Android App Bundle 简介

重要提示：2021 年下半年，新应用需要使用 Android App Bundle (/guide/app-bundle) 才能在 Google Play 中发布。大小超过 150 MB 的新应用必须使用 Play Feature Delivery (/guide/app-bundle/dynamic-delivery) 或 Play Asset Delivery (/guide/app-bundle/asset-delivery)。

Android App Bundle 是一种发布格式，其中包含您应用的所有经过编译的代码和资源，它会将 APK 生成及签名交由 Google Play 来完成。

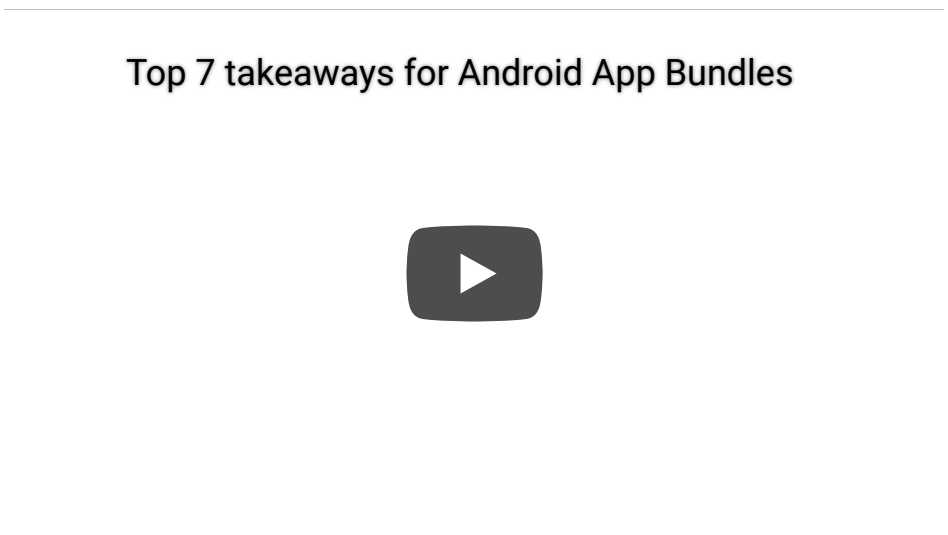
Google Play 会使用您的 App Bundle 针对每种设备配置生成并提供经过优化的 APK，因此只会下载特定设备所需的代码和资源来运行您的应用。您不必再构建、签署和管理多个 APK 来优化对不同设备的支持，而用户也可以获得更小且更优化的下载文件包。

对于大多数应用项目而言，构建 App Bundle 以支持提供经过优化的 APK 并不费力。例如，如果您已经按照既定惯例组织管理应用的代码和资源 (/guide/topics/resources/providing-resources#AlternativeResources)，只需使用 Android Studio 或使用命令行 (/studio/build/building-cmdline)来构建已签名的 Android App Bundle (/studio/publish/app-signing#sign-apk)，并将它们上传到 Google Play (/studio/publish/upload-bundle)。然后，就能提供经过优化的 APK，自动获享其带来的优势了。

使用 App Bundle 格式发布应用时，您也可以选择使用 Play Feature Delivery (/guide/app-bundle/dynamic-delivery)，它可让您向应用项目中添加功能模块。您可以选择在用户首次下载并安装您的应用时排除这些模块中包含的某些功能和资源。使用 Play 核心库 (#playcore)，您的应用日后可以请求下载这些模块。Google Play 只会将该模块的代码和资源提供给设备。

使用 App Bundle 发布应用的游戏开发者可以使用 Play Asset Delivery (/guide/app-bundle/asset-delivery)：它是 Google Play 的解决方案，用于分发大量的游戏资产，为开发者提供了灵活的分发方式和极高的性能。

请观看下面的视频，大致了解为什么应使用 Android App Bundle 发布应用。



使用 Android App Bundle 时，压缩下载大小限制 (#size_restrictions)现在为 150MB。您不能将 app bundle 与 APK 扩展文件一起使用。

如果您使用的是 Android Studio 3.2 或更高版本，只需点击几下即可构建 Android App Bundle。

本页介绍了开始构建 Android App Bundle 的步骤，还介绍了与 app bundle 相关的一些重要概念。

开始构建

App bundle 与 APK 的不同之处在于，您无法将其部署到设备。相反，它是一种发布格式，将您应用的所有经过编译的代码和资源包含在一个构建工件中。因此，在您上传已签名的 App Bundle 后，Google Play 就具备了构建和签署应用 APK 并将其提供给用户所需的一切。

如果您使用的是 Android Studio，只需点击几下即可将您的项目构建为已签名的 App Bundle (/studio/run#reference)。如果您没有使用 IDE，则可以改为从命令行构建 app bundle (/studio/build/building-cmdline#build_bundle)。然后，将您的 app bundle 上传 (/studio/publish/upload-bundle)到 Play 管理中心，以测试或发布您的应用。

如需构建 App Bundle，请按以下步骤操作：

1. 下载 Android Studio 3.2 或更高版本 (/studio) - 这是添加功能模块和构建 App Bundle 最简单的方式。
2. 添加对 Play Feature Delivery 的支持 (/studio/projects/dynamic-delivery)，具体方法是添加基本模块，整理用于配置 APK 的代码和资源，然后按需添加功能模块。
3. 使用 Android Studio 构建 Android App Bundle (/studio/run#reference)。您还可以通过修改运行/调试配置 (/studio/run/rundebugconfig#android-application)并选择**从 app bundle 部署 APK** 的选项，从 app bundle 将应用部署到连接的设备。请注意，与仅构建和部署 APK 相比，使用此选项会使构建时间延长。
    - 如果您没有使用 IDE，可以改为从命令行构建 App Bundle (/studio/build/building-cmdline#build_bundle)。
4. 使用 Android App Bundle 生成部署到设备的 APK 来测试您的 Android App Bundle (#bundletool)。
5. 注册 Play 应用签名服务 (https://support.google.com/googleplay/android-developer/answer/7384423)。否则，您无法将 App Bundle 上传到 Play 管理中心。

6. 将您的 App Bundle 发布到 Google Play (/studio/publish/upload-bundle)。

如需构建包含资源包的 App Bundle，请参阅 Play Asset Delivery 简介 (/guide/app-bundle/asset-delivery)。

测试您的 app bundle

构建 Android App Bundle 后，您必须测试 Google Play 使用该 Android App Bundle 生成 APK 的情形，以及这些 APK 部署到设备上之后的行为。

如需测试 App Bundle，请使用以下任一方法：

- 使用 bundletool 在本地测试 Android App Bundle (/studio/command-line/bundletool)，此测试会根据您的 App Bundle 生成 APK 并将其部署到连接的设备上。
- 通过网址分享应用 (https://support.google.com/googleplay/android-developer/answer/9303479)。通过这种方式，您能够以最快的速度上传 app bundle 并通过 Google Play 商店中的链接将应用分享给受信任的测试人员。此外，这也是测试自定义分发选项（如按需下载功能）的最快方式。
- 设置开放式测试、封闭式测试或内部测试 (https://support.google.com/googleplay/android-developer/answer/3131213)。这是测试自定义分发选项（如按需下载功能）的另一种方法。

使用 Play 核心库下载功能模块

如果您的应用包含功能模块，需要使用 Play 核心库请求、监控和管理功能模块下载。如需了解详情，请转到使用 Play 核心库下载模块 (/guide/app-bundle/playcore)。

如果您想看看该库的实际应用，请试用 Play 核心库示例应用 (https://github.com/android/app-bundle-samples/tree/main/DynamicFeatures)。

关于免安装应用的说明

在 Android Studio 3.2 或更高版本中，只要应用足够小，您就可以向 App Bundle 添加免安装体验 (/topic/google-play-instant/getting-started/instant-enabled-app-bundle)。如需详细了解可创建的各种免安装体验的大小限制，请参阅 Google Play 免安装体验概览 (/topic/google-play-instant/overview#reduce-size)。

压缩下载大小限制

使用 Android App Bundle 发布应用可帮助用户以尽可能最小的下载大小安装您的应用，并将**压缩下载大小上限提高到 150MB**。也就是说，当用户下载您的应用时，安装应用所需的压缩 APK（例如，基本 APK + 配置 APK）的总大小不得超过 150 MB。任何后续下载内容（如按需下载功能模块（及其配置 APK））也必须满足此压缩下载大小限制。Asset Pack 不受此大小限制，但它们有其他大小限制 (/guide/app-bundle/asset-delivery#size-limits)。

上传 app bundle 时，如果 Play 管理中心发现您的应用或其按需功能的可能下载大小超过 150MB，您会收到错误。

请注意，**Android App Bundle 不支持 APK 扩展 (`*.obb`) 文件**。因此，如果您在发布 App Bundle 时遇到此错误，请使用以下某种资源来缩减压缩的 APK 下载大小：

- 请务必为每种类型的配置 APK 设置 `enableSplit = true` 以启用所有配置 APK (/studio/projects/dynamic-delivery#disable_config_apks)。这样可以确保用户只下载在其设备上运行您的应用所需的代码和资源。
- 请务必移除不用的代码和资源以缩减应用大小 (/studio/build/shrink-code)。
- 遵循最佳做法以进一步缩减应用大小 (/topic/performance/reduce-apk-size)。
- 考虑将只有部分用户使用的功能转换为应用可以在日后按需下载的功能模块 (/studio/projects/dynamic-delivery#dynamic_feature_modules)。请注意，这可能需要对您的应用稍微进行重构，因此请务必先尝试上述其他建议。

已知问题

以下介绍了当前已知在使用 Android App Bundle 构建或提供应用时会出现的问题。如果您遇到下文尚未说明的问题，请报告错误 (https://issuetracker.google.com/issues/new?component=398856&template=1084213)。

- 部分安装旁加载的应用（即，未通过 Google Play 商店安装且缺少一个或多个必需拆分 APK 的应用）在所有经过 Google 认证的设备上以及搭载 Android 10（API 级别 29）或更高版本的设备上都会失败。通过 Google Play 商店下载您的应用时，Google 会确保安装应用的所有必需组件。
- 如果您使用会动态修改资源表的工具，则从 app bundle 生成的 APK 可能会行为异常。因此，在构建 app bundle 时，建议您停用此类工具。
- 在功能模块的清单中，您不得引用基本模块中不存在的资源。这是因为，在 Google Play 生成应用的基本 APK 时，会将所有模块的清单合并到基本 APK 的清单中。因此，如果基本 APK 的清单引用了基本 APK 中不存在的资源，资源关联就会断开。
- 从 Android Studio 3.2 Canary 14 开始，当您针对应用的基本模块更改构建变体 (/studio/run#changing-variant)时，系统不会为依赖基本模块的功能模块自动选择相同的构建变体。因此，您可能会在构建应用时收到错误。只需确保为基本模块和依赖于它的其他模块选择相同的构建变体即可。
- 目前可以在功能模块的构建配置中配置与基本模块或其他模块中的属性相冲突的属性。例如，您可以在基本模块中设置 `buildTypes.release.debuggable = true`，而在功能模块中将其设置为 `false`。此类冲突可能会导致构建和运行时问题。请注意，默认情况下，功能模块会从基本模块继承一些构建配置。因此，请务必了解在功能模块构建配置 (/guide/app-bundle/configure#feature_build_config)中应保留哪些配置以及应省略哪些配置。
- 为了下载功能模块，设备必须安装最新版本的 Play 商店应用。因此，如果您的应用包含功能模块，那么很少一部分用户的下载内容可能会回退到单个经过优化的多 APK，这会为搭载 Android 4.4（API 级别 20）及更低版本的设备提供相同的下载体验。

Android App Bundle 格式

Android App Bundle 是您上传到 Google Play 的一种文件（文件扩展名为 `.aab`）。

app bundle 是经过签名的二进制文件，可将应用的代码和资源组织到模块中，如图 1 所示。各个模块的代码和资源的组织方式与 APK 中的相似，这是合理的，因为每个模块都可以作为单独的 APK 生成。然后，Google Play 会使用 app bundle 生成向用户提供的各种 APK，如基本 APK、功能 APK、配置 APK 以及多 APK（多 APK 适用于不支持拆分 APK 的设备）。以蓝色标识的目录（如 `drawable/`、`values/` 和 `lib/` 目录）表示 Google Play 用来为每个模块创建配置 APK 的代码和资源。

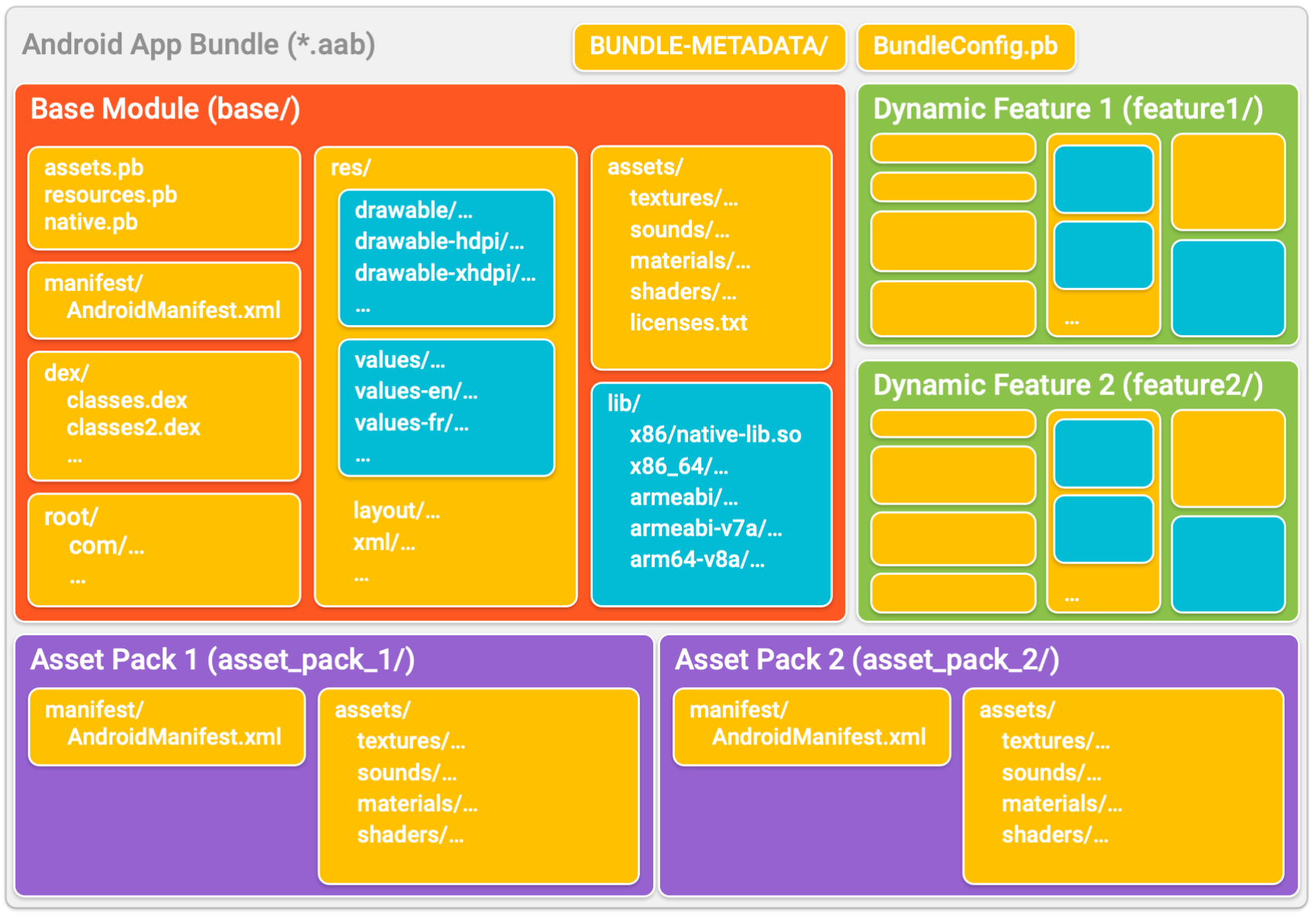


图 1. 包含一个基本模块、两个功能模块和两个资源包的 Android App Bundle 的内容。

注意：您需要为每个唯一的应用或 applicationID 构建一个 App Bundle。也就是说，如果您使用产品变种从单个应用项目创建多个版本的应用，且每个版本都使用唯一的 applicationID (/studio/build/application-id)，则需要为每个应用版本构建单独的 app bundle。

以下列表更详细地介绍了 App Bundle 的部分文件和目录：

- **base/、feature1/ 和 feature2/**：其中每个顶级目录都表示一个不同的应用模块。应用的基本模块始终包含在 App Bundle 的 `base` 目录中。不过，为每个功能模块的目录提供的名称由模块清单中的 `split` 属性指定。如需了解详情，请参阅功能模块清单 (/guide/app-bundle/configure#dynamic_feature_manifest)。
- **asset_pack_1/ 和 asset_pack_2/**：对于需要大量图形处理的大型应用或游戏，您可以将资产模块化处理为资源包。Asset Pack 因大小上限较高而成为游戏的理想之选。您可以按照三种分发模式（即，安装时分发、快速跟进式分发和按需分发）自定义如何以及何时将各个 Asset Pack 下载到设备上。所有 Asset Pack 都在 Google Play 上托管并从 Google Play 提供。如需详细了解如何将 Asset Pack 添加到您的 app bundle，请参阅 Play Asset Delivery 概览 (/guide/app-bundle/asset-delivery)。
- **BUNDLE-METADATA/**：此目录包含元数据文件，其中包含对工具或应用商店有用的信息。此类元数据文件可能包含 ProGuard 映射和应用的 DEX 文件的完整列表。此目录中的文件未打包到您应用的 APK 中。
- **模块协议缓冲区 (`*.pb`) 文件**：这些文件提供了一些元数据，有助于向各个应用商店（如 Google Play）说明每个应用模块的内容。例如，`BundleConfig.pb` 提供了有关 bundle 本身的信息（如用于构建 app bundle 的构建工具版本），`native.pb` 和 `resources.pb` 说明了每个模块中的代码和资源，这在 Google Play 针对不同的设备配置优化 APK 时非常有用。
- **manifest/**：与 APK 不同，app bundle 将每个模块的 `AndroidManifest.xml` 文件存储在这个单独的目录中。
- **dex/**：与 APK 不同，app bundle 将每个模块的 DEX 文件存储在这个单独的目录中。
- **res/、lib/ 和 assets/**：这些目录与典型 APK 中的目录完全相同。当您上传 App Bundle 时，Google Play 会检查这些目录并且仅打包满足目标设备配置需求的文件，同时保留文件路径。
- **root/**：此目录存储的文件之后会重新定位到包含此目录所在模块的任意 APK 的根目录。例如，app bundle 的 `base/root/` 目录可能包含您的应用使用 `Class.getResource()` (/reference/java/lang/Class#getResource(java.lang.String)) 加载的基于 Java 的资源。这些文件之后会重新定位到您应用的基本 APK 和 Google Play 生成的每个多 APK 的根目录。此目录中的路径也会保留下来。也就是说，目录（及其子目录）也会重新定位到 APK 的根目录。

注意：如果此目录中的内容与 APK 根目录下的其他文件和目录发生冲突，则 Play 管理中心会在上传时拒绝整个 app bundle。例如，您不能包含 `root/lib/` 目录，因为它会与每个 APK 已包含的 `lib` 目录发生冲突。

拆分 APK 概览

提供经过优化的应用所需的一个基本组件就是 Android 5.0（API 级别 21）及更高版本上提供的拆分 APK 机制。拆分 APK 与常规 APK 非常相似，其中包含经过编译的 DEX 字节码、资源和 Android 清单。不过，Android 平台能够将安装的多个拆分 APK 视为一个应用。也就是说，您可以安装多个拆分 APK，它们共用代码和资源，并且在设备上看起来像是安装的一个应用。

拆分 APK 的好处是能够将单体式 APK 拆分为更小的独立软件包，这些软件包可“按需”安装在用户设备上。单体式 APK 是指一个 APK 中包含应用所支持的全部功能和设备配置对应的代码和资源。

例如，一个拆分 APK 可能包含只有少数用户需要的附加功能所对应的代码和资源，而另一个拆分 APK 则只包含特定语言或屏幕密度所对应的资源。这些拆分 APK 可以根据用户的请求或设备的需要单独下载和安装。

下面介绍可以一起安装在设备上以形成完整应用体验的不同类型的 APK。本页的后面几节将介绍如何配置应用项目以支持这些 APK。

- **基本 APK**：此 APK 中包含了所有其他拆分 APK 都可以访问的代码和资源，并提供应用的基本功能。当用户请求下载您的应用时，会首先下载并安装该 APK。这是因为只有基本 APK 的清单才包含关于应用的服务、内容提供方、权限、平台版本要求和对系统功能的依赖性的完整声明。Google Play 会根据项目的应用模块（即基本模块）为应用生成基本 APK。如果您想减小应用的初始下载大小，请一定要注意，此模块中包含的所有代码和资源都包含在应用的基本 APK 中。
- **配置 APK**：每个配置 APK 都包含针对特定屏幕密度、CPU 架构或语言的原生库和资源。当用户下载您的应用时，他们的设备只会下载并安装该设备对应的配置 APK。每个配置 APK 都是基本 APK 或功能模块 APK 的依赖项。也就是说，配置 APK 会随它们为之提供代码和资源的 APK 一起下载和安装。与基本模块和功能模块不同，您不需要为配置 APK 单独创建模块。如果您在为基本模块和功能模块组织管理配置专用的备用资源 (/guide/topics/resources/providing-resources#AlternativeResources)时遵循了标准实践，**Google Play 会自动为您生成配置 APK。**
- **功能模块 APK**：每个功能模块 APK 都包含您使用功能模块进行了模块化处理的某项应用功能的代码和资源。您随后可以自定义如何以及何时将该功能下载到设备上。例如，使用 Play 核心库 (/guide/app-bundle/playcore)，可在将基本 APK 安装到设备上之后再按需安装某

些功能，以向用户提供额外的功能。假设我们有一款聊天应用，它仅在用户想要拍摄并发送照片时才下载并安装该功能。由于功能模块在安装时可能不可用，因此您应将所有通用代码和资源包含在基本 APK 中。也就是说，您的功能模块应假定在安装时只有基本 APK 的代码和资源可用。Google Play 会根据项目的功能模块为应用生成功能模块 APK。

假设我们有一款包含三个功能模块并支持多种设备配置的应用。下面的图 1 展示了该应用的各个 APK 的依赖关系树可能是什么样子的。请注意，基本 APK 构成树的头部，所有其他 APK 都依赖于基本 APK。（如果您想知道这些 APK 的模块在 Android App Bundle 中的表示方式，请参阅 Android App Bundle 格式 (/guide/app-bundle/build)。）

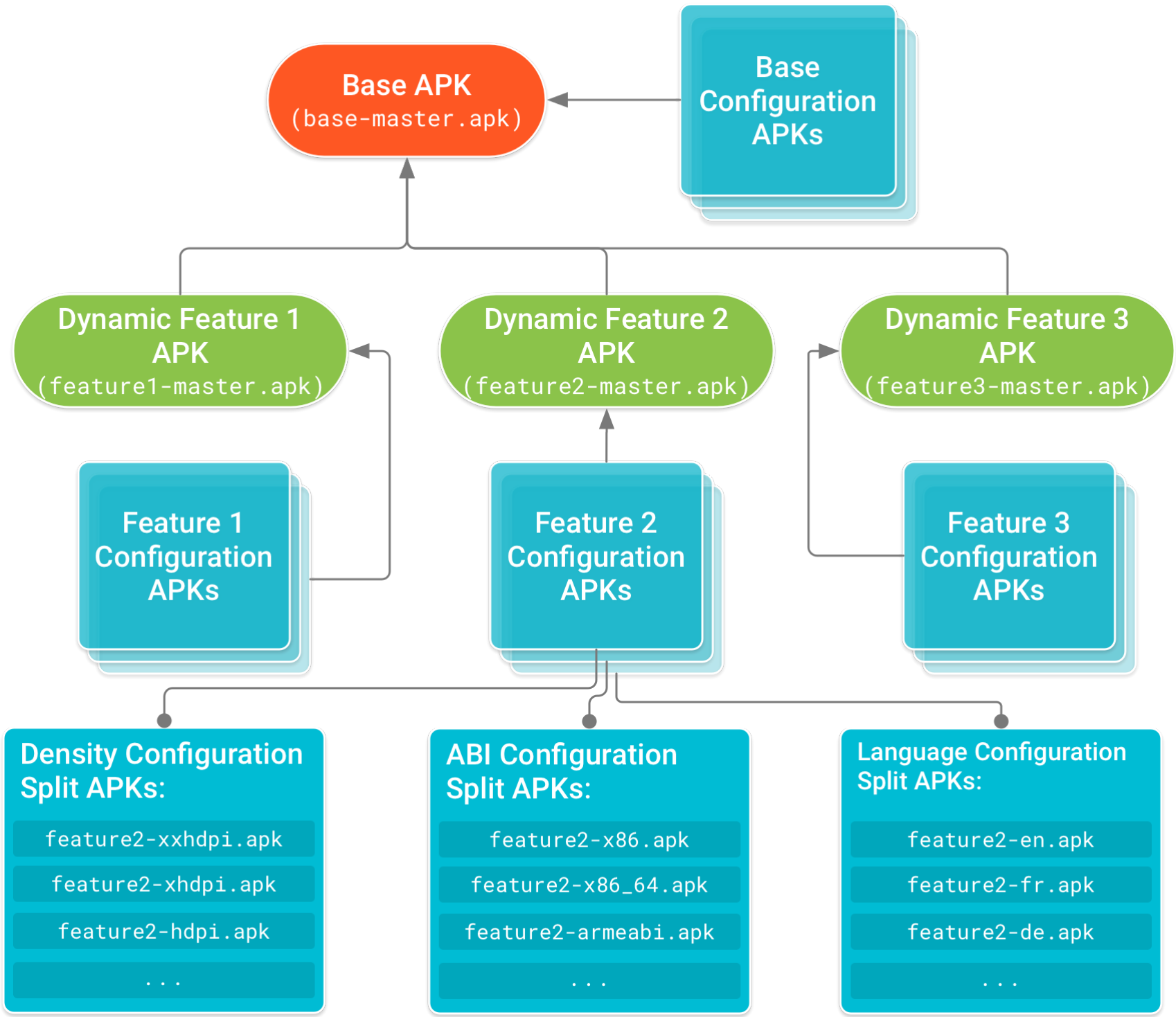


图 1. 使用拆分 APK 构建的应用的依赖关系树

请注意，您无需自己构建这些 APK，Google Play 会根据您通过 Android Studio 构建的单个签名 App Bundle 来为您完成构建。如需详细了解 app bundle 的格式和构建方式，请转到构建、部署和上传 Android App Bundle (/guide/app-bundle/build)。

搭载 Android 4.4（API 级别 19）及更低版本的设备

由于搭载 Android 4.4（API 级别 19）及更低版本的设备不支持下载和安装拆分 APK，因此 Google Play 会为这些设备提供名为 multi-APK 的单个 APK，该 APK 针对相应设备配置进行了优化。也就是说，multi-APK 提供完整的应用体验，但不包含不必要的代码和资源，例如用于其他屏幕密度和 CPU 架构的代码和资源。

但是，它们包含了应用所支持的所有语言的资源。例如，这让用户无需另外下载 multi-APK 即可更改应用的首选语言设置。

multi-APK 无法在以后按需下载功能模块。如需将功能模块包含在此 APK 中，必须在创建功能模块 (/studio/projects/dynamic-delivery#create_dynamic_feature)时停用**按需**选项或启用**融合**选项。

请注意，有了 app bundle，您无需为应用支持的每种设备配置分别构建、签署、上传和管理 APK。您仍然只需为整个应用构建和上传一个 app bundle，剩下的工作可交由 Google Play 处理。因此，无论您是否打算支持搭载 Android 4.4 或更低版本的设备，Google Play 都为您和您的用户提供了灵活的服务机制。

其他资源

要详细了解 Android App Bundle，请参阅以下资源。

示例

- PlayCore API 示例 (https://github.com/android/app-bundle-samples/tree/main/DynamicFeatures)：演示了如何使用 PlayCore API 请求和下载功能模块。
- 动态代码加载示例 (https://github.com/googlesamples/android-dynamic-code-loading)：演示了从安装的功能模块安全访问代码的三种不同方法。

Codelab

- 按需模块 (https://codelabs.developers.google.com/codelabs/on-demand-dynamic-delivery/index.html)：可帮助您打造按需下载和安装功能模块的应用。

博文

- 全新发布格式对 Android 的未来意味着什么
  (https://medium.com/googleplaydev/what-a-new-publishing-format-means-for-the-future-of-android-2e34981793a)
- 能够助力您在 Google Play 上开发和发布产品并拓展业务的新功能
  (https://android-developers.googleblog.com/2019/05/whats-new-in-play.html)
- 最新的 Android App Bundle 更新，包括针对其他语言的 API
  (https://android-developers.googleblog.com/2019/03/the-latest-android-app-bundle-updates.html)
- Patchwork Plaid - 模块化故事 (https://medium.com/androiddevelopers/a-patchwork-plaid-monolith-to-modularized-app-60235d9f212e)
- Google 追踪圣诞老人 - 正改为使用 Android App Bundle
  (https://medium.com/androiddevelopers/google-santa-tracker-moving-to-an-android-app-bundle-dde180716096)

视频

- 借助 App Bundle 实现自定义分发，并轻松分享测试 Build (https://www.youtube.com/watch?v=flhib2krW7U)
- 可让您在 Google Play 上优化应用大小和提升安装量的新工具 (https://www.youtube.com/watch?v=rEuwVWpYBOY)

Last updated 2021-01-27 UTC.