版本号　　：V1.0.0 最后修订日期：2014/09/25

插件框架接口技术规范

版本修改说明

版本	修订日期	修订内容说明
1.0.0	2014-09-22	新建

目　录

1. 概述

1.1.对象

本文档编写的主要对象为Android技术开发人员。

1.2.目的

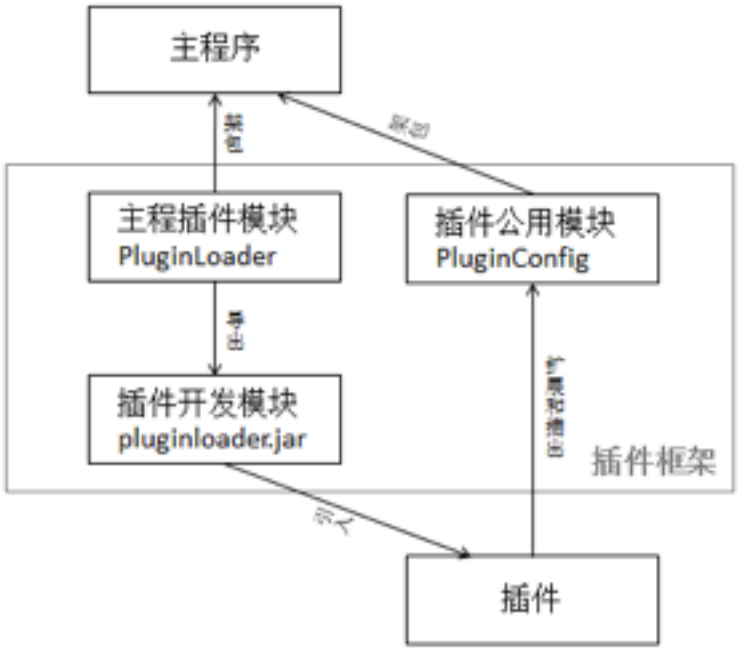
将外部架包和导入包插件化，进行动态加载，提高应用更新灵活性、降低架包依赖度。

1.3.范围

整体架构包括应用主程插件架包、插件开发公用架包、插件开发导入包。接口包括动态加载插件类、动态调用插件函数、动态调用插件res资源、动态启动插件activity等。

2. 插件框架结构说明

插件框架分主程插件模块、插件公用模块、插件开发模块三个部分，其中插件开发模块是主程插件的子模块。具体关系图如下：



2.1.主程插件模块

2.1.1.功能说明

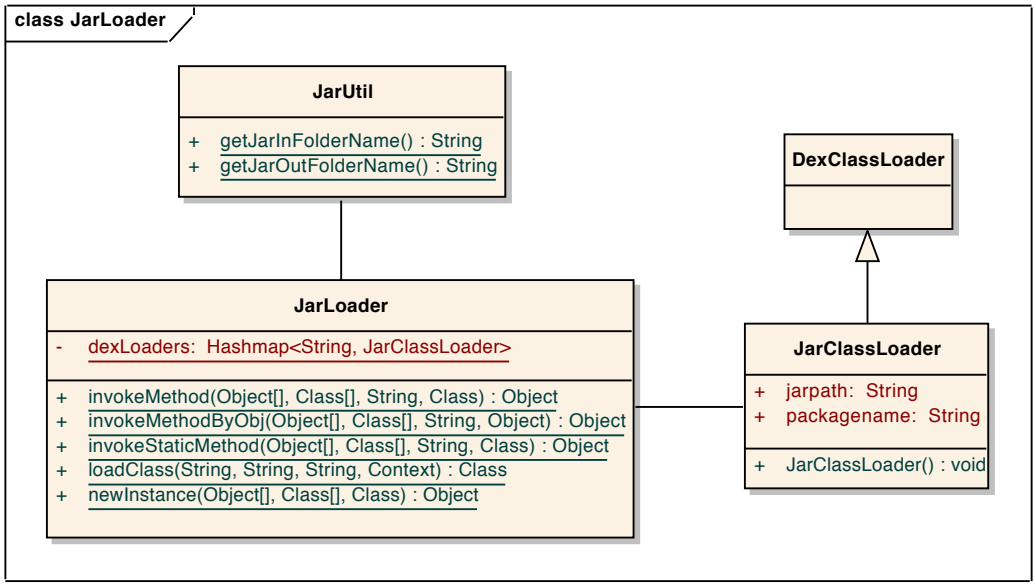
在主程序中进行调用，主要负责插件拷贝、动态加载插件类、动态调用插件函数、实现插件res资源调用、代理方式启动插件activity。

2.1.2.结构说明

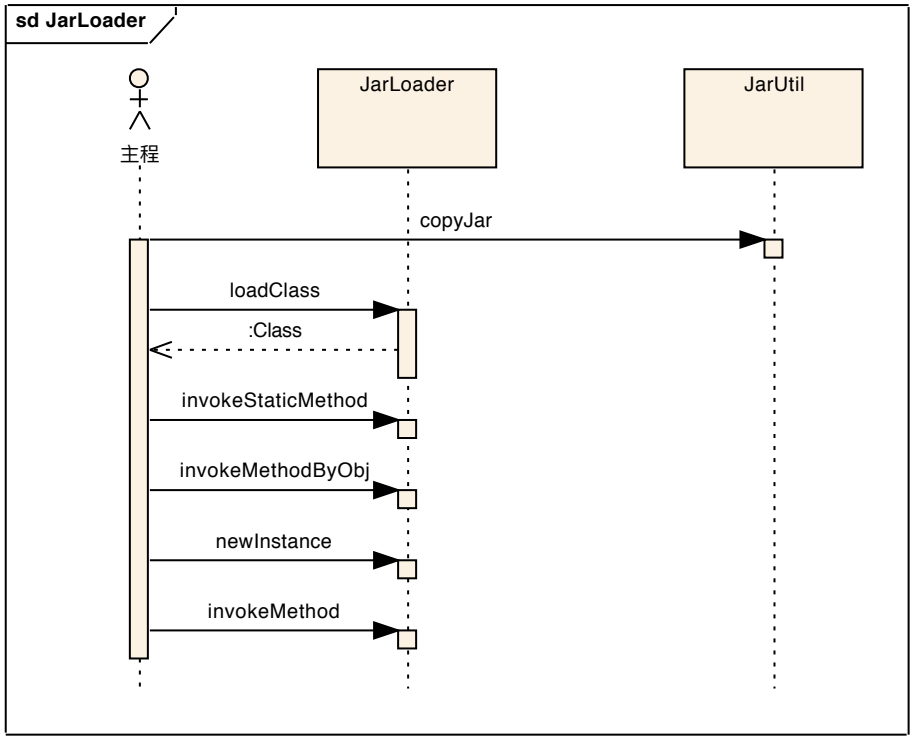
主程插件模块主要分为三个子模块：类的加载、ｒｅｓ资源调用、插件activity的启动。

- 类的动态加载

  类图如下：

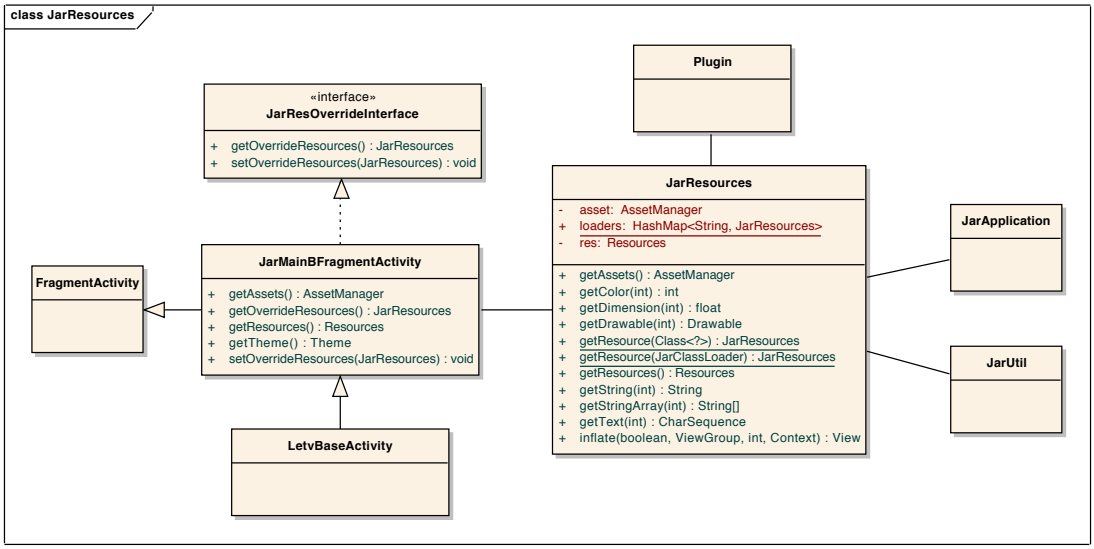


  时序图如下：

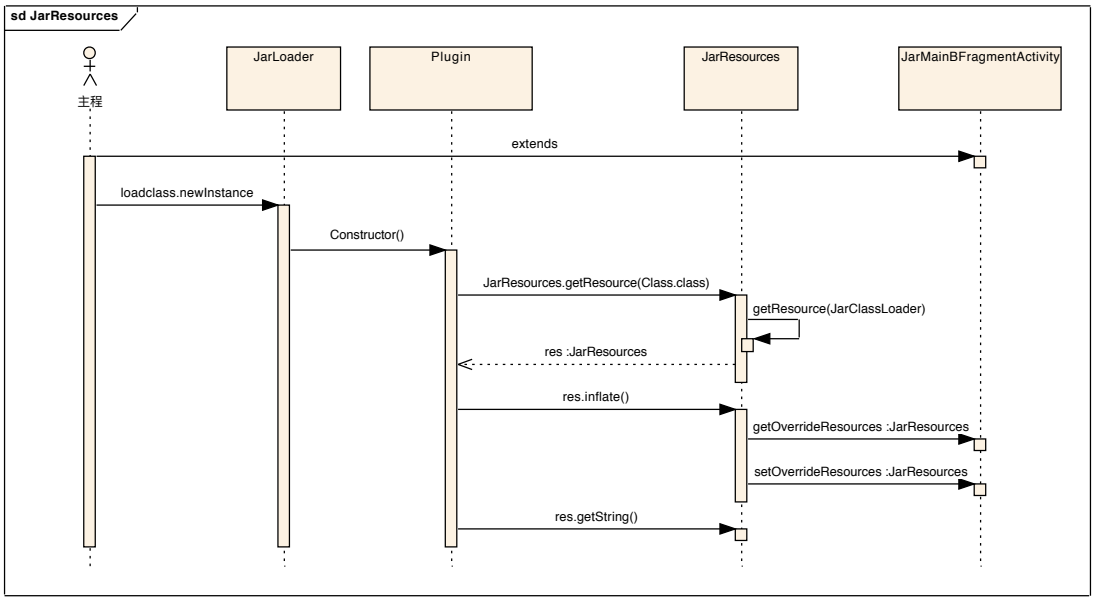


- res资源调用

  类图如下:

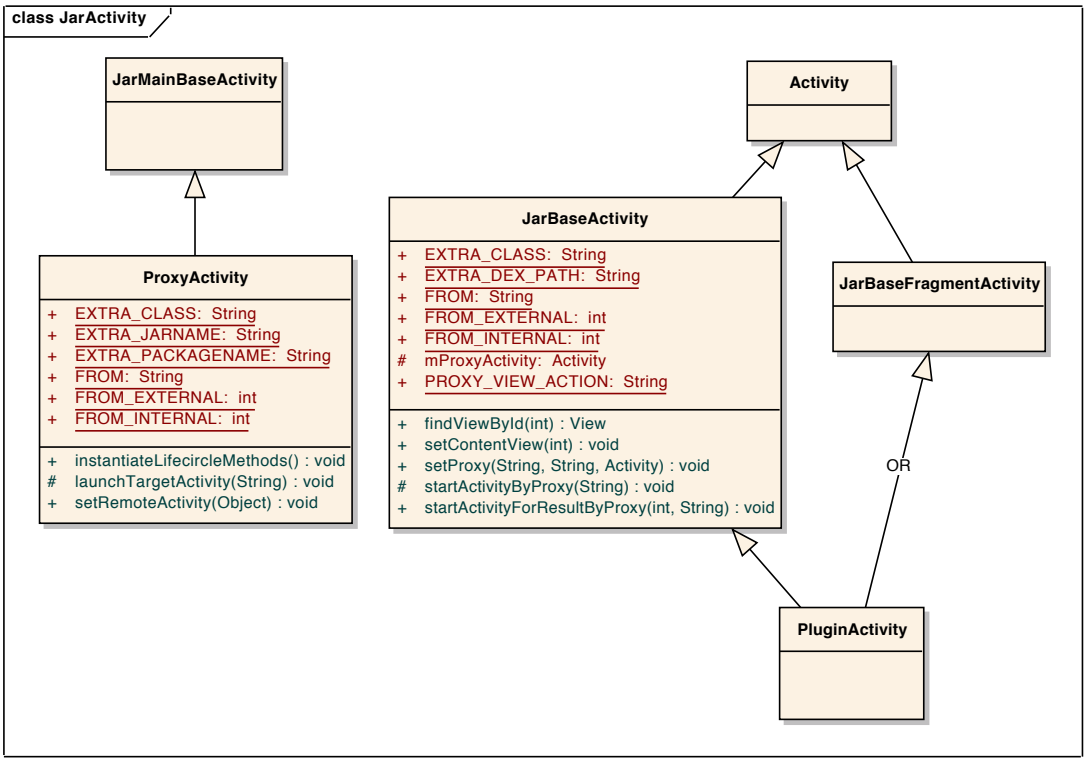


  时序图如下:

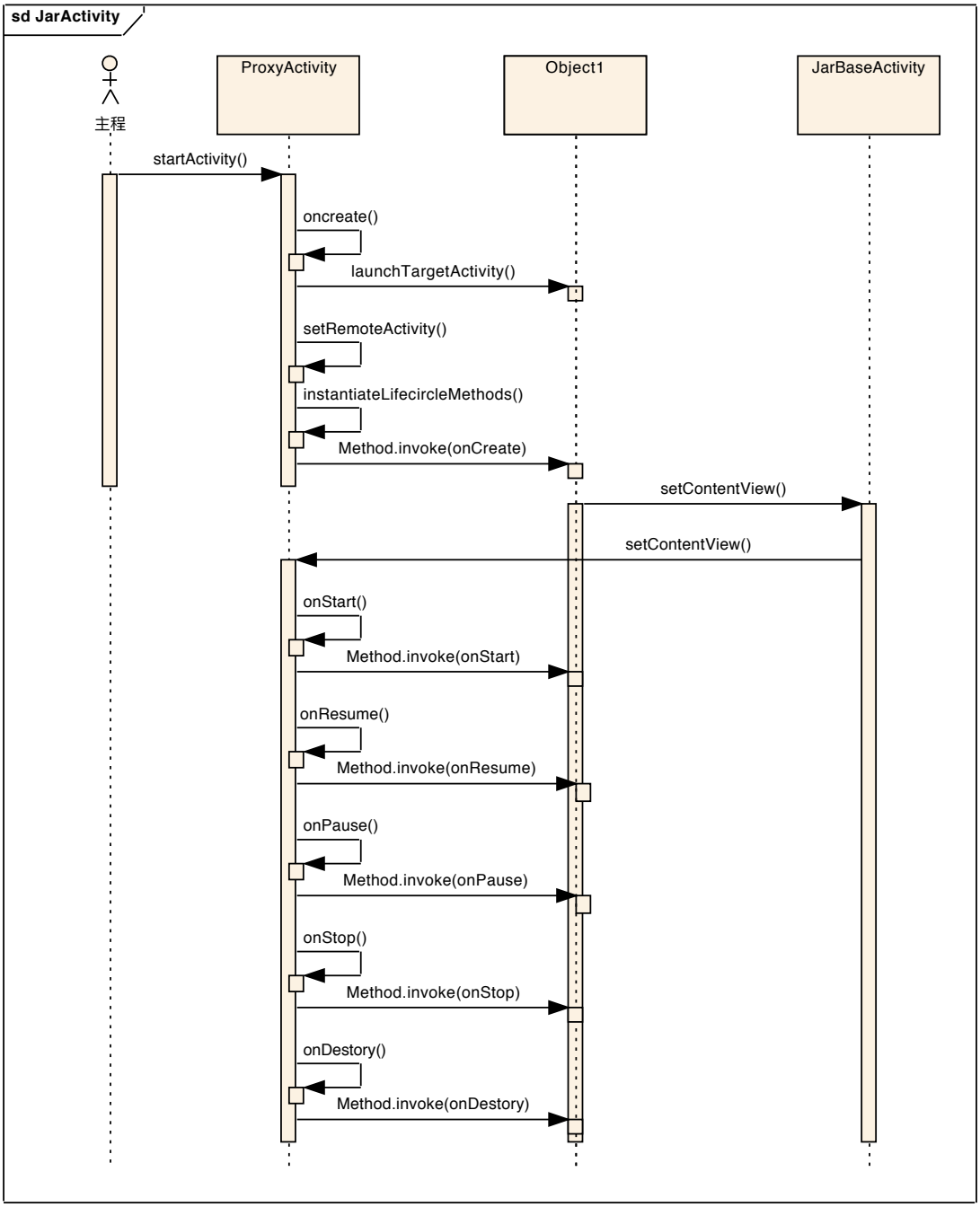


- 插件activity的启动

  类图如下：



  时序图如下：

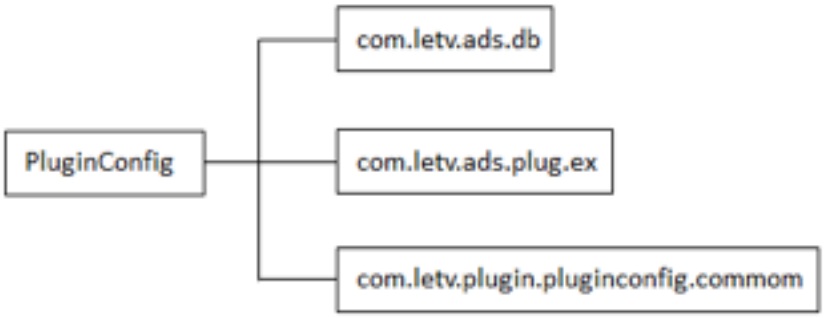


2.2.插件公用模块

2.2.1.功能说明

主要用于存放插件接口、插件数据库代码、插件参数配置。

2.2.2.结构说明



工程名称：PluginConfig

广告插件数据库包名：com.letv.ads.db

广告插件接口扩展包名：com.letv.ads.plug.ex

插件参数配置包：com.letv.plugin.pluginconfig.commom

2.2.3.规范说明

- 插件参数配置包中存放插件配置参数代码。添加插件到主程序时，需要在PluginConstant.java中配置插件名称和插件包名。比如：

```
String LETV_ADS_NAME = "Letv_Ads.apk"; //建议以NAME结尾
String    LETV_ADS_PACKAGENAME    =    "com.letv.ads";//建议以PACKAGENAME结尾
```

- 由于插件是通过反射动态进行调用，所以manifest.xml文件中ContentProvider的配置会失效，所以需要将数据库创建相关代码挪出到PluginConfig工程中。（建议插件中单独包放置数据库操作代码，包名以db结尾，并该包复制到PluginConfig工程下，以方便维护）。

  注意：最后通过Fat Jar（附件一）方式导成jar格式时，去掉PluginConfig工程中已有类。

- 在主程序中通过反射调用插件类会增加代码复杂度，所以可以选用接口技术，将插件调用类继承扩展父类或实现扩展接口的方式，然后将相应的扩展接口包复制到PluginConfig工程中。（建议插件中单独包放置扩展接口，包名以ex结尾，以方便维护）。

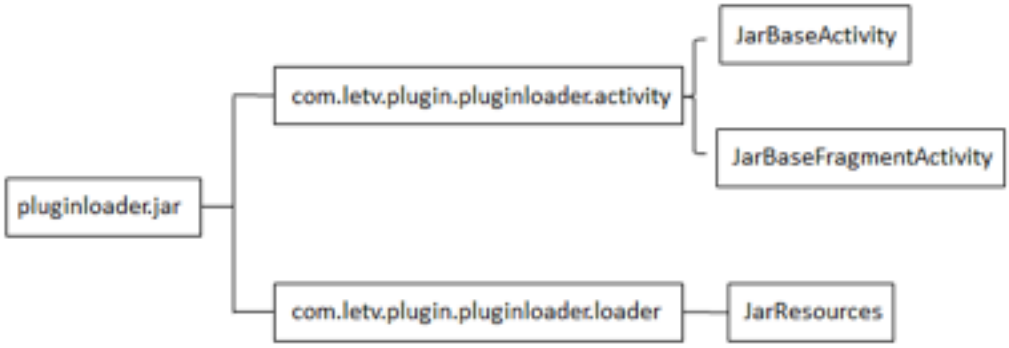
  注意：最后通过Fat Jar方式导成jar格式时，去掉插件ex扩展包。

2.3.插件开发模块

2.3.1.功能说明

主要用于插件中的Activity和res资源的调用。

2.3.2.结构说明



JarBaseActivity插件基类：所有插件中需要继承Activity的类都继承此类。

JarBaseFragmentActivity插件基类：所有插件中需要继承FragmentActivity的类都继承此类。

JarResources资源调用类：插件中出Activity外所有资源调用都采用本资源管理器。

3. 接入流程

3.1.主程序

主程序通过插件框架对插件进行动态调用。接入插件框架步骤如下：

1. 导入PluginConfig和PluginLoader两个架包；
2. 将自定义的LetvApplication继承JarApplication，并删除以下代码：
```
public static JarApplication instance;
//以上代码已在JarApplication中声明。
```
3. 在manifest中配置ProxyActivity修改主程序中所有继承Activity的类，修改对应关系如下：
```
Activity => JarMainBaseActivity
FragmentActivity => JarMainBFragmentActivity
ActivityGroup => JarMainBActivityGroup
```
4. 将插件.apk复制到主程的assets下，并配置插件参数到PluginConstant中，格式如下：
```
//乐视广告插件
String LETV_ADS_NAME = "Letv_Ads.apk";  //***_NAME
String LETV_ADS_PACKAGENAME = "com.letv.ads";//*_PACKAGENAME
```
5. 在主程序中通过PluginLoader的反射调用或PluginConfig的接口方式调用插件类。当调用插件activity时需要用PluginLoader中的代理ProxyActivity启动，代码格式如下：

```
Intent intent = new Intent(this, ProxyActivity.class);
intent.putExtra(ProxyActivity.EXTRA_JARNAME,
PluginConstant.LETV_ADS_NAME);
intent.putExtra(ProxyActivity.EXTRA_PACKAGENAME,
PluginConstant.LETV_ADS_PACKAGENAME);
intent.putExtra(ProxyActivity.EXTRA_CLASS,
"plugin.pActivity");
startActivity(intent);
```

3.2.插件

插件提供给主程序调用的接口需要进行封装，和主程的耦合性尽量低。所有插件的公共配置工程目录是PluginConfig，存放数据库创建、扩展接口、插件配置代码。插件提供给主程序的调用方式有反射和扩展接口两种方式，建议尽量采用扩展接口方式。以扩展接口方式开发插件步骤如下（反射方式雷同）：

1. 导入pluginloader.jar包到插件libs下；
2. 建议将数据库操作代码放到同一包下，包名以.db结尾，并将以manifest配置ContentProvider方式创建的数据库代码复制到PluginConfig下，包名需要与插件保持一致。
3. 建议将扩展插件代码放到同一包下，包名以.ex结尾，并将该包复制到PluginConfig下，包名需要与插件保持一致。
4. 主程序用到的插件自定义接口需要放到扩展接口.ex包下，可统一定义在一个内部类中，参考com.letv.ads.plug.ex包下的AdInterface类，静态方法必须反射方式调用。
5. 插件中的activity需要继承JarBaseActivity，不需要在manifest中声明，插件中res资源通过代理类实例mProxyActivity获取，示例代码：

   ```
   mProxyActivity.getResources().getString(R.string.app_name);
   ```

6. 除去activity，插件类中的资源获取需要通过pluginloader.jar中的JarResources资源管理其进行调用，示例代码：

   ```
   private JarResources res;
   public ImageAdView () { //ImageAdView的构造器
       //ImageAdView需要是主程通过DexClassLoader加载过的类**
       res = JarResources.getResource(ImageAdView.class);
       View view = res.inflate(context, R.layout.ad_view, this,
   true);
   ```

}

7. 由于是反射加载，插件中的自定义View不能在主程序的layout.xml中调用，styleable等配置参数失效，所以反射初始化化自定义View一般调用无AttributeSet构造器，AttributeSet中参数可以通过设置默认值和get、set函数进行设置。

3.3.插件打包

将插件打包并提供给主程序调用。结合apk和Fat Jar两种方式进行插件打包，打包格式为.apk。

1. 去掉插件项目的Is Library选项，运行编译生成插件apk，并复制到临时文件夹中。；
2. 勾选插件项目的Is Library选项，以Fat Jar方式export导出成jar文件，导出jar文件时，需要除去PluginConfig中已存在的数据库创建、扩展接口等包和文件，同时需要除去主程序中已经存在的jar包，最后复制到临时文件夹中；
3. 打开cmd或终端，cd到临时文件夹中使用dx命令将插件.jar中的.class打包转换成dex文件；
4. 用插件.jar中的classes.dex替换掉apk中的classes.dex；
5. 最后apk插件提供给主程序调用。

4. 插件框架接口定义

4.1.面向主程序接口

4.1.1.反射调用

编号	方法名	参数	返回类型	描述
1	getJarClassLoader @class JarLoader @type public static	@param context 句柄Context @param jarname 插件名，jar或apk，String @param packagename 插件包名,String	JarClassLoader	根据插件包名获取JarClassLoader
示例: JarClassLoader jcl = JarLoader.getJarClassLoader(this, jarname, jar_packagename);				
2	loadClass @class JarLoader @type public static	@param context 句柄,Context @param jarname 插件名,String @param packagename 插件包名,String @param classname 类名，不需要.class后缀,String	Class	根据插件包名和类名获取类的反射Class
示例: Class clazz_am = JarLoader.loadClass(this, PluginConstant.LETV_ADS_NAME, PluginConstant.LETV_ADS_PACKAGENAME, "AdsManager");				
3	newInstance @class JarLoader @type public static	@param clazz 反射类, Class @param constructors_class 类构造函数参数类型, Class[] @param constructors_args 类构造函数参数, Object[]	Object	反射生成实例
示例: Object adsManager = JarLoader.invokeStaticMethod(clazz_am, "getInstance", null, null);				

4	invokeMethod @class JarLoader @type public static	@param clazz 反射类, Class @param method_name 方法名, String @param method_class 方法参数类型, Class[] @param method_args 方法参数, Object[]	Object	通过反射类调用函数
示例: int ret = (Integer) JarLoader.invokeMethod(clazz, "Add", new Class[]{int.class, int.class}, new Object[]{20, 21});				
5	invokeMethodByObj @class JarLoader @type public static	@param obj 反射实例,Object @param method_name 方法名, String @param method_class 方法参数类型, Class[] @param method_args 方法参数, Object[]	Object	通过反射实例调用函数
示例: Class clazz_am = JarLoader.loadClass(this, PluginConstant.LETV_ADS_NAME, PluginConstant.LETV_ADS_PACKAGENAME, "AdsManager"); Object adsManager = JarLoader.invokeStaticMethod(clazz_am, "getInstance", null, null); JarLoader.invokeMethodByObj(adsManager, "updateBeginAdInfo", null, null);				
5	invokeStaticMethod @class JarLoader @type public static	@param clazz 反射类,Class @param method_name 方法名, String @param method_class 方法参数类型, Class[] @param method_args 方法参数, Object[]	Object	通过反射类调用静态函数

示例: Class clazz_am = JarLoader.loadClass(this, PluginConstant.LETV_ADS_NAME, PluginConstant.LETV_ADS_PACKAGENAME, "AdsManager"); Object adsManager = JarLoader.invokeStaticMethod(clazz_am, "getInstance", null, null);

4.1.2.JarUtil接口

编号	方法名	参数	返回类型	描述
1	copyJar @class JarUtil @type public static	@param context 句柄Context @param jarname 插件名，jar或apk，String	String 插件存放路径	将apk插件复制到应用目录下存放

示例:
//广告插件
JarUtil.copyJar(this, PluginConstant.LETV_ADS_NAME);				

4.1.3.扩展接口

主程序中调用插件函数可以通过扩展接口进行调用，接口申明的反射实例在动态运行时绑定到具体类。

示例代码:

```
Class clazz_wfm = JarLoader.loadClass(this,
PluginConstant.LETV_WO_NAME,
                PluginConstant.LETV_WO_PACKAGENAME,
"WoFlowManager");
IWoFlowManager woFlowManager=(IWoFlowManager)JarLoader.
invokeStaticMethod(clazz_wfm, "getInstance", null, null);
   woFlowManager.initSDK(this, false);
```

4.1.4.代理类ProxyActivity启动

通过代理类ProxyActivity可以启动插件的activity。启动ProxyActivity时需要传入如下intent参数:

ProxyActivity.EXTRA_JARNAME => 插件名

ProxyActivity.EXTRA_PACKAGENAME => 插件包名

ProxyActivity.EXTRA_CLASS =>需要启动activity类名（类），不要.class后缀

示例代码:

```
Intent intent = new Intent(TestActivity.this,
ProxyActivity.class);
intent.putExtra(ProxyActivity.EXTRA_JARNAME,
PluginConstant.LETV_ADS_NAME);
intent.putExtra(ProxyActivity.EXTRA_PACKAGENAME,
                            PluginConstant.LETV_ADS_PACKAGENAME);
intent.putExtra(ProxyActivity.EXTRA_CLASS,
```

```
"activity.TestActivity");
startActivity(intent);
```

4.2.面向插件接口

4.2.1.资源管理器接口

编号	方法名	参数	返回类型	描述
1	getResource @class JarResources @type public static	@param clazz 类名，Class<?>	JarResources	根据类名获取资源管理器，该类必须是主程序中通过类加载器加载过的。
示例： JarResources res = JarResources.getResource(ADPlayFragment.class); String app_name = res.getResources().getString(R.string.app_name)); View root = res.inflate(getActivity(), R.layout.ad_play, container, false);				
2	getResourceByCl @class JarResources @type public static	@param mcl 类加载器， JarClassLoader	JarResources	根据类加载器获取资源管理器。
示例： JarClassLoader jcl = JarLoader.getJarClassLoader(this, jarname, jar_packagename); JarResources jres = JarResources.getResourceByCl(jcl);				
3	inflate @class JarResources @type public	@param context 句柄，Context @param id 资源ID @param parent 父View @param attachToRoot 是否附加到父view中显示	View	资源加载器通过布局文件生成View
示例： JarResources res = JarResources.getResource(ImageAdView.class); res.inflate(context, R.layout.ad_view, this, true);				
4	getDrawable @class JarResources @type public	@param id 图片资源id	Drawable	资源加载器通过图片资源id获取Drawable
示例： JarResources res = JarResources.getResource(ImageAdView.class); Drawable app_icon = res.getResources().getDrawable(R.drawable.app_icon));				
5	getString @class JarResources @type public	@param id 字符串资源id	String	资源加载器通过字符串资源id获取String

示例: JarResources res = JarResources.getResource(ImageAdView.class); String app_name = res.getResources().getString(R.string.app_name));				
6	getStringArray @class JarResources @type public	@param id 字符串数组资源id	String[]	资源加载器通过字符串数组资源id获取字符串数据
示例: JarResources res = JarResources.getResource(ImageAdView.class); String[] app_name = res.getResources().getStringArray(R.array.app_name));				
7	getColor @class JarResources @type public	@param id 色值资源id	int	资源加载器通过色值资源id获取色值
示例: JarResources res = JarResources.getResource(ImageAdView.class); int app_color = res.getResources().getColor(R.color. app_color));				
8	getDimension @class JarResources @type public	@param id 尺寸资源id	float	资源加载器通过尺寸资源id获取尺寸值
示例: JarResources res = JarResources.getResource(ImageAdView.class); float app_dimen = res.getResources().getDimension(R.dimen.app_dimen));				
8	openRawResource @class JarResources @type public	@param id raw资源id	InputStream	资源加载器通过raw资源id获取raw字节流
9	getRawResource @class JarResources @type public	@param id raw资源id	byte[]	资源加载器通过raw资源id获取raw字节数组
10	getResources @class JarResources @type public	null	Resources	获取独立的Resources
11	getAssets @class JarResources @type public	null	AssetManager	获取独立的AssetManager

4.2.2.Activity接口

插件中FragmentActivity类需要继承JarBaseFragmentActivity，Activity类需要继承JarBaseActivity。以上两父类来自pluginloader.jar

包。

编号	方法名	参数	返回类型	描述
1	setProxy @class JarBaseActivity @type public	@param proxyActivity 代理类的引用， Activity @param jarname 插件名称 @param jar_packagename 插件包名	void	通过 ProxyActivity 传入代理 activity的引 用、插件名 和包名。
2	startActivityByProxy @class JarBaseActivity @type public	@param className 启动的activity类名，包名后部分加类名，不需要.class后缀，String	void	通过代理类 启动 activity。
3	startActivityForResultBy Proxy @class JarBaseActivity @type public	@param className 启动的activity类名，包名后部分加类名，不需要.class后缀，String @param requestCode	void	通过代理类 以ForResult 形式启动 activity。在 启动activity 中重写 onActivityRes ult处理返回 结果。
4	setContentView @class JarBaseActivity @type public	@param layoutResID 布局文件id，int	void	设置布局， 通过 proxyActivity 启动的则设 置代理类布 局。
示例: setContentView(R.layout.test);				
5	setContentView @class JarBaseActivity @type public	@param view 布局view，View	void	设置布局， 通过 proxyActivity 启动的则设 置代理类布 局。

6	setContentView @class JarBaseActivity @type public	@param view 布局view，View @param params 布局view， LayoutParams	void	设置布局，通过proxyActivity启动的则设置代理类布局。
7	addContentView @class JarBaseActivity @type public	@param view 视图view，View @param params 布局view， LayoutParams	void	添加视图，通过proxyActivity启动的则添加视图到代理类。
8	addContentView @class JarBaseActivity @type public	@param view 视图view，View @param params 布局view， LayoutParams	void	添加视图，通过proxyActivity启动的则添加视图到代理类。
9	findViewById @class JarBaseActivity @type public	@param id 组件ID，int	View	通过组件ID获取View，通过代理启动的在代理类中获取。
示例： EditText mEditText = (EditText)findViewById(R.id.editText1);				
9	getResources @class ProxyActivity @type public	null	Resources	通过代理引用mProxyActivity获取插件Resources，通过Resources获取插件资源。
示例： String app_name = mProxyActivity.getResources().getString(R.string.app_name);				

在插件activity中返回ForResult值时需要通过代理activity的引用mProxyActivity，示例代码如下：

```
mProxyActivity.setResult(RESULT_FIRST_USER);
mProxyActivity.finish();
```

（接口规范文件全文完）