Android 接口定义语言 (AIDL)

AIDL（Android 接口定义语言）与您可能使用过的其他 IDL 类似。 您可以利用它定义客户端与服务使用进程间通信 (IPC) 进行相互通信时都认可的编程接口。 在 Android 上，一个进程通常无法访问另一个进程的内存。 尽管如此，进程需要将其对象分解成操作系统能够识别的原语，并将对象编组成跨越边界的对象。 编写执行这一编组操作的代码是一项繁琐的工作，因此 Android 会使用 AIDL 来处理。

**注**：只有允许不同应用的客户端用 IPC 方式访问服务，并且想要在服务中处理多线程时，才有必要使用 AIDL。 如果您不需要执行跨越不同应用的并发 IPC，就应该通过[实现一个 Binder](https://developer.android.com/guide/components/bound-services.html" \l "Binder) 创建接口；或者，如果您想执行 IPC，但根本不需要处理多线程，则[使用 Messenger 类](https://developer.android.com/guide/components/bound-services.html#Messenger)来实现接口。无论如何，在实现 AIDL 之前，请您务必理解[绑定服务](https://developer.android.com/guide/components/bound-services.html)。

在您开始设计 AIDL 接口之前，要注意 AIDL 接口的调用是直接函数调用。 您不应该假设发生调用的线程。 视调用来自本地进程还是远程进程中的线程，实际情况会有所差异。 具体而言：

* 来自本地进程的调用在发起调用的同一线程内执行。如果该线程是您的主 UI 线程，则该线程继续在 AIDL 接口中执行。 如果该线程是其他线程，则其便是在服务中执行您的代码的线程。 因此，只有在本地线程访问服务时，您才能完全控制哪些线程在服务中执行（但如果真是这种情况，您根本不应该使用 AIDL，而是应该通过[实现 Binder 类](https://developer.android.com/guide/components/bound-services.html" \l "Binder)创建接口）。
* 来自远程进程的调用分派自平台在您的自有进程内部维护的线程池。 您必须为来自未知线程的多次并发传入调用做好准备。 换言之，AIDL 接口的实现必须是完全线程安全实现。
* oneway 关键字用于修改远程调用的行为。使用该关键字时，远程调用不会阻塞；它只是发送事务数据并立即返回。接口的实现最终接收此调用时，是以正常远程调用形式将其作为来自 [Binder](https://developer.android.com/reference/android/os/Binder.html) 线程池的常规调用进行接收。 如果 oneway 用于本地调用，则不会有任何影响，调用仍是同步调用。

在实现 AIDL 接口时应注意遵守以下这几个规则：

* 由于不能保证在主线程上执行传入调用，因此您一开始就需要做好多线程处理准备，并将您的服务正确地编译为线程安全服务。
* 默认情况下，RPC 调用是同步调用。如果您明知服务完成请求的时间不止几毫秒，就不应该从 Activity 的主线程调用服务，因为这样做可能会使应用挂起（Android 可能会显示“Application is Not Responding”对话框）— 您通常应该从客户端内的单独线程调用服务。
* 您引发的任何异常都不会回传给调用方。

## 类型支持

AIDL 使用简单语法，使您能通过可带参数和返回值的一个或多个方法来声明接口。 参数和返回值可以是任意类型，甚至可以是其他 AIDL 生成的接口。

您必须使用 Java 编程语言构建 .aidl 文件。每个 .aidl 文件都必须定义单个接口，并且只需包含接口声明和方法签名。

默认情况下，AIDL 支持下列数据类型：

* Java 编程语言中的所有原语类型（如 int、long、char、boolean 等等）
* [String](https://developer.android.com/reference/java/lang/String.html)
* [CharSequence](https://developer.android.com/reference/java/lang/CharSequence.html)
* [List](https://developer.android.com/reference/java/util/List.html)

[List](https://developer.android.com/reference/java/util/List.html) 中的所有元素都必须是以上列表中支持的数据类型、其他 AIDL 生成的接口或您声明的可打包类型。 可选择将 [List](https://developer.android.com/reference/java/util/List.html) 用作“通用”类（例如，List<String>）。另一端实际接收的具体类始终是 [ArrayList](https://developer.android.com/reference/java/util/ArrayList.html)，但生成的方法使用的是 [List](https://developer.android.com/reference/java/util/List.html) 接口。

* [Map](https://developer.android.com/reference/java/util/Map.html)

[Map](https://developer.android.com/reference/java/util/Map.html) 中的所有元素都必须是以上列表中支持的数据类型、其他 AIDL 生成的接口或您声明的可打包类型。 不支持通用 Map（如 Map<String,Integer>形式的 Map）。 另一端实际接收的具体类始终是 [HashMap](https://developer.android.com/reference/java/util/HashMap.html)，但生成的方法使用的是 [Map](https://developer.android.com/reference/java/util/Map.html) 接口。

您必须为以上未列出的每个附加类型加入一个 import 语句，即使这些类型是在与您的接口相同的软件包中定义。

定义服务接口时，请注意：

* 方法可带零个或多个参数，返回值或空值。
* 所有非原语参数都需要指示数据走向的方向标记。可以是 in、out 或 inout（见以下示例）。

原语默认为 in，不能是其他方向。

**注意**：您应该将方向限定为真正需要的方向，因为编组参数的开销极大。

* .aidl 文件中包括的所有代码注释都包含在生成的 [IBinder](https://developer.android.com/reference/android/os/IBinder.html) 接口中（import 和 package 语句之前的注释除外）
* 只支持方法；您不能公开 AIDL 中的静态字段。

## 通过 IPC 传递对象

通过 IPC 接口把某个类从一个进程发送到另一个进程是可以实现的。 不过，您必须确保该类的代码对 IPC 通道的另一端可用，并且该类必须支持[Parcelable](https://developer.android.com/reference/android/os/Parcelable.html) 接口。支持 [Parcelable](https://developer.android.com/reference/android/os/Parcelable.html) 接口很重要，因为 Android 系统可通过它将对象分解成可编组到各进程的原语。

如需创建支持 [Parcelable](https://developer.android.com/reference/android/os/Parcelable.html) 协议的类，您必须执行以下操作：

1. 让您的类实现 [Parcelable](https://developer.android.com/reference/android/os/Parcelable.html) 接口。
2. 实现 [writeToParcel](https://developer.android.com/reference/android/os/Parcelable.html" \l "writeToParcel(android.os.Parcel, int))，它会获取对象的当前状态并将其写入 [Parcel](https://developer.android.com/reference/android/os/Parcel.html)。
3. 为您的类添加一个名为 CREATOR 的静态字段，这个字段是一个实现 [Parcelable.Creator](https://developer.android.com/reference/android/os/Parcelable.Creator.html) 接口的对象。
4. 最后，创建一个声明可打包类的 .aidl 文件（按照下文 Rect.aidl 文件所示步骤）。

如果您使用的是自定义编译进程，切勿在您的编译中添加 .aidl 文件。 此 .aidl 文件与 C 语言中的头文件类似，并未编译。

AIDL 在它生成的代码中使用这些方法和字段将您的对象编组和取消编组。

例如，以下这个 Rect.aidl 文件可创建一个可打包的 Rect 类：

package android.graphics;  
  
// Declare Rect so AIDL can find it and knows that it implements  
// the parcelable protocol.  
parcelable Rect;

调试时的一个蛋疼问题：A App作为Server，B App作为Client，当B bind service后，在Studio中启动A App，此后B将无法再bind Server。