**突破65k限制**

目录

[错误出现 2](#_Toc445311350)

[65k限制 2](#_Toc445311351)

[5.0之前的Multidex 2](#_Toc445311352)

[5.0之后的Multidex 3](#_Toc445311353)

[避免65k限制 3](#_Toc445311354)

[配置Multidex通过gradle 3](#_Toc445311355)

[Multidex支持库的限制 4](#_Toc445311356)

[优化Multidex开发版本 5](#_Toc445311357)

[Build Variants 6](#_Toc445311358)

[测试 Multidex App 7](#_Toc445311359)

# 错误出现

伴随着Android平台的继续增长，Android应用程序的尺寸也在变大。当你的应用程序或者引用库达到一定规模，你会碰到一个build错误：你的应用到达了Android应用构架架构的最大限制。早期版本构建系统报告如下错误：

Conversion to Dalvik format failed:  
Unable to execute dex: method ID not in [0, 0xffff]: 65536

最近版本构建系统会报告不同的错误：

trouble writing output:  
Too many field references: 131000; max is 65536.  
You may try using --multi-dex option.

这两种错误显示一个共同的数字：65536。这个数字表示在一个Dalvik Executable(dex)字节码可以调用的引用总数。如果你创建一个app，并收到这个错误。首先恭喜你，你有很多代码！接下来帮你突破这个限制，继续构建应用程序。

# 65k限制

Android应用程序包含的可执行文件Dex,其中包含用于运行你的应用程序的可执行字节码文件。Dalvik可执行规范限制了可以在单个dex文件内引用65536：包括Android框架方法，库方法，并在自己的代码中实现的方法总数。突破这个限制需要配置你的应用程序构建过程，生成多个dex文件，被称为multidex配置。

# 5.0之前的Multidex

5.0之前的版本使用Dalvik运行时执行应用程序代码。默认情况下，Dalvik限制应用中每个apk的classes.dex字节码文件。为了解决这个限制，可以使用multidex支持库，成为你的应用程序主dex文件的一部分。然后设法获得额外的dex文件和它们包含的代码。

# 5.0之后的Multidex

5.0之后的版本使用ART运行时执行应用程序代码。ART本身支持应用程序apk文件加载多个Dex文件。ART在应用程序执行预编译安装时它会扫描所有的dex文件，并将其编译成由Android设备执行的单个文件：.oat文件。

# 避免65k限制

在配置您的应用程序，以便使用65K或以上方法的引用，您应该采取措施，以减少你的应用程序代码调用引用的总次数，包括您的应用程序代码或包含的库定义的方法。以下策略可以帮助您避免击中DEX参考限值：

* 查看应用的直接和传递依赖 - 确保依赖大的library胜过代码直接被添加到应用程序中使用。一个常见的​​反模式是包含一个非常大的library，只有很少的方法是有用的。减少你的应用程序代码依赖性往往可以帮助你避免DEX参考限值。
* 使用ProGuard删除未使用的代码-为您的应用程序配置ProGuard的设置来运行ProGuard，并将其确定您已经缩水的版本。确保你不发布未使用的代码在APK中 。

使用这些方法可以使你的应用减少引用。可以降低APK 的尺寸，这对其中带宽成本很高的市场是很重要的。

# 配置Multidex通过gradle

在Android SDK中为gradle提供的构建工具 Android SDK Build Tools 21.1和更高版本支持multidex作为构建配置的一部分。在配置之前，请确保更新了Android SDK Build Tools和Android Support Repository 到最新的版本。

设置应用开发项目中使用multidex配置要求做一些修改应用程序开发项目。特别需要执行以下步骤：

* 改变你的gradle构建配置multidex
* 修改你的manifest文件

修改gradle的配置，包括支持库。 在gradle构建文件中指定multiDexEnabled，可以在defaultConfig ，buildType或productFlavor节设置。

android {  
    compileSdkVersion 21  
    buildToolsVersion "21.1.0"  
  
    defaultConfig {  
        ...  
        minSdkVersion 14  
        targetSdkVersion 21  
        ...  
  
        // Enabling multidex support.  
        multiDexEnabled true  
    }  
    ...  
}  
  
dependencies {  
  compile 'com.android.support:multidex:1.0.0'  
}

在mainfest文件中添加MultidexApplication

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.example.android.multidex.myapplication">  
    <application  
        ...  
        android:name="android.support.multidex.MultiDexApplication">  
        ...  
    </application>  
</manifest>

当这些配置设置添加到应用程序， Android编译工具会构建一个主DEX （ classes.dex ）及其配套（ classes2.dex ， classes3.dex ） 。然后，构建系统将它们打包成APK文件。

如果应用程序使用扩展的Application，则可以覆盖attachBaseContext （）方法，并调用MultiDex.install （this） ，以使multidex 。欲了解更多信息，请参阅MultiDexApplication参考文档。

# Multidex支持库的限制

multidex支持库有一些已知的限制，当你将其纳入应用程序构建配置，你应该了解和测试。

* .dex文件启动期间安装到设备的数据分区是复杂的，如果二次DEX文件是大的，可能导致ANR。在这种情况下，你应该使用代码缩减技术使用ProGuard尽量减少DEX文件的大小，并删除代码的未使用的部分。
* 早于Android 4.0的（API级别14）平台版本的设备启动应用程序,使用multidex可能无法运行由于Dalvik的linearAlloc bug（ issue 22586）。如果你是早于14 API平台，确保执行与这些版本的平台测试您的应用程序可以在启动时的问题或当类特殊群体被加载。代码收缩可以减少或可能消除这些潜在的问题。
* 使用multidex配置过的应用，非常大的内存分配请求可能在运行时由于Dalvik的linearAlloc limit（issue 78035 ）崩溃。此分配限制是在Android 4.0的（API级别14 ）增加，但应用仍有可能Android 5.0 （API级别21）之前碰上在Android版本中此限制。
* 很多复杂的需求包括：在基础的dex运行在Dalvik中需要的类文件。Android编译工具的更新处理Android的要求，但它可能是其他包括library有额外的依赖性要求，包括使用内省或Java方法调用的本地代码。某些库可能无法使用，不可以指定必须包括在主DEX文件类，除非multidex构建工具的更新，，。

# 优化Multidex开发版本

一个multidex配置要求显著增加构建的处理时间，因为构建系统必须做出什么类必须包含在主DEX文件，哪些类可以包含在二次DEX文件复杂的决定。这意味着，multidex常规编译通常需要更长的时间，并有可能减缓你的开发过程。  
为了减少编译multidex时间，你应该创建使用Android插件gradle productFlavors的构建输出两个味道：一个开发flavors和产品flavors。  
对于开发味，设置一个最小的SDK版本21.此设置更快产生multidex输出采用ART-支持的格式。对于释放香味，设置符合你的实际的最低支持率的最低SDK版本。此设置生成APK multidex是与更多的设备兼容，但需要更长的时间来建立。

下面构建配置示例演示了如何设置这些味道在摇篮构建文件：

android {  
    productFlavors {  
        // Define separate dev and prod product flavors.  
        dev {  
            // dev utilizes minSDKVersion = 21 to allow the Android gradle plugin  
            // to pre-dex each module and produce an APK that can be tested on  
            // Android Lollipop without time consuming dex merging processes.  
            minSdkVersion 21  
        }  
        prod {  
            // The actual minSdkVersion for the application.  
            minSdkVersion 14  
        }  
    }  
          ...  
    buildTypes {  
        release {  
            runProguard true  
            proguardFiles getDefaultProguardFile('proguard-android.txt'),  
                                                 'proguard-rules.pro'  
        }  
    }  
}  
dependencies {  
  compile 'com.android.support:multidex:1.0.0'  
}

完成此配置更改后，它结合了dev productFlavor和debug buildType的属性devDebug变种。使用此目标创建了ProGuard的disable， multidex enabled应用程序启用，并设置的minSdkVersion到Android API 21级这些设置导致了Android gradle这个插件来做到以下几点:

* 将每个module（包括dependency）创建成单独的dex文件。这个过程叫做pre-dexing.
* 包括不加修改地在每个DEX 文件。
* 最重要的是，模块DEX文件将不会合并，所以长期运行计算来确定主DEX文件的内容是可以避免的。

这些设置会导致快速，增量编译，因为只有修改模块的DEX文件重新计算和重新包装成APK文件。从这些结果APK建立可用于仅5.0设备在Android上进行测试。然而，通过实现配置作为香味后，将保留执行正常的生成与释放，适当的最低水平SDK和ProGuard的设置的能力。  
还可以构建其它变体，包括一个prodDebug变种的构建，这需要更长的时间来建立，但可以用于测试开发之外。在所示的配置中， prodRelease变异将是最后的测试和发布版本。如果您在命令行中执行任务的gradle ，您可以使用带有DevDebug标准命令追加到结尾（例如./gradlew installDevDebug ） 。有关使用香精与摇篮任务的详细信息，请参阅Gradle Plugin用户指南。

你也可以提供一个自定义的清单，或者针对每种口味的自定义应用程序类，允许您使用支持库MultiDexApplication类，或调用MultiDex.install （）只对需要它的变种。

# 配置Build Varians

此页面建立在配置编译概述为您展示如何可以配置版本的变种，从一个单一的项目建立不同版本的应用程序，以及如何正确地管理你的依赖和签名配置。

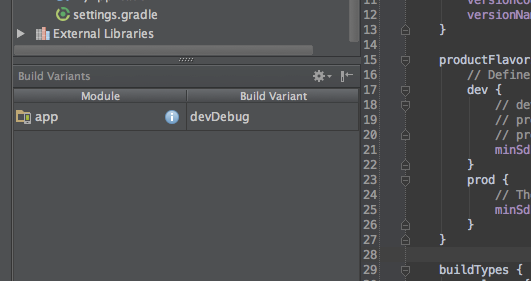
每个构建变量代表了不同版本的应用程序，你可以建立的。他们用的是特定的规则相结合的设置，代码，并在构建种类和口味的产品配置资源的摇篮结果。虽然你没有直接配置构建变种，你做配置构建类型和产品风味形成它们。

例如， “演示”产品风味可以指定不同的功能和设备的要求，如自定义源代码，资源和最低API级别，而“调试”构筑类型采用不同的构建和打包设置，如调试选项和签名键。产生的构建变种是您的应用程序的“ demoDebug ”版本，它包含了配置和资源的组合包含在“演示”产品的风味， “调试”构筑类型，主/源集。

# Build Variants

Android Studio中建立自己的应用程序的“ devDebug ”变种：

打开从左边的侧边栏build variants窗口。该选项旁边的收藏夹。



# 测试 Multidex App

当使用instrumentation测试 multidex应用程序，则需要额外的配置，以使测试仪器。由于代码在multidex应用类的位置不是一个单一的DEX文件内，instrumentation测试不正确除非配置为multidex运行。  
为了测试与仪器测试中multidex应用程序，从multidex测试支持库配置MultiDexTestRunner 。下面的示例的build.gradle文件演示了如何配置您的构建来使用这个测试运行：

android {  
  defaultConfig {  
      ...  
      testInstrumentationRunner "com.android.test.runner.MultiDexTestRunner"  
  }  
}

有了Android插件gradle版本低于1.1时，您需要添加以下依赖于multidex的instrument:

dependencies {  
    androidTestCompile('com.android.support:multidex-instrumentation:1.0.1') {  
         exclude group: 'com.android.support', module: 'multidex'  
    }         
}

你可以直接使用instrumentation测试运行的类或扩展，以满足您的测试需求。或者，您可以在这样现有的仪器仪表覆盖的onCreate ：

public void onCreate(Bundle arguments) {  
    MultiDex.install(getTargetContext());  
    super.onCreate(arguments);  
    ...  
}

目前不支持创建测试APK使用multidex的。