# 12.2 插件开发：平台通道简介

“平台特定”或“特定平台”中的平台指的就是Flutter应用程序运行的平台，如Android或IOS。我们知道一个完整的Flutter应用程序实际上包括原生代码和Flutter代码两部分。由于Flutter本身只是一个UI系统，它本身是无法提供一些系统能力，比如使用蓝牙、相机、GPS等，因此要在Flutter APP中调用这些能力就必须和原生平台进行通信。为此，Flutter中提供了一个平台通道（platform channel），用于Flutter和原生平台的通信。平台通道正是Flutter和原生之间通信的桥梁，它也是Flutter插件的底层基础设施。

Flutter使用了一个灵活的系统，允许您调用特定平台的API，无论在Android上的Java或Kotlin代码中，还是iOS上的ObjectiveC或Swift代码中均可用。

Flutter与原生之间的通信依赖灵活的消息传递方式：

* 应用的Flutter部分通过平台通道（platform channel）将消息发送到其应用程序的所在的宿主（iOS或Android）应用（原生应用）。
* 宿主监听平台通道，并接收该消息。然后它会调用该平台的API，并将响应发送回客户端，即应用程序的Flutter部分。

### 平台通道

使用平台通道在Flutter(client)和原生(host)之间传递消息，如下图所示：

平台通道

当在Flutter中调用原生方法时，调用信息通过平台通道传递到原生，原生收到调用信息后方可执行指定的操作，如需返回数据，则原生会将数据再通过平台通道传递给Flutter。值得注意的是消息传递是异步的，这确保了用户界面在消息传递时不会被挂起。

在客户端，[MethodChannel API](https://docs.flutter.io/flutter/services/MethodChannel-class.html) 可以发送与方法调用相对应的消息。 在宿主平台上，MethodChannel 在[Android API](https://docs.flutter.io/javadoc/io/flutter/plugin/common/MethodChannel.html) 和 [FlutterMethodChannel iOS API](https://docs.flutter.io/objcdoc/Classes/FlutterMethodChannel.html)可以接收方法调用并返回结果。这些类可以帮助我们用很少的代码就能开发平台插件。

**注意**: 如果需要，方法调用(消息传递)可以是反向的，即宿主作为客户端调用Dart中实现的API。 [quick\_actions](https://pub.dartlang.org/packages/quick_actions)插件就是一个具体的例子。

### 平台通道数据类型支持

平台通道使用标准消息编/解码器对消息进行编解码，它可以高效的对消息进行二进制序列化与反序列化。由于Dart与原生平台之间数据类型有所差异，下面我们列出数据类型之间的映射关系。

|  |  |  |
| --- | --- | --- |
| Dart | Android | iOS |
| null | null | nil (NSNull when nested) |
| bool | java.lang.Boolean | NSNumber numberWithBool: |
| int | java.lang.Integer | NSNumber numberWithInt: |
| int, 如果不足32位 | java.lang.Long | NSNumber numberWithLong: |
| int, 如果不足64位 | java.math.BigInteger | FlutterStandardBigInteger |
| double | java.lang.Double | NSNumber numberWithDouble: |
| String | java.lang.String | NSString |
| Uint8List | byte[] | FlutterStandardTypedData typedDataWithBytes: |
| Int32List | int[] | FlutterStandardTypedData typedDataWithInt32: |
| Int64List | long[] | FlutterStandardTypedData typedDataWithInt64: |
| Float64List | double[] | FlutterStandardTypedData typedDataWithFloat64: |
| List | java.util.ArrayList | NSArray |
| Map | java.util.HashMap | NSDictionary |

当在发送和接收值时，这些值在消息中的序列化和反序列化会自动进行。

### 自定义编解码器

除了上面提到的MethodChannel，还可以使用[BasicMessageChannel](https://docs.flutter.io/flutter/services/BasicMessageChannel-class.html)，它支持使用自定义消息编解码器进行基本的异步消息传递。 此外，可以使用专门的[BinaryCodec](https://docs.flutter.io/flutter/services/BinaryCodec-class.html)、[StringCodec](https://docs.flutter.io/flutter/services/StringCodec-class.html)和 [JSONMessageCodec](https://docs.flutter.io/flutter/services/JSONMessageCodec-class.html)类，或创建自己的编解码器。

### 如何获取平台信息

Flutter 中提供了一个全局变量defaultTargetPlatform来获取当前应用的平台信息，defaultTargetPlatform定义在“platform.dart”中，它的类型是TargetPlatform，这是一个枚举类，定义如下：

enum TargetPlatform {  
 android,  
 fuchsia,  
 iOS,  
}

可以看到目前Flutter只支持这三个平台。我们可以通过如下代码判断平台：

if(defaultTargetPlatform==TargetPlatform.android){  
 // 是安卓系统，do something  
 ...  
}  
...

由于不同平台有它们各自的交互规范，Flutter Material库中的一些组件都针对相应的平台做了一些适配，比如路由组件MaterialPageRoute，它在android和ios中会应用各自平台规范的切换动画。那如果我们想让我们的APP在所有平台都表现一致，比如希望在所有平台路由切换动画都按照ios平台一致的左右滑动切换风格该怎么做？Flutter中提供了一种覆盖默认平台的机制，我们可以通过显式指定debugDefaultTargetPlatformOverride全局变量的值来指定应用平台。比如：

debugDefaultTargetPlatformOverride=TargetPlatform.iOS;  
print(defaultTargetPlatform); // 会输出TargetPlatform.iOS

上面代码即在Android中运行后，Flutter APP就会认为是当前系统是iOS，Material组件库中所有组件交互方式都会和iOS平台对齐，defaultTargetPlatform的值也会变为TargetPlatform.iOS。