蚂蚁金服金融科技产品手册

接入 Android

产品版本： AntStack 1.3.0

文档版本： V20190316

蚂蚁金服金融科技文档

商标声明

及其他蚂蚁金服金融科技服务相关的商标均为蚂蚁金服金融科技所有。
本文档涉及的第三方的注册商标，依法由权利人所有。

免责声明

由于产品版本升级、调整或其他原因，本文档内容有可能变更。蚂蚁金服金融科技保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在蚂蚁金服金融科技授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过蚂蚁金服金融科技授权渠道下载、获取最新版的用户文档。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

目录

1 接入方式简介

两种接入方式

模块化是 mPaaS 框架 的核心设计理念。一个基于 mPaaS 框架开发的 App 包括：

- **一个或多个 Bundle 工程** ：一个 Bundle 即是一个业务独立的模块。
- **一个 Portal 工程** ：构建 App 时，需要先构建各 Bundle，再构建 Portal。Portal 负责将各 Bundle 的构建结果合并成一个可运行的 .apk 包。

接入方式分为 2 种：

- **基于 mPaaS 框架**：使用上述 mPaaS 框架，即将 App 分成 Portal 和 Bundle。使用这种方式，接入 mPaaS 组件（如移动网关、移动分析等）更加方便快捷。
- **基于原生框架**：基于原生 Android 框架，不使用上述 mPaaS 框架。

快速选择接入方式

如果 **从零开始开发全新应用**，那么推荐使用 **基于 mPaaS 框架** 的接入方式。请参考 学习路径 和 快速开始 。

如果 **已有 Android 应用**，您有以下两种选择：

- 您可以 **改造应用**，从而使用 **基于 mPaaS 框架** 的接入方式。请参考 迁移原生工程至 mPaaS 框架 。
- 如果 **改造成本大**，您可以使用 **基于原生框架** 的接入方式。请参考 基于原生框架 > 接入流程简介 。

相关链接

- mPaaS 框架介绍 > 简介
- 迁移原生工程至 mPaaS 框架
- 基于 mPaaS 框架 > 学习路径
- 基于 mPaaS 框架 > 快速开始
- 基于原生框架 > 接入流程简介

2 基于 mPaaS 框架

2.1 学习路径

阅读本文前，请您确保已阅读 接入方式简介 。

本文将简述 **基于 mPaaS 框架** 开发的学习路径。

快速开始

快速开始帮助您快速接入 mPaaS 框架，并得到可运行测试的简单应用。

- 配置开发环境
- 在控制台创建应用

- 本地开发

mPaaS 框架介绍

对照快速开始获得的应用代码，阅读该目录下的文档，您可以深入了解：

- mPaaS 框架的特点和优势。
- Portal 和 Bundle 工程的 **代码结构**、**编译打包结果** 以及 **与原生工程的区别**。

使用 mPaaS 插件

该目录下的文档详细介绍了 mPaaS 插件的功能。包括：

- 编译打包等常用功能。
- 热修复等和特定组件相关的功能。您可以在业务需要时参考此类文档。

启动框架

通过阅读该文，您可以了解 mPaaS 应用的启动流程以及定制启动流程的方法。

注册通用组件

模块化是 mPaaS 框架的设计原则之一。模块以 Bundle 的形式存在互不影响，但 Bundle 之间会存在一些关联性，比如跳转到另一个 Bundle 界面，调用另一个 Bundle 中的接口。您可以使用通用组件来实现这些需求。

使用 Material Design

根据业务需求，您可以通过 mPaaS 提供的 appcompat 库使用 Material Design 的图形界面特性。

混淆 Android 文件

该文帮助您在 mPaaS 框架下使用 ProGuard 混淆 Android 文件。

2.2 快速开始

2.2.1 通用步骤说明

如果使用 **基于 mPaaS 框架** 的接入方式，那么您一般需要完成以下通用步骤：

1. 配置开发环境
2. 在控制台创建应用
3. 客户端创建新应用

2.2.2 配置开发环境

2.2.2.1 准备配置

在进行客户端开发之前，您首先需要配置开发环境：

- macOS
- Linux
- Windows

2.2.2.2 Windows

配置 Windows 开发环境的步骤包括：

- 配置 Java 8 环境
- 配置 Gradle 4.4 环境
- 安装并配置 Android Studio 安装 Android Studio安装 Android SDK安装 mPaaS 插件配置 Gradle 构建工具

配置 Java 8 环境

mPaaS 框架只支持 **JDK 8+**：

1. 下载并安装 JDK 8。
2. 配置 JAVA_HOME 环境变量，并将 JAVA_HOME 下的 bin 路径添加到 PATH 环境变量中。
3. 正确配置后，在命令行执行 java -version 命令，您将看到 JDK 版本等信息：

```
d:\>java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
```

配置 Gradle 4.4 环境

mPaaS 框架只支持 **Gradle 4.4**：

1. 下载 Gradle 4.4.zip。
2. 解压 .zip 包，然后将解压路径配置为 GRADLE_HOME 环境变量，并将 GRADLE_HOME 下的 bin 路径添加到 PATH 环境变量中。
3. 正确配置后，在命令行执行 gradle -v 命令，您将看到 Gradle 版本等信息：

```
d:\>gradle -v

------------------------------------------------------------
Gradle 4.4
------------------------------------------------------------

Build time:   2017-12-06 09:05:06 UTC
Revision:     cf7821a6f79f8e2a598df21780e3ff7ce8db2b82

Groovy:       2.4.12
Ant:          Apache Ant(TM) version 1.9.9 compiled on February 2 2017
JVM:          1.8.0_121 (Oracle Corporation 25.121-b13)
OS:           Windows 7 6.1 amd64
```

安装并配置 Android Studio

安装 Android Studio

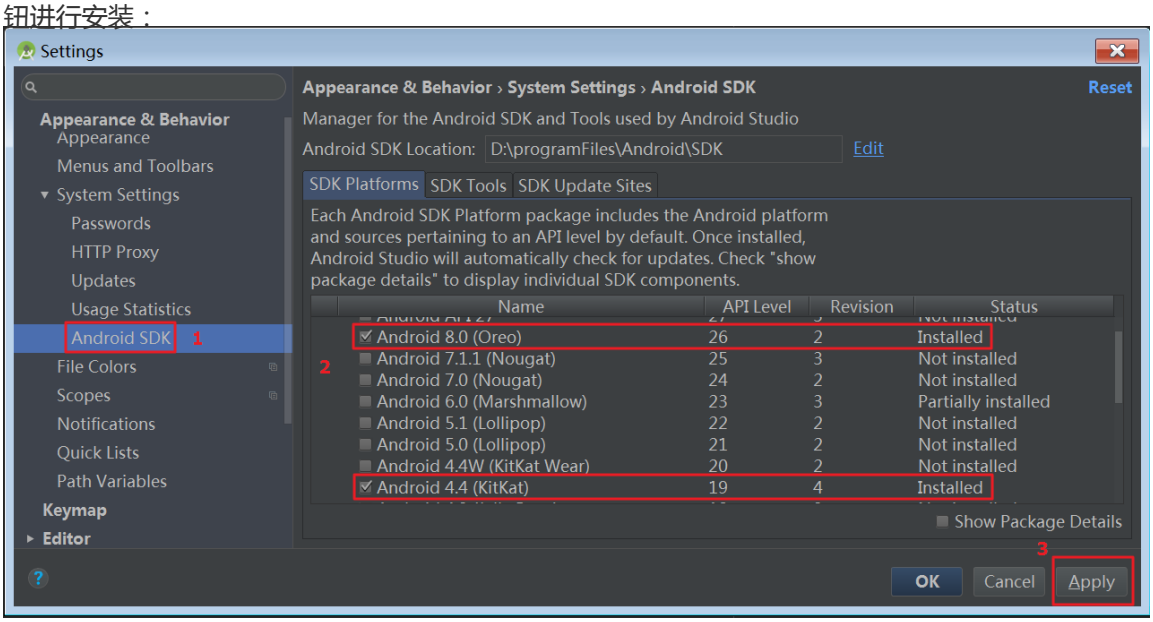
mPaaS 框架只支持 **3.0.x**、**3.1.x**、**3.2** 和 **3.3.x** 版本的 Android Studio。

3.0.1 版本下载地址：Android Studio 3.0.1；更多下载，请参见 官网。

安装 Android SDK

您需要安装 API Level 为 **19** 和 **26** 的 Android SDK：

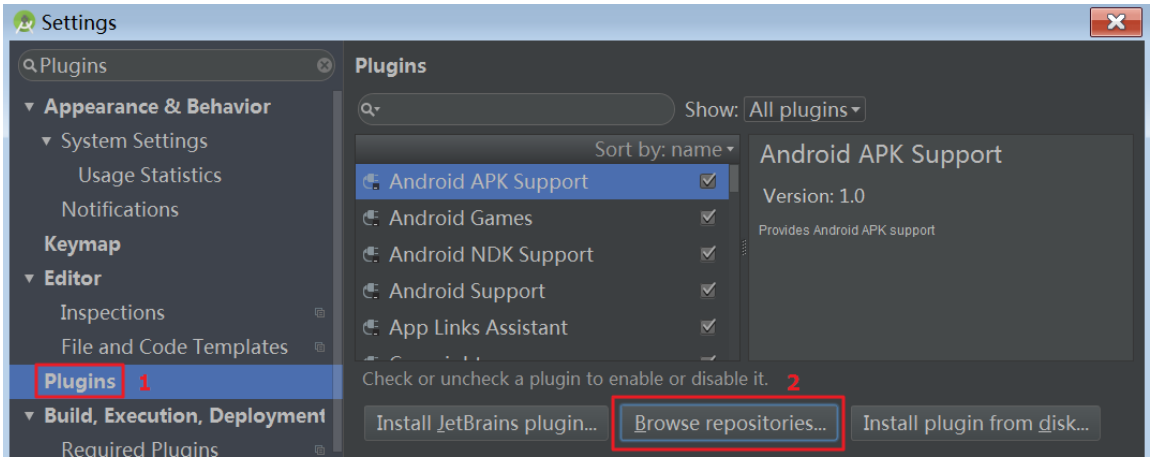
1. 在 Android Studio 中，通过 **File** > **Settings** 打开设置对话框。
2. 如下图所示，在 **Android SDK** 对话框中，勾选 API Level 为 **19** 和 **26** 的 SDK，然后点击 **Apply** 按钮进行安装：



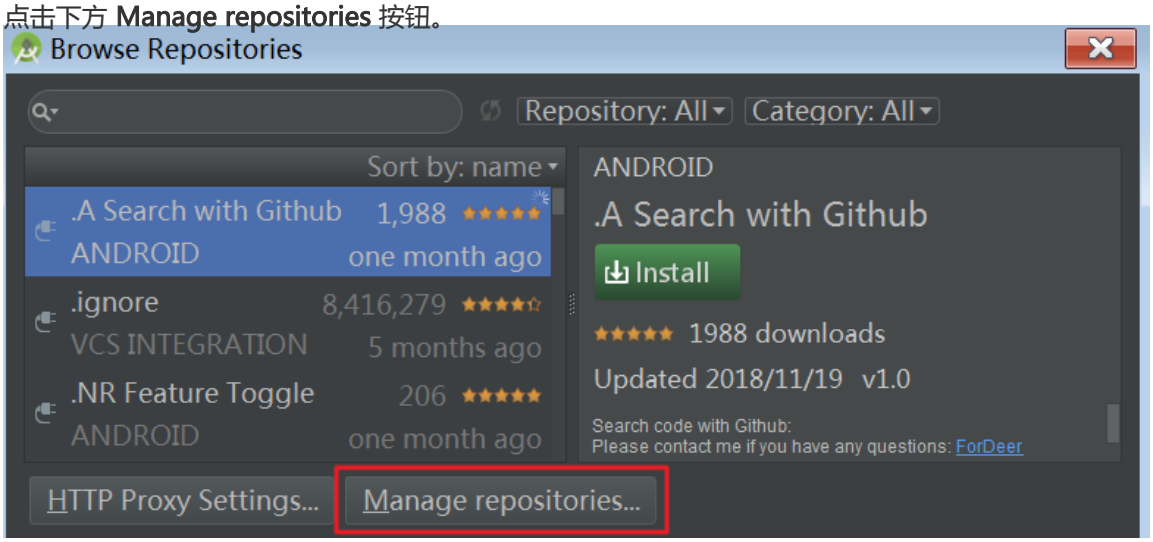
安装 mPaaS 插件

mPaaS 插件提供多种开发辅助功能，包括：新建 mPaaS 工程，添加、删除和升级 mPaaS 组件，构建工程等。安装步骤如下：

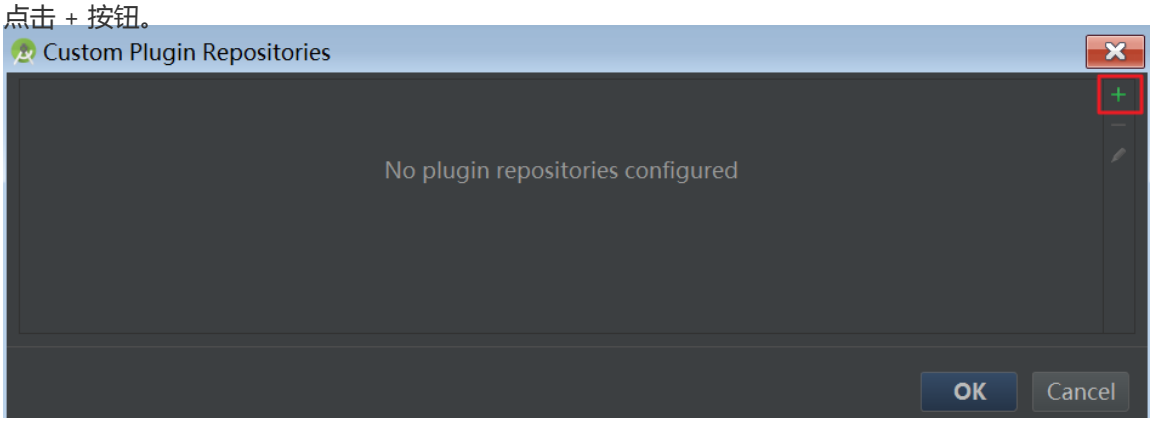
1. 打开 Android Studio 设置对话框。
2. 点击左侧 **Plugins**，然后点击下方 **Browse repositories** 按钮。



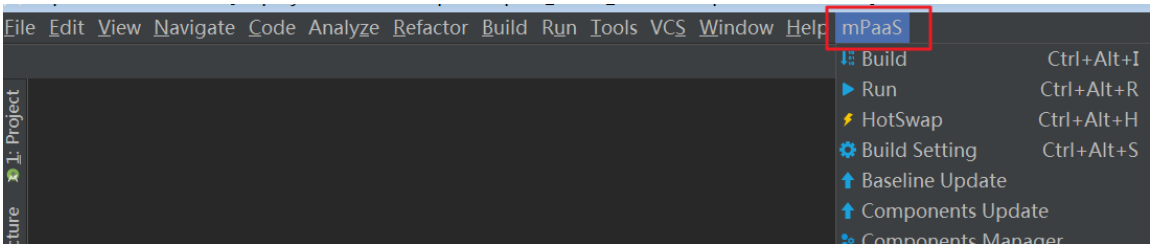
3. 点击下方 **Manage repositories** 按钮。



4. 点击 + 按钮。



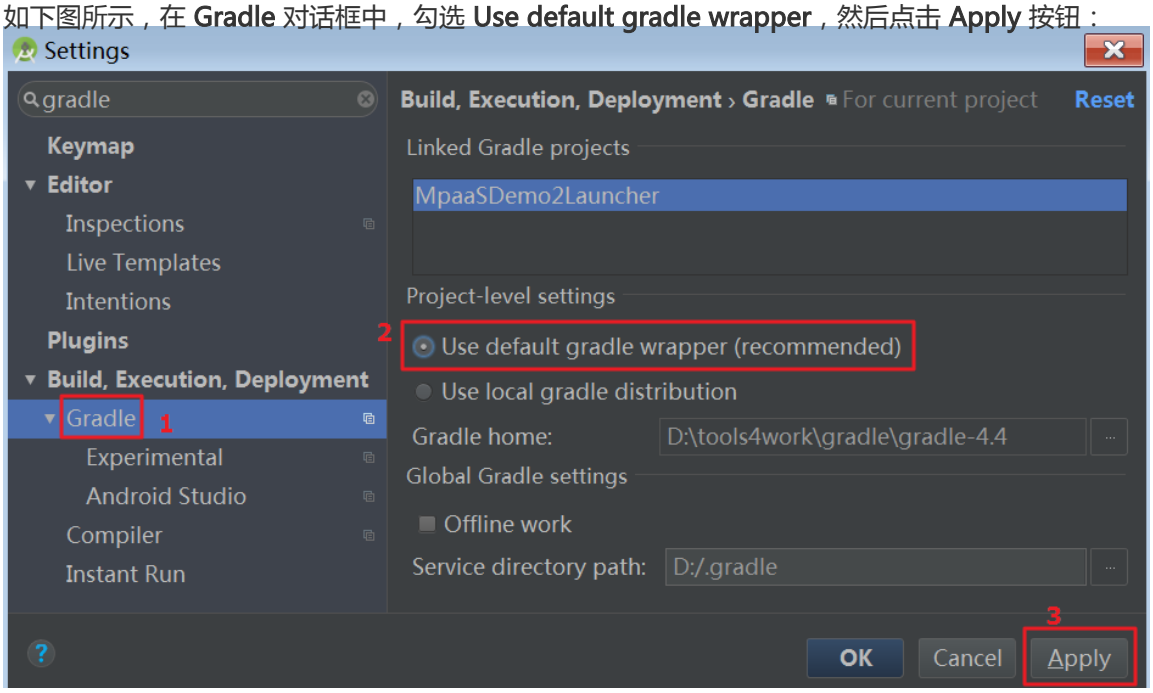
5. 在弹出框 **Repository URL** 中填入 https://mulei.oss-cn-hangzhou.aliyuncs.com/updatePlugins.xml，然后点击 **OK** 按钮。
6. 点击 Custom Plugin Repositories 对话框 **OK** 按钮，回到 Browse repositories 对话框。
7. 在 Browse repositories 对话框左上方的搜索框中输入 mPaaS。等待搜索结束后，点击搜索到的 mPaaS 插件，然后点击 **install** 按钮。
8. 等待插件安装结束，重启 Android Studio，您将看到 mPaaS 插件：



配置 Gradle 构建工具

您需要确保工程构建时使用 **Gradle Wrapper**：

1. 在 Android Studio 中打开任意一个 Android 工程。
2. 打开设置对话框。
3. 如下图所示，在 **Gradle** 对话框中，勾选 **Use default gradle wrapper**，然后点击 **Apply** 按钮：



2.2.2.3 macOS

按照以下描述配置 macOS 开发环境：

- 配置 Java 8 环境
- 配置 Gradle 4.4 环境
- 安装并配置 Android Studio 安装 Android Studio安装 Android SDK安装 mPaaS 插件配置 Gradle 构建工具

配置 Java 8 环境

mPaaS 框架只支持 **JDK 8+**：

1. 下载并安装 JDK 8。
2. 配置 JAVA_HOME 环境变量，并将 JAVA_HOME 下的 bin 路径添加到 PATH 环境变量中。

3. 正确配置后，在命令行执行 java -version 命令，您将看到 JDK 版本等信息：

```
~java -version
java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
```

配置 Gradle 4.4 环境

mPaaS 框架只支持 **Gradle 4.4**：

1. 下载 Gradle 4.4.zip。
2. 解压 .zip 包，然后将解压路径配置为 GRADLE_HOME 环境变量，并将 GRADLE_HOME 下的 bin 路径添加到 PATH 环境变量中。
3. 正确配置后，在命令行执行 gradle -v 命令，您将看到 Gradle 版本等信息：

```
~gradle -v

------------------------------------------------------------
Gradle 4.4
------------------------------------------------------------

Build time:   2017-12-06 09:05:06 UTC
Revision:     cf7821a6f79f8e2a598df21780e3ff7ce8db2b82

Groovy:       2.4.12
Ant:          Apache Ant(TM) version 1.9.9 compiled on February 2 2017
JVM:          1.8.0_201 (Oracle Corporation 25.201-b09)
OS:           Mac OS X 10.14.3 x86_64
```

安装并配置 Android Studio

安装 Android Studio

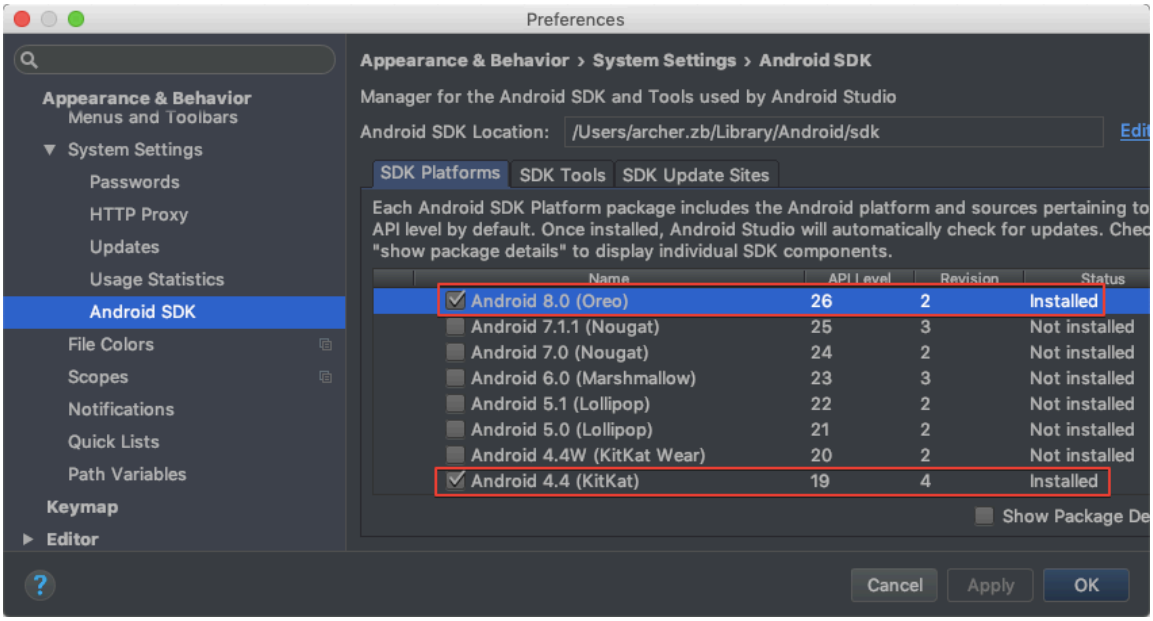
mPaaS 框架只支持 **3.0.x**、**3.1.x**、**3.2** 和 **3.3.x** 版本的 Android Studio。

3.0.1 版本下载地址：Android Studio 3.0.1；更多下载，请参见 官网。

安装 Android SDK

您需要安装 API Level 为 **19** 和 **26** 的 Android SDK：

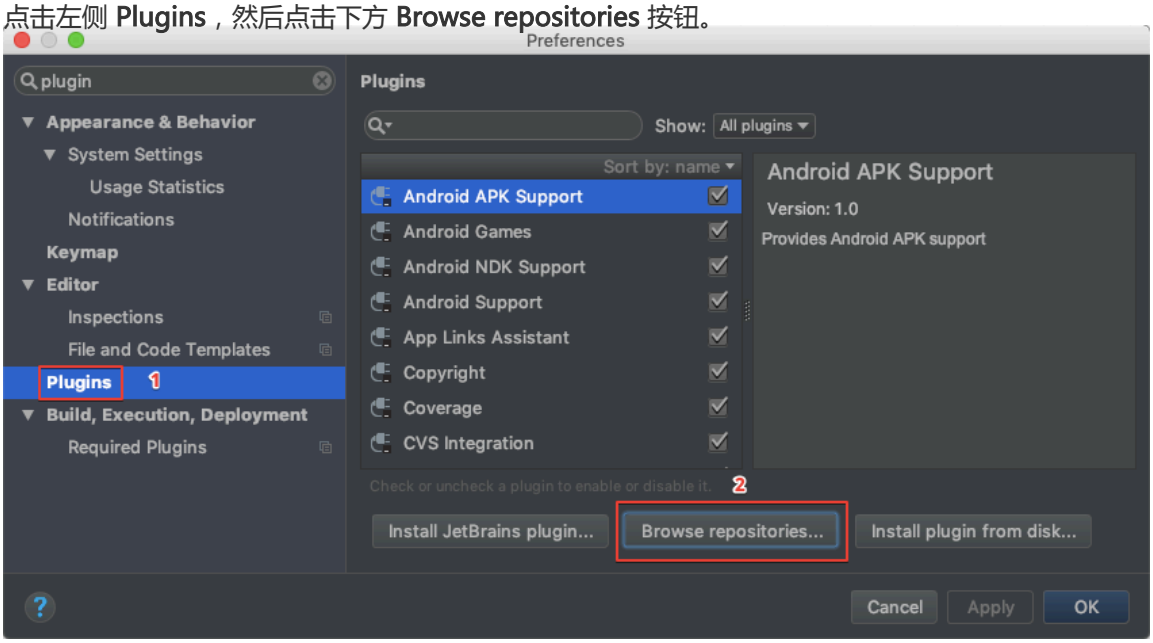
1. 在 Android Studio 中，通过 **Android Studio** > **Preferences** 打开设置对话框。
2. 如下图所示，在 **Android SDK** 对话框中，勾选 API Level 为 **19** 和 **26** 的 SDK，然后点击 **Apply** 按钮进行安装：



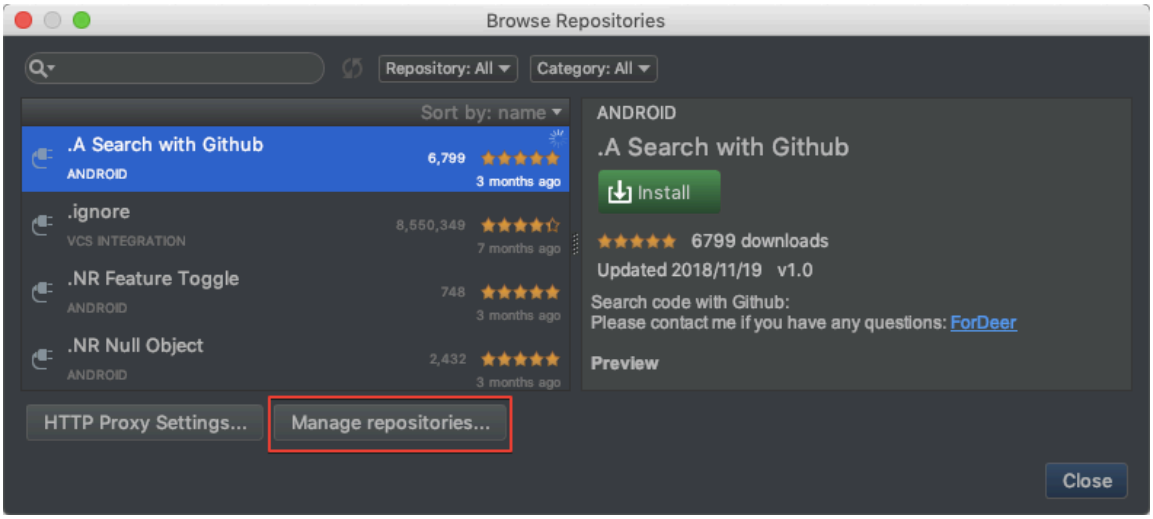
安装 mPaaS 插件

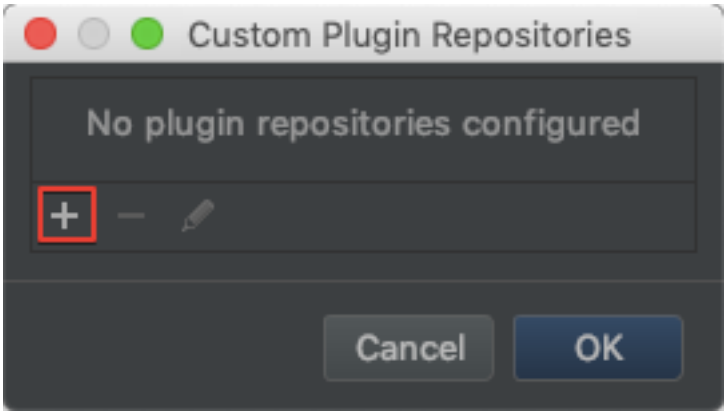
mPaaS 插件提供多种开发辅助功能，包括：新建 mPaaS 工程，添加、删除和升级 mPaaS 组件，构建工程等。安装步骤如下：

1. 打开 Android Studio 设置对话框。
2. 点击左侧 **Plugins**，然后点击下方 **Browse repositories** 按钮。

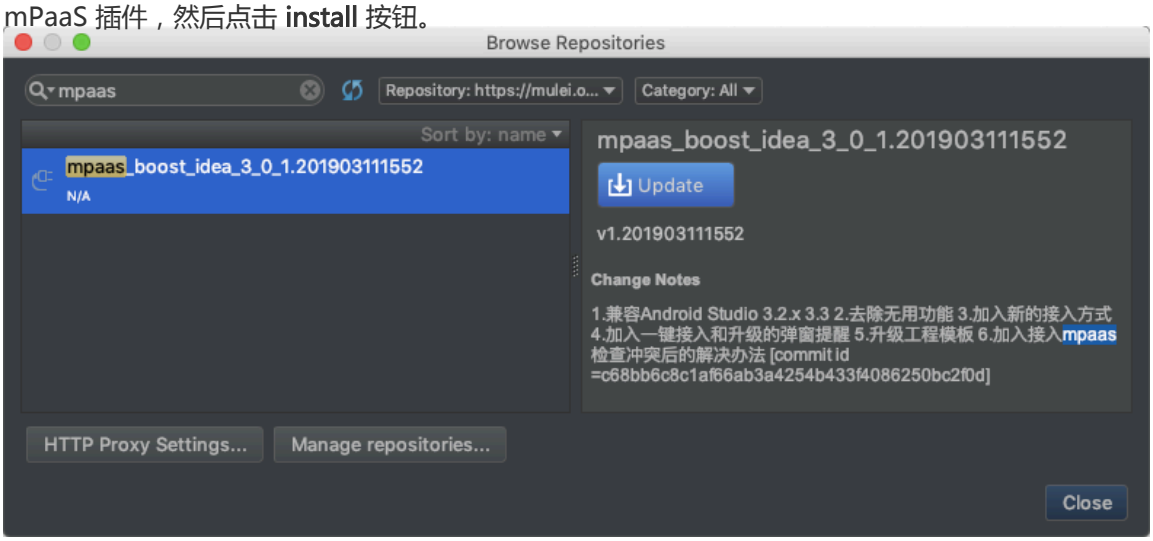


3. 点击下方 **Manage repositories** 按钮。

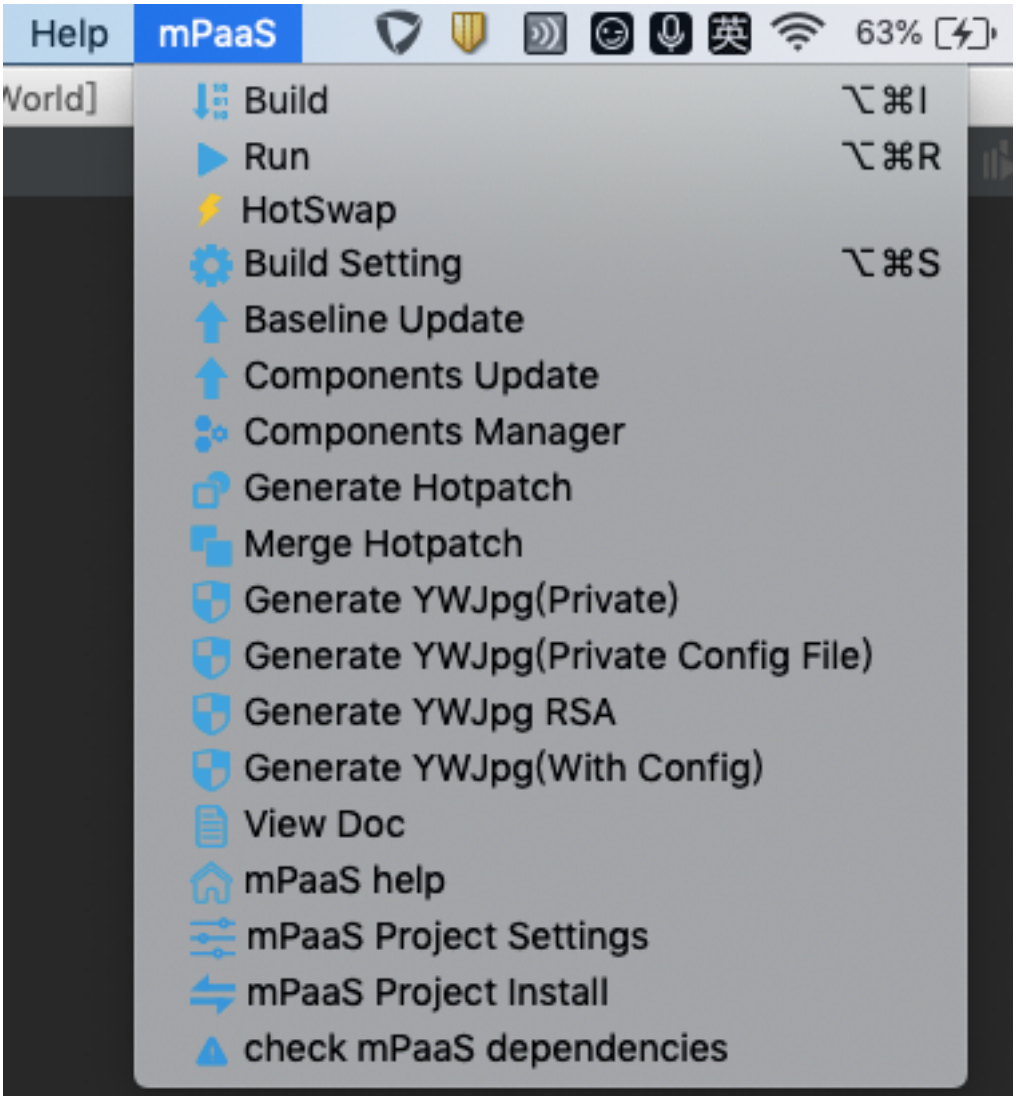




4. 点击 + 按钮。
5. 在弹出框 **Repository URL** 中填入 https://mulei.oss-cn-hangzhou.aliyuncs.com/updatePlugins.xml，然后点击 **OK** 按钮。
6. 点击 Custom Plugin Repositories 对话框 **OK** 按钮，回到 Browse repositories 对话框。
7. 在 Browse repositories 对话框左上方的搜索框中输入 mPaaS。等待搜索结束后，点击搜索到的 mPaaS 插件，然后点击 **install** 按钮。



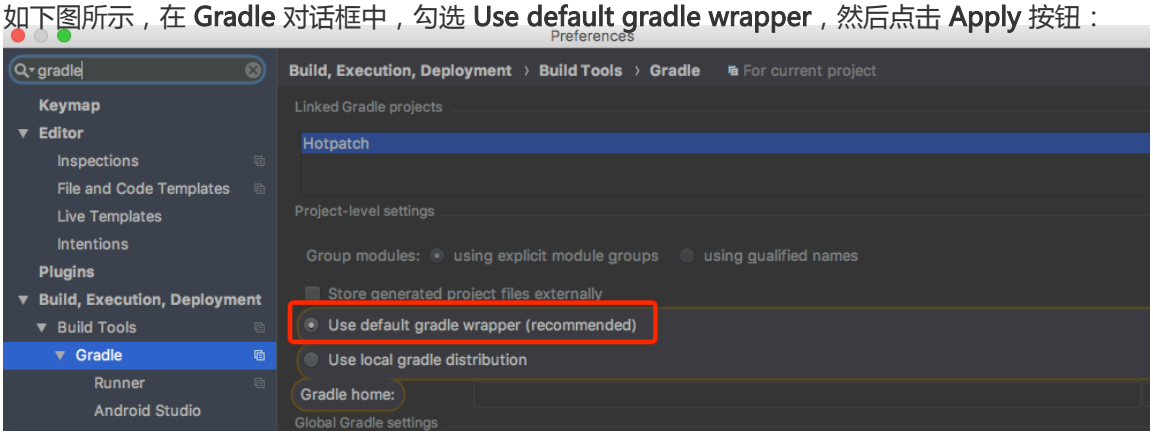
8. 等待插件安装结束，重启 Android Studio，您将看到 mPaaS 插件：



配置 Gradle 构建工具

您需要确保工程构建时使用 **Gradle Wrapper**：

1. 在 Android Studio 中打开任意一个 Android 工程。
2. 打开设置对话框。
3. 如下图所示，在 **Gradle** 对话框中，勾选 **Use default gradle wrapper**，然后点击 **Apply** 按钮：



2.2.2.4 Linux

Linux 操作系统版本较多，本文仅适用于 CentOS 和 Ubuntu 版本。

配置 Linux 开发环境的步骤包括：

- 配置 Java 8 环境
- 配置 Gradle 4.4 环境
- 安装 32 位兼容库
- 安装并配置 Android Studio 安装 Android Studio安装 Android SDK安装 mPaaS 插件配置 Gradle 构建工具

配置 Java 8 环境

mPaaS 框架只支持 **JDK 8+**：

1. 下载并安装 JDK 8。
2. 配置 JAVA_HOME 环境变量，并将 JAVA_HOME 下的 bin 路径添加到 PATH 环境变量中。
3. 正确配置后，在命令行执行 java -version 命令，您将看到 JDK 版本等信息：

```
d:\>java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
```

配置 Gradle 4.4 环境

mPaaS 框架只支持 **Gradle 4.4**：

1. 下载 Gradle 4.4.zip。
2. 解压 .zip 包，然后将解压路径配置为 GRADLE_HOME 环境变量，并将 GRADLE_HOME 下的 bin 路径添加到 PATH 环境变量中。
3. 正确配置后，在命令行执行 gradle -v 命令，您将看到 Gradle 版本等信息：

```
d:\>gradle -v

------------------------------------------------------------
Gradle 4.4
------------------------------------------------------------

Build time:   2017-12-06 09:05:06 UTC
Revision:     cf7821a6f79f8e2a598df21780e3ff7ce8db2b82

Groovy:       2.4.12
Ant:          Apache Ant(TM) version 1.9.9 compiled on February 2 2017
JVM:          1.8.0_121 (Oracle Corporation 25.121-b13)
OS:           Windows 7 6.1 amd64
```

安装 32 位兼容库

CentOS 6、CentOS 7、Ubuntu 等 Linux 发行版默认去除 ia32-lib。所有 64 位 Linux 系统都需安装 32 位兼容库。参照 android-sdk 中的安装方法:

Ubuntu:
sudo apt-get install zlib1g:i386

CentOS:
yum install libstdc++.i686

安装并配置 Android Studio

安装 Android Studio

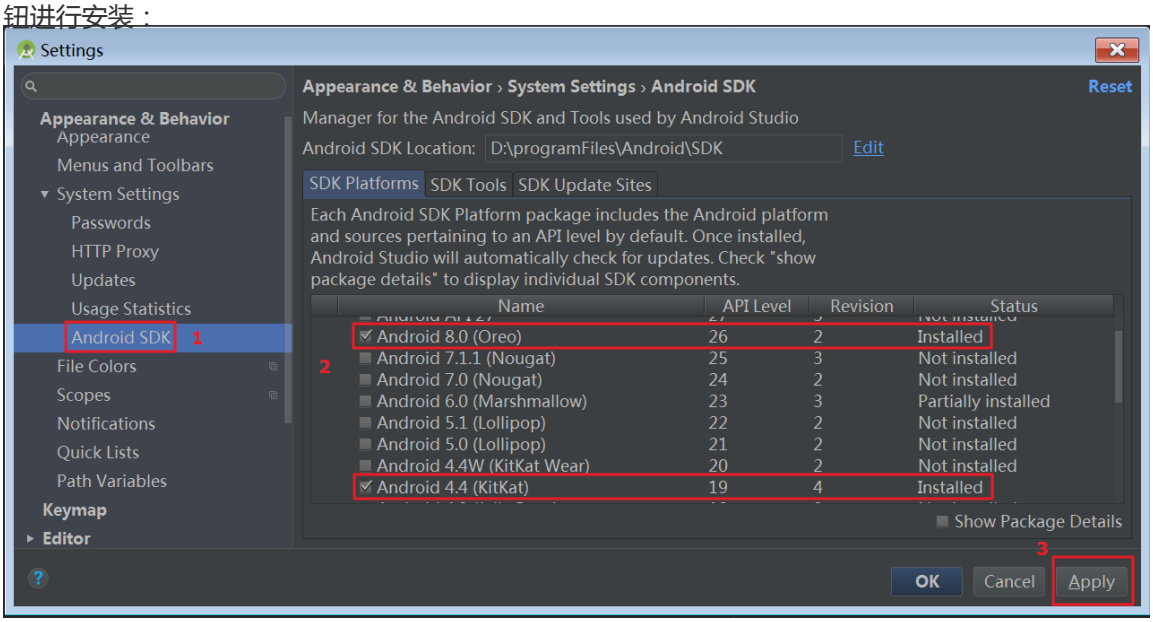
mPaaS 框架只支持 **3.0.x**、**3.1.x**、**3.2** 和 **3.3.x** 版本的 Android Studio：

- 下载地址：Android Studio 3.0.1；更多下载，请参见 官网
- 安装指南

安装 Android SDK

您需要安装 API Level 为 **19** 和 **26** 的 Android SDK：

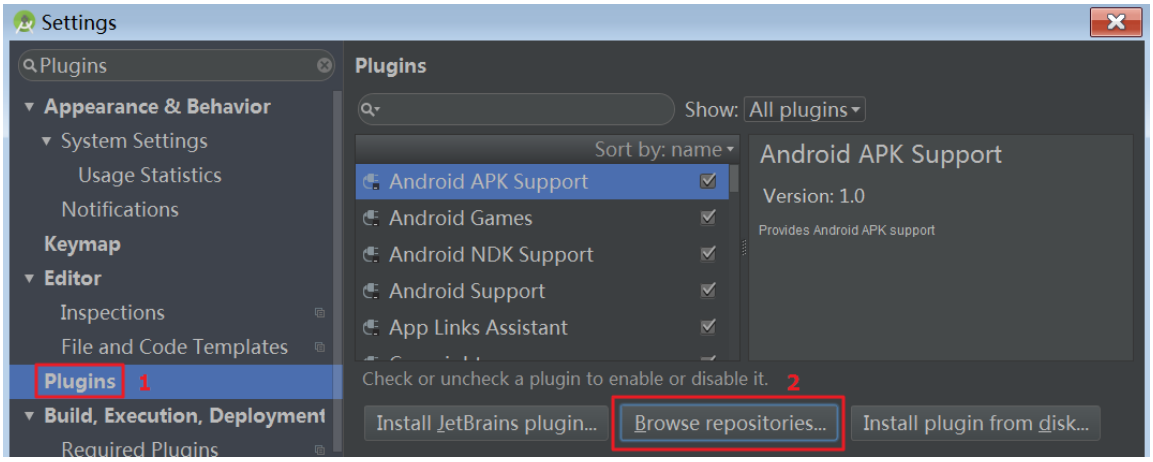
1. 在 Android Studio 中，通过 **File** > **Settings** 打开设置对话框。
2. 如下图所示，在 **Android SDK** 对话框中，勾选 API Level 为 **19** 和 **26** 的 SDK，然后点击 **Apply** 按钮进行安装：



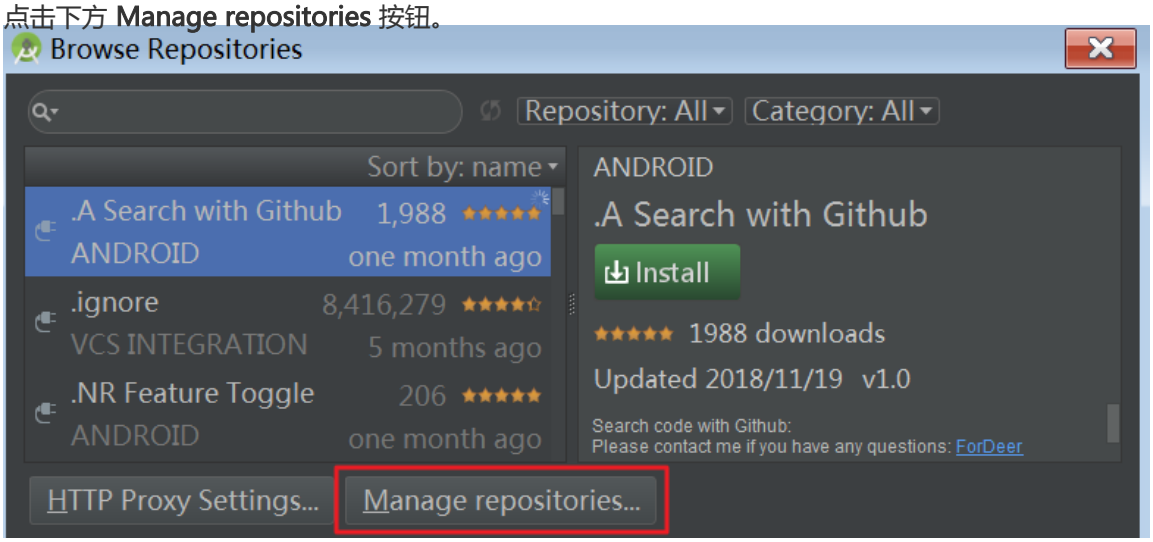
安装 mPaaS 插件

mPaaS 插件提供多种开发辅助功能，包括：新建 mPaaS 工程，添加、删除和升级 mPaaS 组件，构建工程等。安装步骤如下：

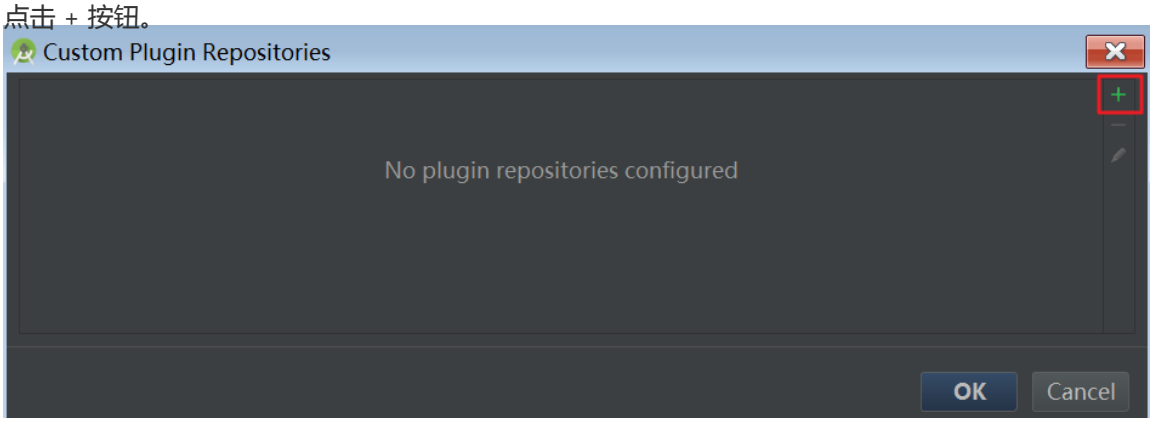
1. 打开 Android Studio 设置对话框。
2. 点击左侧 **Plugins**，然后点击下方 **Browse repositories** 按钮。



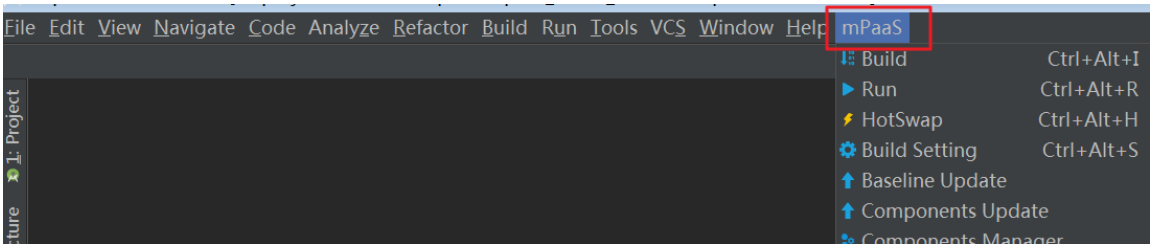
3. 点击下方 **Manage repositories** 按钮。



4. 点击 + 按钮。



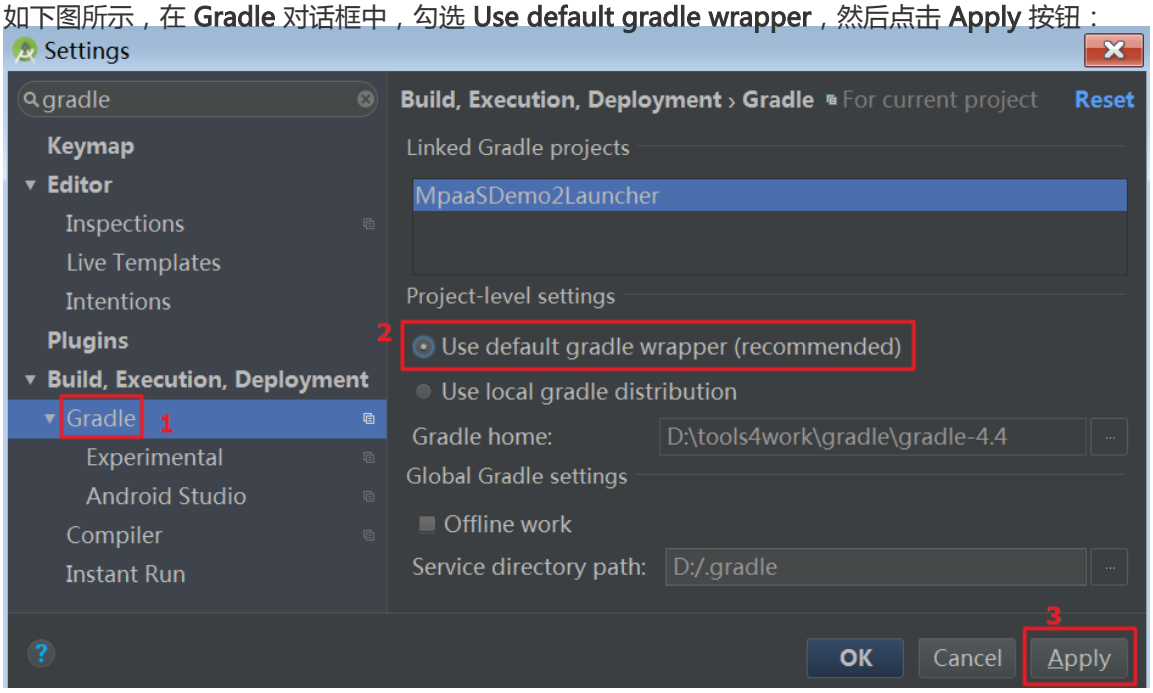
5. 在弹出框 **Repository URL** 中填入 https://mulei.oss-cn-hangzhou.aliyuncs.com/updatePlugins.xml，然后点击 **OK** 按钮。
6. 点击 Custom Plugin Repositories 对话框 **OK** 按钮，回到 Browse repositories 对话框。
7. 在 Browse repositories 对话框左上方的搜索框中输入 mPaaS。等待搜索结束后，点击搜索到的 mPaaS 插件，然后点击 **install** 按钮。
8. 等待插件安装结束，重启 Android Studio，您将看到 mPaaS 插件：



配置 Gradle 构建工具

您需要确保工程构建时使用 **Gradle Wrapper**：

1. 在 Android Studio 中打开任意一个 Android 工程。
2. 打开设置对话框。
3. 如下图所示，在 **Gradle** 对话框中，勾选 **Use default gradle wrapper**，然后点击 **Apply** 按钮：



2.2.3 在控制台创建应用

要使用 mPaaS，您首先需要在 mPaaS 控制台创建应用并下载配置文件。

前置条件

您需要确保拥有开发者账号。账号注册的更多信息，请参见 账号注册 。

> 如果您在其他平台（如蚂蚁金服开放平台）使用 mPaaS，请查看对应平台的账号说明。

创建 mPaaS 应用

登录 mPaaS 控制台。

> 如果您在其他平台（如蚂蚁金服开放平台）使用 mPaaS，请登录对应平台的 mPaaS 控制台。

选择租户和工作空间，然后点击 **确定** 按钮。

3. 点击 **创建mPaaS应用** 按钮： 

4. 完善应用信息：
    - 输入应用名称。示例：mPaaSDemo。
    - 点击 + 上传应用图标。您可以忽略此步骤，此时应用将使用默认图标。

点击 **确定** 按钮，完成应用创建。您可以看到应用列表，列表里包含刚才创建的应用：

下载配置文件

1. 在应用列表页，点击应用名称（如上一步中创建的应用 mPaaSDemo），您将看到以下页面：
    - 左上方展示应用名称，您可以在此切换应用。
    - 页面右侧展示 mPaaS 提供的组件服务列表。

2. 在左侧导航栏，点击 **代码管理** > **代码配置**。
3. 进入 **Android** 标签页，输入 **Package Name**（应用包名）。
4. 部分 mPaaS 组件接入时，需要 APK 签名信息。因此，您需要根据组件接入文档决定是否 **上传签名后的 APK**（应和 **发布版本** 的 APK 使用相同的签名文件）。
5. 点击 **下载配置** 按钮，下载应用配置文件到本地，为后续开发做准备。
    - 如果未上传 APK，您将得到 .config 配置文件，文件内容为 JSON 格式，示例如下：

```
{
"appId":"11111111111",
"appKey":"11111111111_ANDROID",
"base64Code":"xxxx",
"packageName":"com.mpaas.demo",
"rootPath":"mpaas/android/11111111111-default",
"workspaceId":"default",
"rpcGW":"https://cn-hangzhou-mgs-gw.cloud.alipay.com/mgw.htm",
"mpaasapi":"https://cn-hangzhou-component-gw.cloud.alipay.com/mgw.htm",
"pushGW":"cn-hangzhou-mps-link.cloud.alipay.com",
"pushPort":"443",
"logGW":"https://cn-hangzhou-mas-log.cloud.alipay.com",
"syncport":"443",
"syncserver":"cn-hangzhou-mss-link.cloud.alipay.com"
}
```

    - 如果上传了 APK，您将得到 .zip，压缩包中包含上述 .config 文件和名为 yw_1222.jpg 的 加密图片 。

2.2.4 客户端创建新应用

模块化是 mPaaS 框架 的核心设计理念。一个基于 mPaaS 框架开发的 App 包括：

- **一个或多个 Bundle 工程** ：一个 Bundle 即是一个业务独立的模块。
- **一个 Portal 工程** ：Portal 负责把所有的 Bundle 构建结果合并成一个可运行的 .apk 包。因此，您需要先构建各 Bundle，再构建 Portal。

本文将以 Windows 开发环境为例，引导您在本地创建一个全新 App，并编译打包，最终获得可运行的 .apk 包。

前置条件

您首先需要：

1. 配置开发环境
2. 在控制台创建应用

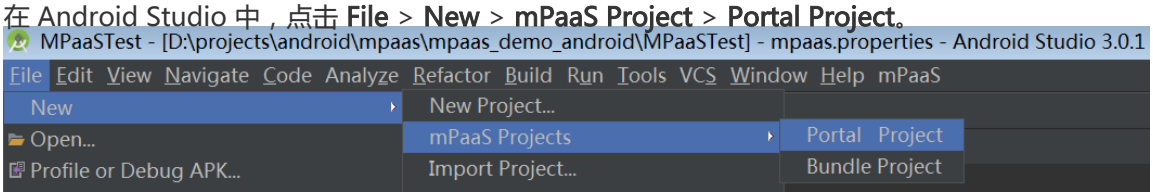
创建 Portal

为了创建新应用，您首先需要创建一个 Portal 工程。

Portal 一般不包含业务代码，仅仅用于将各 Bundle 合并成一个可运行的 .apk 包。因此在创建 Portal 时，默认会创建一个后缀名为 Launcher 的 Bundle 工程。

具体创建步骤如下：

1. 在 Android Studio 中，点击 **File** > **New** > **mPaaS Project** > **Portal Project**。



2. 填写 **Application name**（应用名，假定为 MPaaSDemo）、**Package name**（包名）等信息，然后点击 **Next** 按钮。
3. 在 **mPaaS Config Location** 中选择从控制台 **代码管理** > **代码配置** 中下载到的 .config 配置文件 。然后您将看到解析出来的 App Key、PRC 网关 等信息。点击 **Next** 按钮。
4. 选择 mPaaS SDK 版本，并勾选您需要的模块依赖。点击 **Next** 按钮。

   重要：

   - 请按需勾选模块依赖。具体依赖信息，请参考各组件的接入文档。
   - 您也可以只选择框架必选依赖。在完成应用创建之后，再参考各组件的接入文档，使用 mPaaS 插件 > Components Manager 功能添加所需依赖。

5. 填写 **Group Id**、**Artifact Id**、**Version** 等信息。

   为了快速接入，您可以先忽略更多高级配置。不过如有需要，请参见 Bundle 属性 。

6. 点击 **Finish** 按钮。稍等片刻，您将看到创建成功的 **Portal**（MPaaSDemoPortal）和 **Launcher**（MPaaSDemoLauncher）工程。
7. 在 **Launcher**（MPaaSDemoLauncher）工程中，点击 **mPaaS** > **Build**，构建 Launcher 工程。
8. 在 **Portal**（MPaaSDemoPortal）工程中，点击 **mPaaS** > **Build**，构建 Portal 工程。构建成功后会生成 .apk 包，且会出现安装应用弹出框，您可以将应用安装后进行测试。

创建新 Bundle

mPaaS 框架支持多 Bundle，您可以在 Android Studio 中，通过 **File** > **New** > **mPaaS Project** > **Bundle Project** 创建新 Bundle。创建过程与上文描述类似，不再赘述。

后续步骤

您可以参考各组件的接入文档，接入并使用 mPaaS 组件 。

相关链接

- mPaaS 框架介绍 ：介绍 Portal 和 Bundle 工程的 **代码结构**、**编译打包结果** 以及 **与原生工程的区别** 。

2.3 mPaaS 框架介绍

2.3.1 框架简介

OSGi（Open Service Gateway Initiative）是 Java 动态化模块化系统的一系列规范。mPaaS 框架基于 OSGi 技术把一个 App 划分成业务独立的 Bundle 工程，并对每个 Bundle 工程的生命周期和依赖加以管理。Portal 工程把所有的 Bundle 工程包合并成一个可运行的 .apk 包。

mPaaS 框架适合团队协同开发 App。该框架包含组件初始化、埋点等功能，方便您轻松接入 mPaaS 组件。

Bundle 工程

一个 Bundle 工程一般由一个名为 app 的主 module 和若干个子 module 组成。有关 Bundle 工程的介绍以及 Bundle 与传统 Android 工程的区别，请参见 Bundle 工程 。

Portal 工程

Portal 工程把所有的 Bundle 工程包合并成一个可运行的 .apk 包。有关 Portal 工程的介绍以及 Portal 与传统 Android 工程的区别，请参见 Portal 工程 。

工程依赖

一个 **基于 mPaaS 框架** 的 App 包括 **一个或多个 Bundle** 和 **一个 Portal**。

Bundle 编译打包结果

使用 mPaaS 插件编译打包后，一个 Bundle 会生成一个工程包（是一个 .jar 包）。更多信息，请参考 Bundle 工程 > Bundle 工程包 和 Bundle 工程 > Bundle 接口包 。

工程包会以 groupid:artifactid:version:classifier@type 的形式发布到指定仓库中。发布仓库地址在 Bundle 主 module 中的 build.gradle 中定义，示例如下：

```
uploadArchives {
repositories {
mavenLocal()
}
}
```

上述配置指定发布仓库为本地 Maven 仓库（mavenLocal）。如需修改本地 Maven 仓库地址（默认 ~/.m2）或增加发布仓库，请参见 配置发布仓库 。

添加 Bundle 依赖

您可以在 Portal 中添加 Bundle 依赖，也可以在 Bundle 中添加其他 Bundle 依赖。只需：

1. 在 Portal 或 Bundle 最外层的 build.gradle 中声明依赖仓库地址。依赖仓库应和上文 Bundle 发布仓库相对应。依赖仓库的配置方法，请参见 配置依赖仓库 。
2. 在 Portal 或 Bundle 主 module 的 build.gradle 中声明 dependencies 依赖。如添加 Bundle（quinox）依赖的示例如下：

```
dependencies {
bundle 'com.alipay.android.phone.mobilesdk:quinox-build:2.2.1.161221190158:nolog@jar'
manifest 'com.alipay.android.phone.mobilesdk:quinox-build:2.2.1.161221190158:AndroidManifest@xml'
}
```

相关链接

- OSGi 简介
- mPaaS 插件
- 配置依赖仓库
- 配置发布仓库

2.3.2 Bundle 工程

传统的原生工程由一个主模块或是一个主 module 和若干个子 module 组成，而一个 mPaaS Bundle 工程一般由一个名为 app 的主 module 和若干个子 module 组成。

例如，在支付宝中，一个 Bundle 一般由一个名为 app 的主 module 和以下三个子 module 组成：

- api：纯代码接口，interface 的定义。
- biz：interface 的实现。
- ui：activity，自定义 view 等。

> **重要**：至少有一个名为 api 的子 module。如果没有子 module， 就打不出 Bundle 的接口包，并且该 Bundle 不能被其他 Bundle 依赖。

通过阅读本文，您将从以下方面了解 Bundle 工程：

- Bundle 与传统工程区别
- Bundle 属性
- Bundle 接口包
- Bundle 工程包

Bundle 与传统工程区别

Bundle 本质上也是一个原生工程，只是在 **工程**、**主 Module**、**子 Module** 的 build.gradle 中多了 mPaaS 的 **Apply 插件**，具体差别体现在以下方面：

- 工程根目录
- 主 module 的
- 子 module 的

工程根目录 build.gradle

在工程根目录的 build.gradle 中，增加了对 mPaaS 插件的依赖：

> 重要：因功能迭代，插件版本可能会不断增加。

```
classpath 'com.alipay.android:android-gradle-plugin:2.1.3.3.1.2'
```

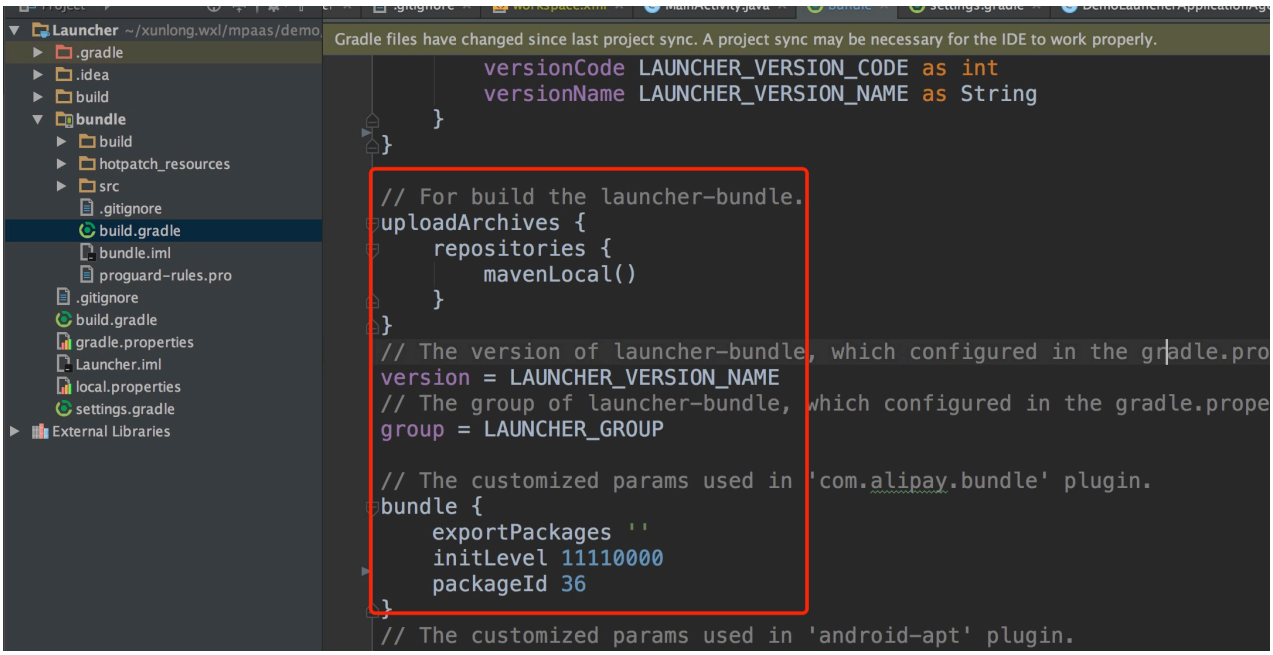
```
buildscript {
    repositories {
        mavenLocal()
        jcenter()
        // The Following repository is mPaaS address:
        maven {
            credentials {
                username "[redacted]"
                password "[redacted]"
            }
            url "http://mvn.cloud.alipay.com/nexus/content/repositories/r
        }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.3'
        // Following two one is mPaaS dependencies:
        classpath 'com.alipay.android:android-gradle-plugin:2.1.3.2.7'
        // Ref: https://bitbucket.org/hvisser/android-apt
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

主 module 的 build.gradle

在主 module 的 build.gradle 中，增加了 mPaaS Bundle **Apply 插件** 的声明，表示该工程为 Bundle 工程，Bundle 配置如下：

```
apply plugin: 'com.alipay.bundle'
```

主 module 的 build.gradle 中还增加了以下配置：



其中：

- version：该 Bundle 的 version。
- group：该 Bundle 的 groupid。
- exportPackages： 描述当前 Bundle 工程所有的类在哪些包名下面，包名可以取合集。非静态链接的 Bundle 必须填写 exportPackages，否则会出现类加载不到的问题。例如，如果所有的代码在 com.alipay.demo 和 com.alipay.bundle 下，那么在 exportPackages 中就可以写 com.alipay，也可以写 com.alipay.demo，com.alipay.bundle。包名不宜过长或过短。
- initLevel：框架启动时加载该 Bundle 的时机。时机范围在 0-100，数字越小表示越早加载。其中 11110000 时为使用时加载，即懒加载。
- packageId：描述当前 Bundle 的资源的 ID，供 aapt 打包时需要。由于是多 Bundle 架构，每个 Bundle 的 packageId 必须唯一，不可与其它 Bundle 的 packageId 重复。目前 mPaaS 已经使用的 packageId 如下：

Bundle	packageId
com.alipay.android.phone.thirdparty:androidsupportrecyclerview-build	28
com.alipay.android.phone.mobilesdk:framework-build	30
com.alipay.android.phone.rome:pushservice-build	35
com.alipay.android.phone.sync:syncservice-build	38
com.alipay.android.phone.wallet:nebulabiz-build	41
com.alipay.android.phone.mobilecommon:share-build	42
com.alipay.android.phone.wallet:nebulacore-build	66
com.alipay.android.mpaas:scan-build	72
com.alipay.android.phone.wallet:nebula-build	76
com.alipay.android.phone.securitycommon:aliupgrade-build	77

在 dependencies 中会添加对 mPaaS 的如下依赖：

```
dependencies {
compile project(":api")
apt 'com.alipay.android.tools:androidannotations:2.7.1@jar'
//mPaaS dependencies
provided 'com.alipay.android.phone.thirdparty:fastjson-api:1.1.45@jar'
provided 'com.alipay.android.phone.thirdparty:androidsupport-api:13.23@jar'
}
```

子 module 的 build.gradle

在子 module 的 build.gradle 中，增加了 mPaaS **Apply 插件** 的声明，表示该工程为 Bundle 的 子 module 工程，最终会打出该 Bundle 的接口包。

```
apply plugin: 'com.alipay.library'
```

在 dependencies 中会添加对 mPaaS 的如下依赖：

```
dependencies {
apt 'com.alipay.android.tools:androidannotations:2.7.1@jar'
//mPaaS dependencies
provided"com.alipay.android.phone.thirdparty:utdid-api:1.0.3@jar"
provided"com.alipay.android.phone.mobilesdk:framework-api:2.1.1@jar"
}
```

Bundle 属性

本框架的 Bundle 属性设计思路参考 OSGi 的 Bundle，但比 OSGi 的 Bundle 更简洁和轻巧。

以下表格列出 Bundle 属性及其解释：

属性	解释
Bundle-Name	Bundle Name，来自于由 build.gradle 文件中的 group 和 settings.gradle 中定义的 name。
Bundle-Version	Bundle Version，来自于 build.gradle 文件中的 version。
Init-Level	Bundle 的加载时机，来自于 build.gradle 文件中定义的 properties：init.level。
Package-Id	Bundle 资源的 packageid，来自于 build.gradle 文件中定义的 properties。
Contains-Dex	是否包含 dex，编译插件自动判断。
Contains-Res	是否包含资源，编译插件自动判断。
Native-Library	包含的 so 文件有哪些，编译插件自动判断。
Component-Name	来自于 AndroidManifest.xml 文件中定义的 Activity，Service，BroadcastReceiver，ContentProvider
exportPackages	该 Bundle 的所有的类所在的包名，参考主 module 的 build.gradle

Bundle 接口包

一个 Bundle 有可能包含多个 **子 Module**，如 biz， api， ui。在编译打包 Bundle 的时候，每个子 module

都会分别生成一个格式为 .jar接口包，这些接口包可以被其他 Bundle 使用。

同时在打包的时候，也会生成一个 Bundle 工程包，这个工程包包含所有的子 module，工程包可以被 Portal 工程使用，工程包在 Portal 中编译，最后生成 .apk 包。

- 由 Bundle 的 **子 module** 打出来的接口包，且 **对外提供接口类**，如 api module。
- 各 Bundle 工程直接通过 Bundle 的接口包互相依赖，需要在 bundle 的 build.gradle 中的 dependency 配置依赖 api 接口。例如，Bundle A 依赖 Bundle B 中的 bapi 子 module，那么在 Bundle A 相应的子 module 的 build.gradle 中的 dependency 配置对 bapi 的依赖。

```
provided"com.alipay.android.phone:bundleA:1.0.1:bapi@jar"
```

- 依赖中涉及的 groupId:artifactid:version:classifier 分别对应 Bundle 中声明的 group，name，version，子 module 的名字。
- Bundle 的 name 默认为主 module 的文件夹名，可以在 settings.gradle 中修改，如下代码所示，其中 app 为主 module 的工程名：

```
include ':api', ':xxxx-build'
project(':xxxx-build').projectDir = new File('app')
```

Bundle 工程包

- 由整个 Bundle 工程打出来的 .jar 包其实是一个 .apk 格式的文件，但是也以 .jar 结尾，如 framework-build.jar。
- 如果要在 Portal 中依赖 Bundle，则在 Portal 主 module 的 build.gradle 中的 dependency 中声明依赖 Bundle 包，如下所示：

```
dependencies {
bundle"com.alipay.android.phone.mobilesdk:framework-build:version@jar"
manifest"com.alipay.android.phone.mobilesdk:framework-build:version:AndroidManifest@xml"
}
```

- 对 Bundle 打包分两种：debug 包和 release 包，在 Portal 依赖 Bundle 的 debug 包时，需要在 debug 包中额外加上 :raw
    - 当 Portal 依赖 bundle 的 debug 包时，bundle"com.alipay.android.phone.mobilesdk:framework-build:version:raw@jar"
    - 当 Portal 依赖 bundle 的 release 包时，bundle"com.alipay.android.phone.mobilesdk:framework-build:version@jar"

注意：

打 Portal 包时，需要确定以下内容：

- 哪些 Bundle 是要打在 app 的主 dex 中。静态链接，有 ContentProvider 的 Bundle 必须放在静态链接中。
- 哪些动态加载。如果 app 不大，建议都在主 dex 中。

如果想把 Bundle 的代码打进主 dex 中，则需要在 Portal 的 slinks 文件中配置当前 Bundle。配置内容为：groupId-artifactId。如果以 -build 结尾，则去掉 -build。例如， groupId 为 com.mpaas.group ，artifactId 为 testBundle-build，则需要在 slinks 文件中添加一行：com.mpaas.group-testBundle。

静态链接：把 Bundle 的代码打进 apk 的 classes.dex 或者 classes1.dex， classes2.dex 等中，以便工程启动时就可以加载 Bundle 中的类。

动态加载：把 Bundle 的代码存放在 lib/armeabi 中。在使用到该 Bundle 的类时，框架自动创建一个 BundleClassLoader 加载，此种情况需要配置 Bundle 的 exportPackages。

2.3.3 Portal 工程

一个把各个 Bundle 合并为一个可以运行的 .apk 的工程称为 Portal。

Portal 与传统工程区别

Portal 和传统开发中的工程的区别体现在 build.gradle:

- 工程根目录
- 主 module 目录

工程根目录 build.gradle

如下图所示，classpath 多了一个 com.alipay.android:android-gradle-plugin:2.1.3.2.7 插件 ：

重要：因功能迭代，插件版本可能会不断增加。

```
buildscript {
    repositories {
        mavenLocal()
        jcenter()
        // The Following repository is mPaaS address:
        maven {
            credentials {
                username "[illegible]"
                password "[illegible]"
            }
            url "http://mvn.cloud.alipay.com/nexus/content/repositories/r
        }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.1.3'
        // Following two one is mPaaS dependencies:
        classpath 'com.alipay.android:android-gradle-plugin:2.1.3.2.7'
        // Ref: https://bitbucket.org/hvisser/android-apt
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

该插件中包含 Portal 插件，Portal 插件可以在打包过程中把各 Bundle 合并。

- 合并 Bundle 的 .jar
- 合并 Bundle 的 AndroidManifest

主 module 目录 build.gradle

多了 mPaaS **Apply Portal 插件** 的声明，表示该工程为 Portal 工程。Portal 配置如下：

```
apply plugin: 'com.alipay.portal'
```

同时，在 dependencies 中添加相应的对 Bundle 的依赖。dependencies中的语句是 bundle 和 manifest 的声明，用来表示 Portal 依赖了哪些 Bundle 或 manifest：

```
dependencies {
    compile 'com.android.support:appcompat-v7:25.0.0'
    compile(name: 'SecurityGuardSDK-external-release-5.1.38', ext: 'aar')
    // launcher-bundle
    bundle "com.mpaas.demo.launcher:bundle:1.1-SNAPSHOT:raw@jar"
    manifest "com.mpaas.demo.launcher:bundle:1.1-SNAPSHOT:AndroidManifest@xml"
    //adapter-bundle
    bundle "com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.1701092045@jar"
    manifest "com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.1701092045:AndroidManifest@xml"

    // basic-bundle -- the component of the basic framework
    bundle "com.alipay.android.phone.mobilesdk:common-build:1.3.0@jar"
    manifest "com.alipay.android.phone.mobilesdk:common-build:1.3.0:AndroidManifest@xml"
    // quinox -- the component of the basic framework
    bundle "com.alipay.android.phone.mobilesdk:quinox-build:2.2.0.161028154501:nolog@jar"
    manifest "com.alipay.android.phone.mobilesdk:quinox-build:2.2.0.161028154501:AndroidManifest@xml"
    // framework -- the component of the basic framework
    bundle "com.alipay.android.phone.mobilesdk:framework-build:2.2.0.161028154723:nolog@jar"
    manifest "com.alipay.android.phone.mobilesdk:framework-build:2.2.0.161028154723:AndroidManifest@xml"
    // android-support-wrapper-bundle
    bundle "com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.161114144707:nolog@jar"
    manifest "com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.161114144707:AndroidManifest@xml"
    // rpc-bundle -- the component of the basic framework
```

注意:

- 通常 Portal 下面不写代码。
- 以下几种在 Bundle 工程中使用的资源（style/drawable/string等），必须放在 Portal 工程中，否则会导致编译/运行时找不到资源：
    - AndroidManifest.xml 中使用的资源。
    - 传递给 NotificaionManager 使用的资源。
    - 通过 getResources().getIdentifier() 方法使用的资源。
    - 引用的第三方 aar 包中如有以上三种情况，也需要解压 aar，将对应的资源复制一份放到 Portal 工程中。

2.3.4 切换工作空间（Workspace）

mPaaS 支持多工作空间（Workspace），假如已有基于 mPaaS 框架开发的 App，本文引导您将之切换到其他工作空间。

前置条件

已有 **基于 mPaaS 框架** 开发的 App。更多信息，请参见 基于 mPaaS 框架 > 快速开始 。

共有云

在公有云环境中，切换工作空间的步骤如下：

确保工程根目录 build.gradle 文件中，有如下依赖：

提示：因功能迭代，以下依赖的版本可能会不断增大。

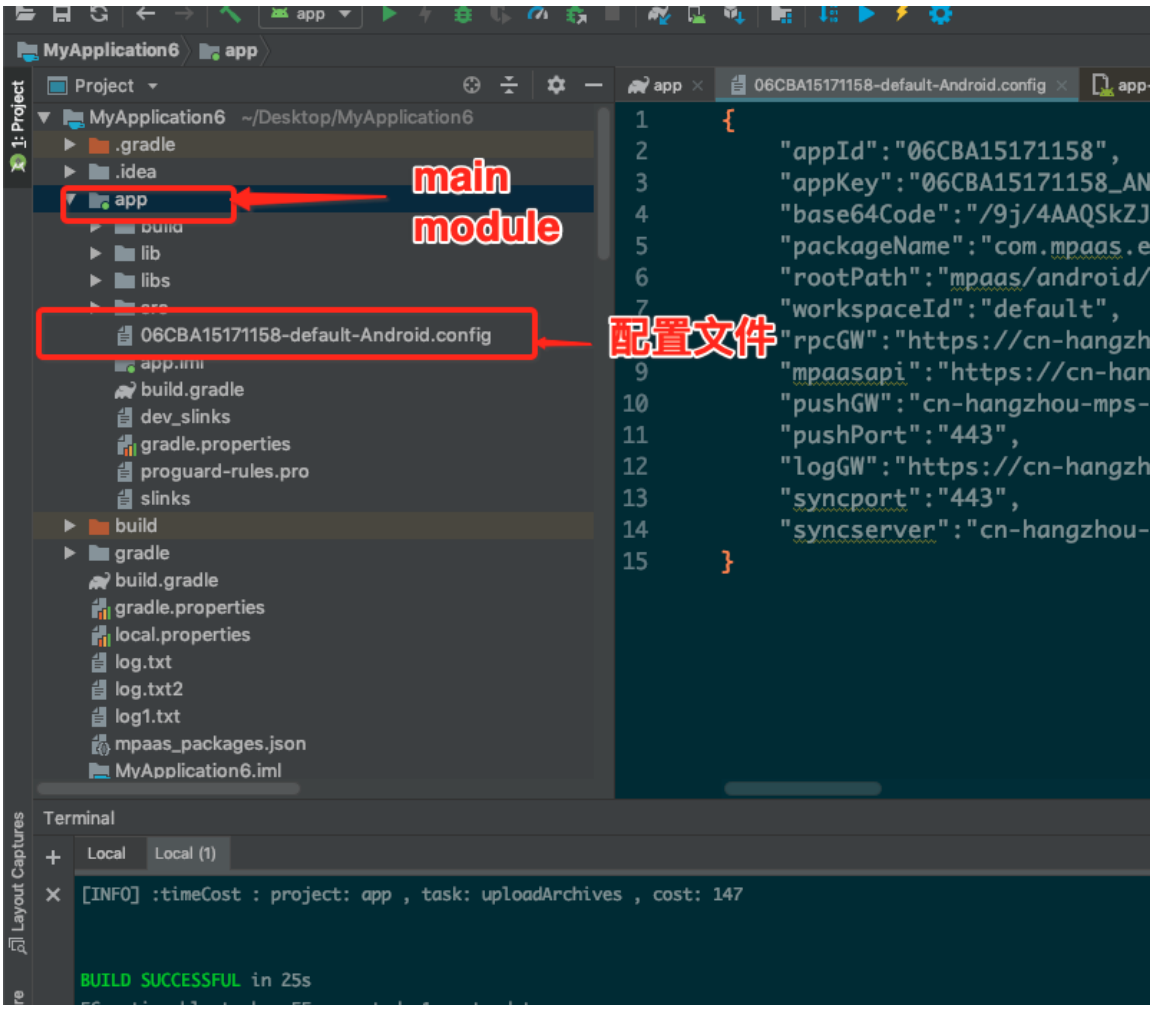
```
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.6.1'
// 版本号必须大于 2.1.0
classpath 'com.android.boost.easyconfig:easyconfig:2.1.0'
```

确保主工程（android main module） 的 build.gradle 中有如下配置（请注意顺序）：

```
apply plugin: 'com.alipay.portal'
//位于 com.alipay.portal 之后即可
apply plugin: 'com.alipay.apollo.baseline.update'
```

从控制台下载对应工作空间（Workspace）的 .config 配置文件。更多信息，请参考 在控制台创建应用 > 下载配置文件 。

将下载到的 .config 配置文件 添加到主工程（android main module）路径下。注意：**仅保留对应工作空间的配置文件即可**。如图所示：



私有云

在私有云环境中，切换工作空间的步骤如下：

确保工程根目录 build.gradle 文件中，有如下依赖：

提示：因功能迭代，以下依赖的版本可能会不断增大。

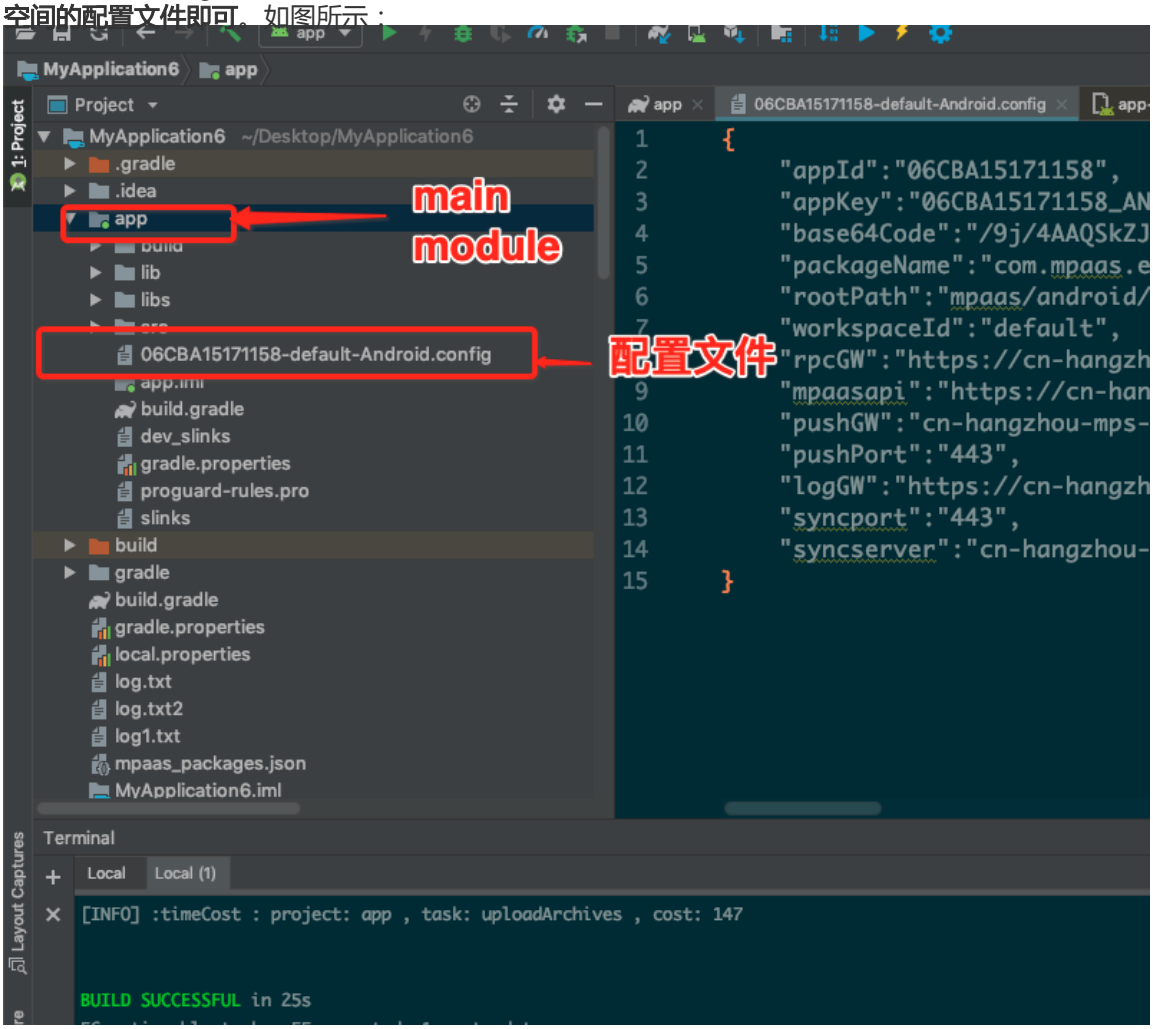
```
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.6.1'
// 版本号必须大于 2.1.0
classpath 'com.android.boost.easyconfig:easyconfig:2.1.0'
```

确保主工程（android main module）的 build.gradle 中有如下配置（请注意顺序）：

```
apply plugin: 'com.alipay.portal'
//位于 com.alipay.portal 之后即可
apply plugin: 'com.alipay.apollo.baseline.update'
```

从控制台下载对应工作空间（Workspace）的 .config 配置文件。更多信息，请参考 在控制台创建应用 > 下载配置文件 。

将下载到的 .config 配置文件 添加到主工程（android main module）路径下。注意：**仅保留对应工作空间的配置文件即可**。如图所示：



使用 mPaaS 插件生成 yw_1222.jpg 加密图片。更多信息，请参见 加密图片（私有云）

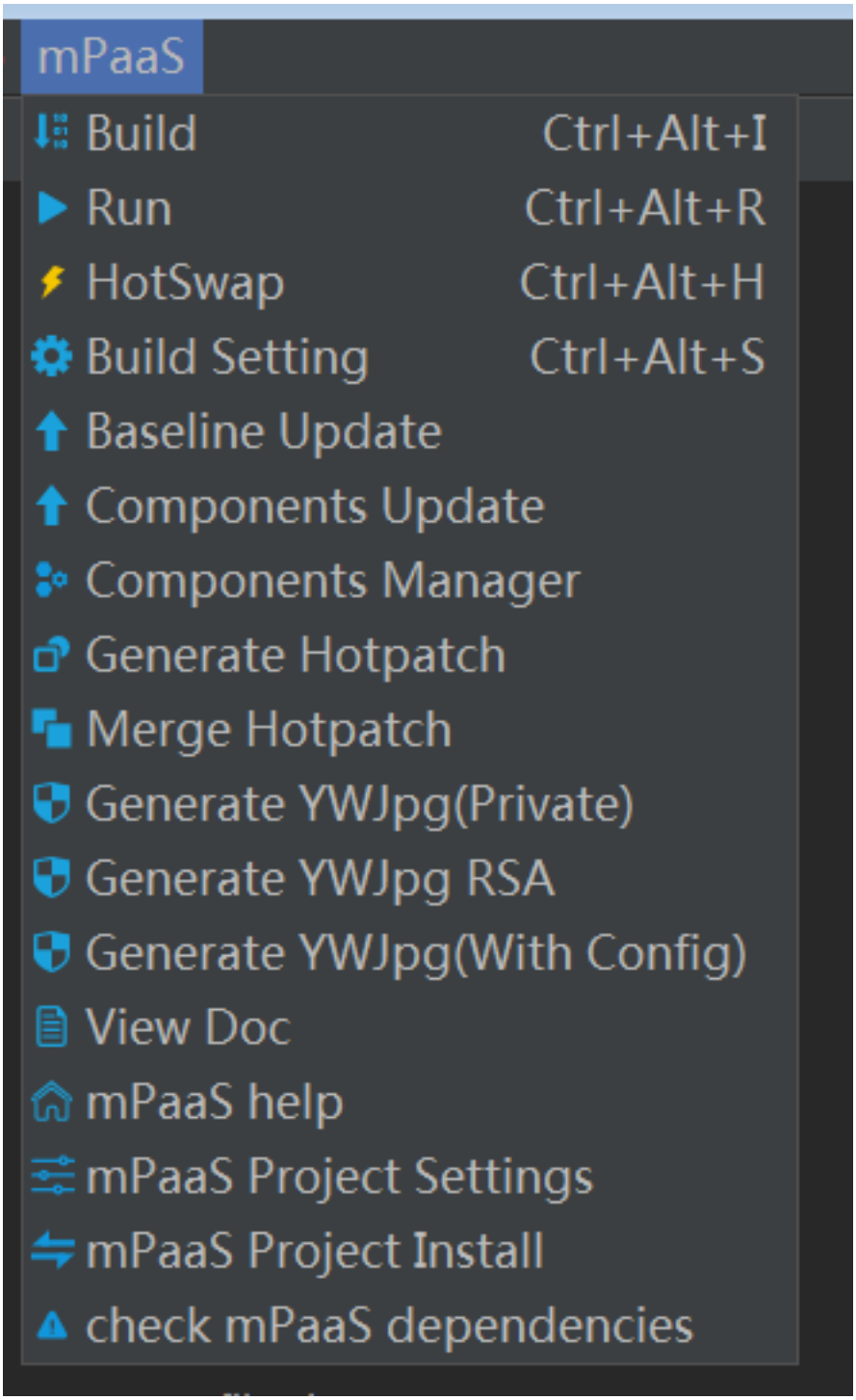
注意事项

easyconfig 工作原理

1. 能够修改AndroidManifest workspace 相关的 meta 属性。
2. 修改 assets 下 mpaas.properties 文件。
3. 如果 mPaaS 工程配置文件中包涵 base64 属性且属性不为空，会生成无线保镖加密图片 yw_1222.jpg 。

2.4 使用 mPaaS 插件

2.4.1 功能介绍



如图，mPaaS 插件提供多种开发辅助功能，包括：

功能	mPaaS 插件
编译打包	Build、Build Settings
管理组件依赖	Components Manager、Components Update、Baseline Update
迁移原生工程至 mPaaS 框架	mPaaS Project Install
热部署	HotSwap
加密图片	Generate YWJpg

热修复包	Generate Hotpatch、Merge Hotpatch

2.4.2 编译打包

本文将介绍与编译打包相关的 mPaaS 插件功能：

- Build ：打包。
- Build Settings ：打包配置。

Build

Build 执行 gradle 命令，用于编译打包。更多信息，请参考 Build Settings > Custom Command。

Build Settings

Build Settings 用于自定义打包参数。

Custom Command

当勾选了 **打debug包** 时，Build 默认执行命令 gradle clean buildDebug；否则默认执行命令 gradle clean buildRelease。
您可以在此自定义打包命令。此时，是否 **打debug包** 勾选框将失效。

Custom Params

默认为 --stacktrace --info -Plog=true。更多信息，请参考 Gradle Command-Line Interface。

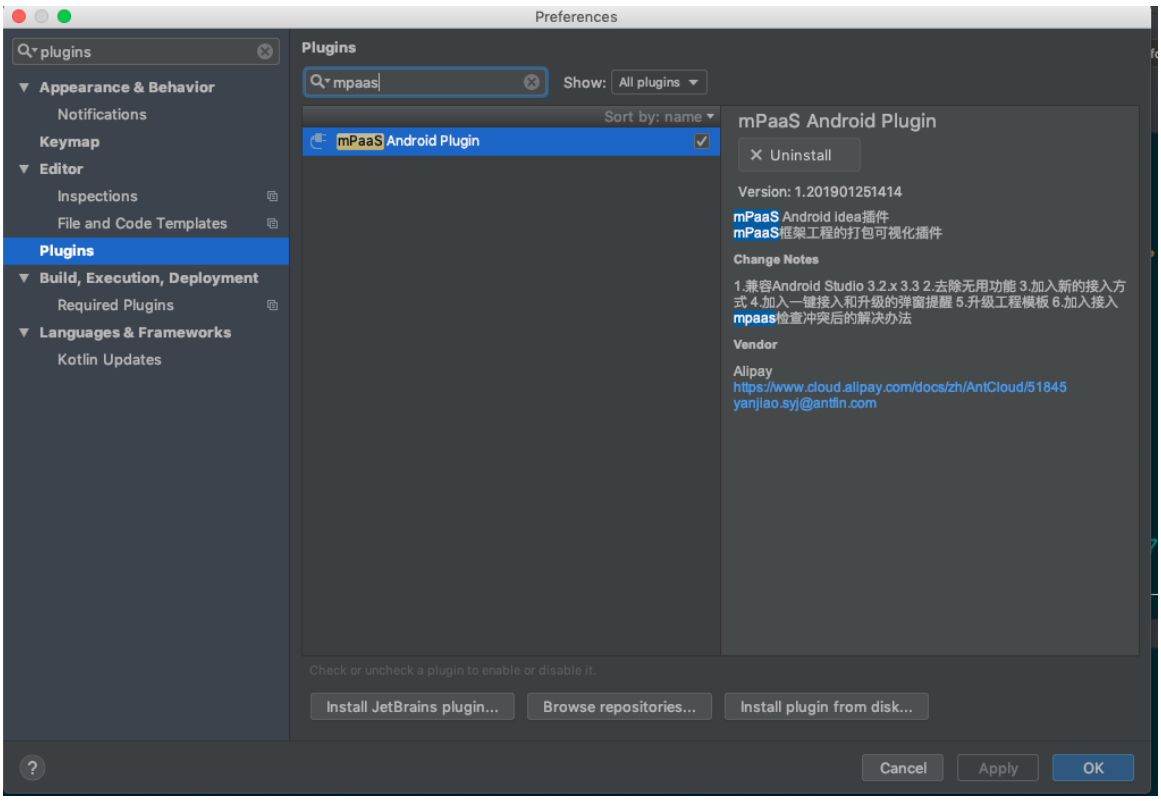
2.4.3 管理组件依赖

- 前置条件
- 增删组件依赖
- 升级 mPaaS SDK 版本
- 升级组件版本

前置条件

本文仅适用于版本号不低于 1.201901251414 的 mPaaS 插件：

在 **Android Studio** > **Settings** > **Plugins** 中查看 mPaaS Android Studio 插件版本号：



关于最新版本插件的安装方法，请参考 配置开发环境 。

增删组件依赖

为了使用 mPaaS 组件 ，您首先需要分别在 Portal 和 Bundle 工程添加该组件的依赖。增删组件依赖的步骤如下：

1. 在 Android Studio 中打开工程。
2. 点击 **mPaaS > Components Manager**，您将看到组件列表。

3. 根据组件状态，进行增删操作：
   - 若组件状态为 **Installed**，则说明已有该组件依赖。

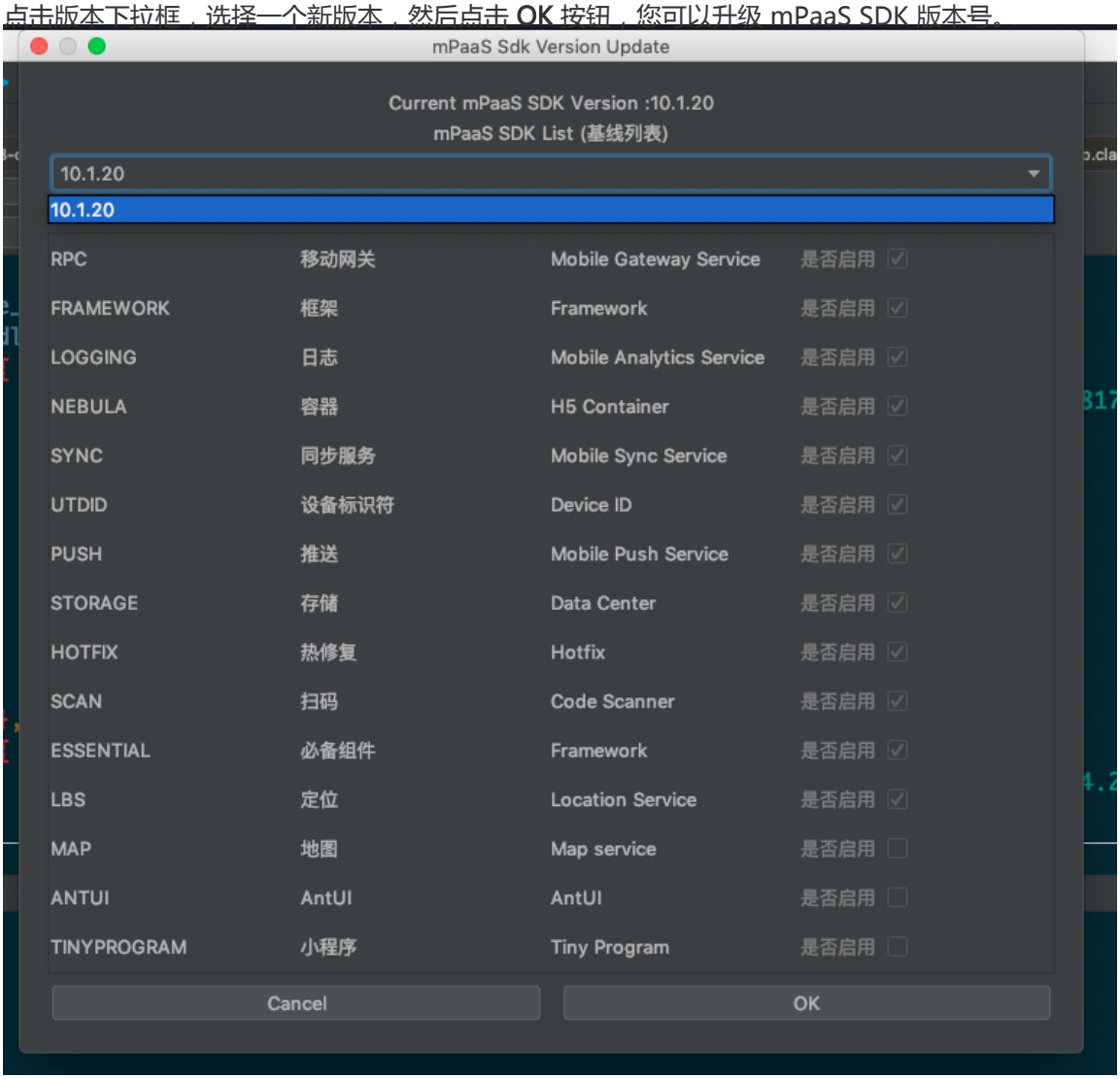
     提示：点击（Installed）状态按钮，状态会变为 **No Installed**，则该组件依赖删除成功。删除操作请小心进行。

   - 否则说明没有该组件依赖。点击（No Installed）状态按钮，状态会变为 **Installed**，则该

组件依赖添加成功。

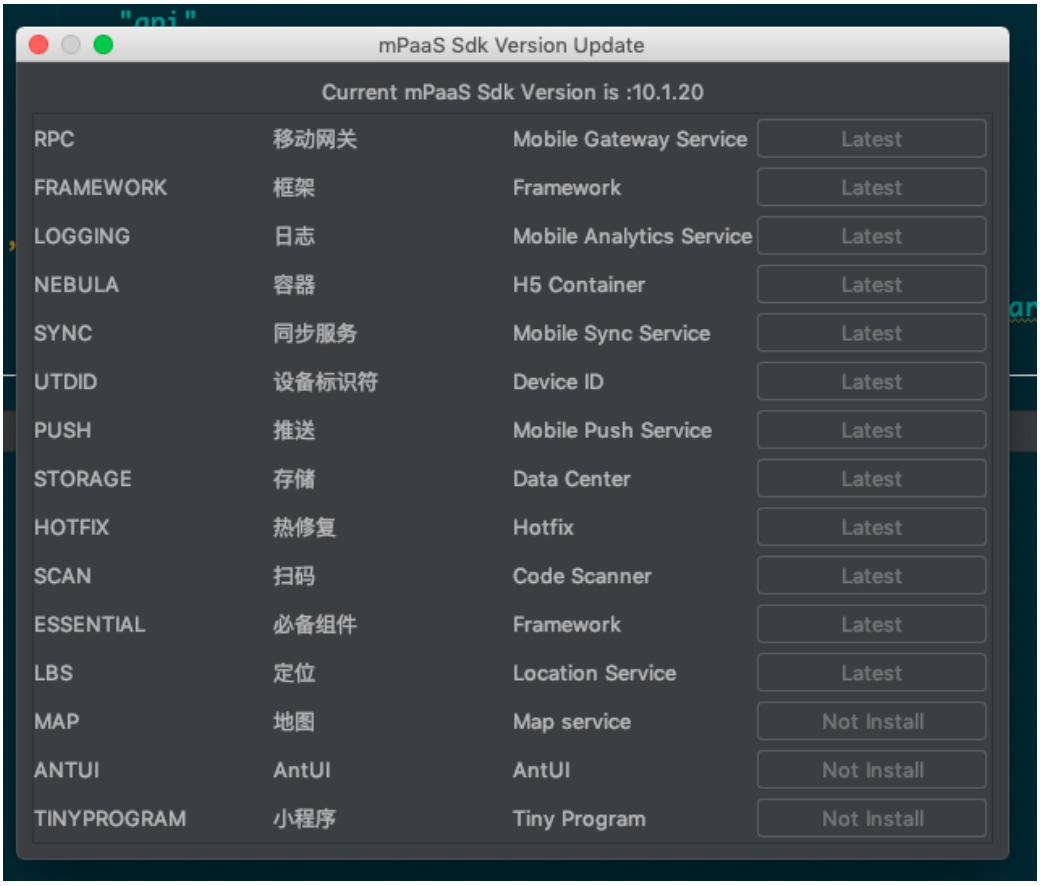
升级 mPaaS SDK 版本

1. 在 Android Studio 中打开工程。
2. 点击 **mPaaS** > **Baseline Update**，您将看到当前 SDK 版本信息。
3. 点击版本下拉框，选择一个新版本，然后点击 **OK** 按钮，您可以升级 mPaaS SDK 版本号。



升级组件版本

1. 在 Android Studio 中打开工程。
2. 点击 **mPaaS** > **Components Update**，您将看到组件列表。
3. 查看组件状态，进行升级操作：
   - 若为 **Lastest**，则说明该组件无需升级。
   - 否则说明该组件有新版本。您可以点击状态按钮，升级该组件。



2.4.4 迁移原生工程至 mPaaS 框架

如果已有基于 Android 原生框架的应用，您可以使用 mPaaS 插件 mPaaS Project Install 功能将之迁移至 mPaaS 框架。

配置本地开发环境

参考 文档，配置本地开发环境。包括安装特定版本 Android Studio 和 安装 mPaaS 插件等。

在控制台创建应用

参考 文档 ，在控制台创建应用，并下载 .config 配置文件。

使用 mPaaS 插件迁移工程

1. 打开 Android Studio。
2. 点击 **mPaaS** > **mPaaS Project Install**。
3. 在弹出框中，选择 **原生工程路径** 和 在控制台下载到的 .config **配置文件路径**。
4. 点击 **OK** 按钮，等待迁移完成。若转换失败，一般是因为已有工程中的依赖与 mPaaS SDK 存在冲突，具体信息请参见日志文件 conflict.json。

2.4.5 热部署

mPaaS 插件的热部署 （**HotSwap**） 功能类似 Android 原生的 Instance Run。热部署 Bundle，仅支持 debug 模式，对应命令 gradle clean hotSwapDebug。

注意：要使用热部署， 必须保证 Portal 的依赖中有如下内容：

```
devbundle"com.alipay.Android.phone.devtool:hotswap:2.1.0.161103185423@jar"
devmanifest"com.alipay.Android.phone.devtool:hotswap:2.1.0.161103185423:AndroidManifest@xml"
```

2.4.6 加密图片

为了安全，mPaaS 某些组件（如热修复、消息推送）访问网络时需要对内容进行加密。具有特殊名称 yw_1222.jpg 的图片为加解密提供密钥信息。mPaaS 组件自动使用该图片进行加解密，您无需额外操作。

本文引导您生成并使用加密图片 yw_1222.jpg。

准备

加密图片与 APK 的签名文件有绑定关系。因此，您需要准备好签名之后的 .apk 文件（应和 **发布版本** 的 APK 使用相同的签名文件）；而且生成的加密图片只能用于该 APK 工程中。

生成

公有云

您可以从 **控制台** 下载加密图片。

1. 登录 mPaaS 控制台。

   如果您在其他平台（如蚂蚁金服开放平台）使用 mPaaS，请登录对应平台的 mPaaS 控制台。

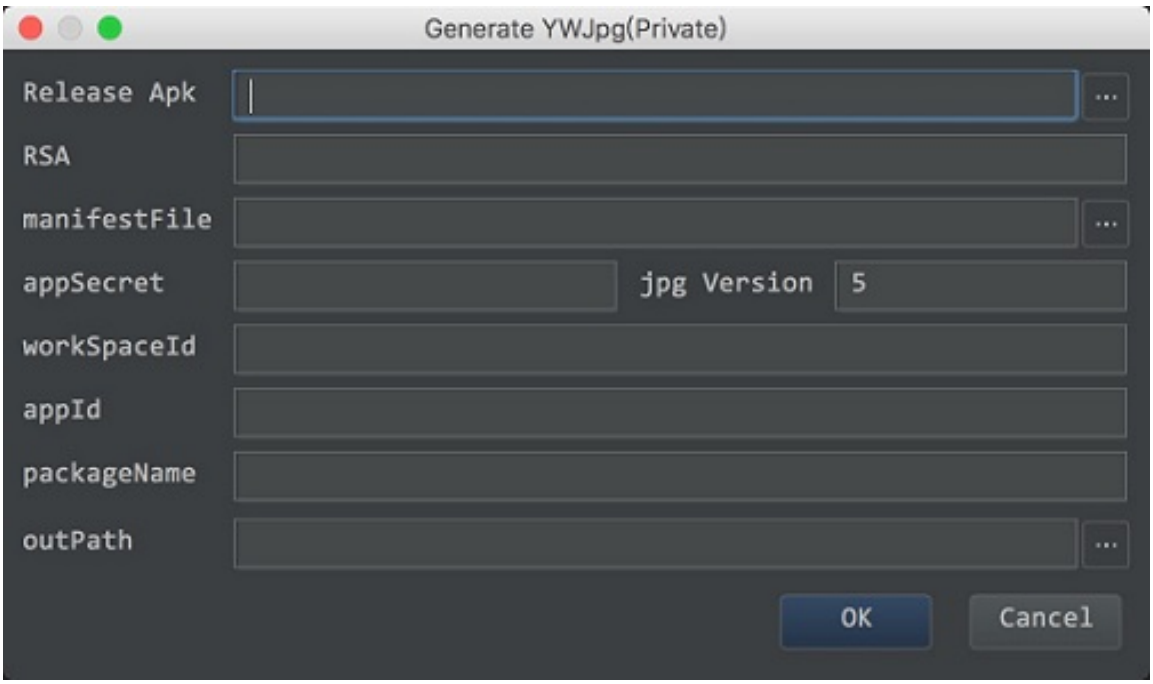
2. 选择应用后，点击左侧导航栏 **代码管理** > **代码配置**。
3. 上传 **签名后** 的 APK。
4. 点击 **下载配置** 按钮，下载 .zip 文件到本地。解压后，您将获得 yw_1222.jpg 加密图片。

更多信息，请参考 下载配置文件。

私有云

您可以通过 **mPaaS 插件** 生成加密图片。

1. 在 Android Studio 中，点击 **mPaaS** > **Generate YWjpg(Private)**。您将看到如下对话框：



2. 点击 **Release Apk** 最右侧的 ...， 选择 Portal 工程签名之后的 apk 文件， **RSA** 会自动填充。
3. 点击 **manifestFile** 最右侧的 ...， 选择 Portal 工程的 AndroidManifest.xml 文件， **workSpaceId**、**appId** 和 **packageName** 会自动填充。如果没有，可以根据工程的 .config 文件中的配置填写到对应输入框中。
4. 填写 **appsecret** 。

   注意：作为服务端管理人员，您可以从控制台中查询 **appid** 所对应的 **appsecret**。

5. 在 **jpg Version** 栏，填写对应的无线保镖图片版本号。

   查看 Portal 工程主 module 下 build.gradle 文件中 securityguard 版本，低于 5.4 的填 **4**（例如基线中给出的 securityguard-build:5.1.38.180402194514），其余填 **5** 。

6. 点击 **outPath** 最右侧的 ...， 选择无线保镖图片 yw_1222.jpg 的输出路径，即加密图片生成的本地路径。
7. 点击 **OK** 生成加密图片。

使用

加密图片的使用步骤如下：

1. 将加密图片 yw_1222.jpg 存放到 Portal 工程的 res/drawable 文件夹中。
2. 如使用 ProGuard，需避免加密图片被混淆。
   - 检查 build.gradle 中是否配置了如下内容：

```
minifyEnabled true
shrinkResources true
```

- 若有如上配置，为了避免加密图片被混淆，需要在 res/raw 下创建 keep.xml 文件。文件内容如下：

```
<?xml version="1.0"encoding="utf-8"?>
<resources xmlns:tools="http://schemas.android.com/tools"
tools:keep="@drawable/yw_1222*"/><!--tools:discard="@layout/unused2"-->
```

2.4.7 热修复包功能

通过 mPaaS 插件，基于不同的场景，生成热修复包或合并热修复包。

- 生成热修复包 ：根据不同的 mPaaS 集成方式，使用 **Generate Hotpatch** 功能生成热修复包。
- 合并热修复包 ：一个 Android App 版本最多只能有一个热修复包在运行。如果客户端某版本有两个 bug，那需要先在本地使用 **Merge Hotpatch** 功能将修正两个 bug 的热修复包合成一个热修复包。

前置条件

使用热修复能力前，首先要让 App 具备热修复的能力。具体内容，查看 热修复管理：添加 Android SDK 。

生成热修复包

使用 mPaaS 插件的 **Generate Hotpatch**，通过以下步骤生成热修复包：

1. 针对不同的 mPaaS 集成方式，选择对应的包，通过 mPaaS 插件的 **Generate Hotpatch** 生成热修复包：
    - **原生工程**：准备以下不同类型的 .apk 包，根据代码的不同，生成热修复包：
        - 有 bug 的线上 .apk 包。
        - 修复后的 .apk 包。

          提示：在 **New bundle** 栏，填写修复后的 apk 包的本地地址。在 **Old bundle** 栏， 填写有 bug 的 apk 包的本地地址。

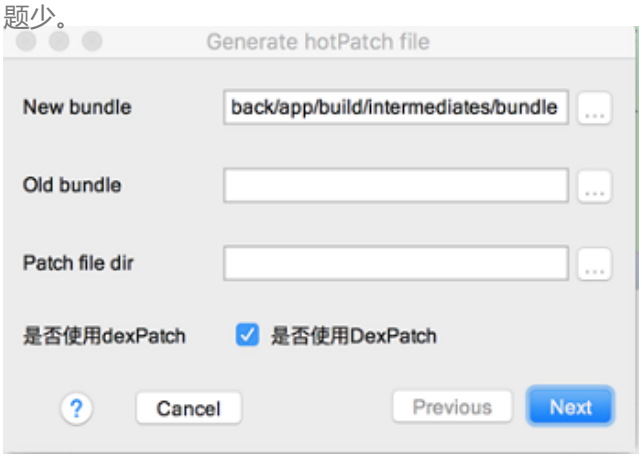
    - **mPaaS 框架**：准备以下类型的 bundle 包:
        - 正在使用的有 bug 的 bundle 包。
        - 修复后的 bundle 包。

          提示：bundle 的输出路径为 bundle 的主 module 目录下的 build/intermediates/bundle/xxxx-raw.jar。如果是 release 包则没有 -raw。其中：

            - **New bundle**：选择修复 bug 的包路径。
            - **Old bundle**：选择有 bug 的包路径。
            - **Patch file dir**：输出的 patch 包路径。
            - **是否使用 dexPatch**：选择是否使用 dexPatch 热修复方式。mPaaS 插件的 **Generate Hotpatch** 功能支持 Andfix 和 dexPatch 两种热修复方式。不论使用哪种热修复方案，在发版本

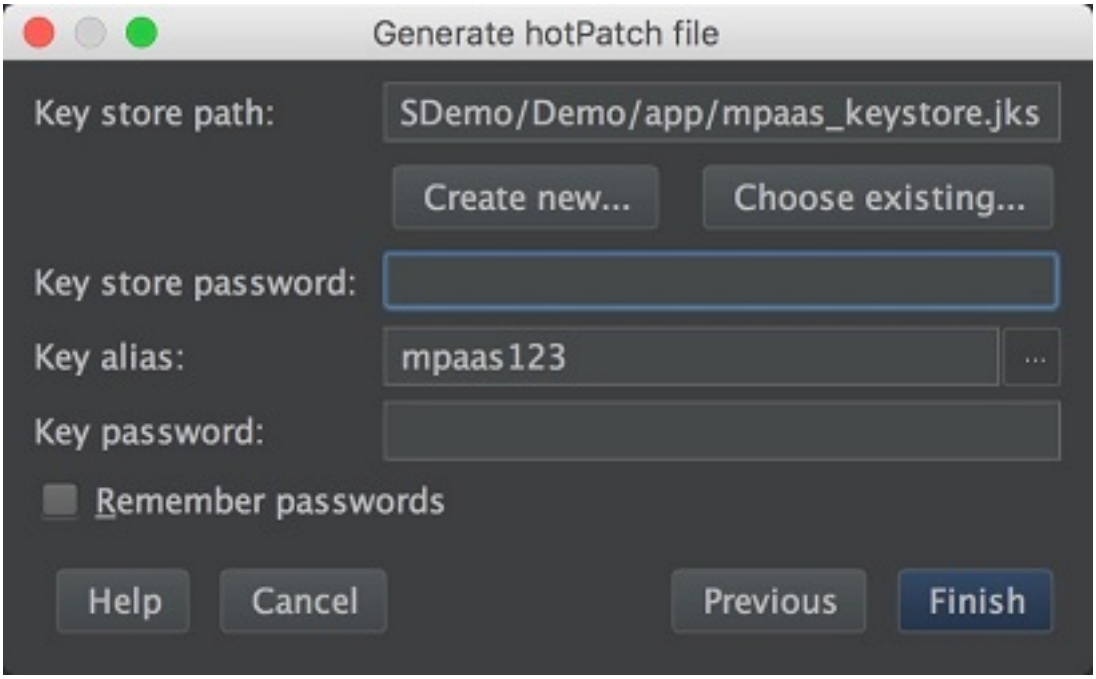
前都需要进行验证，查看热修复包是否能生效：

- 如果不勾选该复选框，那么会生成 Andfix 热修复包，其优点在于补丁立即生效，不需要重启应用即刻生效，缺点在于有机型问题，修复的场景限制较多。
- 如果勾选该复选框，那么会生成 dexPatch 热修复包，dexPatch 不能立即生效，需要杀掉进程后才能生效，但优点在于修复的场景比 Andfix 多，且机型适配问题少。



输入签名信息生成热修复包。

注意：生成热修复包所需要的签名文件必须和运行的 .apk 的签名文件保持一致，并且签名文件和生成图片选择的 .apk 的签名文件也要保持一致。生成的图片需要放在 Portal 工程的 res/drawable 文件夹下面，命名为 yw_1222.jpg。

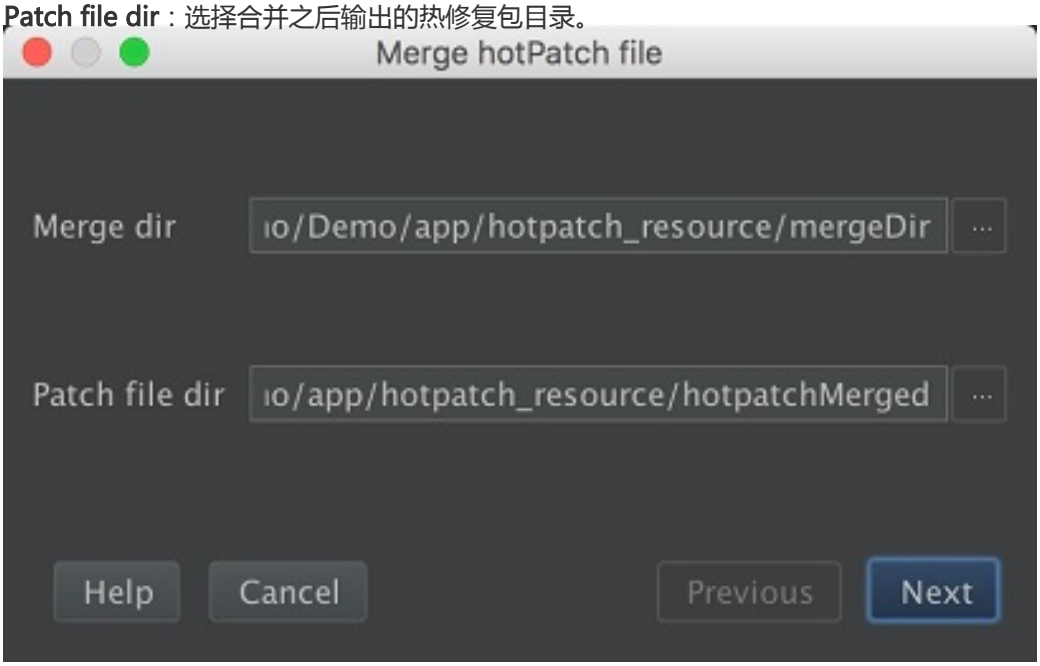


合并热修复包

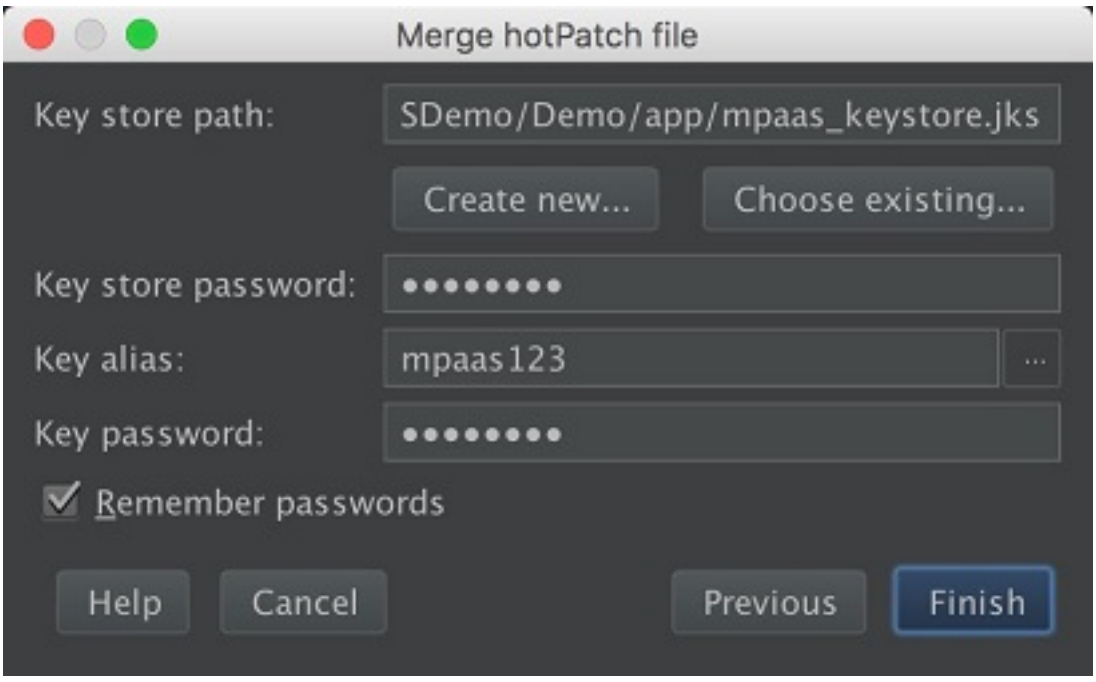
一个 Android App 版本最多只能有一个热修复包在运行。如果客户端某版本有两个 bug，那需要先在本地使用 **Merge Hotpatch** 功能将修正两个 bug 的热修复包合成一个热修复包。例如，针对某一个版本的 App 已经发过热修复包 A， 之后在这个版本上又发现了两外一个问题，那可以在本地生成另外一个热修复包 B，然后合并 A，B 两个热修复包，最后下发到客户端。

使用 mPaaS 插件的 **Merge Hotpatch**，通过以下步骤合并热修复包：

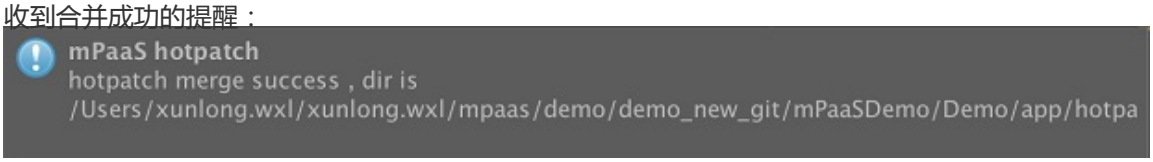
1. 填写热修复包文件夹地址：
    - **Merge dir**：选择合并的 hotpatch 包目录。文件夹下面是所有的需要合并的包，包名需要以 .jar 或者 .apk 结尾。
    - **Patch file dir**：选择合并之后输出的热修复包目录。



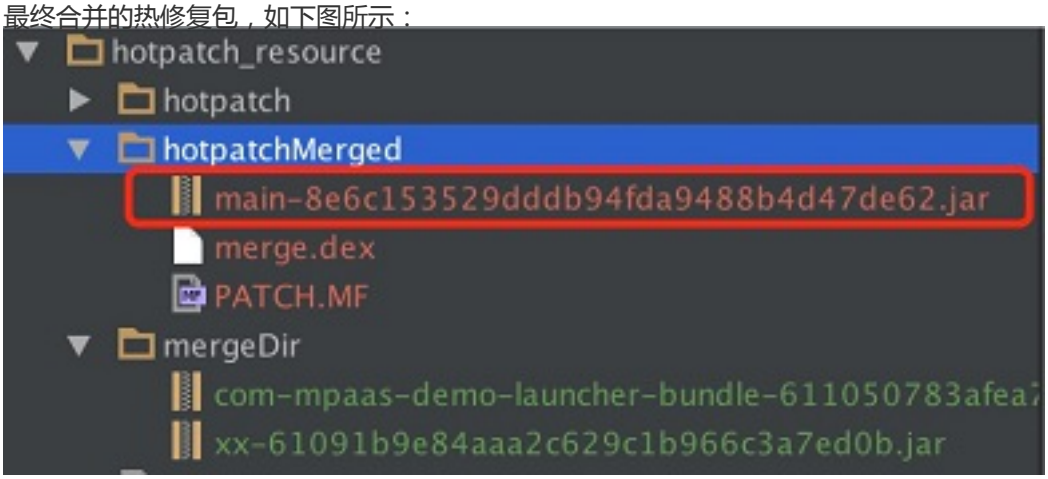
2. 配置签名信息：



3. 收到合并成功的提醒：



4. 最终合并的热修复包，如下图所示：



2.5 启动框架

2.5.1 创建 Portal

一个把各个 Bundle 合并为一个可以运行的 .apk 的工程称为 Portal。使用 mPaaS 插件创建 Portal。

操作步骤

打开 mPaaS 插件，选择 **File** > **New** > **mPaaS Projects** > **Portal Project** 打开创建 Portal 的引导。具体操作步骤，查看 本地开发：基于 mPaaS 框架开发 。

相关链接

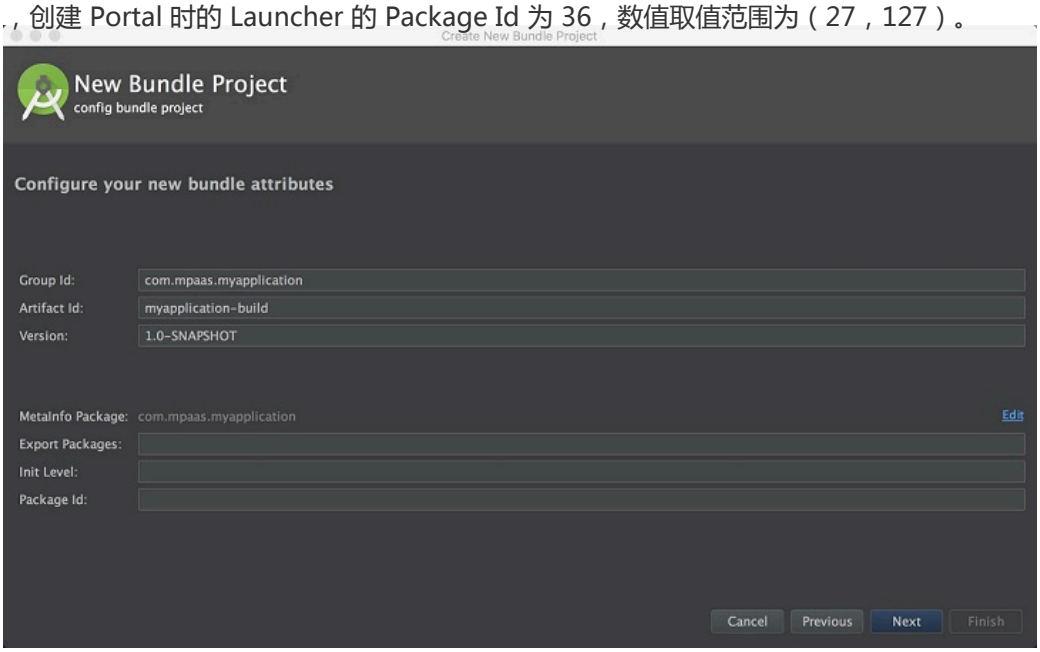
- mPaaS 插件

2.5.2 创建 Bundle

一个 mPaaS Bundle 工程一般由一个名为 app 的主 module 和若干个子 module 组成。使用 mPaaS 插件创建 Bundle。

操作步骤

1. 打开 mPaaS 插件，选择 **File** > **New** > **mPaaS Projects** > **Bundle Project** 打开创建 Bundle 的引导。
2. 配置 Bundle 的属性，注意以下配置项：
   - **Init Level**：框架启动时加载该 Bundle 的时机，懒加载设置为 11110000，可选参数 0 - 100。
   - **Package Id**：该 Bundle 的资源 Id 的起始两位 Id。原则上所有的 Bundle 不允许重复，创建 Portal 时的 Launcher 的 Package Id 为 36，数值取值范围为（27，127）。



结果

创建完工程后，在 main module 的 build.gradle 中记录了当前 Bundle 的如下属性配置：

```
bundle {
exportPackages ''
initLevel 11110000
packageName ''
packageId 36
}
```

相关链接

- mPaaS 插件
- 本地开发：基于 mPaaS 框架开发

2.5.3 加载框架与定制

mPaaS Android 框架提供了一整套的加载逻辑。基于此框架，研发团队可以进行多业务线开发。本文描述框架的启动流程以及如何在框架下添加自己的代码以对接启动。

启动流程

Application

传统 Android apk 运行时首先加载 AndroidManifest 文件 application 节点中 android:name 配置的 Application。

由于 mPaaS Android 框架重写了加载流程， android:name 中配置 mPaaS Android 框架的 com.alipay.mobile.quinox.LauncherApplication 类。

```
<application
android:name="com.alipay.mobile.quinox.LauncherApplication"
android:allowBackup="true"
android:debuggable="true"
android:hardwareAccelerated="false"
android:icon="@drawable/appicon"
android:label="@string/name"
android:theme="@style/AppThemeNew">
</application>
```

启动页

由于框架加载 bundle 可能会比较耗时，因此需要一个启动页等待框架启动完成之后再跳转到程序主页，所以在 AndroidManifest 文件中配置了框架提供的 com.alipay.mobile.quinox.LauncherActivity 应用启动页。

配置如下：

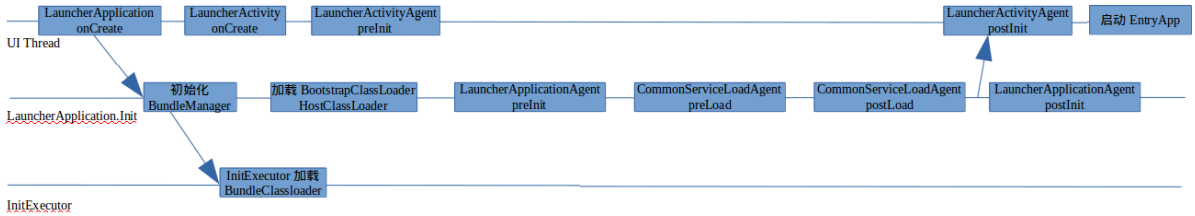
```
<activity
android:name="com.alipay.mobile.quinox.LauncherActivity"
android:configChanges="orientation | keyboardHidden | navigation"
android:screenOrientation="portrait"
android:windowSoftInputMode="stateAlwaysHidden">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

为方便开发人员对启动过程的理解，也为了避免启动过程被误改误删、被干扰，mPaaS 的启动过程被适度封装。因此，上述的 LauncherApplication 和 LauncherActivity 对开发人员完全不可见。

为了让客户端 App 在启动过程中实现自身的初始化逻辑，mPaaS 设计了 LauncherApplicationAgent 和 LauncherActivityAgent 代理。作为开发人员，您可以通过继承这两个类，在相应的回调中实现自身的初始化逻辑。

启动流程图

mPaaS Android 框架加载流程如下：



1. 框架启动后主线程会创建启动页 LauncherActivity ，然后回调 LauncherActivityAgent 的preInit 方法。
2. 框架进行 multidex。过程中会回调 LauncherApplicationAgent 的 preInit 方法，读取当前 .apk 中每个 bundle 的描述文件，并对每个 bundle 创建对应的类加载器，加载其中的资源文件。
3. 初始化完成后，回调 LauncherActivityAgent 和 LauncherApplicationAgent 的 postInit 方法。

定制

实际上，框架已在 Launcher 工程中自动创建了两个类 MockLauncherApplicationAgent 和 MockLauncherActivityAgent 继承了 LauncherApplicationAgent 和 LauncherActivityAgent 这两个回调接口。在框架的初始化过程中，它们分别在 LauncherAppliction 和 LauncherActivity 中被调用。

在 Portal 的 AndroidManifest.xml 中配置如下。开发人员也可在 Bundle 中自行实现这两个代理类，在以上配置中修改对应 meta-data 的 value 值：

```
<application
android:name="com.alipay.mobile.quinox.LauncherApplication">

<!-- Application 的回调配置 -->
<meta-data
android:name="agent.application"
android:value="com.mpaas.demo.launcher.framework.MockLauncherApplicationAgent"/>

<!-- Activity 的回调配置 -->
<meta-data
android:name="agent.activity"
android:value="com.mpaas.demo.launcher.framework.MockLauncherActivityAgent"/>
<!-- 启动页的布局配置 -->
<meta-data
android:name="agent.activity.layout"
android:value="layout_splash"/>

</application>
```

代理类

agent.application 配置的是启动过程代理 ApplicationAgent，如下所示：

```
public class MockLauncherApplicationAgent extends LauncherApplicationAgent {
@Override
protected void preInit() {
super.preInit();
//框架初始化前
}

@Override
protected void postInit() {
super.postInit();
//框架初始化后
}
}
```

客户端 App 可以在 LauncherApplicationAgent 的实现类中，进行一些 Application 级别的初始化工作。preInit() 回调发生在框架初始化之前，因此不要在这里调用框架（MicroApplicationContext）的相关接口。而 postInit() 回调发生在框架初始化完成之后，是可以使用的。

agent.activity 配置的是启动 Activity 的代理，如下所示：

```
public class MockLauncherActivityAgent extends LauncherActivityAgent {

@Override
public void preInit(Activity activity) {
super.preInit(activity);
//Launcher Activity 启动前
}

@Override
public void postInit(final Activity activity) {
super.postInit(activity);
//Launcher Activity 启动后
跳转程序首页的逻辑
startActivity(activity,YOUR_ACTIVITY);
}
}
```

同上述的 LauncherApplicationAgent 类似，LauncherActivityAgent 的两个回调也分别发生在框架初始化之前和之后，使用方法也类似。

修改启动页布局

在 Portal 的 AndroidManifest.xml 中还配置了启动页的布局文件，如下所示：

```
<application
android:name="com.alipay.mobile.quinox.LauncherApplication">
<!-- 启动页的布局配置 -->
<meta-data
android:name="agent.activity.layout"
android:value="layout_splash"/>
```

```
</application>
```

修改 value 值为自定义的布局文件名。

注意：需要把布局文件和引用的相关资源放置在 Portal 工程里。

相关链接

代码示例

2.6 注册通用组件

2.6.1 简介

模块化是 mPaaS 框架的设计原则之一，业务模块的低耦合与高内聚有利于业务的扩展和维护。

业务模块以 Bundle 的形式存在互不影响，但 Bundle 之间会存在一些关联性，比如跳转到另一个 Bundle 界面，调用另一个 Bundle 中的接口，或者Bundle 中的一些操作需要在初始化的过程中完成等。

因此，mPaaS 设计了 metainfo 通用组件注册机制，各个 Bundle 将需要注册的组件在 metainfo.xml 中声明。

框架目前支持以下组件:

- ActivityApplication（Application）
- ExternalService（Service）
- BroadcastReceiver
- Pipeline

metainfo.xml 格式如下：

```
<?xml version="1.0"encoding="UTF-8"?>
<metainfo>
<broadcastReceiver>
<className>com.mpaas.demo.broadcastreceiver.TestBroadcastReceiver</className>
<action>com.mpaas.demo.broadcastreceiver.ACTION_TEST</action>
</broadcastReceiver>
<application>
<className>com.mpaas.demo.activityapplication.MicroAppEntry</className>
<appId>33330007</appId>
</application>
</metainfo>
```

2.6.2 Application 组件

ActivityApplication 是 mPaaS 框架设计的组件，起到 Activity 容器的角色。ActivityApplication 组件的主要作用在于管理和组织各个 Activity，专门用于解决跳转到另一个Bundle 界面的问题。因而，调用方只需关心业务方在框架中注册的 ActivityApplication 信息以及约定的参数。

关于此任务

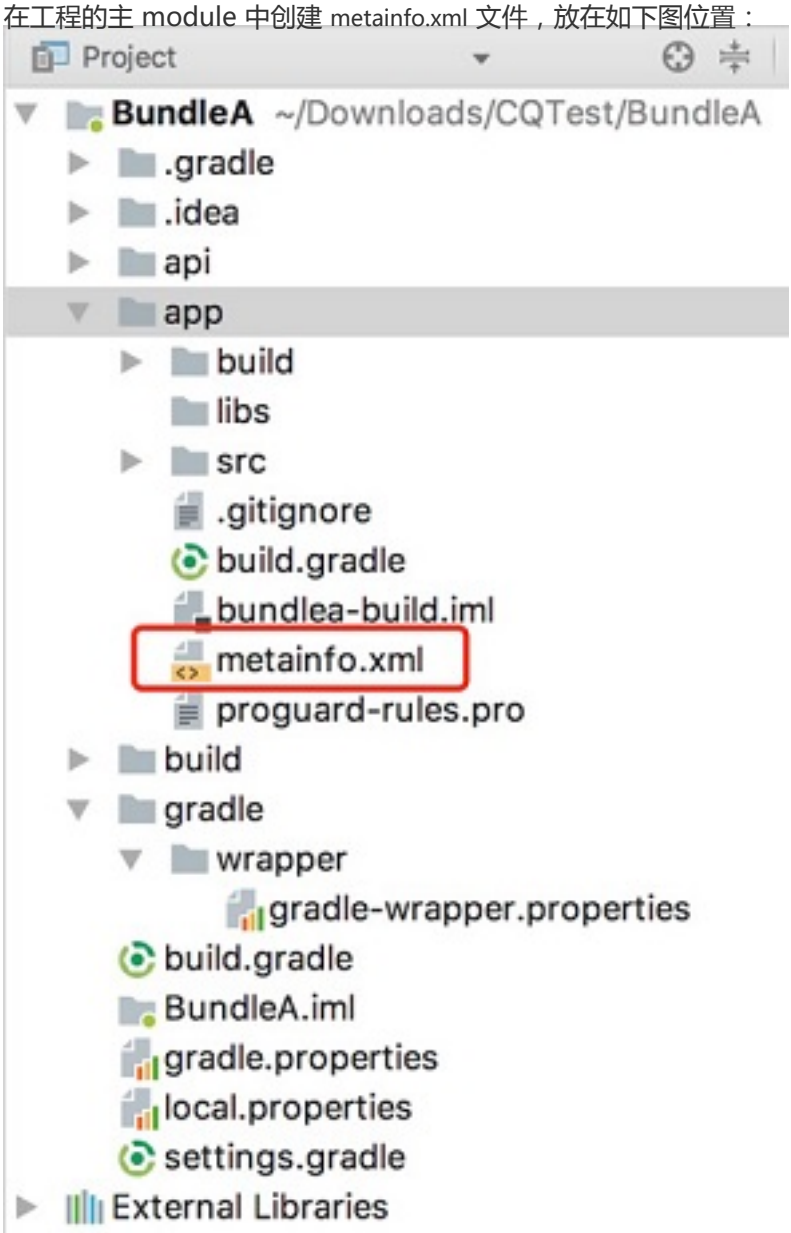
ActivityApplication 的创建、销毁等一系列逻辑完全由 mPaaS 框架来管理。业务方只需要处理其收到的参数并管理自己业务下的 Activity，这样业务方和调用方之间被有效的隔离开来。业务方和调用方只需要协调调用的参数，使得依赖更加轻量。

基于 mPaaS 框架开发的 Android 客户端应用，Activity 须继承自 BaseActivity 或 BaseFragmentActivity，以便能够被 ActivityApplication 类所管理。

提示：您可以下载代码示例，示例中包含跳转到另一个 Bundle 的 Activity。有关下载地址、使用方法及注意事项，查看 获取代码示例 。

操作步骤

1. 在工程的主 module 中创建 metainfo.xml 文件，放在如下图位置：



在 metainfo.xml 中写入如下的配置，其中：

- className 配置的类名用于提供跳转的类名称和定义各阶段的行为。具体类的定义，参见步骤 3 的代码。框架通过 className 定义的名称加载相应的类，所以该类一定不能被混淆，需要在混淆文件中保留。

  appId：业务的唯一标识。业务方只需要知道该业务的 appId 就能完成跳转。appId 与 ActivityApplication 的映射由框架层处理。

```
<?xml version="1.0"encoding="UTF-8"?>
<metainfo>
<application>
<className>com.mpaas.demo.hotpatch.HotpatchMicroApp</className>
<appId>33330002</appId>
</application>
</metainfo>
```

如果 metainfo 通过 className 指定的类只是完成简单的跳转，使用以下代码实现：

```
/**
* 场景一：
* 若只可能会跳转到某一个Activity界面，那么需要重载getEntryClassName和onRestart，前者返回Activity的classname，后者需要调用getMicroApplicationContext().startActivity(this, getEntryClassName());
* 场景二：
* 若需要根据需求跳转到不同的Activity界面那么需要重载onStart和onRestart，根据bundle中的参数跳转到指定的界面
* Created by mengfei on 2018/7/23.
*/
public class MicroAppEntry extends ActivityApplication {

@Override
public String getEntryClassName() {
//场景一：只可能跳转到某一个Activity在此返回classname即可
//return MainActivity.class.getName();
//场景二：根据参数跳转到某一界面，需要返回null
return null;
}

/**
* Application被创建时被调用，实现类可以在这里做些初始化的工作
*
* @param bundle
*/
@Override
protected void onCreate(Bundle bundle) {
doStartApp(bundle);
}

/**
* 启动Application时被调用
* 如果Application还没有被创建，会先去执行create方法，然后再执行onStart()回调
*/
@Override
protected void onStart() {
```

```
}

/**
* 当Application被销毁时，调用此回调
*
* @param bundle
*/
@Override
protected void onDestroy(Bundle bundle) {

}

/**
* 启动Application时，如果Application已经被start过了，则不调用onStart()而是调用onRestart()回调
*
* @param bundle
*/
@Override
protected void onRestart(Bundle bundle) {
//针对场景一：需要在此调用getMicroApplicationContext().startActivity(this, getEntryClassName());
doStartApp(bundle);
}

/**
* 当一个新的Application被start时，当前的Application将被暂停，此方法被回调
*/
@Override
protected void onStop() {

}

private void doStartApp(Bundle bundle) {
String dest = bundle.getString("dest");
if ("main".equals(dest)) {
Context ctx = LauncherApplicationAgent.getInstance().getApplicationContext();
ctx.startActivity(new Intent(ctx, MainActivity.class));
} else if ("second".equals(dest)) {
Context ctx = LauncherApplicationAgent.getInstance().getApplicationContext();
ctx.startActivity(new Intent(ctx, SecondActivity.class));
}
}
}
```

4. 作为调用者，您需要通过框架封装的 MicroApplicationContext 中提供的接口进行跳转。curId 参数也可以传 null：

```
MicroApplicationContext context = LauncherApplicationAgent.getInstance().getMicroApplicationContext();
String curId ="";
ActivityApplication curApp = context.getTopApplication();
if (null != curApp) {
curId = curApp.getAppId();
}
String appId ="目标ApplicationActivity的id";
Bundle bundle = new Bundle(); // 附加参数，也可以不传
```

```
context.startApp(curId, appId, bundle);
```

2.6.3 Service 组件

mPaaS 设计了 Service 组件解决跨 Bundle 调用接口。Service 组件用于将一些逻辑以服务的形式提供出来，供其他模块使用。

关于此任务

Service 组件的特点如下：

- 没有用户界面的限制。
- 在设计上，遵循接口与实现分离。

原则上只有接口类对调用者可见，所以接口类应定义在接口 module 中（在生成 Bundle project 时，默认会生成一个接口 module，名字是 api），实现定义在主 module 中。

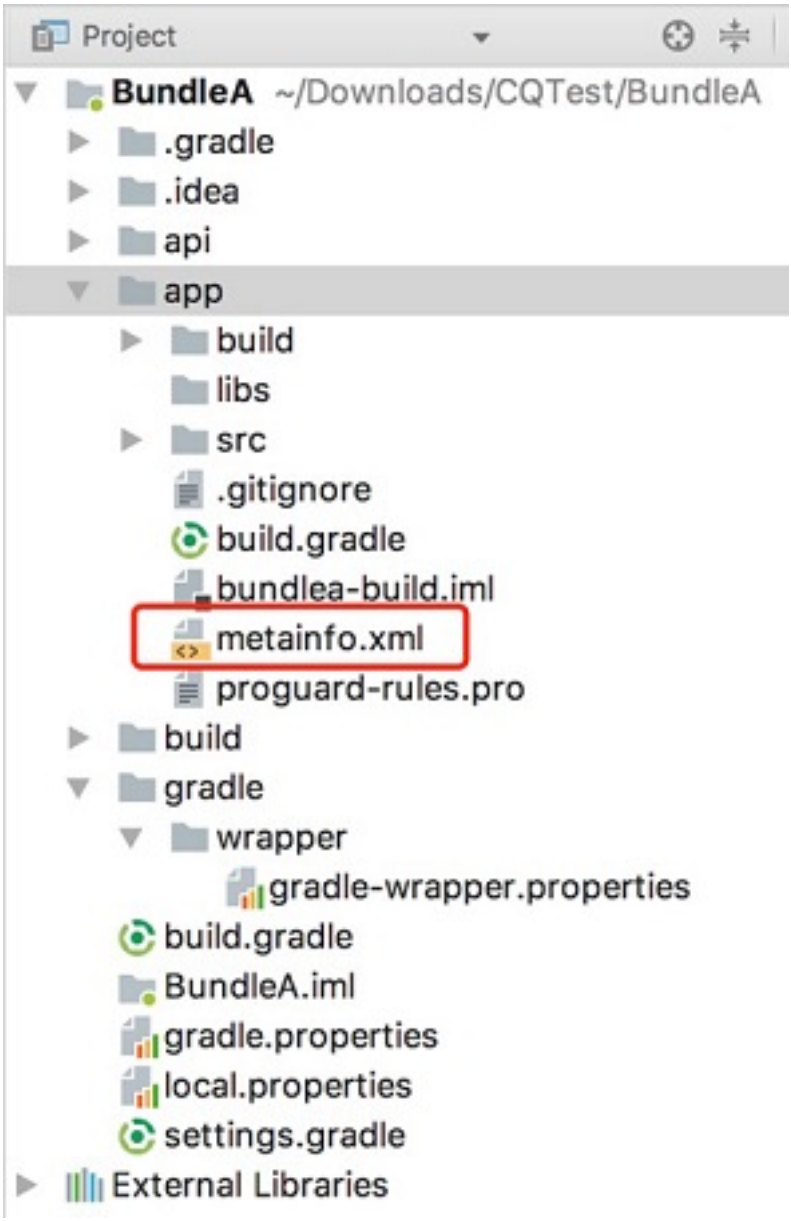
外部调用都通过 MicroApplicationContext 的 findServiceByInterface 接口，通过interfaceName 获取相应的服务。对于 Bundle 使用来说，只暴露服务抽象接口类，即 interfaceName 中定义的类，抽象接口类会定义在接口包中。

提示：您可以下载代码示例，示例中包含调用另一个 Bundle 的接口示例。有关下载地址、使用方法及注意事项，查看 获取代码示例 。

操作步骤

通过以下步骤注册 Service 组件：

定义 metainfo.xml 位置，如下图所示：



在 metainfo.xml 中写入如下的配置。框架将 interfaceName 作为 key，className 作为 value，记录两者的映射关系。其中，className 为具体接口的实现类， interfaceName 为抽象接口类：

```
<metainfo>
<service>
<className>com.mpaas.cq.bundleb.MyServiceImpl</className>
<interfaceName>com.mpaas.cq.bundleb.api.MyService</interfaceName>
<isLazy>true</isLazy>
</service>
</metainfo>
```

抽象接口类定义如下：

```
public abstract class MyService extends ExternalService {
```

```
public abstract String funA();
}
```

接口类实现定义如下：

```
public class MyServiceImpl extends MyService {
@Override
public String funA() {
return"这是 BundleB 提供的接口 by service";
}

@Override
protected void onCreate(Bundle bundle) {

}

@Override
protected void onDestroy(Bundle bundle) {

}
}
```

- 外部调用方式如下：

```
MyService myservice =
LauncherApplicationAgent.getInstance().getMicroApplicationContext().findServiceByInterface(MyService.class.getName());
myservice.funA();
```

2.6.4 BroadcastReceiver 组件

BroadcastReceiver 是 android.content.BroadcastReceiver 的封装，但区别在于 mPaaS 框架采用了 android.support.v4.content.LocalBroadcastManager 来注册和反注册 BroadcastReciever，因此，这些广播仅用于当前应用程序内部，除此之外，mPaas框架内部内置了一系列的广播事件，供使用者监听。

关于此任务

您可以下载包含该通用组件的代码示例。有关下载地址、使用方法及注意事项，查看 获取代码示例 。

mPaaS内置广播事件

mPaaS 定义了多种广播事件，主要用于监听当前应用的状态，注册监听与原生开发没有任何区别，但有一点需要特别注意，这些状态只有在主进程才能监听到。示例代码如下：

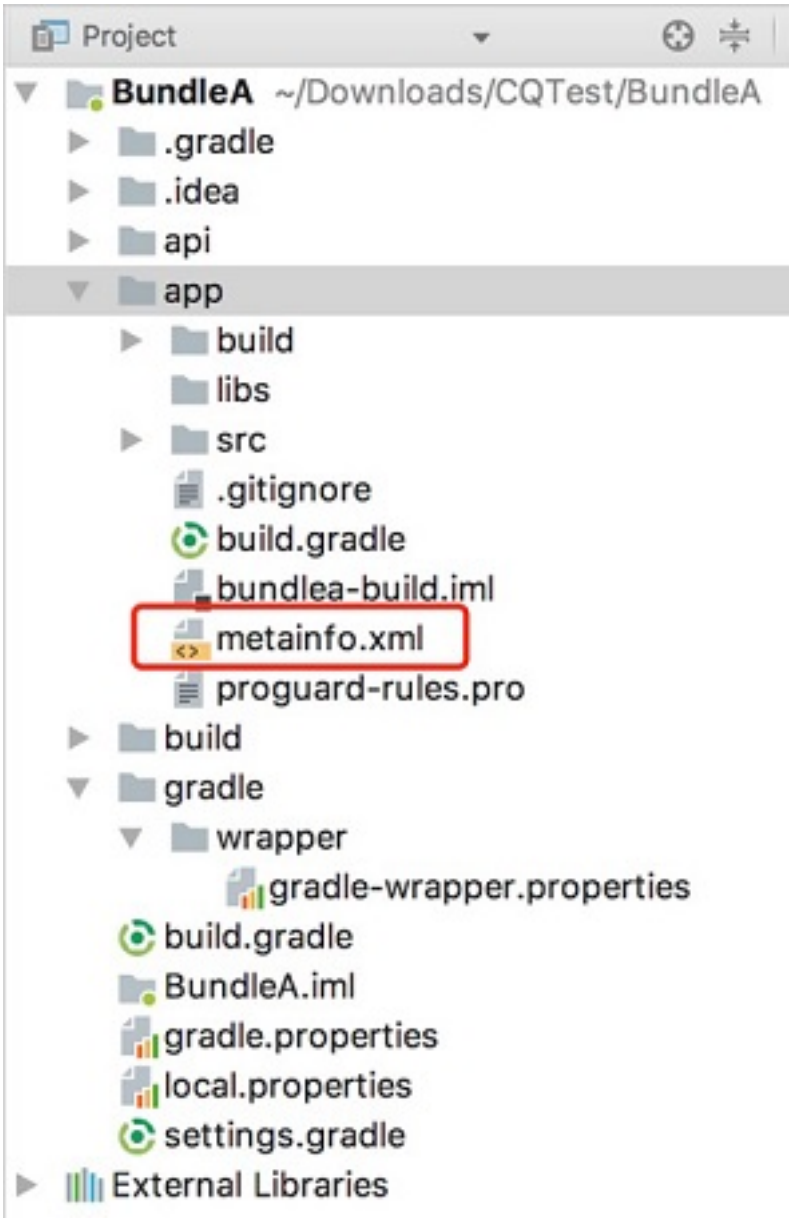
```
IntentFilter filter = new IntentFilter();
filter.addAction(MsgCodeConstants.FRAMEWORK_BROUGHT_TO_FOREGROUND);
filter.addAction(MsgCodeConstants.FRAMEWORK_ACTIVITY_USERLEAVEHINT);
LocalBroadcastManager.getInstance(this).registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        LoggerFactory.getTraceLogger().debug( s: "demo",  s1: "Received action:" + intent.getAction());
    }
}, filter);
```

内置的广播事件如下：

```
public interface MsgCodeConstants {
String FRAMEWORK_ACTIVITY_CREATE ="com.alipay.mobile.framework.ACTIVITY_CREATE";
String FRAMEWORK_ACTIVITY_RESUME ="com.alipay.mobile.framework.ACTIVITY_RESUME";
String FRAMEWORK_ACTIVITY_PAUSE ="com.alipay.mobile.framework.ACTIVITY_PAUSE";
// 用户离开的广播，压后台广播
String FRAMEWORK_ACTIVITY_USERLEAVEHINT ="com.alipay.mobile.framework.USERLEAVEHINT";
// 所有 Activity 全都 Stop 的广播，可能代表压后台，但目前没有用相同的判断逻辑
String FRAMEWORK_ACTIVITY_ALL_STOPPED ="com.alipay.mobile.framework.ACTIVITY_ALL_STOPPED";
String FRAMEWORK_WINDOW_FOCUS_CHANGED ="com.alipay.mobile.framework.WINDOW_FOCUS_CHANGED";
String FRAMEWORK_ACTIVITY_DESTROY ="com.alipay.mobile.framework.ACTIVITY_DESTROY";
String FRAMEWORK_ACTIVITY_START ="com.alipay.mobile.framework.ACTIVITY_START";
String FRAMEWORK_ACTIVITY_DATA ="com.alipay.mobile.framework.ACTIVITY_DATA";
String FRAMEWORK_APP_DATA ="com.alipay.mobile.framework.APP_DATA";
String FRAMEWORK_IS_TINY_APP ="com.alipay.mobile.framework.IS_TINY_APP";
String FRAMEWORK_IS_RN_APP ="com.alipay.mobile.framework.IS_RN_APP";
// 用户回到前台的广播
String FRAMEWORK_BROUGHT_TO_FOREGROUND ="com.alipay.mobile.framework.BROUGHT_TO_FOREGROUND";
}
```

自定义广播事件

定义 metainfo.xml 位置，如下图所示：



在 metainfo.xml 中写入如下配置：

```
<?xml version="1.0"encoding="UTF-8"?>
<metainfo>
<broadcastReceiver>
<className>com.mpaas.demo.broadcastreceiver.TestBroadcastReceiver</className>
<action>com.mpaas.demo.broadcastreceiver.ACTION_TEST</action>
</broadcastReceiver>
</metainfo>
```

自定义 Receiver 实现

```
public class TestBroadcastReceiver extends BroadcastReceiver {
private static final String ACTION_TEST ="com.mpaas.demo.broadcastreceiver.ACTION_TEST";
```

```
@Override
public void onReceive(Context context, Intent intent) {
String action = intent.getAction();
if (ACTION_TEST.equals(action)) {
//TODO
}
}
}
```

发送广播

```
LocalBroadcastManager.getInstance(LauncherApplicationAgent.getInstance().getApplicationContext()).sendBroadcast(new Intent("com.mpaas.demo.broadcastreceiver.ACTION_TEST"));
```

2.6.5 Pipeline 组件

mPaaS 框架有一个比较明显的启动过程，Pipeline 机制允许业务线将自己的运行逻辑封装成 Runnable 放到 Pipeline。框架在适当的阶段启动适当的 Pipeline。

以下为定义的 Pipeline 时机：

- com.alipay.mobile.framework.INITED: 框架初始化完成。进程在后台启动，框架也会初始化。
- com.alipay.mobile.client.STARTED: 客户端开始启动。必须等到界面出现，例如，欢迎界面。
- com.alipay.mobile.TASK_SCHEDULE_SERVICE_IDLE_TASK：优先级最低，当没有其他高优化级的操作时才会得到执行

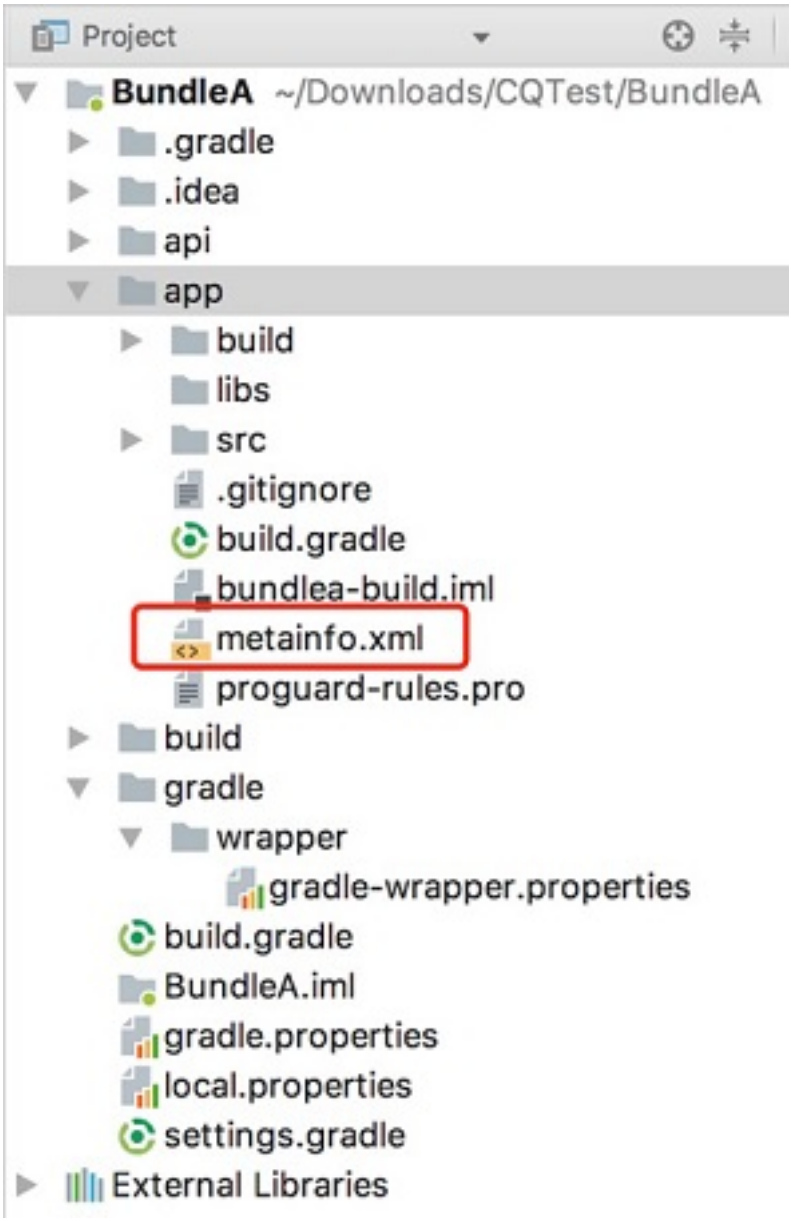
因为 Pipeline 的调用是由框架触发，使用者只需要在 metainfo 里指定相应的时机。

关于此任务

您可以下载包含该通用组件的代码示例。有关下载地址、使用方法及注意事项，查看 获取代码示例 。

操作步骤

1. 定义 metainfo.xml 位置，如下图所示：



在 metainfo.xml 中写入如下的配置：

```
<?xml version="1.0"encoding="UTF-8"?>
<metainfo>
<valve>
<className>com.mpaas.demo.pipeline.TestPipeLine</className>
<!--pipelineName就是用于指定执行所在的阶段-->
<pipelineName>com.alipay.mobile.client.STARTED</pipelineName>
<threadName>com.mpaas.demo.pipeline.TestPipeLine</threadName>
<!--weight指定了操作的优化级，值越小，代表越会优先得到执行-->
<weight>10</weight>
</valve>
</metainfo>
```

3. 实现 Pipeline：

```
public class TestPipeLine implements Runnable {
@Override
public void run() {
PreferenceManager.getDefaultSharedPreferences(LauncherApplicationAgent.getInstance().getApplicationContext()).
edit().putString(Constants.KEY_PIPELINE_RUN_TIMESTAMP,"Pipeline running timestamp:" +
System.currentTimeMillis()).apply();
}
}
```

2.7 使用 Material Design

2.7.1 配置工程

由于 mPaaS 框架的特殊性，若直接在项目中引入 AppCompat 相关库，编译时会报错，提示资源找不到。为了解决该问题，mPaaS 提供了自定义的 AppCompat 库。要使用 mPaaS 自定义的 AppCompat 库，首先要配置 Portal 工程和 Bundle 工程。

关于此任务

mPaaS AppCompat 库基于原生 Android 23 版本开发，包含以下组件：

- appcompat
- animated-vector-drawable
- cardview
- design
- recyclerview
- support-vector-drawable

由于该自定义的 AppCompat 库是基于原生 Android 23 版本编译，和原生并无区别，只是解决了引入原生库的一系列编译问题。

使用资源的情况主要包括 **使用另一个 Bundle 中的资源** 、**对外提供资源**、**在 AndroidManifest 中使用自定义资源**。由于 mPaaS 框架的特殊性，您需要了解使用资源不同情况下的注意事项。要了解具体内容，查看 使用资源 。

操作步骤

1. 配置 Portal 工程

在调用 mPaaS AppCompat 库前，完成以下操作配置 Portal 工程：

1. 确保 Gradle 脚本中依赖的 quinox 库使用以下版本：

```
bundle"com.alipay.android.phone.mobilesdk:quinox-build:2.3.6.171123113218@jar"
manifest"com.alipay.android.phone.mobilesdk:quinox-build:2.3.6.171123113218:AndroidManifest@xml"
```

2. 运行以下命令将 Gradle 打包插件（Alipay Plugin for Gradle）版本替换为以下版本：

```
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.5.68'
```

3. 移除 Gradle 脚本中原先依赖的 AppCompat 库。
4. 在 Gradle 脚本中添加以下 AppCompat 依赖：

```
bundle 'com.mpaas.android.res.base:mpaas-baseresjar:1.0.0.180626203034@jar'
manifest 'com.mpaas.android.res.base:mpaas-baseresjar:1.0.0.180626203034:AndroidManifest@xml'
```

5. 配置完成后，为了让 Bundle 工程调用 AppCompat 组件，同步 Portal 工程。

2. 配置 Bundle 工程

1. 在需要使用 AppCompat 组件的 Bundle 工程中，将 Gradle 打包插件（Alipay Plugin for Gradle）修改为以下版本：

```
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.5.68'
```

2. 根据您使用组件的情况，选择需要依赖的子组件。以下为添加 recyclerview 的示例语句：

```
provided'com.mpaas.android.res.base:mpaas-baseresjar:1.0.0.180626203034:recyclerview@jar'
```

2.7.2 使用资源

Material Design 常见的资源包括 String、Color、Style 等。使用资源的情况主要包括：

- 检测 Package ID 是否重复
- 使用另一个 Bundle 中的资源
- 对外提供资源
- 在 AndroidManifest 中使用自定义资源

检测 Package ID 是否重复

如果按照本文的说明使用资源时，出现资源找不到的情况，您需要查看 Package ID 是否重复。Package ID 定义在 build.gradle 中，其 ID 值与生成的资源 ID 有关。重复定义 Package ID 会导致资源找不到的情况。

您可以通过以下两种方式检测 Package ID 是否重复：

方式一：通过 gradle task 自动检测

前置条件：

android-gradle-plugin 版本号为 3.0.0.5.45 及以上。如：

```
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.5.45'
```

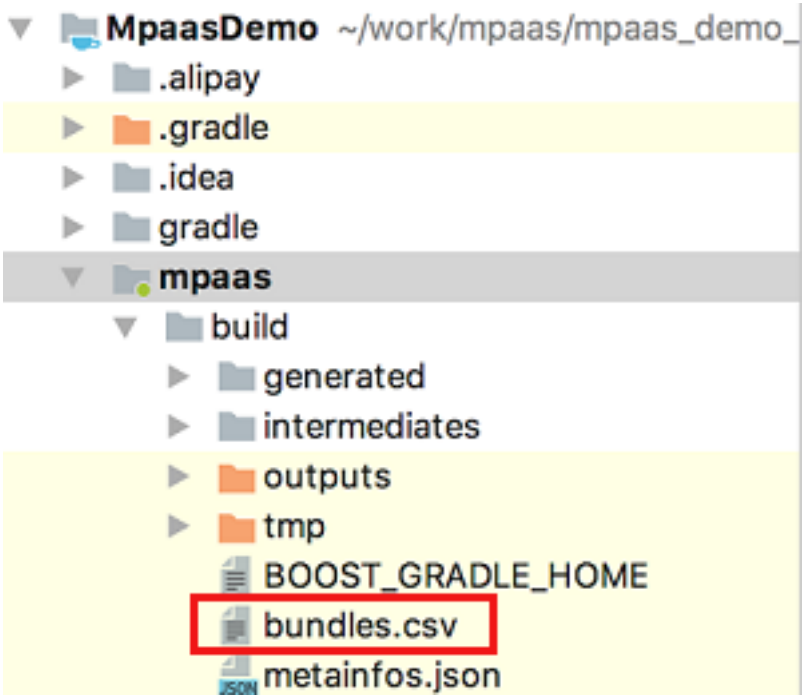
检测步骤：

- 在 Portal 工程根目录下，执行以下命令：
  - Windows 系统：执行 gradlew.bat checkBundleIds。
  - 其它操作系统：执行 gradlew checkBundleIds。
- 检查输出结果：
  - 若输出结果包含 No duplicate bundle ids found，说明 Package ID 没有重复。
  - 若输出结果包含 There are duplicate bundle ids，说明 Package ID 有重复。您可以根据输出结果进一步判断具体是哪些 Package ID 重复。

方式二：手动检测

手动检测适用于任何情况，具体步骤如下：

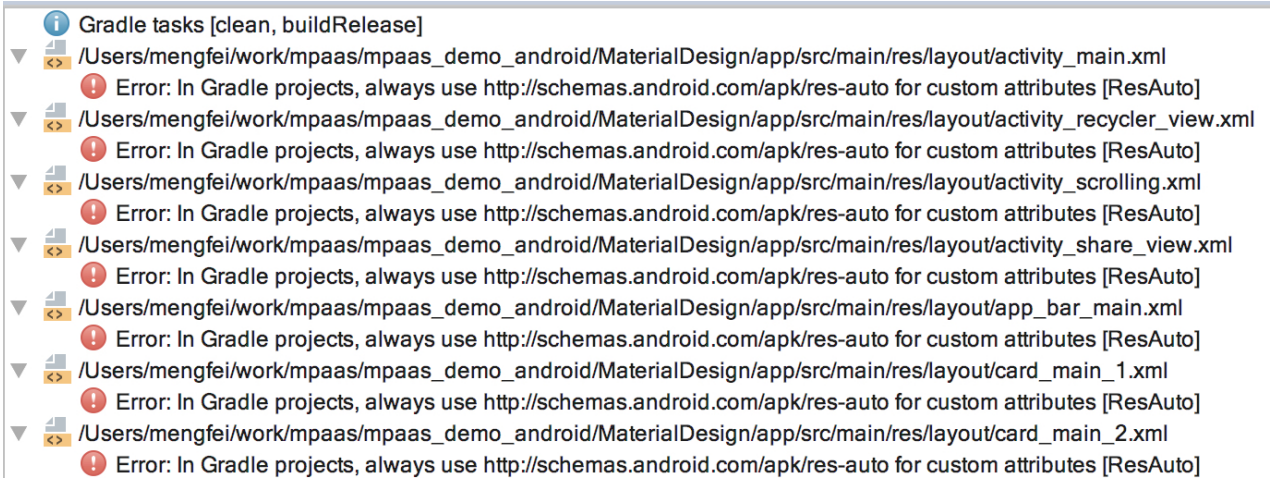
在 Portal 工程以下位置打开 bundles.csv 纯文本文件。



2. 将 PackageId 列进行排序，查看有无重复的 Package ID；确保没有重复的 Package ID 后再重新编译。

使用另一个 Bundle 中的资源

使用 mpaas-baseresjar 中的资源属于该情况。在使用另一个 Bundle 中的资源时，必须要加上资源的命名空间。命名空间为资源所在 Bundle 的 applicationID，构建 release 包时可能会出现如下图的错误：



解决方法是在 build.gradle 下配置 lintOptions，配置方法如下图所示：

```
android {
    compileSdkVersion 23
    buildToolsVersion '26.0.2'

    defaultConfig {
        applicationId "com.mpaas.demo.materialdesign"
        minSdkVersion 18
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    lintOptions {
        disable 'ResAuto'
    }
}
```

只要引用该 Bundle 中的资源（包括在 layout 中引用、在自定义 style 中引用等），就必须加入命名空间作为前缀。否则，会出现资源找不到的编译错误。

代码示例：在 layout 中引用

以在 layout 中引用另一个 Bundle 中的资源为例，使用的代码示例如下所示：

```
<?xml version="1.0"encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res/com.mpaas.android.res.base"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fitsSystemWindows="true">

<android.support.design.widget.AppBarLayout
android:id="@+id/app_bar_scrolling"
```

```
android:layout_width="match_parent"
android:layout_height="@dimen/app_bar_height_image_view"
android:fitsSystemWindows="true"
android:theme="@style/AppTheme.AppBarOverlay"
android:background="@color/blue">

<android.support.design.widget.CollapsingToolbarLayout
android:id="@+id/collapsing_toolbar_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fitsSystemWindows="true"
app:contentScrim="?com.mpaas.android.res.base:attr/colorPrimary"
app:layout_scrollFlags="scroll|exitUntilCollapsed">

<ImageView
android:id="@+id/image_scrolling_top"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fitsSystemWindows="true"
android:scaleType="fitXY"
android:src="@drawable/material_design_3"
app:layout_collapseMode="parallax"/>

<android.support.v7.widget.Toolbar
android:id="@+id/toolbar"
android:layout_width="match_parent"
android:layout_height="?com.mpaas.android.res.base:attr/actionBarSize"
app:layout_collapseMode="pin"
app:popupTheme="@style/AppTheme.PopupOverlay"/>

</android.support.design.widget.CollapsingToolbarLayout>
</android.support.design.widget.AppBarLayout>

<android.support.design.widget.FloatingActionButton
android:id="@+id/fab_scrolling"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_margin="@dimen/big_activity_fab_margin"
android:src="@drawable/ic_share_white_24dp"
app:layout_anchor="@id/app_bar_scrolling"
app:layout_anchorGravity="bottom|end"/>

<include layout="@layout/content_scrolling"/>

</android.support.design.widget.CoordinatorLayout>
```

代码示例：在自定义 style 中引用

以在自定义 style 中使用另一个 Bundle 中的资源为例，使用的代码示例如下所示：

```
<style name="AppTheme"parent="@com.mpaas.android.res.base:style/Theme.AppCompat.NoActionBar">
<!-- Customize your theme here. -->
<item name="com.mpaas.android.res.base:colorPrimary">@color/colorPrimary</item>
<item name="com.mpaas.android.res.base:colorPrimaryDark">@color/colorPrimaryDark</item>
<item name="com.mpaas.android.res.base:colorAccent">@color/colorAccent</item>
```

```
</style>
```

对外提供资源

1. 作为使用者，为保证编译通过，您需要引入资源 Bundle 的 jar 包，以 provide 形式引入，如下所示：

```
provided 'com.mpaas.demo.materialdesign:materialdesign-build:1.0-SNAPSHOT:raw@jar'
```

2. 配置 Portal：在 Portal 中需要引入资源 Bundle 的信息，如下所示：

```
manifest"com.mpaas.demo.materialdesign:materialdesign-build:1.0-SNAPSHOT:AndroidManifest@xml"
bundle"com.mpaas.demo.materialdesign:materialdesign-build:1.0-SNAPSHOT:raw@jar"
`
```

定义资源。完成以下步骤定义资源，以使得资源可被其他的 Bundle 或者 Portal 引用：

将需要对外提供的资源 id 定义在 public.xml 中，以达到固定资源 id 的目的。该能力由 Android 提供。资源 id 值可以从 R.java 中拷贝，示例代码如下：

```
<?xml version="1.0"encoding="utf-8"?>
<resources>
<public name="AppTheme"id="0x1f030000"type="style"/>
<public name="AppTheme.AppBarOverlay"id="0x1f030001"type="style"/>
<public name="AppTheme.NoActionBar"id="0x1f030002"type="style"/>
<public name="AppTheme.NoActionBar.StatusBar"id="0x1f030003"type="style"/>
<public name="AppTheme.PopupOverlay"id="0x1f030004"type="style"/>
<public name="DialogFullscreen"id="0x1f030005"type="style"/>
<public name="DialogFullscreenWithTitle"id="0x1f030006"type="style"/>

<public name="title_activity_login"id="0x1f0c0081"type="string"/>
<public name="title_activity_recycler_view"id="0x1f0c0082"type="string"/>
<public name="title_activity_share_view"id="0x1f0c0085"type="string"/>
<public name="title_activity_scrolling"id="0x1f0c0083"type="string"/>
<public name="title_activity_settings"id="0x1f0c0084"type="string"/>
<public name="title_activity_about"id="0x1f0c007f"type="string"/>
<public name="activity_donate"id="0x1f0c000e"type="string"/>
<public name="activity_my_apps"id="0x1f0c000f"type="string"/>

</resources>
```

- 外部在使用时，在资源前加上包名作为前缀。具体内容，参考 使用另一个 Bundle 中的资源 。

在 AndroidManifest 中使用自定义资源

如果您在 Bundle 工程的 AndroidManifest 定义了主题，如下代码所示：

```
<activity
android:name=".activity.MainActivity"
android:launchMode="singleTop"
android:theme="@com.mpaas.demo.materialdesign:style/AppTheme.NoActionBar"
android:windowSoftInputMode="stateHidden|stateUnchanged">
</activity>
```

请您在 Portal 工程的主工程路径下添加 res_slinks 文件，并且将 Bundle 名 ，逐行添加到 res_slinks 文件中。同时在 build.gradle 中去掉该 Bundle 的 manifest依赖 。如下代码所示：

```
manifest 'com.mpaas.demo.materialdesign:materialdesign-build:1.0.0:AndroidManifest@xml'
```

2.8 混淆 Android 文件

mPaaS Android 客户端开发的应用程序是通过 Java 代码编写而成，而 Java 代码易被反编码，因此为了保护 Java 源代码，需要使用 ProGuard 混淆 Android 文件。

ProGuard 是一个压缩、优化和混淆 Java 字节码文件的工具。

- **压缩** 指检测以及删除没有用到的类、字段、方法以及属性。
- **优化** 指分析以及优化方法的字节码。
- **混淆** 指使用无意义的短变量，对类、变量、方法进行重命名。

使用 ProGuard 可以让代码更精简，更高效，也更难被逆向或破解。

前置条件

您已经配置 mPaaS 工程。

关于此任务

mPaaS 工程采用组件化方案，每一个 bundle 的编译产物都是一个已经混淆的 dex 文件，所以配置混淆文件是以 bundle 工程为单位而进行的。

portal 工程通常没有代码，不需要开启混淆。

代码示例

Gradle 配置

```
android {
compileSdkVersion 23
buildToolsVersion"19.1.0"

defaultConfig {
applicationId"com.youedata.xionganmaster.launcher"
minSdkVersion 15
```

```
targetSdkVersion 23
versionCode 1
versionName"1.0"
}
buildTypes {
release {
// 混淆开关，是否混淆
minifyEnabled true
// 混淆文件列表，混淆规则配置
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
}
lintOptions {
checkReleaseBuilds false
// Or, if you prefer, you can continue to check for errors in release builds,
// but continue the build even when errors are found:
abortOnError false
}
}
```

混淆文件示例

下列混淆是一个基本示例（如果要添加额外的第三方库，需要加入其它混淆，通常配置文件可在第三方库的官网中找到）：

```
Add project specific ProGuard rules here.
By default, the flags in this file are appended to flags specified
in /Users/xunlong.wxl/Library/Android/sdk/tools/proguard/proguard-android.txt
You can edit the include path and order by changing the proguardFiles
directive in build.gradle.

For more details, see [Shrink your code and
resources](http://developer.android.com/guide/developing/tools/proguard.html)。

Add any project specific keep options here:

If your project uses WebView with JS, uncomment the following
and specify the fully qualified class name to the JavaScript interface
class:
-keepclassmembers class fqcn.of.javascript.interface.for.webview {
public *;
}
-optimizationpasses 5
-dontusemixedcaseclassnames
-dontskipnonpubliclibraryclasses
-dontpreverify
-verbose
-ignorewarnings
-optimizations !code/simplification/arithmetic,!field/*,!class/merging/*

-keep public class * extends android.app.Activity
-keep public class * extends android.app.Application
-keep public class * extends android.app.Service
-keep public class * extends android.content.BroadcastReceiver
-keep public class * extends android.content.ContentProvider
```

```
-keep public class com.android.vending.licensing.ILicensingService
-keep public class com.alipay.mobile.phonecashier.*
-keepnames public class *
-keepattributes SourceFile,LineNumberTable
-keepattributes *Annotation*

-keepclasseswithmembernames class * {
native <methods>;
}

-keepclasseswithmembernames class * {
public <init>(android.content.Context, android.util.AttributeSet);
}

-keepclasseswithmembernames class * {
public <init>(android.content.Context, android.util.AttributeSet, int);
}

-keepclassmembers enum * {
public static **[] values();
public static ** valueOf(java.lang.String);
}

-keep class * extends java.lang.annotation.Annotation { *; }
-keep interface * extends java.lang.annotation.Annotation { *; }

-keep class * implements android.os.Parcelable {
public static final android.os.Parcelable$Creator *;
}

-keep public class * extends android.view.View{
!private <fields>;
!private <methods>;
}

-keep class android.util.**{
public <fields>;
public <methods>;
}

-keep public class com.squareup.javapoet.**{
!private <fields>;
!private <methods>;
}
-keep public class javax.annotation.**{
!private <fields>;
!private <methods>;
}
-keep public class javax.inject.**{
!private <fields>;
!private <methods>;
}
-keep interface **{
!private <fields>;
!private <methods>;
}
```

```
# for dagger
-keep class * extends dagger.internal.Binding
-keep class * extends dagger.internal.ModuleAdapter

-keep class **$$ModuleAdapter
-keep class **$$InjectAdapter
-keep class **$$StaticInjection

-keep class dagger.** { *; }

-keep class javax.inject.**{ *; }
-keep class * extends dagger.internal.Binding
-keep class * extends dagger.internal.ModuleAdapter
-keep class * extends dagger.internal.StaticInjection

# for butterknife
-keep class butterknife.* { *; }
-keep class butterknife.** { *; }
-dontwarn butterknife.internal.**
-keep class **$$ViewBinder { *; }

-keepclasseswithmembernames class * {
@butterknife.* <fields>;
}

-keepclasseswithmembernames class * {
@butterknife.* <methods>;
}
```

避免混淆通用组件

如果在metainfo.xml中注册了通用组件，编译时会检查这些组件是否存在，请避免这些组件被混淆，否则会编译失败。例如注册了以下组件：

```
<metainfo>
<service>
<className>com.mpaas.cq.bundleb.MyServiceImpl</className>
<interfaceName>com.mpaas.cq.bundleb.api.MyService</interfaceName>
<isLazy>true</isLazy>
</service>
</metainfo>
```

请在混淆配置中添加：

```
-keep class com.mpaas.cq.bundleb.MyServiceImpl
-keep class com.mpaas.cq.bundleb.api.MyService
```

相关链接

ProGuard 帮助手册

3 基于原生框架

3.1 接入流程简介

阅读本文前，请确保已阅读 接入方式简介 。

基于原生框架开发的流程如下：

1. 在控制台创建应用并下载配置文件
2. 通用基础配置
3. 参考各组件文档完成组件接入。如：
    - 热修复
    - 消息推送
    - 移动分析

3.2 通用基础配置

为了使用 mPaaS 组件，您需要完成通用基础配置。

前置条件

您已 在控制台创建应用并下载配置文件 。

通用基础配置

在工程最外层的 build.gradle 文件中配置仓库地址。其中，仓库 username 和 password 请通过 工单 获取。

```
buildscript {
repositories {
mavenLocal()
maven {url 'http://maven.aliyun.com/nexus/content/groups/public/'}
// 金融云repo
maven {
credentials {
username"工单获取"
password"工单获取"
}
url"http://mvn.cloud.alipay.com/nexus/content/repositories/releases/"
}
maven {url 'http://maven.aliyun.com/nexus/content/repositories/google/'}

}
dependencies {
classpath 'com.android.tools.build:gradle:3.0.1'
// 请加入如下插件，简化配置
classpath 'com.android.boost.dependency:easyaar:3.3.6'
}
}
allprojects {
```

```
repositories {
maven {url 'http://maven.aliyun.com/nexus/content/groups/public/'}
// 金融云 repo
maven {
credentials {
username"工单获取"
password"工单获取"
}
url"http://mvn.cloud.alipay.com/nexus/content/repositories/releases/"
}
maven {url 'http://maven.aliyun.com/nexus/content/repositories/google/'}
}
}
```

2. 在工程 app (即主工程) 的 build.gradle 文件中添加如下内容：

```
apply plugin: 'com.android.application'
// 注意位于 'com.android.application' apply 之后
apply plugin: 'com.alipay.apollo.baseline.platform'
defaultConfig {
ndk {
abiFilters"armeabi"
}
}
```

3. 将在控制台下载的 配置文件 （ 以 .config 结尾 ）拷贝到 app (即主工程)下。示例如下:

4. 禁用 aapt2。在 gradle.properties 文件中添加如下内容：

```
android.enableAapt2=false
```

禁用 aapt2 的原因：aapt2 对 AndroidManifest.xml 转义符号支持尚不完善。

后续操作

接入组件。更多信息，请参见各组件的接入文档。

相关链接

接入流程简介

4 参考

4.1 基线列表

4.1.1 基线 10.1.20

基线 是指一系列功能的稳定版本，是进一步开发的基础。对于 mPaaS，基线是 mPaaS 的 SDK 列表。由于 mPaaS 产品是基于支付宝的某个特定版本开发的，因此基于该版本的 SDK 集合称之为基线。

随着 mPaaS 产品的不断升级，会出现多个版本的基线。本文介绍 10.1.20 基线，该基线是基于支付宝 10.1.20 版本。相较于 10.1.10 版本，10.1.20 基线增加了以下特性：

- 修复若干Bug，提高稳定性
- 小程序增加对蓝牙支持

基线列表

模块	Bundle	备注
框架	com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.180926192302	
	com.alipay.android.phone.mobilesdk:framework-build:2.5.7.180320154023	
	com.alipay.android.phone.mobilesdk:quinox-build:2.5.5.180327225510	
基础服务	com.alipay.android.phone.thirdparty:nineoldandroids-build:2.4.0.151028164451	
	com.alipay.android.phone.thirdparty:androidsupportpalette-build:7.22	
	com.alipay.mobileapi:mobileapp-build:1.0.0.20170307	
	com.alipay.android.phone.thirdparty:androidsupportrecyclerview-build:7.23.2.170814173705	
	com.alipay.android.phone.thirdparty:fastjson-build:1.1.66.171117210532	
	com.alipay.mobileapi:mobileappcommon-build:1.0.0.20171204	
	com.alipay.android.phone.thirdparty:utdid-build:2.1.1.171212154524	
	com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.171222165440	
	com.alipay.android.phone.thirdparty:androidannotations-build:9.6.6.180102221133	
	com.alipay.android.phone.mobilesdk:storage-build:1.7.0.180308143113	

	com.alipay.android.phone.mobilesdk:commonservice-build:1.9.0.180319220933	
	com.alipay.android.phone.mobilecommon:ui-build:10.1.18.180320200212	
	com.alipay.android.phone.wallet:antui-build:10.1.20.180321211516	
	com.alipay.android.phone.mobilesdk:common-build:1.8.2.180326200857	
	com.mpaas.mpaasadapter:commonbiz-build:1.0.0.180420210915	
	com.alipay.android.phone.mobilesdk:commonbizservice-build:1.7.0.180507114202	
日志监控	com.alipay.android.phone.mobilesdk:forerunner-build:1.0.0.170616150043	
	com.alipay.android.phone.mobilesdk:aspect-build:1.4.1.180212140351	
	com.alipay.android.phone.mobilesdk:autotracker-build:1.0.0.180320154841	
	com.alipay.android.phone.mobilesdk:tianyanadapter-build:1.0.4.180320161437	
	com.alipay.android.phone.mobilesdk:logging-build:2.0.2.180322162837	
移动网关	com.alipay.android.phone.thirdparty:wire-build:1.5.3.571018171133	
	com.alipay.android.phone.mobilesdk:netsdkextdependapi-build:1.0.1.180102162159	
	com.alipay.android.phone.mobilesdk:netsdkextdepend-build:1.0.1.180102162159	
	com.alipay.android.phone.mobilesdk:crypto-build:1.0.1.180309141623	
	com.alipay.android.phone.mobilesdk:rpc-build:1.8.1.180320105342	
	com.alipay.android.phone.mobilesdk:transportext-build:1.8.1.180320110007	
	com.alipay.android.phone.mobilesdk:transport-build:1.8.1.181114200639	
	com.alipay.android.phone.mobilesdk:monitor-build:2.1.4.180502135527	
H5容器	com.alipay.android.phone.wallet:nebulaucsdk-build:1.0.0.180322110653	
	com.alipay.android.phone.wallet:nebula-build:1.6.2.180322181148	
	com.alipay.android.phone.wallet:nebulaappcenter-build:1.6.2.180326120533	
	com.alipay.android.phone.wallet:nebulauc-build:1.6.0.180404103447	
	com.alipay.android.phone.wallet:nebulaappproxy-build:1.6.2.180413172504	
	com.mpaas.nebula.adapter:mpaasnebulaadapter-build:10.1.20.190104175650	
多媒体	com.alipay.android.phone.wallet:basicstl-build:1.0.0.161020105915	
	com.alipay.android.phone.mobilecommon:multimediaext-build:1.11.0.180103194218	
	com.alipay.android.phone.mobilecommon:multimedia-build:1.19.0.180320163427	
	com.alipay.multimedia.base:basic-build:1.19.0.180320163427	
	com.alipay.android.phone.mobilecommon:multimediabiz-build:1.19.0.180326192800	
小程序	com.alipay.android.phone.mobilecommon:map-build:1.8.0.170406170552	
	com.alipay.android.phone.mobilecommon:falconlooks-build:1.6.64.171229002058	
	com.alipay.android.phone.mobilecommon:falconrecmanager-build:1.0.0.180102132516	
	com.alipay.android.phone.mobilecommon:falcon-build:1.6.41.180123212839	
	com.alipay.android.phone.mpaas:tinyappcustom-build:1.0.0.180207204431	
	com.alipay.android.phone.thirdparty:amap3dmap-build:3.3.3.180320163433	
	com.alipay.android.phone.mobilecommon:mapbiz-build:1.8.0.180321115405	

	com.alipay.android.phone.mobilecommon:adaptermap-build:1.0.0.180324142935	
	com.alipay.android.phone.wallet:apble-build:1.0.0.180408111111	
	com.alipay.android.phone.mobilesdk:liteprocess-build:1.0.0.180409142018	
	com.alipay.android.phone.wallet:beehive-build:10.1.22.180419184647	
	com.alipay.android.phone.wallet:tinyappcommon-build:1.0.0.180420181653	
sync	com.alipay.android.phone.sync:syncservice-build:2.0.0.180731121835	
热修复	com.alipay.android.phone.mobilecommon:dynamicrelease-build:2.4.2.180326164409	
	com.alipay.android.phone.mobilecommon:cloudfix-build:2.4.5.180327205120	
定位	com.alipay.android.phone.mobilecommon:lbs-build:1.9.0.171219202112	
扫码	com.alipay.android.phone.scancode:mascanengine-build:2.0.0.180402111033	
	com.alipay.android.phone.scancode:bqcscanservice-build:2.0.0.180402111033	
	com.alipay.android.phone.wallet:scanexport-build:1.0.0.181116172939	
无线保镖	com.alipay.android.phone.thirdparty:securityguard-build:5.4.2008.171127213741	

注意：消息推送，升级，分享组件的基线请参考各组件的使用文档配置

```
bundle 'com.alipay.android.phone.thirdparty:nineoldandroids-build:2.4.0.151028164451@jar'
manifest 'com.alipay.android.phone.thirdparty:nineoldandroids-build:2.4.0.151028164451:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidsupportpalette-build:7.22@jar'
manifest 'com.alipay.android.phone.thirdparty:androidsupportpalette-build:7.22:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:basicstl-build:1.0.0.161020105915@jar'
manifest 'com.alipay.android.phone.wallet:basicstl-build:1.0.0.161020105915:AndroidManifest@xml'

bundle 'com.alipay.mobileapi:mobileapp-build:1.0.0.20170307@jar'

bundle 'com.alipay.android.phone.mobilecommon:map-build:1.8.0.170406170552@jar'
manifest 'com.alipay.android.phone.mobilecommon:map-build:1.8.0.170406170552:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:forerunner-build:1.0.0.170616150043@jar'
manifest 'com.alipay.android.phone.mobilesdk:forerunner-build:1.0.0.170616150043:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidsupportrecyclerview-build:7.23.2.170814173705@jar'
manifest 'com.alipay.android.phone.thirdparty:androidsupportrecyclerview-
build:7.23.2.170814173705:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:wire-build:1.5.3.571018171133@jar'
manifest 'com.alipay.android.phone.thirdparty:wire-build:1.5.3.571018171133:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:fastjson-build:1.1.66.171117210532@jar'
manifest 'com.alipay.android.phone.thirdparty:fastjson-build:1.1.66.171117210532:AndroidManifest@xml'

bundle 'com.alipay.mobileapi:mobileappcommon-build:1.0.0.20171204@jar'

bundle 'com.alipay.android.phone.thirdparty:utdid-build:2.1.1.171212154524@jar'
manifest 'com.alipay.android.phone.thirdparty:utdid-build:2.1.1.171212154524:AndroidManifest@xml'
```

```
bundle 'com.alipay.android.phone.mobilecommon:lbs-build:1.9.0.171219202112@jar'
manifest 'com.alipay.android.phone.mobilecommon:lbs-build:1.9.0.171219202112:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.171222165440@jar'
manifest 'com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.171222165440:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:falconlooks-build:1.6.64.171229002058@jar'
manifest 'com.alipay.android.phone.mobilecommon:falconlooks-build:1.6.64.171229002058:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:falconrecmanager-build:1.0.0.180102132516@jar'
manifest 'com.alipay.android.phone.mobilecommon:falconrecmanager-
build:1.0.0.180102132516:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:netsdkextdependapi-build:1.0.1.180102162159@jar'
manifest 'com.alipay.android.phone.mobilesdk:netsdkextdependapi-
build:1.0.1.180102162159:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:netsdkextdepend-build:1.0.1.180102162159@jar'
manifest 'com.alipay.android.phone.mobilesdk:netsdkextdepend-build:1.0.1.180102162159:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidannotations-build:9.6.6.180102221133@jar'
manifest 'com.alipay.android.phone.thirdparty:androidannotations-
build:9.6.6.180102221133:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:multimediaext-build:1.11.0.180103194218@jar'
manifest 'com.alipay.android.phone.mobilecommon:multimediaext-
build:1.11.0.180103194218:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:falcon-build:1.6.41.180123212839@jar'
manifest 'com.alipay.android.phone.mobilecommon:falcon-build:1.6.41.180123212839:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mpaas:tinyappcustom-build:1.0.0.180207204431@jar'
manifest 'com.alipay.android.phone.mpaas:tinyappcustom-build:1.0.0.180207204431:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:aspect-build:1.4.1.180212140351@jar'
manifest 'com.alipay.android.phone.mobilesdk:aspect-build:1.4.1.180212140351:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:storage-build:1.7.0.180308143113@jar'
manifest 'com.alipay.android.phone.mobilesdk:storage-build:1.7.0.180308143113:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:crypto-build:1.0.1.180309141623@jar'
manifest 'com.alipay.android.phone.mobilesdk:crypto-build:1.0.1.180309141623:AndroidManifest@xml'

bundle 'com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.180926192302@jar'
manifest 'com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.180926192302:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:commonservice-build:1.9.0.180319220933@jar'
manifest 'com.alipay.android.phone.mobilesdk:commonservice-build:1.9.0.180319220933:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:rpc-build:1.8.1.180320105342@jar'
manifest 'com.alipay.android.phone.mobilesdk:rpc-build:1.8.1.180320105342:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:transportext-build:1.8.1.180320110007@jar'
manifest 'com.alipay.android.phone.mobilesdk:transportext-build:1.8.1.180320110007:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:framework-build:2.5.7.180320154023@jar'
```

manifest 'com.alipay.android.phone.mobilesdk:framework-build:2.5.7.180320154023:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:autotracker-build:1.0.0.180320154841@jar'
manifest 'com.alipay.android.phone.mobilesdk:autotracker-build:1.0.0.180320154841:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:tianyanadapter-build:1.0.4.180320161437@jar'
manifest 'com.alipay.android.phone.mobilesdk:tianyanadapter-build:1.0.4.180320161437:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:multimedia-build:1.19.0.180320163427@jar'
manifest 'com.alipay.android.phone.mobilecommon:multimedia-build:1.19.0.180320163427:AndroidManifest@xml'

bundle 'com.alipay.multimedia.base:basic-build:1.19.0.180320163427@jar'
manifest 'com.alipay.multimedia.base:basic-build:1.19.0.180320163427:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:amap3dmap-build:3.3.3.180320163433@jar'
manifest 'com.alipay.android.phone.thirdparty:amap3dmap-build:3.3.3.180320163433:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:ui-build:10.1.18.180320200212@jar'
manifest 'com.alipay.android.phone.mobilecommon:ui-build:10.1.18.180320200212:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:mapbiz-build:1.8.0.180321115405@jar'
manifest 'com.alipay.android.phone.mobilecommon:mapbiz-build:1.8.0.180321115405:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:antui-build:10.1.20.180321211516@jar'
manifest 'com.alipay.android.phone.wallet:antui-build:10.1.20.180321211516:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulaucsdk-build:1.0.0.180322110653@jar'
manifest 'com.alipay.android.phone.wallet:nebulaucsdk-build:1.0.0.180322110653:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:logging-build:2.0.2.180322162837@jar'
manifest 'com.alipay.android.phone.mobilesdk:logging-build:2.0.2.180322162837:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebula-build:1.6.2.180322181148@jar'
manifest 'com.alipay.android.phone.wallet:nebula-build:1.6.2.180322181148:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:adaptermap-build:1.0.0.180324142935@jar'
manifest 'com.alipay.android.phone.mobilecommon:adaptermap-build:1.0.0.180324142935:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulaappcenter-build:1.6.2.180326120533@jar'
manifest 'com.alipay.android.phone.wallet:nebulaappcenter-build:1.6.2.180326120533:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:transport-build:1.8.1.181114200639@jar'
manifest 'com.alipay.android.phone.mobilesdk:transport-build:1.8.1.181114200639:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:dynamicrelease-build:2.4.2.180326164409@jar'
manifest 'com.alipay.android.phone.mobilecommon:dynamicrelease-build:2.4.2.180326164409:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:multimediabiz-build:1.19.0.180326192800@jar'
manifest 'com.alipay.android.phone.mobilecommon:multimediabiz-build:1.19.0.180326192800:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:common-build:1.8.2.180326200857@jar'
manifest 'com.alipay.android.phone.mobilesdk:common-build:1.8.2.180326200857:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:cloudfix-build:2.4.5.180327205120@jar'

```
manifest 'com.alipay.android.phone.mobilecommon:cloudfix-build:2.4.5.180327205120:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:quinox-build:2.5.5.180327225510@jar'
manifest 'com.alipay.android.phone.mobilesdk:quinox-build:2.5.5.180327225510:AndroidManifest@xml'

bundle 'com.alipay.android.phone.scancode:mascanengine-build:2.0.0.180402111033@jar'
manifest 'com.alipay.android.phone.scancode:mascanengine-build:2.0.0.180402111033:AndroidManifest@xml'

bundle 'com.alipay.android.phone.scancode:bqcscanservice-build:2.0.0.180402111033@jar'
manifest 'com.alipay.android.phone.scancode:bqcscanservice-build:2.0.0.180402111033:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:scanexport-build:1.0.0.181116172939@jar'
manifest 'com.alipay.android.phone.wallet:scanexport-build:1.0.0.181116172939:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulauc-build:1.6.0.180404103447@jar'
manifest 'com.alipay.android.phone.wallet:nebulauc-build:1.6.0.180404103447:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:apble-build:1.0.0.180408111111@jar'
manifest 'com.alipay.android.phone.wallet:apble-build:1.0.0.180408111111:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:liteprocess-build:1.0.0.180409142018@jar'
manifest 'com.alipay.android.phone.mobilesdk:liteprocess-build:1.0.0.180409142018:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulaappproxy-build:1.6.2.180413172504@jar'
manifest 'com.alipay.android.phone.wallet:nebulaappproxy-build:1.6.2.180413172504:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:beehive-build:10.1.22.180419184647@jar'
manifest 'com.alipay.android.phone.wallet:beehive-build:10.1.22.180419184647:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:tinyappcommon-build:1.0.0.180420181653@jar'
manifest 'com.alipay.android.phone.wallet:tinyappcommon-build:1.0.0.180420181653:AndroidManifest@xml'

bundle 'com.mpaas.mpaasadapter:commonbiz-build:1.0.0.180420210915@jar'
manifest 'com.mpaas.mpaasadapter:commonbiz-build:1.0.0.180420210915:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:monitor-build:2.1.4.180502135527@jar'
manifest 'com.alipay.android.phone.mobilesdk:monitor-build:2.1.4.180502135527:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:commonbizservice-build:1.7.0.180507114202@jar'
manifest 'com.alipay.android.phone.mobilesdk:commonbizservice-build:1.7.0.180507114202:AndroidManifest@xml'

bundle 'com.mpaas.nebula.adapter:mpaasnebulaadapter-build:10.1.20.190104175650@jar'
manifest 'com.mpaas.nebula.adapter:mpaasnebulaadapter-build:10.1.20.190104175650:AndroidManifest@xml'

bundle 'com.alipay.android.phone.sync:syncservice-build:2.0.0.180731121835@jar'
manifest 'com.alipay.android.phone.sync:syncservice-build:2.0.0.180731121835:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:securityguard-build:5.4.2008.171127213741@jar'
manifest 'com.alipay.android.phone.thirdparty:securityguard-build:5.4.2008.171127213741:AndroidManifest@xml'
```

4.1.2 基线 10.1.10

基线 是指一系列功能的稳定版本，是进一步开发的基础。对于 mPaaS，基线是 mPaaS 的 SDK 列表。由于 mPaaS 产品是基于支付宝的某个特定版本开发的，因此基于该版本的 SDK 集合称之为基线。

随着 mPaaS 产品的不断升级，会出现多个版本的基线。本文介绍 **10.1.10** 基线，该基线是基于支付宝 **10.1.10**

版本。相较于 **10.0.18** 版本，**10.1.10** 基线增加了以下特性：

- 小程序
- 热修复支持 V8.1 系统

基线列表

模块	Bundle	备注
框架	com.alipay.android.phone.mobilesdk:framework-build:2.5.6.180108111953	
	com.alipay.android.phone.mobilesdk:quinox-build:2.5.1.180108105517	
	com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.180312175325	
基础服务	com.alipay.android.phone.thirdparty:nineoldandroids-build:2.4.0.151028164451	
	com.alipay.android.phone.thirdparty:androidsupportpalette-build:7.22	
	com.alipay.android.phone.thirdparty:androidannotations-build:9.6.6.160512173203	
	com.alipay.android.phone.thirdparty:androidsupportrecyclerview-build:7.23.2.170814173705	
	com.alipay.mobileapi:mobileapp-build:1.0.0.20170307	
	com.alipay.android.phone.thirdparty:fastjson-build:1.1.66.171117210532	
	com.alipay.mobileapi:mobileappcommon-build:1.0.0.20171204	
	com.alipay.android.phone.thirdparty:utdid-build:2.1.1.171212154524	
	com.alipay.android.phone.wallet:antui-build:10.1.10.171214213652	
	com.alipay.android.phone.mobilesdk:storage-build:1.7.0.171219153847	
	com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.171222165440	
	com.alipay.android.phone.mobilesdk:commonservice-build:1.9.0.180104111219	
	com.mpaas.mpaasadapter:commonbiz-build:1.0.0.180105115556	
	com.alipay.android.phone.mobilecommon:ui-build:10.1.0.180108153750	
	com.alipay.android.phone.mobilesdk:common-build:1.8.2.180108165108	
	com.alipay.android.phone.mobilesdk:commonbizservice-build:1.7.0.180505084216	
多媒体	com.alipay.android.phone.wallet:basicstl-build:1.0.0.161020105915	
	com.alipay.android.phone.mobilecommon:multimediaext-build:1.11.0.171212162356	
	com.alipay.android.phone.mobilecommon:multimedia-build:1.19.0.180115174539	
	com.alipay.android.phone.mobilecommon:multimediabiz-build:1.19.0.180115193820	
	com.alipay.multimedia.base:basic-build:1.19.0.180118202741	
Sync	com.alipay.android.phone.rome:syncmpaasapi-build:1.3.5.160714203203	
	com.alipay.android.phone.sync:syncservice-build:2.0.0.180731121835	
日志监控	com.alipay.android.phone.mobilesdk:aspect-build:1.4.1.161220194837	
	com.alipay.android.phone.mobilesdk:forerunner-build:1.0.0.170616150043	
	com.alipay.android.phone.mobilesdk:autotracker-build:1.0.0.171117105342	
	com.alipay.android.phone.mobilesdk:tianyanadapter-build:1.0.4.171220171704	

	com.alipay.android.phone.mobilesdk:monitor-build:2.1.4.180427205526	
	com.alipay.android.phone.mobilesdk:logging-build:2.0.2.180504235416	
扫码	com.alipay.android.phone.scancode:mascanengine-build:2.0.0.180402111033	
	com.alipay.android.phone.scancode:bqcscanservice-build:2.0.0.180402111033	
	com.alipay.android.phone.wallet:scanexport-build:1.0.0.181116172939	
移动网关	com.alipay.android.phone.thirdparty:wire-build:1.5.3.571018171133	
	com.alipay.android.phone.mobilesdk:netsdkextdepend-build:1.0.0.171207214345	
	com.alipay.android.phone.mobilesdk:netsdkextdependapi-build:1.0.0.171207214345	
	com.alipay.android.phone.mobilesdk:transportext-build:1.7.8.180108115957	
	com.alipay.android.phone.mobilesdk:crypto-build:1.1.0.180307114229	
	com.alipay.android.phone.mobilesdk:transport-build:1.7.6.180613192713	
	com.alipay.android.phone.mobilesdk:rpc-build:1.7.8.180705171115	
H5容器	com.alipay.android.phone.wallet:nebula-build:1.6.2.180104154031	
	com.alipay.android.phone.wallet:nebulauc-build:1.0.0.180104154045	
	com.alipay.android.phone.wallet:nebulaappcenter-build:1.6.2.180106142052	
	com.alipay.android.phone.wallet:nebulacore-build:1.6.0.180112123348	
	com.mpaas.nebula.adapter:mpaasnebulaadapter-build:10.1.10.180614165059	
小程序	com.alipay.android.phone.mobilecommon:falcon-build:1.6.38.171207151140	
	com.alipay.android.phone.mobilecommon:falconlooks-build:1.6.64.171207154512	
	com.alipay.android.phone.wallet:beehive-build:10.1.10.171214105858	
	com.alipay.android.phone.thirdparty:amap3dmap-build:3.3.3.171219202112	
	com.alipay.android.phone.mobilecommon:mapbiz-build:1.0.0.171219202651	
	com.alipay.android.phone.wallet:apble-build:1.0.0.171220115746	
	com.alipay.android.phone.mobilecommon:map-build:1.0.0.171219202651	
	com.alipay.android.phone.mobilecommon:adaptermap-build:1.0.0.180102105934	
	com.alipay.android.phone.mpaas:tinyappcustom-build:1.0.0.180108175103	
	com.alipay.android.phone.wallet:tinyappcommon-build:1.0.0.180119172154	
热修复	com.alipay.android.phone.mobilecommon:dynamicrelease-build:2.3.7.171214184817	
	com.alipay.android.phone.mobilecommon:cloudfix-build:2.4.5.171222001944	
定位	com.alipay.android.phone.mobilecommon:lbs-build:1.9.0.171219202112	
无线保镖	com.alipay.android.phone.thirdparty:securityguard-build:5.4.2008.171127213741	

注意：消息推送，升级，分享组件的基线请参考各组件的使用文档配置

```
bundle 'com.alipay.android.phone.thirdparty:nineoldandroids-build:2.4.0.151028164451@jar'
manifest 'com.alipay.android.phone.thirdparty:nineoldandroids-build:2.4.0.151028164451:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidsupportpalette-build:7.22@jar'
```

manifest 'com.alipay.android.phone.thirdparty:androidsupportpalette-build:7.22:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidannotations-build:9.6.6.160512173203@jar'
manifest 'com.alipay.android.phone.thirdparty:androidannotations-build:9.6.6.160512173203:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidsupportrecyclerview-build:7.23.2.170814173705@jar'
manifest 'com.alipay.android.phone.thirdparty:androidsupportrecyclerview-build:7.23.2.170814173705:AndroidManifest@xml'

bundle 'com.alipay.android.phone.rome:syncmpaasapi-build:1.3.5.160714203203@jar'
manifest 'com.alipay.android.phone.rome:syncmpaasapi-build:1.3.5.160714203203:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:basicstl-build:1.0.0.161020105915@jar'
manifest 'com.alipay.android.phone.wallet:basicstl-build:1.0.0.161020105915:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:aspect-build:1.4.1.161220194837@jar'
manifest 'com.alipay.android.phone.mobilesdk:aspect-build:1.4.1.161220194837:AndroidManifest@xml'

bundle 'com.alipay.mobileapi:mobileapp-build:1.0.0.20170307@jar'

bundle 'com.alipay.android.phone.scancode:bqcscanservice-build:2.0.0.170612143649@jar'
manifest 'com.alipay.android.phone.scancode:bqcscanservice-build:2.0.0.170612143649:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:forerunner-build:1.0.0.170616150043@jar'
manifest 'com.alipay.android.phone.mobilesdk:forerunner-build:1.0.0.170616150043:AndroidManifest@xml'

bundle 'com.alipay.android.phone.scancode:mascanengine-build:2.0.0.170616184613@jar'
manifest 'com.alipay.android.phone.scancode:mascanengine-build:2.0.0.170616184613:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:wire-build:1.5.3.571018171133@jar'
manifest 'com.alipay.android.phone.thirdparty:wire-build:1.5.3.571018171133:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:autotracker-build:1.0.0.171117105342@jar'
manifest 'com.alipay.android.phone.mobilesdk:autotracker-build:1.0.0.171117105342:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:fastjson-build:1.1.66.171117210532@jar'
manifest 'com.alipay.android.phone.thirdparty:fastjson-build:1.1.66.171117210532:AndroidManifest@xml'

bundle 'com.alipay.android.phone.rome:pushsdk-build:1.7.0.171130154840@jar'
manifest 'com.alipay.android.phone.rome:pushsdk-build:1.7.0.171130154840:AndroidManifest@xml'

bundle 'com.alipay.mobileapi:mobileappcommon-build:1.0.0.20171204@jar'

bundle 'com.alipay.android.phone.mobilecommon:falcon-build:1.6.38.171207151140@jar'
manifest 'com.alipay.android.phone.mobilecommon:falcon-build:1.6.38.171207151140:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:falconlooks-build:1.6.64.171207154512@jar'
manifest 'com.alipay.android.phone.mobilecommon:falconlooks-build:1.6.64.171207154512:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:netsdkextdepend-build:1.0.0.171207214345@jar'
manifest 'com.alipay.android.phone.mobilesdk:netsdkextdepend-build:1.0.0.171207214345:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:netsdkextdependapi-build:1.0.0.171207214345@jar'
manifest 'com.alipay.android.phone.mobilesdk:netsdkextdependapi-build:1.0.0.171207214345:AndroidManifest@xml'

```
bundle 'com.alipay.android.phone.thirdparty:utdid-build:2.1.1.171212154524@jar'
manifest 'com.alipay.android.phone.thirdparty:utdid-build:2.1.1.171212154524:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:multimediaext-build:1.11.0.171212162356@jar'
manifest 'com.alipay.android.phone.mobilecommon:multimediaext-
build:1.11.0.171212162356:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:beehive-build:10.1.10.171214105858@jar'
manifest 'com.alipay.android.phone.wallet:beehive-build:10.1.10.171214105858:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:dynamicrelease-build:2.3.7.171214184817@jar'
manifest 'com.alipay.android.phone.mobilecommon:dynamicrelease-
build:2.3.7.171214184817:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:antui-build:10.1.10.171214213652@jar'
manifest 'com.alipay.android.phone.wallet:antui-build:10.1.10.171214213652:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:storage-build:1.7.0.171219153847@jar'
manifest 'com.alipay.android.phone.mobilesdk:storage-build:1.7.0.171219153847:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:lbs-build:1.9.0.171219202112@jar'
manifest 'com.alipay.android.phone.mobilecommon:lbs-build:1.9.0.171219202112:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:amap3dmap-build:3.3.3.171219202112@jar'
manifest 'com.alipay.android.phone.thirdparty:amap3dmap-build:3.3.3.171219202112:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:mapbiz-build:1.0.0.171219202651@jar'
manifest 'com.alipay.android.phone.mobilecommon:mapbiz-build:1.0.0.171219202651:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:map-build:1.0.0.171219202651@jar'
manifest 'com.alipay.android.phone.mobilecommon:map-build:1.0.0.171219202651:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:apble-build:1.0.0.171220115746@jar'
manifest 'com.alipay.android.phone.wallet:apble-build:1.0.0.171220115746:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:tianyanadapter-build:1.0.4.171220171704@jar'
manifest 'com.alipay.android.phone.mobilesdk:tianyanadapter-build:1.0.4.171220171704:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:cloudfix-build:2.4.5.171222001944@jar'
manifest 'com.alipay.android.phone.mobilecommon:cloudfix-build:2.4.5.171222001944:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.171222165440@jar'
manifest 'com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.171222165440:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:adaptermap-build:1.0.0.180102105934@jar'
manifest 'com.alipay.android.phone.mobilecommon:adaptermap-build:1.0.0.180102105934:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:commonservice-build:1.9.0.180104111219@jar'
manifest 'com.alipay.android.phone.mobilesdk:commonservice-build:1.9.0.180104111219:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebula-build:1.6.2.180104154031@jar'
manifest 'com.alipay.android.phone.wallet:nebula-build:1.6.2.180104154031:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulauc-build:1.0.0.180104154045@jar'
manifest 'com.alipay.android.phone.wallet:nebulauc-build:1.0.0.180104154045:AndroidManifest@xml'
```

```
bundle 'com.mpaas.mpaasadapter:commonbiz-build:1.0.0.180105115556@jar'
manifest 'com.mpaas.mpaasadapter:commonbiz-build:1.0.0.180105115556:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulaappcenter-build:1.6.2.180106142052@jar'
manifest 'com.alipay.android.phone.wallet:nebulaappcenter-build:1.6.2.180106142052:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:quinox-build:2.5.1.180108105517@jar'
manifest 'com.alipay.android.phone.mobilesdk:quinox-build:2.5.1.180108105517:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:framework-build:2.5.6.180108111953@jar'
manifest 'com.alipay.android.phone.mobilesdk:framework-build:2.5.6.180108111953:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:transportext-build:1.7.8.180108115957@jar'
manifest 'com.alipay.android.phone.mobilesdk:transportext-build:1.7.8.180108115957:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:ui-build:10.1.0.180108153750@jar'
manifest 'com.alipay.android.phone.mobilecommon:ui-build:10.1.0.180108153750:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:common-build:1.8.2.180108165108@jar'
manifest 'com.alipay.android.phone.mobilesdk:common-build:1.8.2.180108165108:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mpaas:tinyappcustom-build:1.0.0.180108175103@jar'
manifest 'com.alipay.android.phone.mpaas:tinyappcustom-build:1.0.0.180108175103:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulacore-build:1.6.0.180112123348@jar'
manifest 'com.alipay.android.phone.wallet:nebulacore-build:1.6.0.180112123348:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:multimedia-build:1.19.0.180115174539@jar'
manifest 'com.alipay.android.phone.mobilecommon:multimedia-build:1.19.0.180115174539:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:multimediabiz-build:1.19.0.180115193820@jar'
manifest 'com.alipay.android.phone.mobilecommon:multimediabiz-
build:1.19.0.180115193820:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:scanexport-build:1.0.0.181116172939@jar'
manifest 'com.alipay.android.phone.wallet:scanexport-build:1.0.0.181116172939:AndroidManifest@xml'

bundle 'com.alipay.multimedia.base:basic-build:1.19.0.180118202741@jar'
manifest 'com.alipay.multimedia.base:basic-build:1.19.0.180118202741:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:tinyappcommon-build:1.0.0.180119172154@jar'
manifest 'com.alipay.android.phone.wallet:tinyappcommon-build:1.0.0.180119172154:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:crypto-build:1.1.0.180307114229@jar'
manifest 'com.alipay.android.phone.mobilesdk:crypto-build:1.1.0.180307114229:AndroidManifest@xml'

bundle 'com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.180312175325@jar'
manifest 'com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.180312175325:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:securityguard-build:5.4.2008.171127213741@jar'
manifest 'com.alipay.android.phone.thirdparty:securityguard-build:5.4.2008.171127213741:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:monitor-build:2.1.4.180427205526@jar'
manifest 'com.alipay.android.phone.mobilesdk:monitor-build:2.1.4.180427205526:AndroidManifest@xml'
```

```
bundle 'com.alipay.android.phone.mobilesdk:logging-build:2.0.2.180504235416@jar'
manifest 'com.alipay.android.phone.mobilesdk:logging-build:2.0.2.180504235416:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:commonbizservice-build:1.7.0.180505084216@jar'
manifest 'com.alipay.android.phone.mobilesdk:commonbizservice-build:1.7.0.180505084216:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:transport-build:1.7.6.180613192713@jar'
manifest 'com.alipay.android.phone.mobilesdk:transport-build:1.7.6.180613192713:AndroidManifest@xml'

bundle 'com.mpaas.nebula.adapter:mpaasnebulaadapter-build:10.1.10.180614165059@jar'
manifest 'com.mpaas.nebula.adapter:mpaasnebulaadapter-build:10.1.10.180614165059:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:rpc-build:1.7.8.180705171115@jar'
manifest 'com.alipay.android.phone.mobilesdk:rpc-build:1.7.8.180705171115:AndroidManifest@xml'

bundle 'com.alipay.android.phone.sync:syncservice-build:2.0.0.180731121835@jar'
manifest 'com.alipay.android.phone.sync:syncservice-build:2.0.0.180731121835:AndroidManifest@xml'
```

4.1.3 基线 10.0.18

基线 是指一系列功能的稳定版本，是进一步开发的基础。对于 mPaaS，基线是 mPaaS 的 SDK 列表。由于 mPaaS 产品是基于支付宝的某个特定版本开发的，因此基于该版本的 SDK 集合称之为基线。

随着 mPaaS 产品的不断升级，会出现多个版本的基线。本文介绍 **10.0.18** 基线，该基线是基于支付宝 **10.0.18** 版本，提供以下功能：

- 埋点日志
- 热修复
- 应用升级
- 扫码
- 定位
- H5容器
- RPC网络组件
- 社交分享
- 数据同步（Sync）
- 消息推送
- 存储组件

基线信息

模块	Bundle	说明
框架	com.alipay.android.phone.mobilesdk:framework-build:2.5.0.170621123127	
	com.alipay.android.phone.mobilesdk:quinox-build:2.3.6.171214183540	
	com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.171231135940	
基础服务	com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.161114144707	
	com.alipay.android.phone.thirdparty:androidannotations-build:9.6.6.160512173203	

	com.alipay.android.phone.thirdparty:androidsupportpalette-build:7.22	
	com.alipay.mobileapi:mobileapp-build:1.0.0.20170307	
	com.alipay.android.phone.thirdparty:utdid-build:1.1.5.170410144530	
	com.alipay.android.phone.mobilesdk:common-build:1.8.2.170509174053	
	com.alipay.mobileapi:mobileappcommon-build:1.0.0.20170519	
	com.alipay.android.phone.mobilesdk:storage-build:1.7.0.170609123631	
	com.alipay.android.phone.mobilesdk:commonservice-build:1.9.0.170613221028	
	com.alipay.android.phone.mobilesdk:commonbizservice-build:1.7.0.171020173544	
	com.alipay.android.phone.thirdparty:fastjson-build:1.1.65.171103193330	
日志监控	com.alipay.android.phone.mobilesdk:aspect-build:1.4.1.161220194837	
	com.alipay.android.phone.mobilesdk:forerunner-build:1.0.0.170616150043	
	com.alipay.android.phone.mobilesdk:monitor-build:2.1.0.170620210603	
	com.alipay.android.phone.mobilesdk:autotracker-build:1.0.0.171221144233	
	com.alipay.android.phone.mobilesdk:logging-build:2.0.2.171023152737	
	com.alipay.android.phone.mobilesdk:tianyanadapter-build:1.0.4.171230174611	
Sync	com.alipay.android.phone.sync:syncservice-build:2.0.0.170418193042	
移动网关	com.alipay.android.phone.thirdparty:wire-build:1.5.3.370506012228	
	com.alipay.android.phone.mobilesdk:transportext-build:1.7.3.170612110614	
	com.alipay.android.phone.mobilesdk:rpc-build:1.7.6.170908160217	
	com.alipay.android.phone.mobilesdk:transport-build:1.7.6.180307114046	
	com.alipay.android.phone.mobilesdk:crypto-build:1.1.0.180307114229	
多媒体	com.alipay.android.phone.mobilecommon:multimedia-build:1.19.0.170605144942	
扫码	com.alipay.android.phone.scancode:mascanengine-build:2.0.0.180402111033	
	com.alipay.android.phone.scancode:bqcscanservice-build:2.0.0.180402111033	
	com.alipay.android.phone.wallet:scanexport-build:1.0.0.181116172939	
定位	com.alipay.android.phone.mobilecommon:lbs-build:1.9.0.170621020655	
H5容器	com.alipay.android.phone.wallet:nebulaconfig-build:1.6.0.170915172454	
	com.alipay.android.phone.wallet:nebulaappcenter-build:1.6.2.171122164541	
	com.mpaas.nebula.adapter:mpaasnebulaadapter-build:1.0.0.171128150408	
	com.alipay.android.phone.wallet:nebulabiz-build:1.6.0.171215214205	
	com.alipay.android.phone.wallet:nebula-build:1.6.2.171225124652	
	com.alipay.android.phone.wallet:nebulacore-build:1.6.0.180321155244	
	com.alipay.android.phone.wallet:nebulauc-build:1.6.0.180611134305	
热修复	com.alipay.android.phone.mobilecommon:dynamicrelease-build:2.3.3.171120200824	
	com.alipay.android.phone.mobilecommon:cloudfix-build:2.4.3.171129174442	
无线保镖	com.alipay.android.phone.thirdparty:securityguard-build:5.4.2008.171127213741	

注意：消息推送，升级，分享组件的基线请参考各组件的使用文档配置

```
bundle 'com.alipay.android.phone.thirdparty:androidsupportpalette-build:7.22@jar'
manifest 'com.alipay.android.phone.thirdparty:androidsupportpalette-build:7.22:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidannotations-build:9.6.6.160512173203@jar'
manifest 'com.alipay.android.phone.thirdparty:androidannotations-
build:9.6.6.160512173203:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.161114144707@jar'
manifest 'com.alipay.android.phone.thirdparty:androidsupport-build:13.23.2.161114144707:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:aspect-build:1.4.1.161220194837@jar'
manifest 'com.alipay.android.phone.mobilesdk:aspect-build:1.4.1.161220194837:AndroidManifest@xml'

bundle 'com.alipay.mobileapi:mobileapp-build:1.0.0.20170307@jar'

bundle 'com.alipay.android.phone.thirdparty:utdid-build:1.1.5.170410144530@jar'
manifest 'com.alipay.android.phone.thirdparty:utdid-build:1.1.5.170410144530:AndroidManifest@xml'

bundle 'com.alipay.android.phone.sync:syncservice-build:2.0.0.170418193042@jar'
manifest 'com.alipay.android.phone.sync:syncservice-build:2.0.0.170418193042:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:wire-build:1.5.3.370506012228@jar'
manifest 'com.alipay.android.phone.thirdparty:wire-build:1.5.3.370506012228:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:common-build:1.8.2.170509174053@jar'
manifest 'com.alipay.android.phone.mobilesdk:common-build:1.8.2.170509174053:AndroidManifest@xml'

bundle 'com.alipay.mobileapi:mobileappcommon-build:1.0.0.20170519@jar'

bundle 'com.alipay.android.phone.mobilecommon:multimedia-build:1.19.0.170605144942@jar'
manifest 'com.alipay.android.phone.mobilecommon:multimedia-build:1.19.0.170605144942:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:storage-build:1.7.0.170609123631@jar'
manifest 'com.alipay.android.phone.mobilesdk:storage-build:1.7.0.170609123631:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:transportext-build:1.7.3.170612110614@jar'
manifest 'com.alipay.android.phone.mobilesdk:transportext-build:1.7.3.170612110614:AndroidManifest@xml'

bundle 'com.alipay.android.phone.scancode:mascanengine-build:2.0.0.180402111033@jar'
manifest 'com.alipay.android.phone.scancode:mascanengine-build:2.0.0.180402111033:AndroidManifest@xml'

bundle 'com.alipay.android.phone.scancode:bqcscanservice-build:2.0.0.180402111033@jar'
manifest 'com.alipay.android.phone.scancode:bqcscanservice-build:2.0.0.180402111033:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:scanexport-build:1.0.0.181116172939@jar'
manifest 'com.alipay.android.phone.wallet:scanexport-build:1.0.0.181116172939:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:commonservice-build:1.9.0.170613221028@jar'
manifest 'com.alipay.android.phone.mobilesdk:commonservice-build:1.9.0.170613221028:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:forerunner-build:1.0.0.170616150043@jar'
manifest 'com.alipay.android.phone.mobilesdk:forerunner-build:1.0.0.170616150043:AndroidManifest@xml'
```

```
bundle 'com.alipay.android.phone.mobilesdk:monitor-build:2.1.0.170620210603@jar'
manifest 'com.alipay.android.phone.mobilesdk:monitor-build:2.1.0.170620210603:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:lbs-build:1.9.0.170621020655@jar'
manifest 'com.alipay.android.phone.mobilecommon:lbs-build:1.9.0.170621020655:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:framework-build:2.5.0.170621123127@jar'
manifest 'com.alipay.android.phone.mobilesdk:framework-build:2.5.0.170621123127:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:rpc-build:1.7.6.170908160217@jar'
manifest 'com.alipay.android.phone.mobilesdk:rpc-build:1.7.6.170908160217:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulaconfig-build:1.6.0.170915172454@jar'
manifest 'com.alipay.android.phone.wallet:nebulaconfig-build:1.6.0.170915172454:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:commonbizservice-build:1.7.0.171020173544@jar'
manifest 'com.alipay.android.phone.mobilesdk:commonbizservice-build:1.7.0.171020173544:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:logging-build:2.0.2.171023152737@jar'
manifest 'com.alipay.android.phone.mobilesdk:logging-build:2.0.2.171023152737:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:fastjson-build:1.1.65.171103193330@jar'
manifest 'com.alipay.android.phone.thirdparty:fastjson-build:1.1.65.171103193330:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:dynamicrelease-build:2.3.3.171120200824@jar'
manifest 'com.alipay.android.phone.mobilecommon:dynamicrelease-
build:2.3.3.171120200824:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulaappcenter-build:1.6.2.171122164541@jar'
manifest 'com.alipay.android.phone.wallet:nebulaappcenter-build:1.6.2.171122164541:AndroidManifest@xml'

bundle 'com.mpaas.nebula.adapter:mpaasnebulaadapter-build:1.0.0.171128150408@jar'
manifest 'com.mpaas.nebula.adapter:mpaasnebulaadapter-build:1.0.0.171128150408:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilecommon:cloudfix-build:2.4.3.171129174442@jar'
manifest 'com.alipay.android.phone.mobilecommon:cloudfix-build:2.4.3.171129174442:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:quinox-build:2.3.6.171214183540@jar'
manifest 'com.alipay.android.phone.mobilesdk:quinox-build:2.3.6.171214183540:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulabiz-build:1.6.0.171215214205@jar'
manifest 'com.alipay.android.phone.wallet:nebulabiz-build:1.6.0.171215214205:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:autotracker-build:1.0.0.171221144233@jar'
manifest 'com.alipay.android.phone.mobilesdk:autotracker-build:1.0.0.171221144233:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebula-build:1.6.2.171225124652@jar'
manifest 'com.alipay.android.phone.wallet:nebula-build:1.6.2.171225124652:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:tianyanadapter-build:1.0.4.171230174611@jar'
manifest 'com.alipay.android.phone.mobilesdk:tianyanadapter-build:1.0.4.171230174611:AndroidManifest@xml'

bundle 'com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.171231135940@jar'
manifest 'com.mpaas.mpaasadapter:mpaasadapter-build:1.0.0.171231135940:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:transport-build:1.7.6.180307114046@jar'
```

```
manifest 'com.alipay.android.phone.mobilesdk:transport-build:1.7.6.180307114046:AndroidManifest@xml'

bundle 'com.alipay.android.phone.mobilesdk:crypto-build:1.1.0.180307114229@jar'
manifest 'com.alipay.android.phone.mobilesdk:crypto-build:1.1.0.180307114229:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulacore-build:1.6.0.180321155244@jar'
manifest 'com.alipay.android.phone.wallet:nebulacore-build:1.6.0.180321155244:AndroidManifest@xml'

bundle 'com.alipay.android.phone.thirdparty:securityguard-build:5.4.2008.171127213741@jar'
manifest 'com.alipay.android.phone.thirdparty:securityguard-build:5.4.2008.171127213741:AndroidManifest@xml'

bundle 'com.alipay.android.phone.wallet:nebulauc-build:1.6.0.180611134305@jar'
manifest 'com.alipay.android.phone.wallet:nebulauc-build:1.6.0.180611134305:AndroidManifest@xml'
```

4.2 代码示例

本代码示例基于 mPaaS 开发框架开发，集成了多个组件，可以参考相关的代码片段来了解框架和组件的用法。

有关代码示例的下载地址和使用方法，查看 获取代码示例 。

4.3 管理 Gradle 依赖

4.3.1 配置依赖仓库

Gradle 提供配置依赖仓库的功能。mPaaS 常见依赖仓库示例如下：

```
allprojects {
repositories {
mavenLocal()
flatDir {
dirs 'libs'
}
maven {
credentials {
username"*******"
password"*******"
}
url"http://mvn.cloud.alipay.com/nexus/content/repositories/releases/"
}
maven{ url 'http://maven.aliyun.com/nexus/content/groups/public/'}
maven {url 'http://maven.aliyun.com/nexus/content/repositories/google'}
}
}
```

- **mavenLocal**：Maven 本地仓库。您可以 **修改本地仓库的路径**，更多信息请参见 Local Maven repository。
- **flatDir**：工程 libs 目录下的依赖。关于 flatDir 类型仓库的更多信息，请参见 Flat directory repository。
- **maven**：示例中包含 蚂蚁金服金融科技（mvn.cloud.alipay.com）和 阿里云（maven.aliyun.com）的 Maven 仓库。关于 maven 类型仓库的更多信息，请参见 Custom Maven repositories。

您可以在 repositories 下 **新增依赖仓库**。更多信息，请参见 Repository Types。

4.3.2 配置发布仓库

Gradle 提供配置发布仓库的功能。本文将简述发布仓库常见示例，帮助您修改本地 Maven 仓库路径（默认 ~/.m2）、增加自定义发布仓库。

发布仓库示例

一般地，build.gradle 文件中有如下配置：

```
uploadArchives {
repositories {
mavenLocal()
}
}
```

这意味着发布仓库为 **本地 Maven 仓库**，即工程打出的 .jar 包等会自动发布到本地 Maven 仓库。

修改本地 Maven 仓库路径

本地 Maven 仓库（mavenLocal）默认路径为 ~/.m2，您可以自定义修改。更多信息请参见 Local Maven repository。

自定义发布仓库

您可以根据实际情况增加自定义发布仓库，示例如下：

```
uploadArchives {
mavenDeployer {
mavenLocal()
repository(url:"your_repository_url") {
authentication(userName: '*****', password: '*****')
}
snapshotRepository(url:"your_repository_url") {
authentication(userName: '*****', password: '*****')
}
}
}
```

5 常见问题

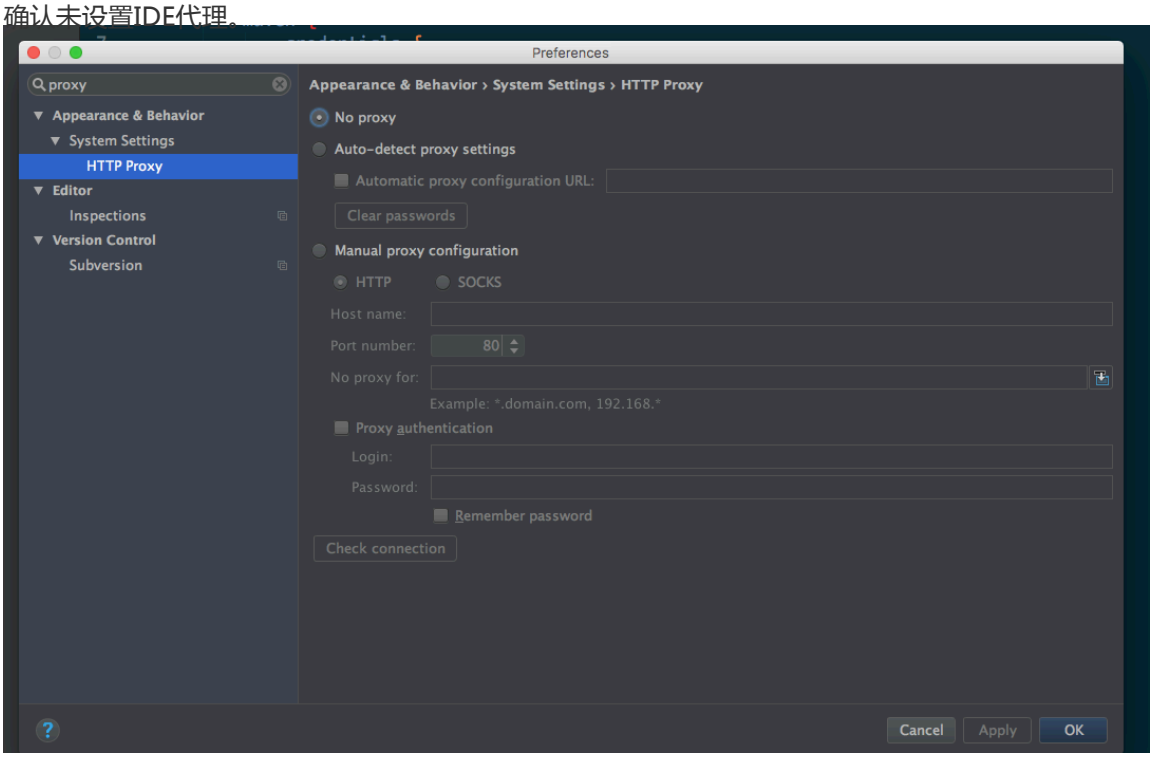
5.1 编译失败或卡顿

在编译打包 Android 文件时，您可能遇到无网络连接、编译失败、编译卡顿等问题。本文将针对这些问题向您介绍排错步骤与解决方法。

无网络连接

在编译文件时，如果没有网络，很有可能造成编译失败。通过以下步骤，确认编译环境的网络已连接：

1. 确认已连接到互联网。
2. 确认未连接网络代理，包括浏览器代理设置、第三方网络代理软件等。
3. 确认未设置IDE代理。



4. 在 gradle.properties 文件中，确认未设置 Gradle 代理，即未设置 **systemProp.http.proxyHost** 和 **systemProp.http.proxyPort** 属性。如果有设置，删除相关属性即可。

编译失败

如果程序编译失败，可通过以下步骤进行排错与解决：

1. 根据 上文步骤 ，确认编译环境网络已正常连接。
2. 检查 Gradle 执行记录，确认新增的依赖有效。
3. 检查依赖的 GAV（group, artifact, version）参数设置正确。

```
//引用 debug 包group:artifact:version:raw@jar
bundle"com.app.xxxxx.xxxx:test-build:1.0-SNAPSHOT:raw@jar"
//引用 release 包group:artifact:version@jar
bundle"com.app.xxxxx.xxxx:test-build:1.0-SNAPSHOT@jar"
manifest"com.app.xxxxx.xxxx:test-build:1.0-SNAPSHOT:AndroidManifest@xml"
```

4. 在系统自带的命令行工具中，执行以下命令，导出 Gradle 执行记录：

```
// 执行命令前，确认未定义 productflavor 属性。否则，命令会运行失败。
// 以下命令将执行记录导出至 log.txt 文件中。
gradle buildDebug --info --debug -Plog=true > log.txt
```

5. 查看步骤 4 中导出的记录文件，在最新生成的记录中，会看到类似如下记录，表示新增的依赖不存在：

```
Caused by: org.gradle.internal.resolve.ArtifactNotFoundException: Could not find nebulacore-build-AndroidManifest.xml (com.alipay.android.phone.wallet:nebulacore-build:1.6.0.171211174825).
Searched in the following locations:
http://mvn.cloud.alipay.com/nexus/content/repositories/releases/com/alipay/android/phone/wallet/nebulacore-build/1.6.0.171211174825/nebulacore-build-1.6.0.171211174825-AndroidManifest.xml
at org.gradle.internal.resolve.result.DefaultBuildableArtifactResolveResult.notFound(DefaultBuildableArtifactResolveResult.java:38)
at org.gradle.api.internal.artifacts.ivyservice.ivyresolve.CachingModuleComponentRepository$LocateInCacheRepositoryAccess.resolveArtifactFromCache(CachingModuleComponentRepository.java:260)
```

6. 访问该记录中的 http 链接（如上一步所列记录中的第 3 行）并登陆，查看 Maven 仓库。

   说明：您可以在 build.gradle 文件中查看登录时需要提供的账户名和密码。

7. 执行以下命令刷新 gradle 缓存：

```
gradle clean --refresh-dependencies
```

8. 如果 Maven 仓库有对应依赖，删除个人目录下 Gradle 缓存，然后重新编译。

   说明：Gradle 缓存的删除方法：

   - 在 MacOS、Linux、Unix等系统中，运行以下命令：

```
cd ~
cd .gradle
cd caches
rm -rf modules-2
```

- 在 Windows 系统中，默认情况下，路径定位到 C:\Users\{用户名}\.gradle\caches，删除 modules-2 文件夹。

编译卡顿

如果编译过程卡顿（等待超过 20 分钟），您可以通过以下步骤提高编译效率：

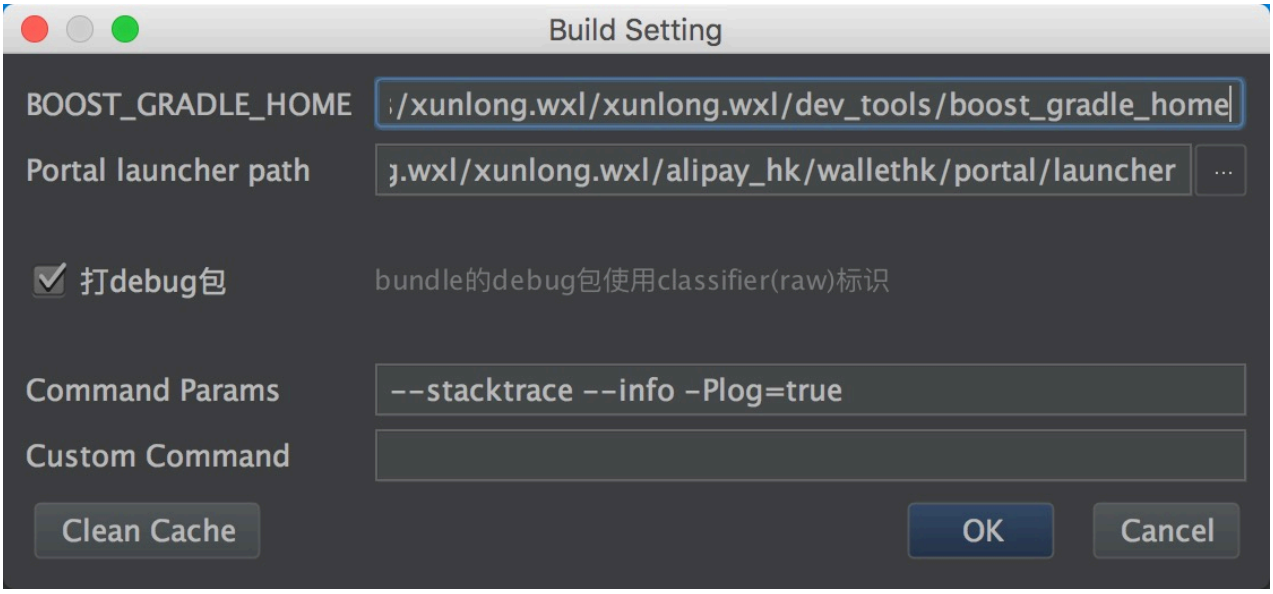
1. 根据 上文步骤 ，确认编译环境网络已正常连接。
2. 确认防火墙已关闭。
3. 确认未开启 IntelliJ IDEA 编译器的网络配置。
4. 在代码中，提前加载 Maven 镜像。例如，以下是阿里云加载 Maven 镜像的代码：

```
apply plugin: 'maven'
buildscript {
repositories {
mavenLocal()

// 开始先加载 Maven 镜像
maven{ url 'http://maven.aliyun.com/nexus/content/groups/public/'}

maven {
credentials {
username"请使用已知用户"
password"请使用已知密码"
}
url"http://mvn.cloud.alipay.com/nexus/content/repositories/releases/"
}
}
dependencies {
classpath 'com.android.tools.build:gradle:2.1.3'
classpath 'com.alipay.android:android-gradle-plugin:2.1.3.3.3'
classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
}
}
allprojects {
repositories {
flatDir {
dirs 'libs'
}
mavenLocal()
maven {
credentials {
username"xxxxxxxxx"
password"xxxxxxx"
```

```
}
url"http://mvn.cloud.alipay.com/nexus/content/repositories/releases/"
}
maven{ url 'http://maven.aliyun.com/nexus/content/groups/public/'}
}
}
```

5.2 如何清除 Gradle 缓存

打开 Gradle 插件的设置界面，点击 **Clean Cache** 按钮，即可删除 Gradle 插件的所有缓存数据。



5.3 如何调试应用

开发过程中需要调试代码，本文介绍两种调试方式：

- 以调试模式启动应用
- 应用运行后调试

以调试模式启动应用

- **使用场景**

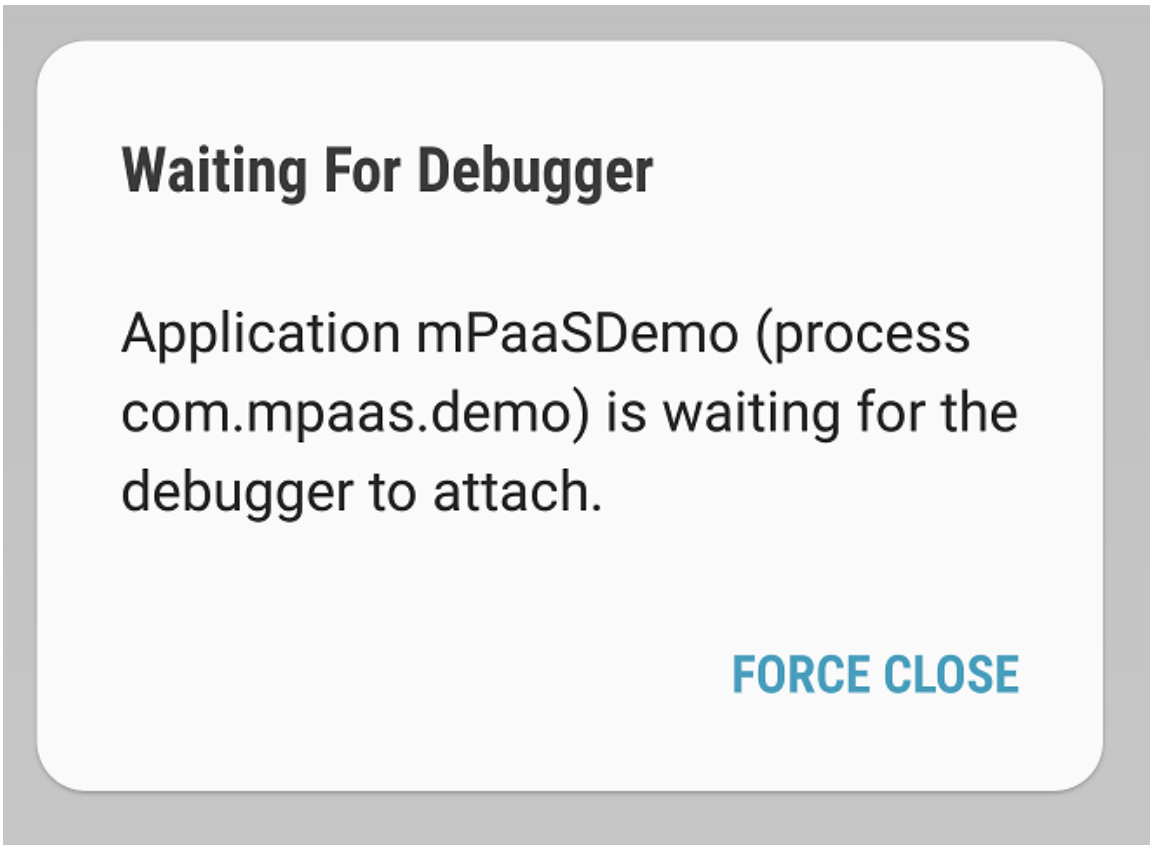
希望调试应用启动时的最初代码，比如在 application init 时初始化代码。

- **操作步骤：**

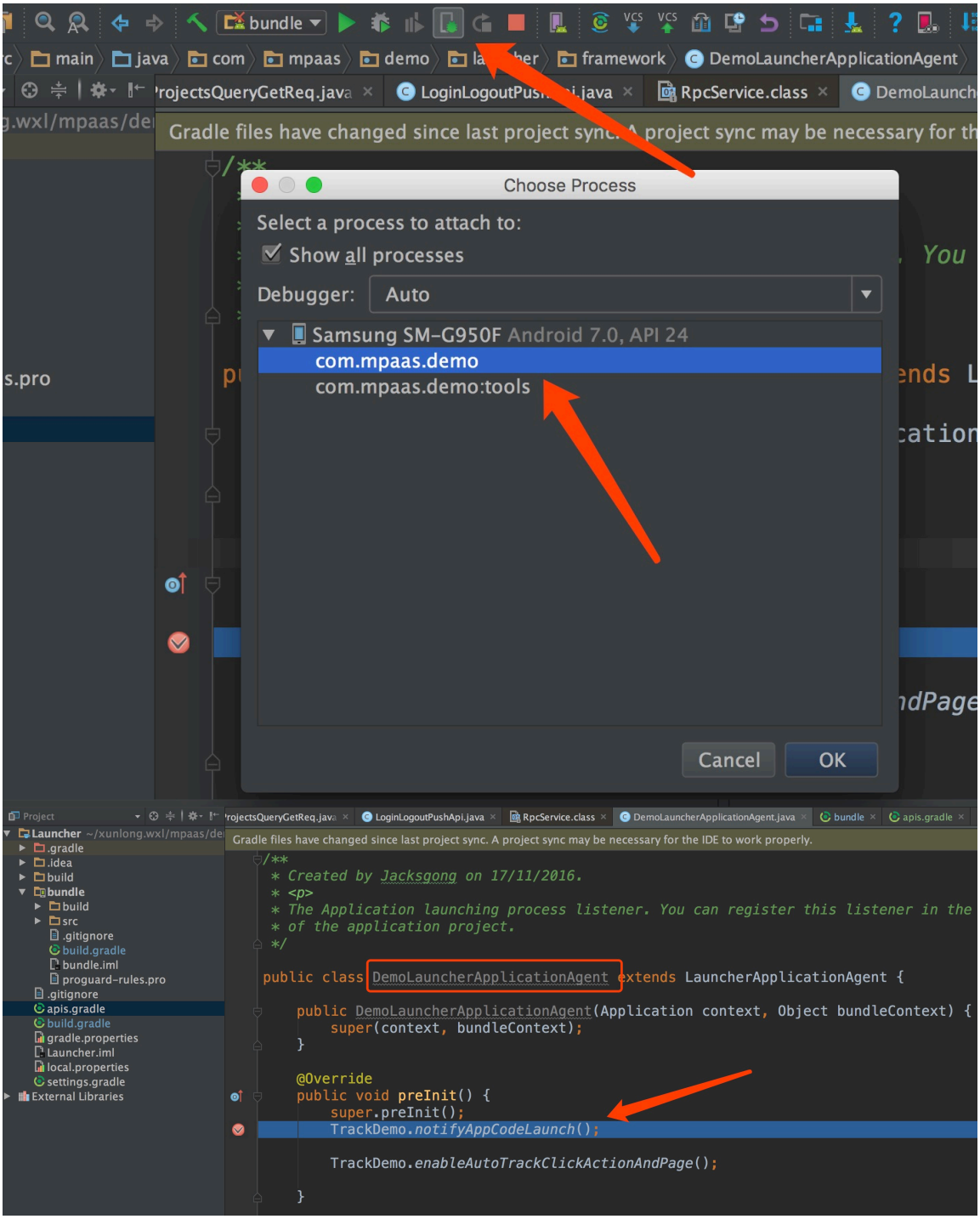
  执行命令 adb shell am start -W -S -D 应用包名/应用第一个启动的页面类名。例如，mPaaS Demo 的包名是 com.mpaas.demo，应用第一个启动的页面类名是com.alipay.mobile.quinox.LauncherActivity，那么可以使用命令行 adb shell am start -W -S -D com.mpaas.demo/com.alipay.mobile.quinox.LauncherActivity 以调试模式启动应用。第一个启动的类名如下图所示：

```
<application
    android:name="com.alipay.mobile.quinox.LauncherApplication"
    android:allowBackup="false"
    android:icon="@mipmap/ic_launcher"
    android:label="mPaaSDemo"
    android:theme="@style/AppTheme"
    tools:replace="android:label">
    <activity
        android:name="com.alipay.mobile.quinox.LauncherActivity"
        android:windowSoftInputMode="stateAlwaysHidden">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
```

执行命令之后，手机会弹出如下对话框：



对希望调试的代码行设置断点，然后附着到应用所在进程即可，如图：



应用运行后调试

- **使用场景**

在触发某个事件之后进行调试，比如点击某个按钮或者跳转某个页面才需要调试。

- **操作步骤：**

在应用运行后，点击附着进程（  ）按钮，或者在执行上述命令后，再点击附着按钮开始调试。


蚂蚁金服
ANT FINANCIAL
金融科技
FINANCIAL TECHNOLOGY