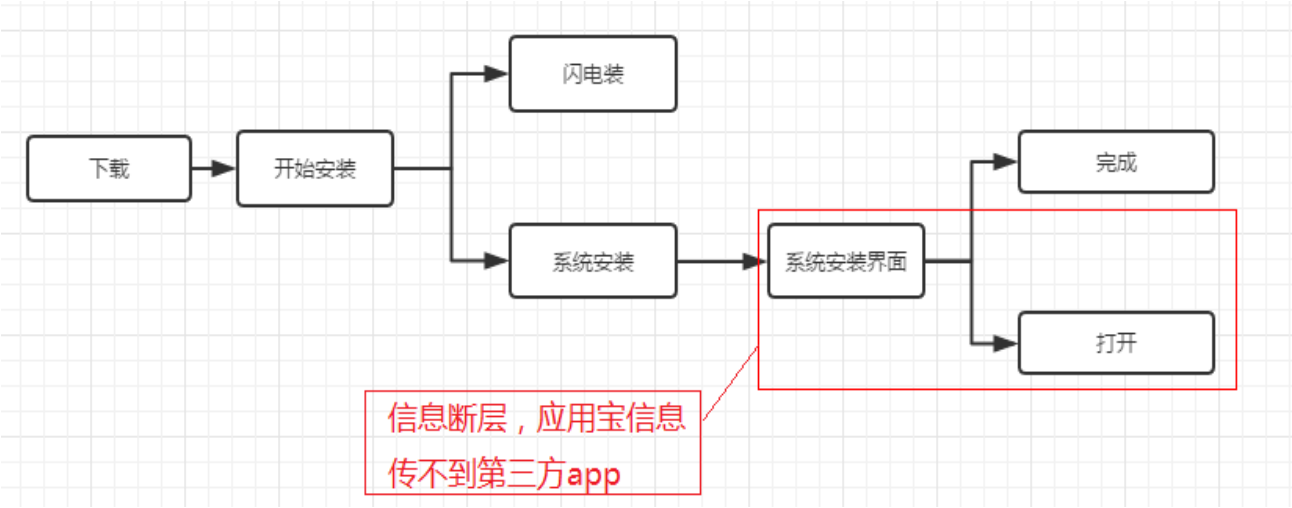
● Android 客户端接入流程

应用宝下载到打开流程如下：



在框出来的流程里会出现信息断层，应用宝没有办法给第三方应用输出参数，导致第三方 app 在这个 case 下不能接收到通过 scheme 传输的参数，这里有2中方法可以能保持体验一致，

第一种，，使用 contentprovider 访问应用宝开放的 db 完成参数的传递，

1, 在 mainfest 中需要申明应用宝设置的权限，否则无法使用
权限名称：
com.tencent.applink.sdk.permission.APPLINK_READ_PERMISSION
com.tencent.applink.sdk.permission.APPLINK_WRITE_PERMISSION

2， 通过 ContentResolver 直接获取到 applink 的参数

Uri:content://com.tencent.android.qqdownloader.applink.provider/applink_provider_task
key：url

示例代码：

```
String packageName = getPackageName();
String value = "";
ContentResolver resolver = getContentResolver();
if(resolver != null) {
    Uri contentUri = Uri.parse("content://com.tencent.android.qqdownloader.applink.provider/applink_provider_task");
    Cursor cursor = null;
    try {
        cursor = resolver.query(contentUri, null,"packageName=?", new String[]{packageName}, null);

        if(cursor != null && cursor.moveToFirst() ) {
            do{
                value = cursor.getString(cursor.getColumnIndexOrThrow("uri"));
            } while(cursor.moveToNext());
        }
    }catch(Throwable ex){
        ex.printStackTrace();
    }finally{
        if (cursor != null) {
            cursor.close();
        }
    }
}
```

第二种：接入应用宝提供的 sdk

客户端 sdk 的工作原理，就是在开始下载的时候应用宝把需要通过 scheme 传输的参数存入 db，，第三方 app 在启动应用的时候通过 sdk 提供的接口去查找 db 获取到参数，

客户端接入流程：

1，获取到 tmassistantapplinksdk_1.1.jar， 并导入工程

2, 在 mainfest 中需要申明应用宝设置的权限，接入 sdk 的应用必须申明 provider 的读写权限，否则 sdk 所有的接口都会无效，

权限名称：
com.tencent.applink.sdk.permission.APPLINK_READ_PERMISSION
com.tencent.applink.sdk.permission.APPLINK_WRITE_PERMISSION

接入代码（直接在 manifest 中申明）：

```
<uses-permission android:name="com.tencent.applink.sdk.permission.APPLINK_READ_PERMISSION"/>
<uses-permission android:name="com.tencent.applink.sdk.permission.APPLINK_WRITE_PERMISSION" />
```

3， 调用 sdk 提供的接口：现在 SDK 提供了 2 个接口如下：

a， public static String getSDCardActionTask(Context context)；

这个接口主要功能是获取安装的时候存入的 applink 跳转任务的信息，返回值是指定跳转到 app 某个页面的 URI 链接。

b， public static void deteleActionTask(Context context)；

负责删除缓存中的跳转任务信息，对使用过的链接进行删除处理，避免在每次打开 app 时候都能获取到这个跳转信息，

备注：这里提供工程自定义 scheme 实例：主要分2步注册和使用

Android 自定义 URI 方案

自定义 URI 方案就是使用其他应用程序启动对应的应用程序。自定义 URI 支持依赖于 Android 清单中指定的 intent-filter。要想自己的程序能被其他应用程序打开，首先必须要注册一个 url scheme。在 Android 系统中注册了 url scheme 且安装了那个应用程序，通过浏览器的方式就可以启动应用程序，如果没有注册或没有安装那个应用程序，就没有任何效果。

若要使用自定义 URI，请将 intent-filter 添加到应用程序描述符的 <android> 区块内。必须指定以下示例中的 intent-filter 元素。编辑 <data android:scheme="myschme"/> 语句以反映自定义方案的 URI 字符串。如：

```
<activity
    android:name="com.example.com.helloworld.LinkProxyActivity"
    android:screenOrientation="portrait"
    android:exported="true" >

    <!-- Test for URL scheme -->
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="myscheme" />
    </intent-filter>
</activity>
```

通过 intent 传过来的参数需在 activity 端接收数据并处理(事例中的 activity（LinkProxyActivity）没有 ui，，仅仅是接收外部跳转的人口）：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Uri uri= getIntent().getData();
    if(uri != null){
        forward(uri);
    }

    finish();
}

private void forward(Uri uri){
    if(uri != null) {
        String schme = uri.getScheme();
        if (!TextUtils.isEmpty(schme) && schme.equalsIgnoreCase("myscheme")) {
            String hostName = uri.getHost();
            Intent intent = new Intent(this, MainActivity.class);
            if (hostName != null) {
                if (hostName.equalsIgnoreCase("SecondActivity")) {
                    intent = new Intent(this, SecondActivity.class);
                } else if (hostName.equalsIgnoreCase("FirstActivity")) {
                    intent = new Intent(this, MainActivity.class);
                }

                startActivity(intent);
            }
        }
    }
}
```