猎宝 SDK 对接说明文档
Version 5.1.0

人员：猎宝 Android 开发团队

日期：2016 年 1 月 1 日

公司：江苏猎宝网络科技有限公司

日期	版本	兼容	作者	说明
<2016-1-1>	<5.1.0.>	<5.1.0>	<韦家郑>	1.增加功能（切换帐号、退出、注销帐号、数据统计、升级）

目录

猎宝 SDK 对接要求（重点）

如遇文档与 DEMO 参数不一致，以 DEMO 为准；包名后缀需要添加.lb；不需要提供更新，需要有角标。

1.1 编写目的

本文档提供给游戏合作商对接的同学使用，文档主要分十大部分：

- 集成 SDK
- 初始化操作
- 对接登录系统
- 对接切换帐号
- 对接注销帐号
- 对接退出
- 对接升级
- 对接数据统计
- 对接充值系统
- 充值服务器回调接口

备注：游戏合作商（统称“合作商”）

运营方（LB）

1.2 集成 SDK

组成（拷贝资源的方式）

- LBSDK_DEMO 工程

- LB_SDK_lib.jar
- LB_SDK_Framework.jar
- AndroidManifest.xml 清单文件
- res 资源文件
- assets 资源文件

1.2.1 配置注意事项

1. 把 AndroidManifest.xml 清单文件中的必要 xml 配置剪切到目标项目，具体内容如下：
   - 权限信息

```
<!--获取internet访问权限 -->
<uses-permission android:name="android.permission.INTERNET"/>

<!-- 允许程序改变Wi-Fi连接状态 -->
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />

<!-- 允许使用PowerManager的WakeLocks保持进程在休眠时从屏幕消失 -->
<uses-permission android:name="android.permission.WAKE_LOCK" />

<!-- 允许程序访问有关GSM网络信息 -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<!-- 读取电话状态 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />

<!-- 允许一个程序初始化一个电话拨号不需通过拨号用户界面需要用户确认 -->
<uses-permission  android:name="android.permission.CALL_PHONE" />

<!-- 允许一个程序打开窗口使用TYPE_SYSTEM_ALERT，显示在其他所有程序的顶层 -->
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />

<!-- 允许一个程序获取信息有关当前或最近运行的任务，一个缩略的任务状态，是否活动等等 -->
<uses-permission android:name="android.permission.GET_TASKS" />
```

```
<!-- 在SDCard中创建与删除文件权限(允许挂载和反挂载文件系统可移动存储) -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>

<!-- 允许程序访问Wi-Fi网络状态信息 -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>

<!-- 允许一个程序访问精良位置 -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>

<!-- 可以读写SDCARD -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

<!-- 手机震动 -->
<uses-permission android:name="android.permission.VIBRATE" />
```

- 默认的参数配置

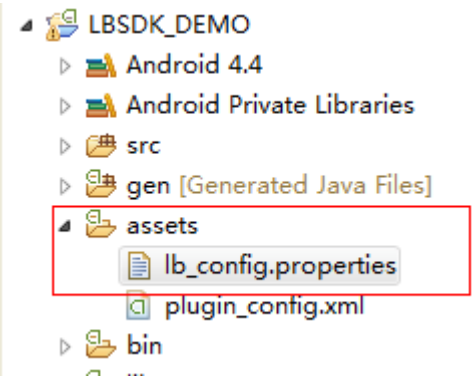
1. 注意：请联系猎宝团队获取配置参数

```
<!-- meta-data需要写在application中 -->
<meta-data android:name="LB_APPID" android:value="322" />
<meta-data android:name="LB_GAMEID" android:value="336" />
<meta-data android:name="LB_AGENT" android:value="default" />

<!--http请求地址-->
<meta-data android:name="LB_HTTP_ADDR" android:value="http://sdk.51508.com/"/>
<!-- 升级 -->
<meta-data android:name="LB_VERSION_HTTP_ADDR" android:value="http://gamelist.51508.com/"/>
<!-- sdk接入层的请求域名 -->
<meta-data android:name="LB_SDK_HTTP_ADDR" android:value="http://sdk.51508.com/"/>
<!-- php页面 -->
<meta-data android:name="LB_HTTP_ADDR_PHP" android:value="http://yt.51508.com/"/>
```

```
<!-- 屏幕适配 -->
<supports-screens
    android:smallScreens="true"
    android:normalScreens="true"
    android:largeScreens="true"
    android:resizeable="true"
    android:anyDensity="true" />
```

2. 以下主要配置应用编号、游戏编号以及渠道号（默认即可）：



Demo 中的 assets 文件夹下的 lb_config.properties 配置文

件主要包含应用编号、游戏编号以及渠道号，这些信息主要用于 sdk 与服务端的交互使用，cp 在对接的时候需要将所对应的值换成 cp 自己真实的应用编号和游戏编号即可，并且必须保证这里所配置的应用编号和游戏编号与清单文件中的对应的信息一致。

注：建议如果 cp 内部需要用到应用编号和游戏编号可直接取自 assets 文件夹下的 lb_config.properties 配置文件中对应的信息。

- Activity 和 Service 等组件的配置

注意：需要配置 application 的 name 值,如截图所示

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
    android:name="com.lb.sdk.LBApplication">
```

```
<activity
    android:name="com.liebao.sdk.ui.LoginActivity"
    android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"
    android:launchMode="singleTask"
    android:screenOrientation="behind"
    android:configChanges="screenSize|keyboardHidden|orientation|layoutDirection"/>

<activity
    android:name="com.liebao.sdk.ui.PayActivity"
    android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"
    android:screenOrientation="behind"
    android:launchMode="singleTask"
    android:configChanges="screenSize|keyboardHidden|orientation|layoutDirection"/>

<!-- webview -->
<activity android:name="com.liebao.sdk.ui.FloatWebActivity"
    android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"
    android:screenOrientation="behind"
    android:configChanges="screenSize|keyboardHidden|orientation|layoutDirection"/>

<activity
    android:name="com.alipay.sdk.auth.AuthActivity"
    android:configChanges="orientation|keyboardHidden|navigation"
    android:exported="false"
    android:screenOrientation="behind" />
```

```
<activity
    android:name="com.alipay.sdk.app.H5PayActivity"
    android:configChanges="orientation|keyboardHidden|navigation"
    android:exported="false"
    android:screenOrientation="behind"
    android:windowSoftInputMode="adjustResize|stateHidden" />

<service android:name="com.liebao.sdk.LBAppService" android:exported="false">
    <intent-filter android:priority="1000"></intent-filter>
</service>
```

1.3 初始化操作

1.3.1 初始化生命周期

在 Activity 对应的生命周期中加上如下的方法：

```
LBSDK.getInstance().onCreate();
LBSDK.getInstance().onStart();
LBSDK.getInstance().onStop();
LBSDK.getInstance().onRestart();
LBSDK.getInstance().onResume();
LBSDK.getInstance().onBackPressed();
LBSDK.getInstance().onDestroy();
LBSDK.getInstance().onNewIntent(intent);
LBSDK.getInstance().onActivityResult(requestCode, resultCode, intent);
LBSDK.getInstance().onConfigurationChanged(newConfig);
```

1.3.2 初始化 sdk

初始化 SDK 相关操作，具体方法如下：

```
LBSDK.getInstance().init(this);
```

1.3.3 监听 SDK 各种事件回调

```
LBSDK.getInstance().setSDKListener(new ILBSDKListener() {

    @Override
    public void onResult(Response res) {
        T.showShort(MainActivity.this, "=======普通回调=========");
    }

    @Override
    public void onPayResult(Response res) {
        T.showShort(MainActivity.this, "=======支付回调=========");
    }

    @Override
    public void onLogout(Response res) {
        T.showShort(MainActivity.this, "=======注销回调=========");
    }

    @Override
    public void onLoginResult(LoginResult res) {
        T.showShort(MainActivity.this, "=======登录回调=========");
    }
```

```
@Override
public void onExit(Response res) {
    if (res.getCode() == Constants.RESULT_CODE_SUCCESS){
        finish();
        System.exit(0);
    }
}
```

```
@Override
public void onSwitchAccount(Response arg0, ILoginPage arg1) {
    //
    // 游戏退出当前帐号，返回登录界面（不需要自动登录）
    //
    if (arg0.getCode() == Constants.RESULT_CODE_SUCCESS){
        Response res = new Response();
        res.setCode(Constants.RESULT_CODE_SUCCESS);
        res.setMsg("游戏处理相关操作成功");
        if (null != arg1){
            arg1.login(res);
        }
    }
}
```

1. Public void onResult(Response res);

这个回调主要用于一般的反馈信息，不需要过多的处理，一般用

于提醒而用。

2. Public void onPayResult(Response res);

这个回调主要用于用户支付的回调，Response 对象中有 code 属性，当 code 值等于 Constants.RESULT_CODE_SUCCESS 时，表示支付成功，其余为支付失败。

3. Public void onLogout(Response res);

这个回调主要用于注销帐号的时候，给 cp 一个反馈，当 code 值等于 Constants.RESULT_CODE_SUCCESS 时，表示注销帐号成功，cp 需要做注销当前帐号的功能。其他情况为注销帐号失败。

4. Public void onLoginResult(LoginResult res);

这个回调主要用于 cp 调用登录接口的反馈信息，当 code 值等于 Constants.RESULT_CODE_SUCCESS 时，表示用户登录成功，反之，用户登录失败。

5. Public void onSwitchAccount(Response res);

这个回调主要用于切换帐号功能的反馈，当 code 值等于 Constants.RESULT_CODE_SUCCESS 时，表示切换帐号成功，cp 需要做退出当前的帐号处理。其他情况切换帐号失败。ILoginPage 参数的含义是让 cp 唤醒当前 sdk 的登录界面。需要在游戏处理相关操作后调用，具体如上图所示。如有不清楚的，见 DEMO 流程。

6. Public void onExit(Response res);

这个回调主要用于退出帐号界面的回调

1.4 对接登录系统

合作商调用相关接口获取登录数据后，需要做**验签处理**，具体如下：

1. 调用 LBUser.getInstance().login()的方法显示登录页面。使用方法如下：

```
/**
 * 登录
 */
private void login(){
    LBSDK.getInstance().runOnMainThread(new Runnable() {

        @Override
        public void run() {
            LBUser.getInstance().login();
        }
    });
}
```

2. 成功登录的信息体 LoginResult 解析：

参数名	类型	参数说明
code	int	Constants.RESULT_CODE_SUCCESS 表示成功
username	String	登录成功后，用户的用户名
logintime	String	用户登录的时间戳
sign	String	用来登录验签对比（签名生成规则）

登录失败信息体 LoginResult 解析

参数名	类型	参数说明

code	int	不等于 Constants.RESULT_CODE_SUCCESS 为登录失败
msg	String	登录失败的消息提示

参数签名规则

参数名	类型	参数说明	签名顺序
username	String	用户帐号(需要转换成小写签名)	1
appkey	String	应用 appkey	2
logintime	String	登录时间，时间戳格式	3

签名字符串示例：(username 的值为小写用户名)

sign = MD5(“username=t315688&appkey=91bac46a9b70bd2db563cc483d443ba3&logintime=1395634100”)

1.5 对接切换帐号

1. 调用 LBUser.getInstance().switchAccount()显示切换帐号界面，具体使用的方法如下：

```
/**
 * 切换帐号
 */
private void switchAccount(){
    LBSDK.getInstance().runOnMainThread(new Runnable() {

        @Override
        public void run() {
            LBUser.getInstance().switchAccount();
        }
    });
}
```

2. 回调信息中的处理，具体说明如下：

```
@Override
public void onSwitchAccount(Response arg0, ILoginPage arg1) {
    //
    // 游戏退出当前帐号，返回登录界面（不需要自动登录）
    //
    if (arg0.getCode() == Constants.RESULT_CODE_SUCCESS){
        Response res = new Response();
        res.setCode(Constants.RESULT_CODE_SUCCESS);
        res.setMsg("游戏处理相关操作成功");
        if (null != arg1){
            arg1.login(res);
        }
    }
}
```

根据 Response 中的 code 值来判断，如果 code 值为 Constants.RESULT_CODE_SUCCESS 时表示切换帐号成功，cp 需要退出当前游戏帐号，返回游戏的首界面。ILoginPage 参数的含义是让 cp 唤醒当前 sdk 的登录界面。需要在游戏处理相关操作后调用，具体如上图所示。如有不清楚的，见 DEMO 流程。

1.6 对接注销帐号

1. 调用 LBUser.getInstance().logout()注销当前帐号，具体使用的方法

如下：

```
/**
 * 注销帐号
 */
private void logout(){
    LBSDK.getInstance().runOnMainThread(new Runnable() {

        @Override
        public void run() {
            LBUser.getInstance().logout();
        }
    });
}
```

2. 回调信息中的处理，具体说明如下：

```
@Override
public void onLogout(Response res) {
    T.showShort(MainActivity.this, "=======注销回调=========");
}
```

根据 Response 中的 code 值来判断，详见初始化操作中的事件回调说明

1.7 对接退出

1. 调用 LBUser.getInstance().exit()调出退出界面，具体使用的方法如下：

```
/**
 * 退出
 */
private void exit(){
    LBSDK.getInstance().runOnMainThread(new Runnable(){

        @Override
        public void run() {
            if (LBUser.getInstance().isSupport("exit")){
                LBUser.getInstance().exit();
            }else{
                //cp自己定义退出界面
            }
        }

    });
}
```

说明：退出中有个判断，cp 可以选择自己的退出界面。或者使用 SDK 自带的退出界面进行退出处理并在回调信息中做退出游戏处理，具体说明如下：

```
@Override
public void onExit(Response res) {
    if (res.getCode() == Constants.RESULT_CODE_SUCCESS){
        finish();
        System.exit(0);
    }
}
```

根据 Response 中的 code 值来判断，详见初始化操作中的事件回调说明

1.8 对接检测升级

1. 调用 LBVersion.getInstance().checkVersion()调出 SDK 升级功能，具体使用的方法如下：

```
/**
 * 升级
 */
private void version(){
    LBSDK.getInstance().runOnMainThread(new Runnable() {

        @Override
        public void run() {
            LBVersion.getInstance().checkVersion();
        }
    });
}
```

注：cp 需要在初始化接口（LBSDK.getInstance().init(this)）后调用升级接口，以唤醒 sdk 的升级界面。

1.9 对接数据统计

1. 调用 LBData.getInstance().submitUserData()调出 SDK 数据统计功能，具体使用的方法如下：

```
/**
 * 提交数据
 */
private void submitUserData(){
    LBSDK.getInstance().runOnMainThread(new Runnable() {

        @Override
        public void run() {
            UserExtraData userData = new UserExtraData();
            userData.setDataType(1);//数据统计类型 1：登录 2：注册 3：登出 4：创建角色 5：角色升级
            userData.setMoneyNum(10);//玩家剩余金币
            userData.setRoleID("124124");//角色编号
            userData.setRoleLevel("1");//角色等级
            userData.setRoleName("角色名称");//角色名称
            userData.setServerID(1);//区服编号
            userData.setServerName("区服");//区服名称
            userData.setUid("d214125125");//玩家编号
            userData.setAttach("12345678");//扩展字段
            LBData.getInstance().submitUserData(userData);
        }
    });
}
```

说明：cp 需要在 datatype 不同类型进行调用该数据统计接口。

参数名	类型	是否必须	参数说明

DataType	Int	是	数据统计类型： 1 登陆，2 注册，3 登出，4 创建角色，5 角色升级
MoneyNum	Int	是	玩家剩余金币
RoleID	String	是	角色编号
RoleLevel	String	是	角色等级，填写数字
RoleName	String	是	角色名称
ServerID	Int	是	区服编号
ServerName	String	是	区服名称
Uid	String	是	玩家编号
Attach	String	是	没有请填“0”

2.0 对接充值系统

1. 调用 LBPay.getInstance().pay(params)调出 SDK 充值功能，具体使用的方法如下：

```
/**
 * 支付
 */
private void pay(){
    LBSDK.getInstance().runOnMainThread(new Runnable() {

        @Override
        public void run() {
            PayParams params = new PayParams();
            params.setRoleId("测试");  //角色编号
            params.setPrice("1");    //充值金额（整型）
            params.setAttach("attch");//扩展字段（订单号）
            params.setProductDesc("商品描述");//商品描述
            params.setProductName("商品名称");//商品名称
            params.setServerId("123456");//区服编号
            LBPay.getInstance().pay(params);
        }
    });
}
```

显示充值界面。参数说明如下：

参数名	类型	是否必须	参数说明
roleid	String	是	角色 ID
price	String	是	充值金额（元）
serverid	String	是	合作商服务器 id
productname	String	是	产品名称
productdesc	String	是	产品描述
attach	String	是	订单号

2. 充值回调信息体 Response 解析：

参数名	类型	参数说明
msg	String	充值结果描述
code	int	成功：Constants.RESULT_CODE_SUCCESS

		其他情况为充值失败
data	Object	额外信息

2.1 充值服务器回调接口开发

合作商需要开发一个接收充值结果的系统，提供该系统的 URL 给 LB，由 LB 发起请求，合作商响应请求。请求的协议采用 **http post**

2.1.1 通知充值结果

玩家在游戏中使用 LB 手游 SDK 进行充值，LB 在处理完充值流程之后，主动请求通知游戏方，游戏方未返回游戏接收成功（接收成功判断请参考合作商响应请求），LB 会启动系统重发机制（一共发送五次，如果五次没有响应将由工作人员补单）。

注意：只有充值成功才会回调通知结果，充值失败的不会通知

合作商接收到请求后，需要对所有 String 类型的参数值做 URLEncode 处理，URLEncode 编码统一为 UTF-8，请求参数具体字段说明如下：

参数名	类型	长度	参数说明	签名顺序
orderid	String	35	LB 订单号	1
username	String	30	LB 登录帐号	2
gameid	Int	11	游戏 ID	3
roleid	String	30	游戏角色 ID	4
serverid	Int	11	服务器 ID	5
paytype	String	10	支付类型，支付类型参数说	6

			明	
amount	Int	11	成功充值金额，单位（元）	7
paytime	Int	11	玩家充值时间，时间戳形式，如 1394087000	8
attach	String		商户拓展参数	9
sign	String	32	参数签名（用于验签对比）	10

2.1.2 签名说明

合作商接收到 LB 的请求后，需要获取相关参数并做**验签处理**，验签规则如下：以请求参数字段说明中的签名顺序签名，规则为 key=value 的形式，参数之间以“&”符号相连，组成一串字符串。目前共 10 个签名字段。

签名算法为 MD5，统一使用 32 位 UTF-8 加密算法。编码后与参数中的 sign 进行对比，如果相同则签名通过，否则失败。

签名字符串示例：

sign =

MD5(“orderid=100000&username=zhangsan&gameid=6&roleid=zhangsanfeng&serverid=1&paytype=1&amount=1&paytime=20130101125612&attach=test&appkey=1231231232213”)

注意：验签参数值为 URLEncode 后的内容，如果参数没有数据值，请以“key=”的形式进行签名，例如：paytime=&attach=自定义。要注意字段的大小写。另外，appkey 由 LB 提供，登录，充值使用同一个

appkey。

示例：

```
/**
 * 说明：只有appkey需要URLEncode编码
 */
String str = "orderid=14501002441631362&username=ewanceshi&gameid=38"+
 "&roleid=%24%23645_1135EWANCESHI&serverid=7540007&paytype=zfb&amount=6&paytime=1450100244" +
 "&attach=20151214213651-1135-5652584484&appkey="+
 URLEncoder.encode("033ce2249e662a110aa6a9179fd7321e", "UTF-8");
System.out.println("===>"+URLEncoder.encode(Md5Util.md5(str), "UTF-8"));
```

2.1.3 合作商相应请求

合作商接收到 LB 发出的请求后，以纯字符串的形式返回下列代码，例如：out.print(“success”);

代码	代码描述
success	接收成功
errorSign	签名错误
error	未知错误

注意：错误信息必须明确，成功必须回传给我们“success”，验签失败传“errorSign”，其他错误传“error”。正常业务操作成功的话，必须回传“success”，如果合作商未返回“success”，LB 会启用系统重发机制。

2.1.4 支付编码表

支付编码	支付类型描述
yibao	易宝支付
Zfb	支付宝快捷支付

注意：订单状态为失败的也需要返回“success”，如果合作商未返回“success”，LB 会启用系统重发机制。