**Android HIDL 在Java 中使用**

**前言：**

HIDL 接口主要通过本机代码使用，因此 HIDL 专注于自动生成高效的 C++ 代码。不过，HIDL 接口也必须能够直接通过 Java 使用，因为有些 Android 子系统（如 Telephony）很可能具有 Java HIDL 接口。本文介绍了 HIDL 接口的 Java 前端，详细说明了如何创建、注册和使用服务，以及使用 Java 编写的 HAL 和 HAL 客户端如何与 HIDL RPC 系统进行交互。

**作为客户端：**

以HelloWorld 为例（详细看[Android HIDL 实例](https://blog.csdn.net/jingerppp/article/details/86530600) 一文），需要将hidl 的java 库导入，在Android.mk 中添加：

1. LOCAL\_JAVA\_LIBRARIES := \
2. android.hardware.helloworld-V1.0-java

添加代码：

1. private void test() {
2. IHelloWorld service = null;
3. try {
4. service = IHelloWorld.getService(true);
5. } catch (RemoteException e) {
6. // TODO Auto-generated catch block
7. e.printStackTrace();
8. }
10. if (service == null) {
11. Log.e(TAG, "test failed, service is null...");
12. }
13. }

**注意**：不含参数的 Java getService 将不会等待服务启动。

**作为服务端：**

Java 中的框架代码可能需要提供接口才能接收来自 HAL 的异步回调。

**注意**：

* 请勿用 Java 实现驱动程序 (HAL)。我们强烈建议您用 C++ 实现驱动程序。
* Java 驱动程序必须与其客户端处于不同的进程中（不支持同一进程通信）。

对于 1.0 版软件包 android.hardware.foo 中的接口 IFooCallback，您可以按照以下步骤用 Java 实现接口。

1. 用 HIDL 定义您的接口。
2. 打开 /tmp/android/hardware/foo/IFooCallback.java 作为参考。
3. 为您的 Java 实现创建一个新模块。
4. 检查抽象类 android.hardware.foo.V1\_0.IFooCallback.Stub，然后编写一个新类以对其进行扩展，并实现抽象方法。

**查看自动生成的文件**

要查看自动生成的文件，请运行以下命令：

1. hidl-gen -o /tmp -Ljava \
2. -randroid.hardware:hardware/interfaces \
3. -randroid.hidl:system/libhidl/transport android.hardware.foo@1.0

这些命令会生成目录 /tmp/android/hardware/foo/1.0。对于文件 hardware/interfaces/foo/1.0/IFooCallback.hal，这会生成文件 /tmp/android/hardware/foo/1.0/IFooCallback.java，其中包含 Java 接口、代理代码和存根（代理和存根均与接口吻合）。

-Lmakefile 会生成在构建时运行此命令的规则，并允许您包含 android.hardware.foo-V1.0-java 并链接到相应的文件。您可以在 hardware/interfaces/update-makefiles.sh 中找到自动为充满接口的项目执行此操作的脚本。 本示例中的路径是相对路径；硬件/接口可能是代码树下的一个临时目录，让您能够先开发 HAL 然后再进行发布。

**运行服务**

HAL 提供了一个接口 IFoo，它必须通过接口 IFooCallback 对框架进行异步回调。IFooCallback 接口不按名称注册为可检测到的服务；相反，IFoo 必须包含一个诸如 setFooCallback(IFooCallback x) 的方法。

要通过软件包 android.hardware.foo 版本 1.0 设置 IFooCallback，请将 android.hardware.foo-V1.0-java 添加到 Android.mk 中。运行服务的代码为：

1. import android.hardware.foo.V1\_0.IFoo;
2. import android.hardware.foo.V1\_0.IFooCallback.Stub;
3. ....
4. class FooCallback extends IFooCallback.Stub {
5. // implement methods
6. }
7. ....
8. // Get the service you will be receiving callbacks from.
9. // This also starts the threadpool for your callback service.
10. IFoo server = IFoo.getService(true /\* retry \*/); // throws NoSuchElementException if not available
11. ....
12. // This must be a persistent instance variable, not local,
13. // to avoid premature garbage collection.
14. FooCallback mFooCallback = new FooCallback();
15. ....
16. // Do this once to create the callback service and tell the "foo-bar" service
17. server.setFooCallback(mFooCallback);