle bundle = new Bundle();
    // Interface-related parameter; whether or not 360SDK screen is displayed in Landscape mode.
    bundle.putBoolean(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandScape);
    // Whether or not the background is transparent
    bundle.putBoolean(ProtocolKeys.IS_LOGIN_BG_TRANSPARENT, isBgTransparent);
    // Required. 360 account id, integer.
    bundle.putString(ProtocolKeys.QIHOO_USER_ID, qihooUserId);
    // Required. Use the 360SDK real name registration module.
    bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_REAL_NAME_REGISTER);

    Intent intent = new Intent(this, ContainerActivity.class);
    intent.putExtras(bundle);
    return intent;
}

Json data format of callback:
Return (no data, only notification of the end of the process of the calling side)

null

callback example:

    // Real name registration callback
    private IDispatcherCallback mRealNameRegisterCallback = new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {
            // TODO do your job
        }
    }
5.1.5 Authority self-check interface – 360 SDK Interface, Application

Client Calling

Function description:
Used to check if the authority to use 360SDK has been added to the game or application. This interface can help developers to improve development efficiency, so it is recommended to use it. The function is called during the game accesses SDK debugging process, after Matrix calls init. After the check passes, please block the calling of this interface before the game is published.

Interface format:

```java
public static void execute(Context context, Intent intent, IDispatcherCallback callback)
```

Interface example:

```java
protected void doSdkSelfCheck() {
    Intent intent = getSelfCheckIntent();
    Matrix.execute(this, intent, new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {
            // TODO parse data
        }
    });
}
```

Intent parameter and example:

```java
/**
 * Generating the Intent parameter for 360SDK self-check
 *
 * @return Intent
 */
private Intent getSelfCheckIntent() {
    Bundle bundle = new Bundle();
    // Required. Using the 360SDK self-check module.
    bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_SELF_CHECK);
    return bundle;
}
```
Intent intent = new Intent(this, ContainerActivity.class);
intent.putExtras(bundle);
return intent;
}

**Json data format of callback:**

Returned upon success

```json
{"error_code":0,"error_msg":"permissions OK"}
```

Returned upon failure

```json
{"error_code":1,"error_msg":"missing permissions: android.permission.SEND_SMS"}
```

<table>
<thead>
<tr>
<th>error_code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>Failure</td>
</tr>
</tbody>
</table>

**callback example:**

```java
new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        Log.d("demo", "self check result = ", data);
        try {
            JSONObject result = new JSONObject(data);
            Toast.makeText(SdkUserBaseActivity.this, result.optString("error_msg"),
            Toast.LENGTH_LONG).show();
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

### 5.1.6 Pro-packet info inquiry interface—360 SDK Interface,

**Application Client Calling**
**Function description:**
This interface is used to query the basic information of the pro package, such as the current SDK version number. It can facilitate to query which version of SDK has access problems and can help developers to improve development efficiency. Developers can use it according to their actual needs.

**Interface format:**

```java
public static void execute(Context context, Intent intent, IDispatcherCallback callback)
```

**Interface example:**

```java
/**
 * The realization of callback in this method is only used for test. In practice, the game developers need to take care of this
 */
protected void doSdkProInfoQuery() {
    Intent intent = getOutSdkInfoQueryIntent();
    Matrix.execute(this, intent, new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {
            // TODO parse data
        }
    });
}
```

**Intent parameter and example:**

```java
/**
 * Generating the Intent parameter for SDK inquiry
 * @return Intent
 */
private Intent getProInfoQueryIntent() {
    Bundle bundle = new Bundle();
    // Required. Using the 360SDK pro-packet inquiry module
    bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_OUT_SDK_INFO);
    Intent intent = new Intent(this, ContainerActivity.class);
    intent.putExtras(bundle);
```
Json data format of callback:

Returned

```
{"content":{"ver_name":"0.7","ver_code":10},"error_code":"0"}
```

<table>
<thead>
<tr>
<th>error_code</th>
<th>0 Success Other values mean failure.</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>Returned json object</td>
</tr>
<tr>
<td>ver_name</td>
<td>versionName of the pro-packet</td>
</tr>
<tr>
<td>ver_code</td>
<td>versionCode of the pro-packet</td>
</tr>
</tbody>
</table>

callback example:

```java
new IDispatcherCallback() {
    @Override
    public void onFinished(String data) {
        Log.d("demo, self check result = ", data);
        try {
            JSONObject result = new JSONObject(data);
            int errCode = result.optInt("error_code");
            if (errCode == 0) {
                JSONObject content = result.optJSONObject("content");
                String verName = null;
                int verCode = 0;
                if (content != null) {
                    verName = content.optString("ver_name");
                    verCode = content.optInt("ver_code");
                }
                Toast.makeText(SdkUserBaseActivity.this,
                                "version name: " + verName + ", version code: " + verCode,
                                Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(SdkUserBaseActivity.this, result.optString("error_msg"),
                                Toast.LENGTH_LONG).show();
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```
5.1.7 Cellphone number binding interface—360 SDK Interface, Application Client Calling

Function description:
This interface provides for the user the function to bind his mobile phone number and can improve the account security.

Interface format:
```
public static void invokeActivity(Context context, Intent intent, IDispatcherCallback callback)
```

Interface example:
```
/**
 * Bound cellphone number
 *
 * @param isLandScape displays the login page in Landscape mode or not
 * @param isBgTransparent displays the login page with transparent background or not
 */
protected void doSdkBindPhoneNum(boolean isLandScape) {
```
```java
Intent intent = getBindPhoneNumIntent(isLandScape);
Matrix.invokeActivity(this, intent, mBindPhoneNumCallback);
}

// Interface related parameter, if 360SDK screen is displayed in Landscape mode.
bundle.putBoolean(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandScape);
// Required. Using the 360SDK cellphone number binding module.
bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_BIND_PHONE_NUM);

Intent intent = new Intent(this, ContainerActivity.class);
intent.putExtras(bundle);
return intent;
}

5.1.8 Suspension window setting interface – 360 SDK Interface,
Application Client Calling

Function description:
The suspension window setting interface integrates 360 BBS, customer services and cellphone number
binding function. While using the interface, please be sure to call Matrix.destroy(this) in OnDestroy of the
main Activity to free resources.
For details, please refer to the description of the below interface example and the description of the code in
demo.
**Interface format:**

```java
public static void execute(Context context, Intent intent, IDispatcherCallback callback)
```

**Interface example:**

```java
/**
 * The suspension window setting
 *
 */
protected void doSdkSettings(boolean isLandScape) {
    Intent intent = getSettingIntent(isLandScape);
    Matrix.execute(this, intent, new IDispatcherCallback() {
        @Override
        public void onFinished(String data) {

        }
    });
}
Intent parameter and example:

```java
/**
 * Generating Intent for setting interface
 *
 * @return Intent
 */
private Intent getSettingIntent(boolean isLandScape) {
    Bundle bundle = new Bundle();
    // Interface related parameter, if 360SDK screen is displayed in Landscape mode.
    bundle.putBoolean(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandScape);
    bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_SETTINGS);
    Intent intent = new Intent(this, ContainerActivity.class);
    intent.putExtras(bundle);
    return intent;
}
```

5.1.9 SMS payment interface - 360 SDK Interface, Application Client Calling

Function description:
Provide SMS payment function. This interface is applicable to small-amount, weakly networked games. The game party needs to go to the China Game Base to apply for the corresponding SMS billing code. After the application succeeds, the game party then takes relevant parameters to call this interface. Note: Currently, the interface only supports mobile SMS payment. For details, please refer to the description of the below interface example and the description of the code in demo.

Interface format:

```java
public static void execute(Context context, Intent intent, IDispatcherCallback callback)
```

Interface example:
```java
/**
 * Call SMS payment; the caller needs to set relevant SMS billing parameters, and SDK will help make orders and send text messages.
 */
protected void doSdkSmsPay() {
    Intent intent = getSmsPayIntent();
    Matrix.execute(this, intent, null);
}

Intent parameter and example:

```
bundle.putString(ProtocolKeys.PRODUCT_ID, pay.getProductId());

// Required; payment result notification uri provided by the application side, having a maximum of 255 characters. 360 Server will call back the payment interface to the uri. As for the specific agreement, please refer to “Payment result notification interface - Interface provided by the application server” in the documentation.
bundle.putString(ProtocolKeys.NOTIFY_URI, pay.getNotifyUri());

// Required; game or application name; maximum: 16 Chinese characters.
bundle.putString(ProtocolKeys.APP_NAME, pay.getAppName());

// Required; the user’s name within the application, such as a game character name. If 360 account and the application account are bound within the application, the 360 user name can be used; maximum: 16 characters. (Recharge doesn’t distinguish regional servers. Pay is charged to a unified user account. Roles in various regional servers can be used).
bundle.putString(ProtocolKeys.APP_USER_NAME, pay getAppUserName());

// Required; a user id within the application
// If 360 account and the application account are bound within the application, recharge doesn’t distinguish regional servers, pay is charged to a unified user account and roles in various regional servers can be used, the 360 user ID can be used, having a maximum of 32 characters.
bundle.putString(ProtocolKeys.APP_USER_ID, pay getAppUserId());

// Required; the amount of money used to purchase goods, using cent as the unit. Here is the money amount represented by the SMS payment code. For example, if the money amount is 2 yuan, 200 should be input here.
// Note: This parameter must be filled out correctly; otherwise, inconsistency may occur between the order and the actual mobile SMS charging.
bundle.putString(ProtocolKeys.AMOUNT, "200");

// SMS payment must set the following 3 billing code parameters, which are applied for by the game party itself from China Mobile
bundle.putString(ProtocolKeys.SMS_PAY_CP_ID, "C00111");

bundle.putString(ProtocolKeys.SMS_PAY_CP_SERVICE_ID, "120123002000");

bundle.putString(ProtocolKeys.SMS_PAY_CONSUMER_CODE, "120123002001");

bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_SMS_PAY);

Intent intent = new Intent(this, ContainerActivity.class);
intent.putExtras(bundle);
5.2.0 Screenshots posting interface - 360 SDK Interface, Application

Client Calling

Function description:
The interface can send screenshots to 360BBS and help users share experience with each other. The application party needs to make screenshots by themselves. Then, the application can call the screenshots posting interface, input the path of the screenshot picture and call the posting page to post the picture to the BBS. For details, please refer to the description of the below interface example and the description of the code in demo.

Interface format:

```java
public static void execute(Context context, Intent intent, IDispatcherCallback callback)
```

Interface example:

```java
/**
 * Use 360SDK screenshots posting interface
 * @param isLandScape Whether to display the page in Landscape mode
 */
protected void doSdkBBSPost(boolean isLandScape){
    Intent intent = getBBSPostIntent(isLandScape);
    Matrix.invokeActivity(this, intent, null);
}
```

Intent parameter and example:

```java
/**
 * Generate Intent for screenshots posting
 * @param isLandScape
 * @return
 */
private Intent getBBSPostIntent(boolean isLandScape) {
    Bundle bundle = new Bundle();

    // Interface-related parameter; whether or not 360SDK screen is displayed in Landscape mode
```
bundle.putBoolean(ProtocolKeys.IS_SCREEN_ORIENTATION_LANDSCAPE, isLandScape);

bundle.putInt(ProtocolKeys.FUNCTION_CODE, ProtocolConfigs.FUNC_CODE_BBS_POSTS);

// Pass the path of your screenshots here, such as:
bundle.putString(ProtocolKeys.BBS_POST_EXTRA_SNAP_PATH, Environment.getExternalStorageDirectory() + "/DCIM/screenshot/20130621152522.png");

Intent intent = new Intent(this, ContainerActivity.class);
intent.putExtras(bundle);
return intent;
}

6. Appendix:

6.1 Signing Algorithm:

The signing algorithm does not distinguish the front or back end, the following algorithm is deployed no matter where the signature is needed:

1. The required parameter must be specified and not null or 0. The character set is utf-8

2. All parameters that are not null or 0 need to be included in the signature. The parameter must be the original value before undergoing urlencode. For example, the Chinese “金币” is encoded as “%E9%87%91%E5%B8%81” during parameter transferring, but needs to remain as its original Chinese value “金币” (please note that the character set is UTF-8) in the signature

3. Sort the not-null parameters by parameter name in the alphabetic ascending order (e.g. the ksort function of php)

4. Add # to the end of the sorted parameters, link the application app_secret using #, sign it using md5, and the calculation is the sign parameter. Please note that for some language the md5 result is in upper-cases and need to be converted into lower-cases

5. Integrate URL for Web transfer, and the parameters need to undergo urlencode

php: sample is as follows

```
// Prepare the signature parameters
```
$input = array(…);

/* Remove the null fields */
foreach($input as $k=>$v)
{
    if(empty($v)) {
        unset($input[$k]);
    }
}
ksort($input); // Sort the parameters by the key

$sign_str = implode('#', $input); // Step 4

$sign_str = $sign_str.'#'.$sign_key; // integrate the key (if it is signature, key is the one after agreed processing)

$sign = md5($sign_str);

$input['sign'] = $sign; // the signature is generated

/* Step 5 getting the URL transfer */

// this address is just a demonstrated interface address
$url = 'http://testapp.com/notify?'.http_build_query($input);
...

---

### 6.2 Generating PrivateKey algorithm

Please apply for app_key and app_id at 360 Application Open Platform http://open.app.360.cn/?from=open_dt

Obtain private_key using the following algorithm

Note: app_secret is not allowed in the client-end, and only app_key and private_key are stored in AndroidManifest.xml.

```java
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class CalPrivateKey {

    public static void main(String[] args) {
        String appKey = "8689e00460eabb1e66277eb4232fde6f";
        String appSecret = "c33cbf893271ab9a90906ebbbacbtf5b8";
        String privateKey = getHash(appSecret + "#" + appKey);
        System.out.println(privateKey);
    }
}
```
public static String getHash(String uri) {
    MessageDigest mDigest;
    try {
        mDigest = MessageDigest.getInstance("MD5");
        mDigest.update(uri.getBytes());
        byte d[] = mDigest.digest();
        return toHexString(d);
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    return uri;
}

private static final char HEX_DIGITS[] = {
    '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'
};

public static String toHexString(byte[] b) { // String to byte
    StringBuilder sb = new StringBuilder(b.length * 2);
    for (int i = 0; i < b.length; i++) {
        sb.append(HEX_DIGITS[(b[i] & 0xf0) >>> 4]);
        sb.append(HEX_DIGITS[b[i] & 0x0f]);
    }
    return sb.toString();
}

6.3 Introduction to Demo Engineering

The project structure of 360SDK is as shown in the picture:
Introduction of the main packets and classes in Demo project:
The login and payment process in practice are demonstrated in the class in com.test.sdk.activity. When the game accesses SDK for joint debugging, please mainly refer to the following classes:

- **SdkWelcomeActivity.java**
  - Provides the entry point for demonstration page and interface display page

- **SdkMainActivity.java**
  - Activity corresponding to the welcome page, demonstrating the interfaces required by the welcome page.

- **SdkGuestActivity.java**
  - Activity corresponding to the game trial guest page, demonstrating the interfaces required by the game trial guest page.

- **SdkUserActivity.java**
  - Activity corresponding to the formal user page, demonstrating the interfaces required by the formal user page.

- **SdkShowAllActivity.java**
  - Looks up Activity of all interfaces
FlowTestLoginActivity.java for flow test—configuring the login parameters during login. FlowTestPayActivity.java for flow test—configuring the payment parameters during payment.

Classes of com.test.sdk.common is used for the using of 360SDK. If there is no particular demand, they can be used directly. SdkUserBaseActivity is an abstract class; Activity inheriting the class can call the 360SDK corresponding interface by using the doSdkxxx method.

SdkHttpTask used for http access in demo.

Classes in com.test.sdk.appserver demonstrate the necessary communication with the application server. Its development requires coordination between the access party client-end and server-end engineers.

TokenInfoTask, this class requests the application server for TokenInfo by calling doRequest.
QihooUserInfoTask, this class requests the application server for QihooUserInfo by calling doRequest.

6.3.1 Look up all the interfaces

![Figure 13. All Interfaces](image)

6.3.2 Demo UI
6.3.3 Flow test——configurable login FlowTestLoginActivity.java

FlowTestLoginActivity file provides the configurable login page in the “flow test”, and only the code mode login with higher security is supported, as shown in the following picture:
Login Configurable Parameter Descriptions:

<table>
<thead>
<tr>
<th>Configurable Parameter</th>
<th>Parameter Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>client id</td>
<td>Application app key</td>
</tr>
<tr>
<td>response type</td>
<td>Login mode, only “code” is supported</td>
</tr>
</tbody>
</table>

### 6.3.4 Flow test —— Configurable payment FlowTestPayActivity.java

FlowTestPayActivity file provides the configurable payment page in the “flow test”, as shown in the following picture:

![FlowTestPayActivity Configurable Payment Page](image)

**Figure 16. FlowTestPayActivity Configurable Payment Page**

Description of Payment Configurable Parameter:

<table>
<thead>
<tr>
<th>Configurable Parameter</th>
<th>Parameter Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>product name</td>
<td>The product name on sale, prop name, for example</td>
</tr>
<tr>
<td>money amount</td>
<td>Payment amount. For value other than 0, initiate fixed payment page; for value of 0, initiate the recharge page</td>
</tr>
<tr>
<td>app user name</td>
<td>User name in the game</td>
</tr>
<tr>
<td>app name</td>
<td>Game name</td>
</tr>
<tr>
<td>app user id</td>
<td>User ID in the game</td>
</tr>
<tr>
<td>qihoo user id</td>
<td>Qihu User ID</td>
</tr>
<tr>
<td>product id</td>
<td>The product ID on sale</td>
</tr>
<tr>
<td>notify uri</td>
<td>Notification return address of the payment result</td>
</tr>
<tr>
<td>app key</td>
<td>Unique key value in the game application</td>
</tr>
<tr>
<td>private key</td>
<td>Md5 value calculated with app key and secret key</td>
</tr>
<tr>
<td>app ext1</td>
<td>Extra information 1 that the application needs to send to SDK, the 360 server will</td>
</tr>
<tr>
<td><strong>app ext2</strong></td>
<td>Extra information 2 that the application needs to send to SDK, the 360 server will send this field to the game server during payment</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>access token</strong></td>
<td>Access token obtained after successful login</td>
</tr>
<tr>
<td><strong>app order id</strong></td>
<td>Order ID. Not required if the game does not generate order ID</td>
</tr>
<tr>
<td><strong>exchange rate</strong></td>
<td>The exchange rate between RMB yuan and the game coin, for example 2 means that 1 yuan can exchange for 2 coins. Not required.</td>
</tr>
</tbody>
</table>