蚂蚁金服金融科技产品手册

接入 Android

产品版本： V20200101

文档版本： V20200101

蚂蚁金服金融科技文档

蚂蚁金服金融科技版权所有 © 2019 ，并保留一切权利。

未经蚂蚁金服金融科技事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

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

将安卓开发工程接入移动开发平台 mPaaS有以下三种接入方式：

- 组件化接入
- mPaaS Inside 接入
- 原生接入

组件化接入

组件化 是指 mPaaS基于 OSGi（Open Service Gateway Initiative，开放服务网关倡议）技术将把一个 App 划分成业务独立的一个或多个 Bundle 工程以及一个 Portal 工程的框架。mPaaS 会对每个 Bundle 工程的生命周期和依赖加以管理，使用 Portal 工程把所有的 Bundle 工程包合并成一个可运行的 .apk 包。

mPaaS Inside 接入

mPaaS Inside 的接入方式与原生工程接入 aar 的方式几乎一致，适合于对 Bundle 组件化方案没有特别需求却急需使用 mPaaS 能力的客户，降低了开发者的接入成本，让开发者更轻松地使用 mPaaS。
说明：mPaaS Inside 的接入方式自 10.1.60-beta 起开始支持。

原生接入

原生框架则是指采用原生 Android 框架，不使用 mPaaS 框架。

选择接入方式

如果是从零开始开发全新应用，那么推荐您使用 mPaaS Inside 的接入方式。
如果已有 Android 应用，那么您有以下两种选择：

- 您可以使用 **组件化接入** 或 **mPaaS Inside** 的接入方式对应用进行改造，以此更便捷地接入 mPaaS 组件。更多详情请参考 迁移原生工程至组件化框架 和 迁移原生工程至 mPaaS Inside 。
- 您可以使用 **基于原生框架** 的接入方式以避免过高的改造成本。在此种方式下，您依然可以使用 mPaaS 提供的组件的功能。更多详情请参考 基于原生框架 > 接入流程简介 。

相关链接

- 组件化接入方式简介
- mPaaS Inside 接入方式简介
- 原生接入流程简介

2 配置开发环境

在进行客户端开发之前，您首先需要配置开发环境：

- 配置 Windows 开发环境
- 配置 macOS 开发环境
- 配置 Linux 开发环境

配置 Windows 开发环境

参考以下说明配置 Windows 开发环境：

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**。

使用 Gradle Wrapper（推荐）

1. 如果您的项目原先已经使用 Gradle Wrapper 进行构建，则建议在项目目录 /gradle/wrapper/gradle.properties 下把版本号修改为**4.4**。
2. 如果您的项目没有使用 Gradle Wrapper，建议先使用全局的 Gradle（4.4）版本，然后调用 gradle wrapper --gradle-version=4.4 安装一个 gradle wrapper。完成上述步骤后，您只需要使用 ./gradlew 的形式构建，这样的方式能最小的影响您的开发环境。

使用独立 gradle

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

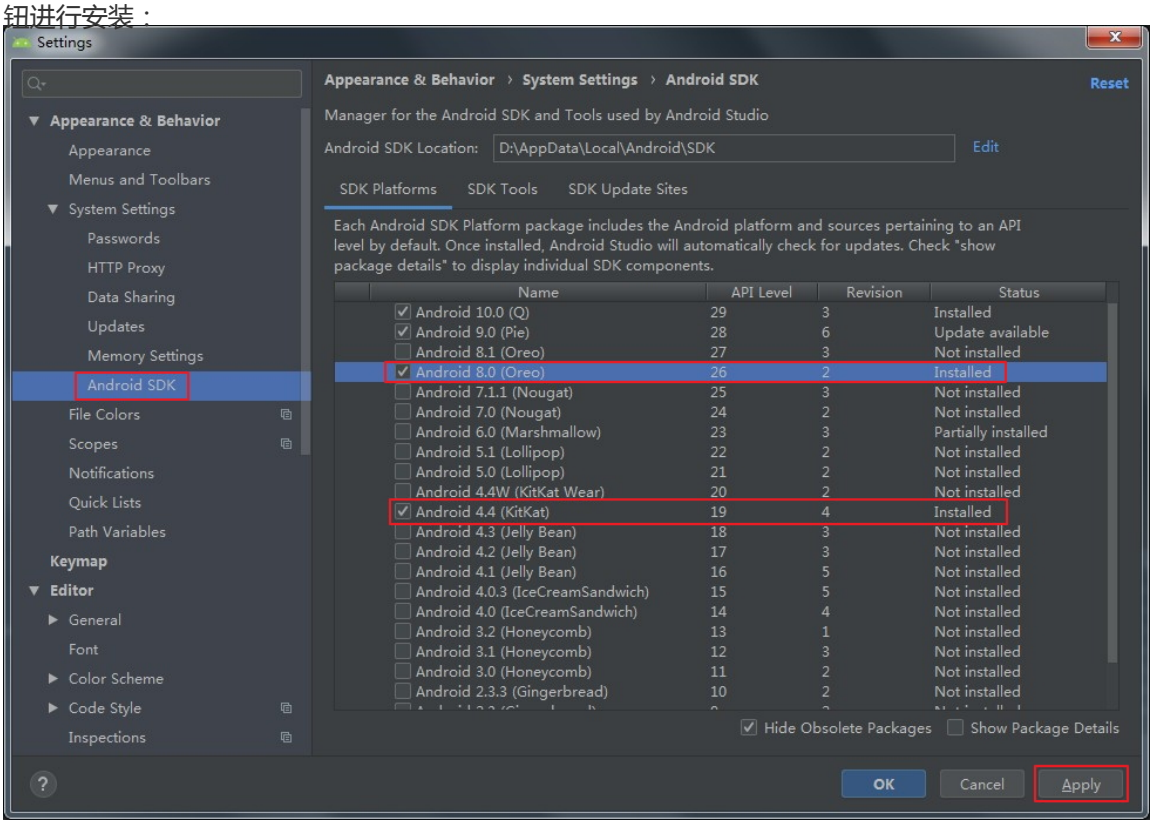
mPaaS 框架支持 **3.2**、**3.3.1**、**3.3.2**、**3.4.2** 和 **3.5**（推荐）版本的 Android Studio。

- 关于 Android Studio 下载，请参见 Android Developers。
- 安装指南。
- 如果您之前使用的是低版本 Android Studio 并已经安装了 mPaaS 插件，那么在您从低版本 Android Studio 升级至 3.5 版本的 Android Studio 之后，您只需要升级 mPaaS 插件至最新版本即可。详情请参见 更新 mPaaS 插件 。

安装 Android SDK

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

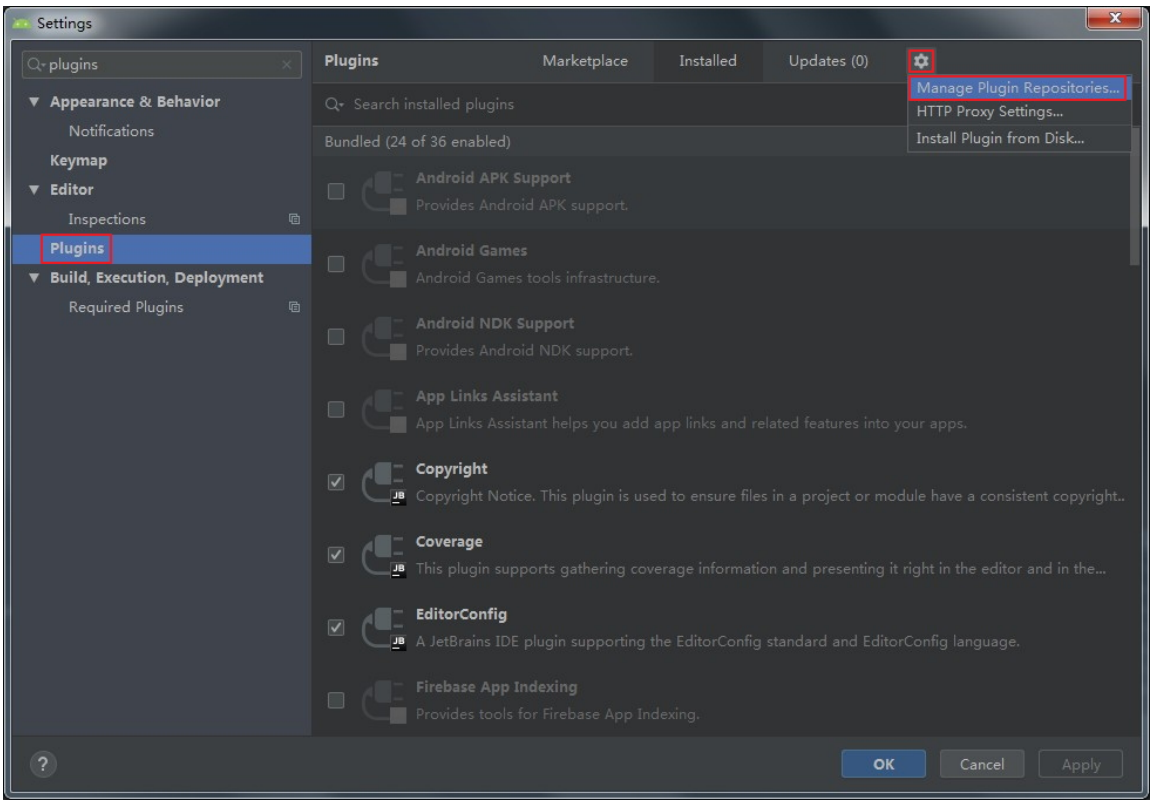
1. 在 Android Studio 中，通过 **File** > **Settings** 打开设置对话框。
2. 如下图所示，在 **Android SDK** 对话框中，勾选 API Level 为 **19** 和 **26** 的 SDK，然后点击 **Apply** 按钮进行安装：



安装 mPaaS 插件

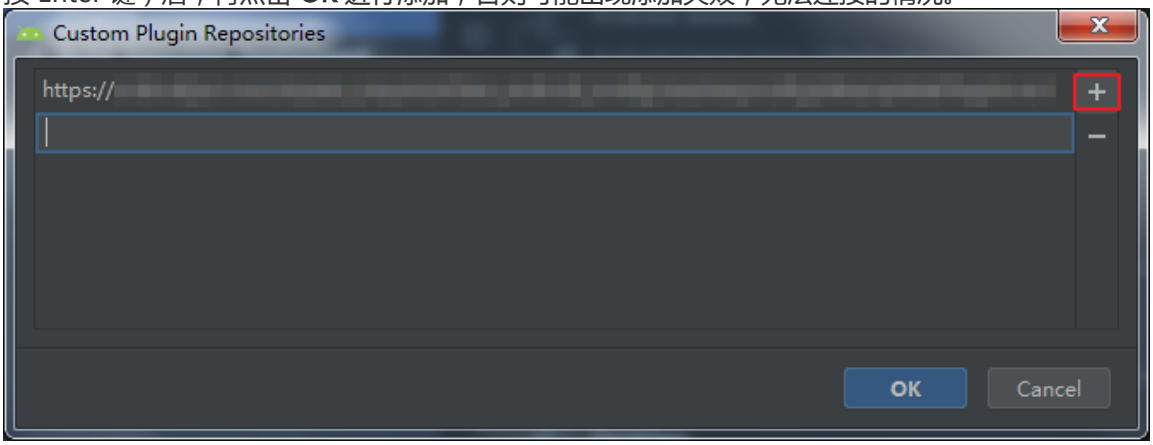
mPaaS 插件提供多种开发辅助功能，包括：新建 mPaaS 工程，添加、删除和升级 mPaaS 组件，构建工程等。安装步骤如下：

1. 在 Android Studio 中，通过 **File** > **Settings** 打开设置对话框。
2. 点击左侧 **Plugins**，然后点击右上方的  ，并在下拉菜单中选择 **Manage Plugin Repositories**。

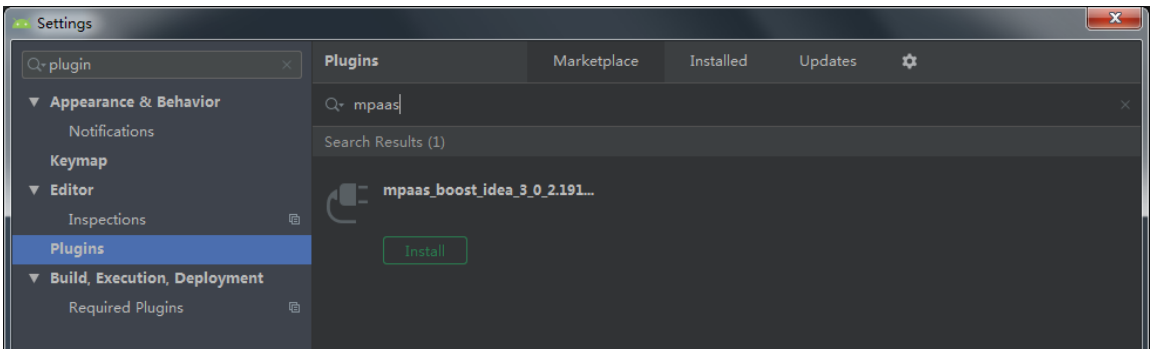


3. 在弹出的 **Custom Plugin Repositories** 窗口中，点击 **+** 按钮，在新增的行中输入 https://mulei.oss-cn-hangzhou.aliyuncs.com/updatePlugins.xml，然后点击 **OK**。

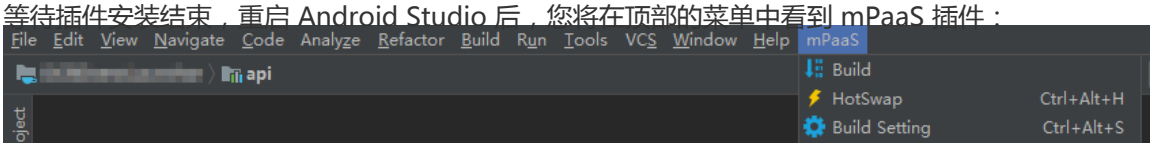
   说明：对于个别版本的 Android studio，可能需要在输入仓库地址后，鼠标单击下方空白区域（或按 Enter 键）后，再点击 **OK** 进行添加，否则可能出现添加失败，无法连接的情况。



4. 返回 **Plugins** 窗口后，在 **Marketplace** 选项卡下搜索 **mpaas**。在搜索结果中，点击搜索到的 mPaaS 插件下方的 **Install** 安装该插件。



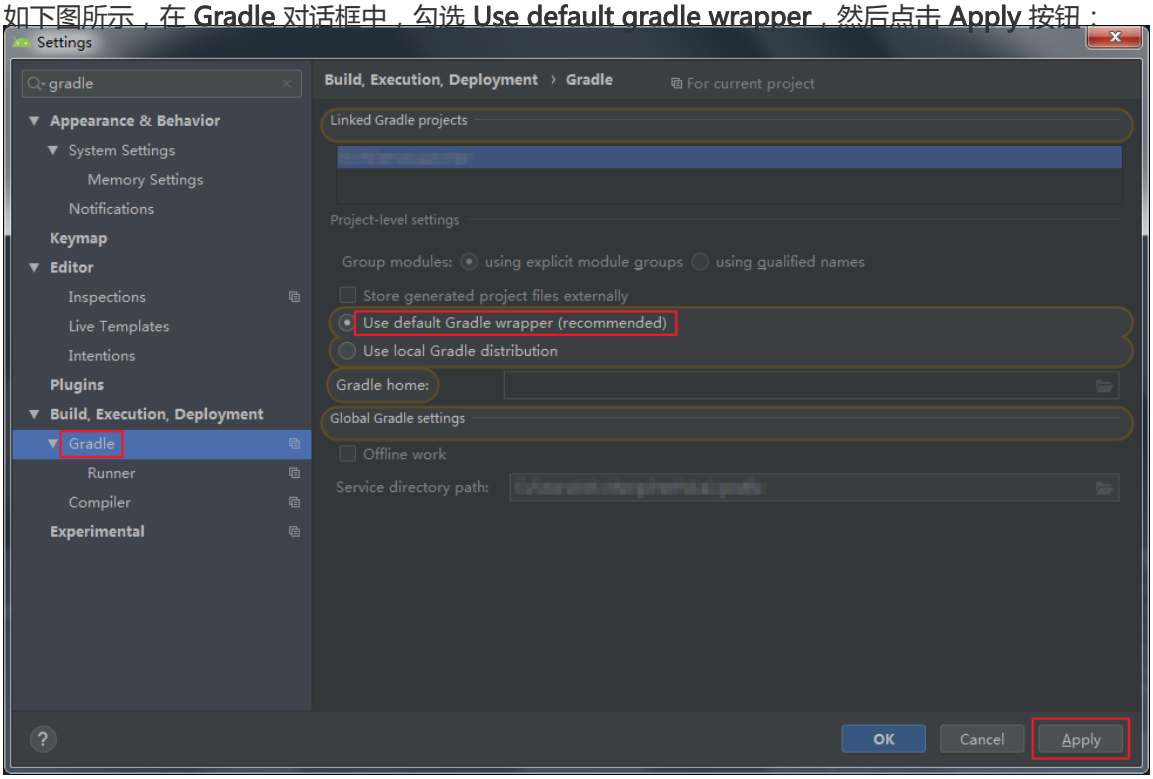
5. 等待插件安装结束，重启 Android Studio 后，您将在顶部的菜单中看到 mPaaS 插件：



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，通过 **File** > **Settings** 打开设置对话框。
2. 如下图所示，在 **Gradle** 对话框中，勾选 **Use default gradle wrapper**，然后点击 **Apply** 按钮：



配置 macOS 开发环境

按照以下描述配置 macOS 开发环境：

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：

配置 Gradle 4.4 环境

mPaaS 框架只支持 **Gradle 4.4**。

使用 Gradle Wrapper（推荐）

1. 如果您的项目原先已经使用 Gradle Wrapper 进行构建，则建议在项目目录 /gradle/wrapper/gradle.properties 下把版本号修改为**4.4**。
2. 如果您的项目没有使用 Gradle Wrapper，建议先使用全局的 Gradle（4.4）版本，然后调用 gradle wrapper --gradle-version=4.4 安装一个 gradle wrapper。完成上述步骤后，您只需要使用 ./gradlew 的形式构建，这样的方式能最小的影响您的开发环境。

使用独立 gradle

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：

安装并配置 Android Studio

安装 Android Studio

mPaaS 框架支持 **3.2**、**3.3.1**、**3.3.2**、**3.4.2** 和 **3.5**（推荐）版本的 Android Studio。

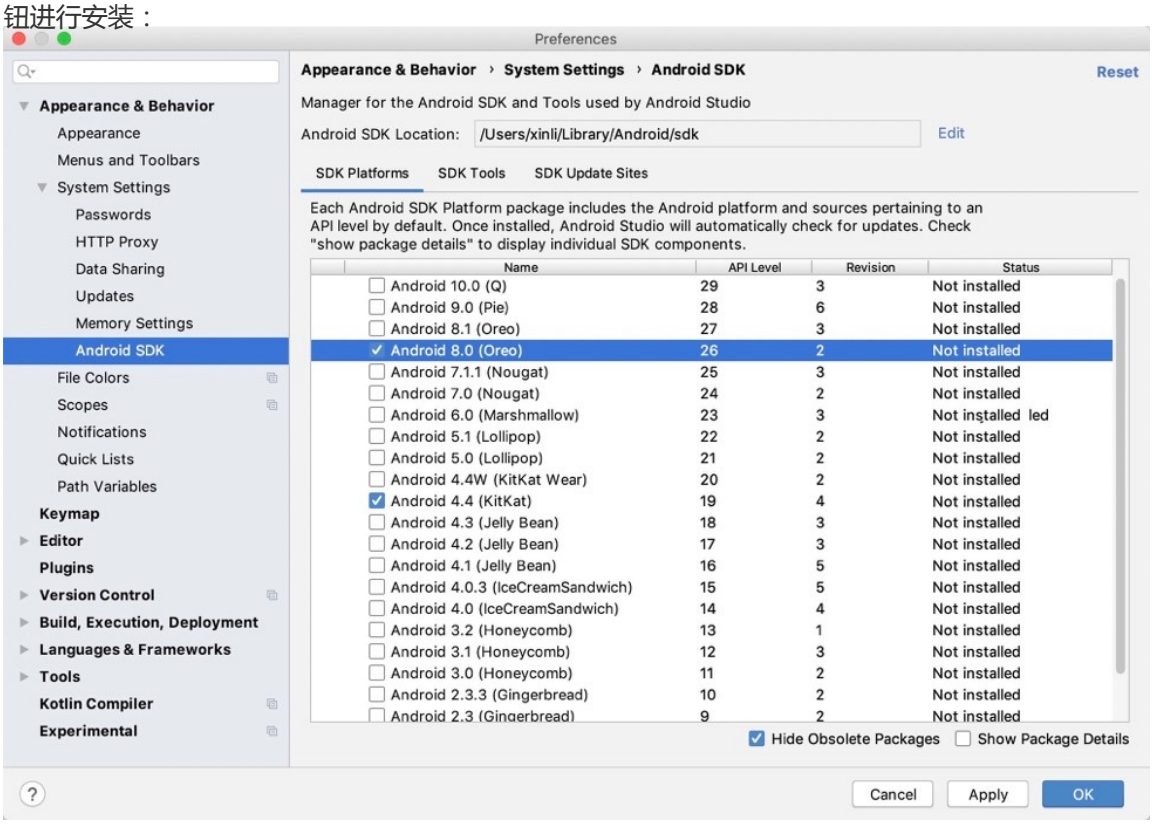
- 关于 Android Studio 下载，请参见 Android Developers。
- 关于 Android Studio 安装，请参见 安装指南。
- 如果您之前使用的是低版本 Android Studio 并已经安装了 mPaaS 插件，那么在您从低版本

Android Studio 升级至 3.5 版本的 Android Studio 之后，您只需要升级 mPaaS 插件至最新版本即可。详情请参见 更新 mPaaS 插件 。

安装 Android SDK

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

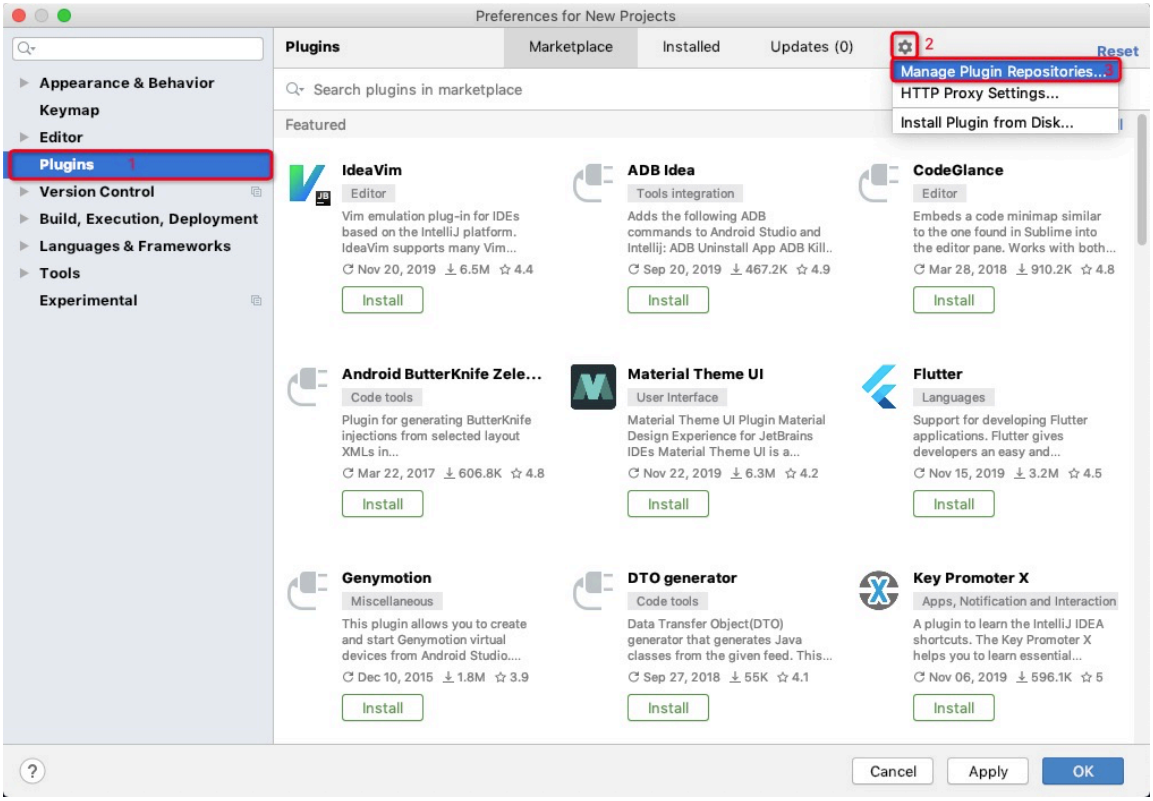
1. 在 Android Studio 中，通过 **Android Studio** > **Preferences** 打开设置对话框。
2. 如下图所示，在 **Android SDK** 对话框中，勾选 API Level 为 **19** 和 **26** 的 SDK，然后点击 **Apply** 按钮进行安装：



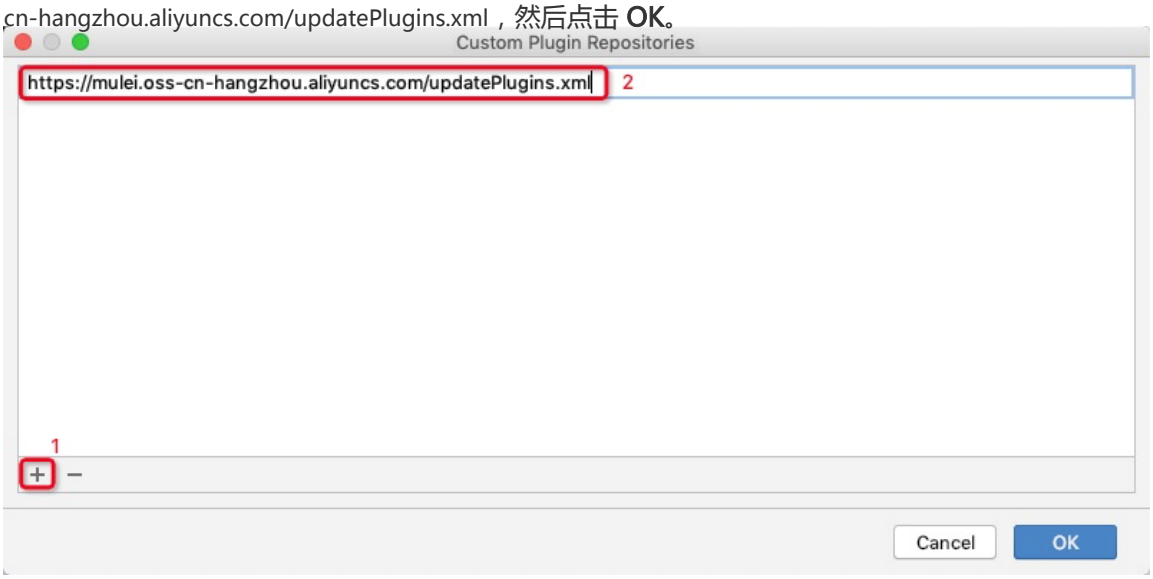
安装 mPaaS 插件

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：

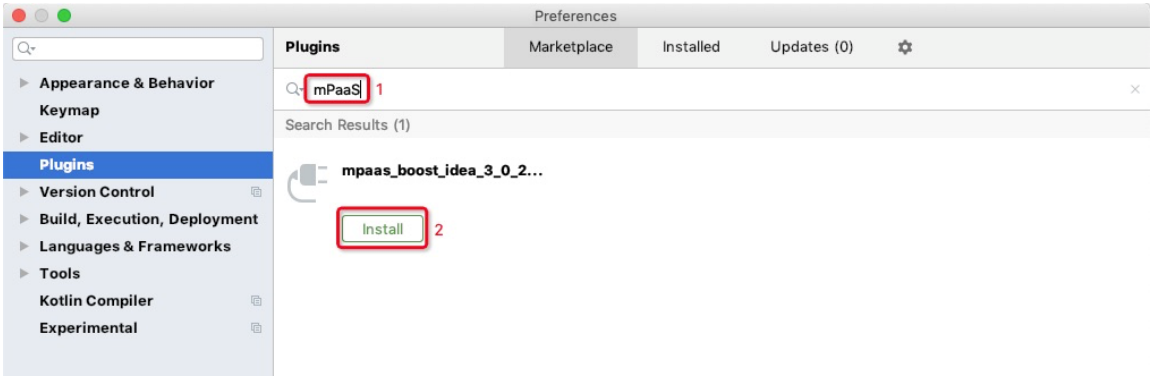
1. 在 Android Studio 中，通过 **Android Studio** > **Preferences** 打开设置对话框。
2. 点击左侧 **Plugins**，然后点击右上方的  ，并在下拉菜单中选择 **Manage Plugin Repositories**。



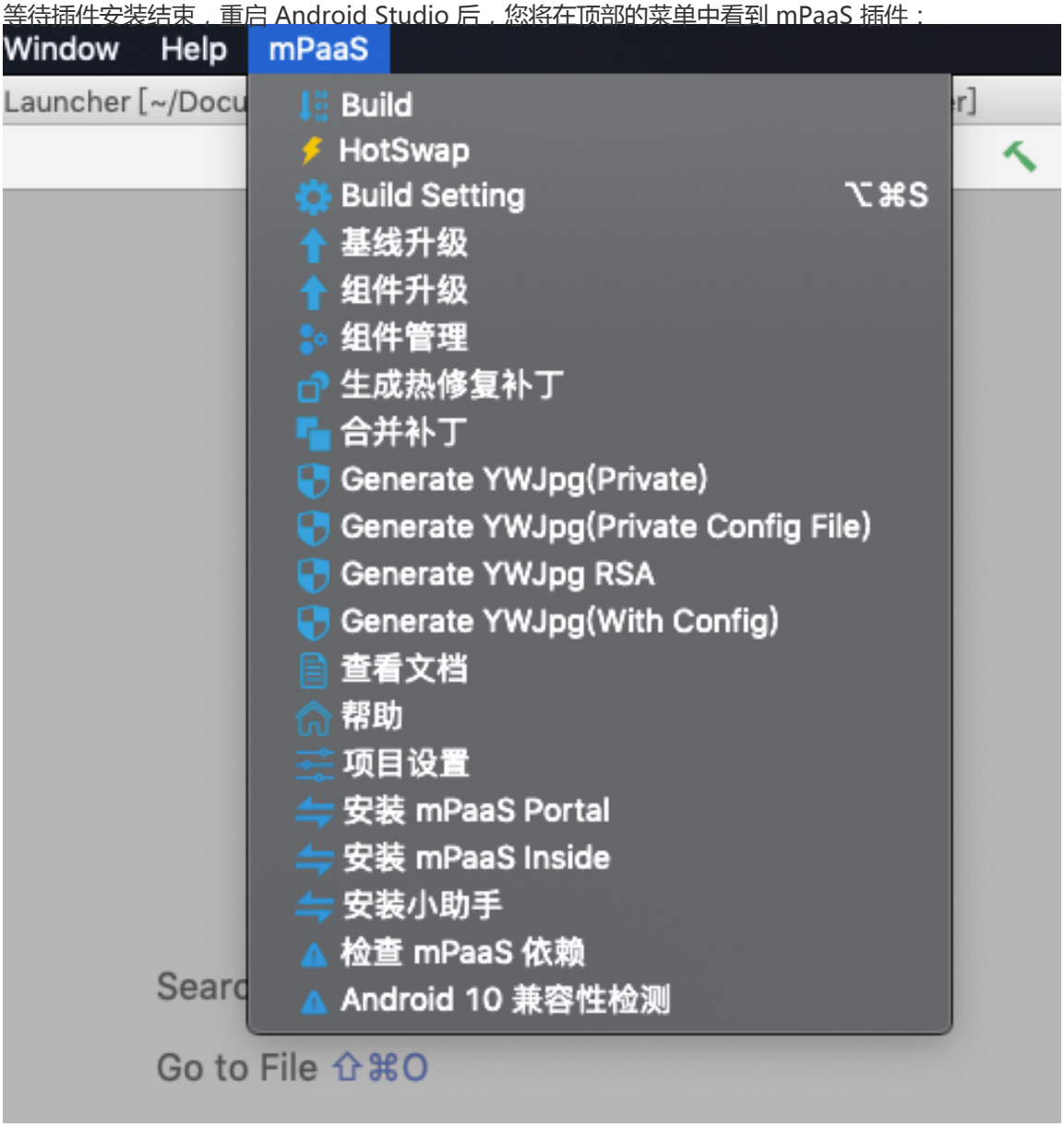
3. 在弹出的 **Custom Plugin Repositories** 窗口中，点击 **+** 按钮，在新增的行中输入 https://mulei.oss-cn-hangzhou.aliyuncs.com/updatePlugins.xml，然后点击 **OK**。



4. 返回 **Plugins** 窗口后，在 **Marketplace** 选项卡下搜索 **mpaas**。在搜索结果中，点击搜索到的 mPaaS 插件下方的 **Install** 安装该插件。



5. 等待插件安装结束，重启 Android Studio 后，您将在顶部的菜单中看到 mPaaS 插件：

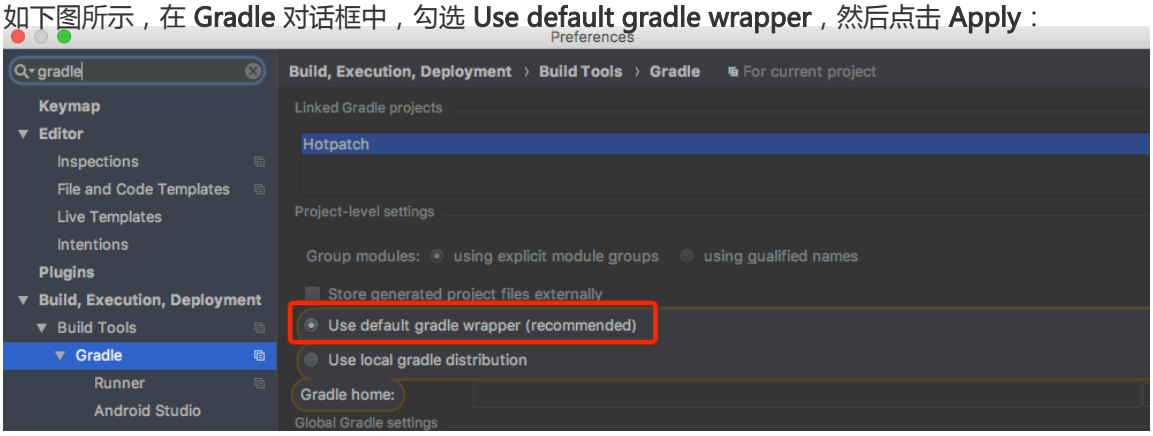


配置 Gradle 构建工具

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**：



配置 Linux 开发环境

按照以下说明配置 Linux 开发环境。
说明：Linux 操作系统版本较多，本文仅适用于 CentOS 和 Ubuntu 版本。

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

使用 Gradle Wrapper（推荐）

1. 如果您的项目原先已经使用 Gradle Wrapper 进行构建，则建议在项目目录 /gradle/wrapper/gradle.properties 下把版本号修改为**4.4**。
2. 如果您的项目没有使用 Gradle Wrapper，建议先使用全局的 Gradle（4.4）版本，然后调用 gradle wrapper --gradle-version=4.4 安装一个 gradle wrapper。完成上述步骤后，您只需要使用 ./gradlew 的形式构建，这样的方式能最小的影响您的开发环境。

使用独立 gradle

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

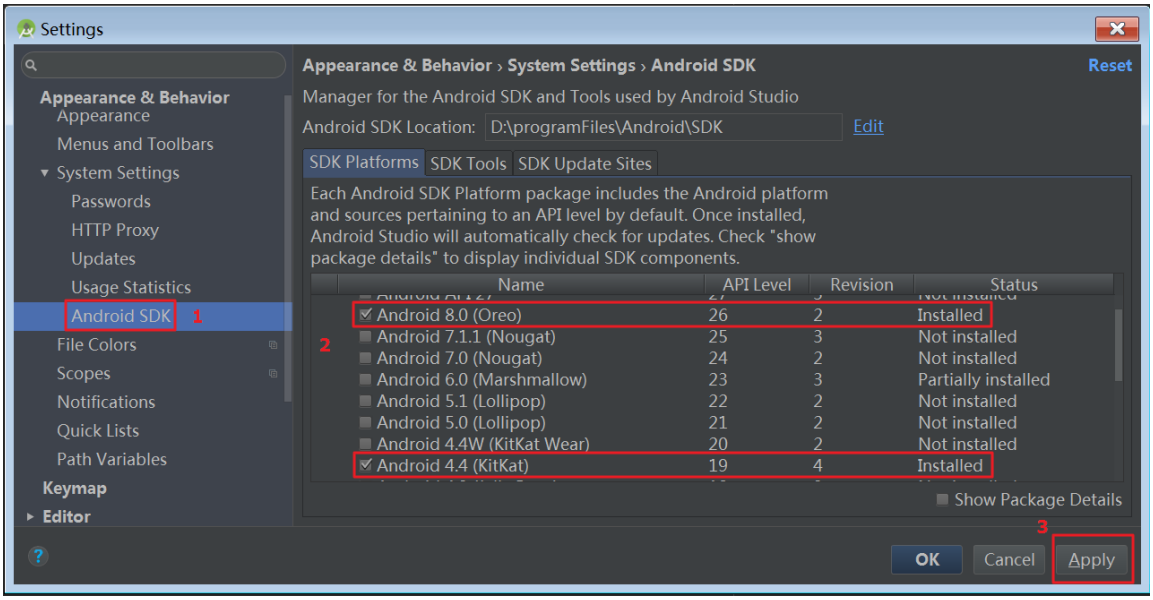
mPaaS 框架支持 **3.2**、**3.3.1**、**3.3.2**、**3.4.2** 和 **3.5**（推荐）版本的 Android Studio。

- 关于 Android Studio 下载，请参见 Android Developers。
- 安装指南。
- 如果您之前使用的是低版本 Android Studio 并已经安装了 mPaaS 插件，那么在您从低版本 Android Studio 升级至 3.5 版本的 Android Studio 之后，您只需要升级 mPaaS 插件至最新版本即可。详情请参见 更新 mPaaS 插件 。

安装 Android SDK

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

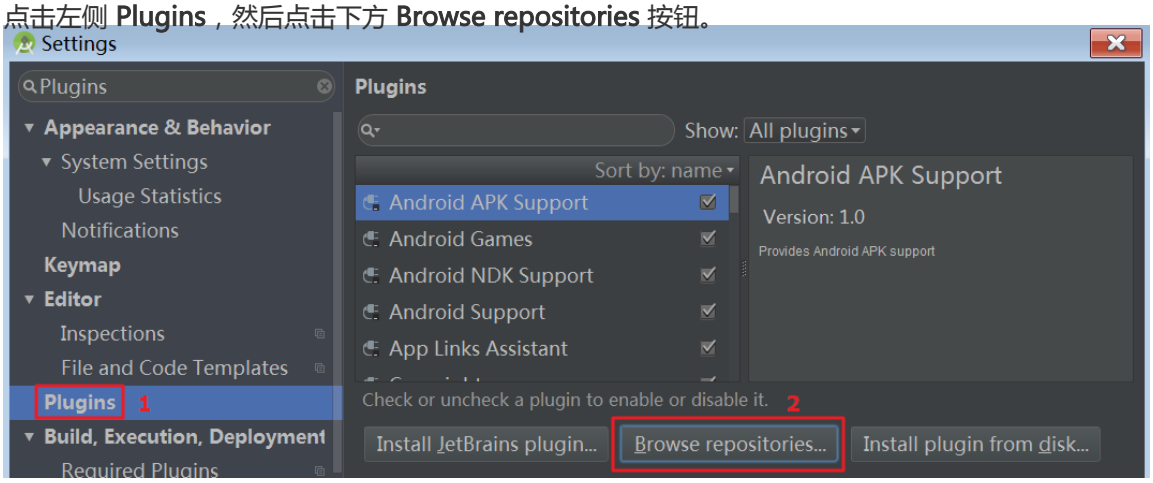
1. 在 Android Studio 中，通过 **File** > **Settings** 打开设置对话框。
2. 如下图所示，在 **Android SDK** 对话框中，勾选 API Level 为 **19** 和 **26** 的 SDK，然后点击 **Apply** 按钮进行安装：



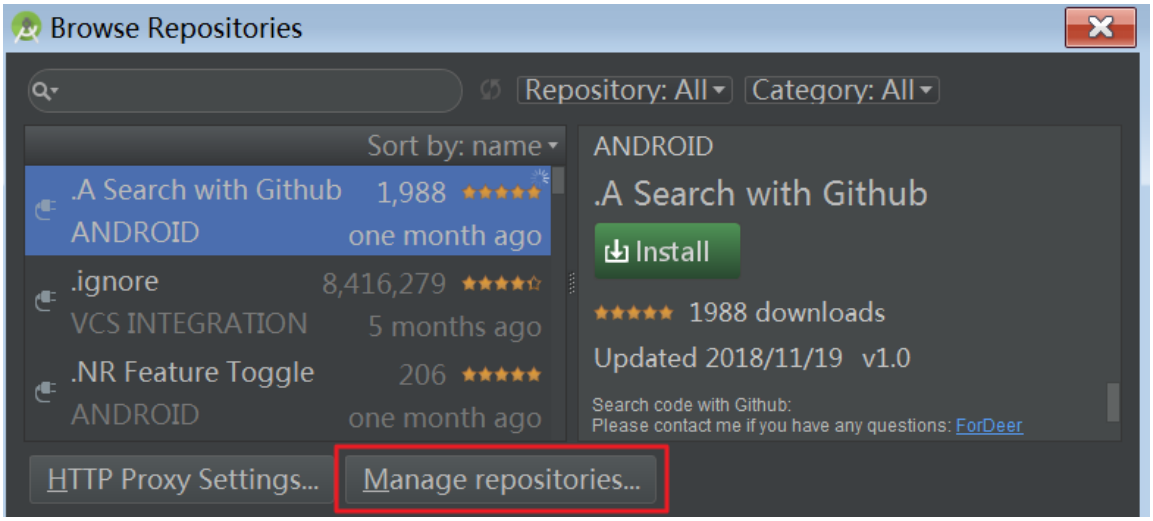
安装 mPaaS 插件

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：

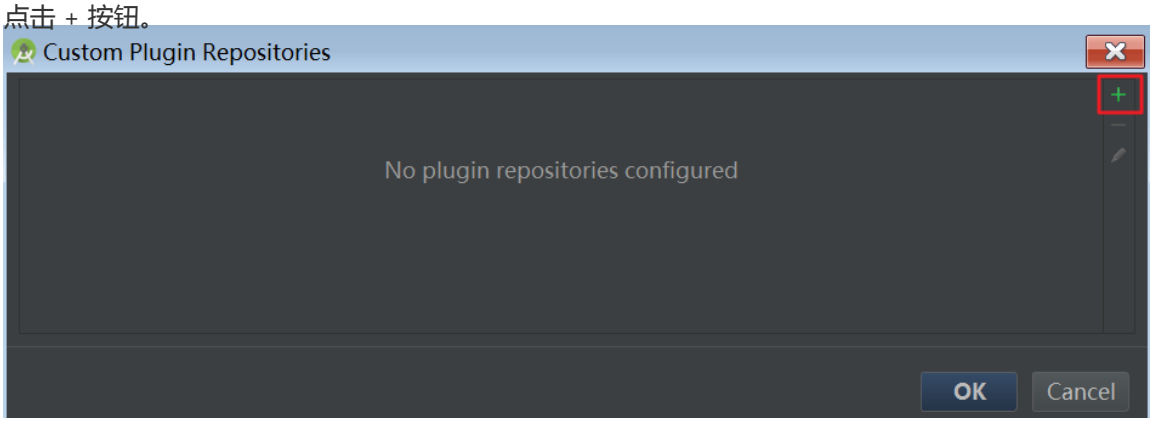
1. 打开 Android Studio 设置对话框。
2. 点击左侧 **Plugins**，然后点击下方 **Browse repositories** 按钮。



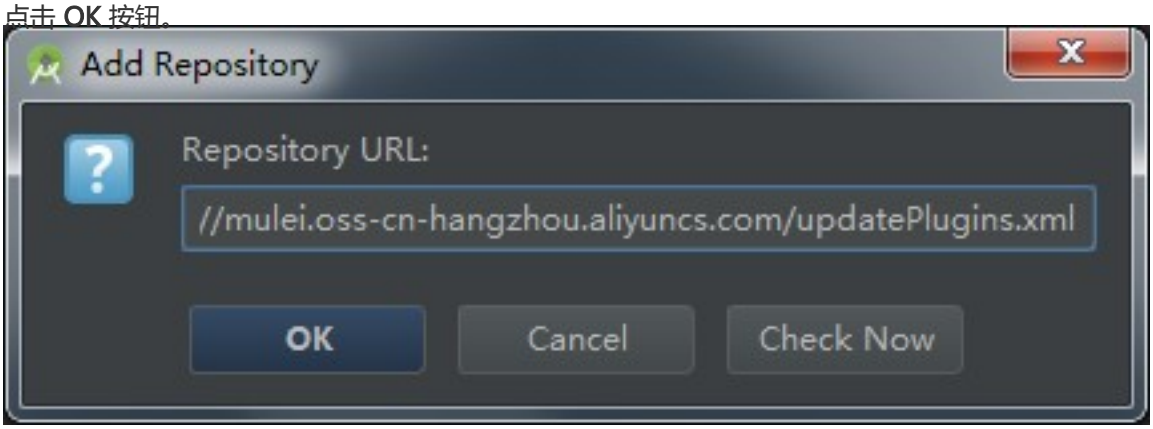
3. 点击下方 **Manage repositories** 按钮。



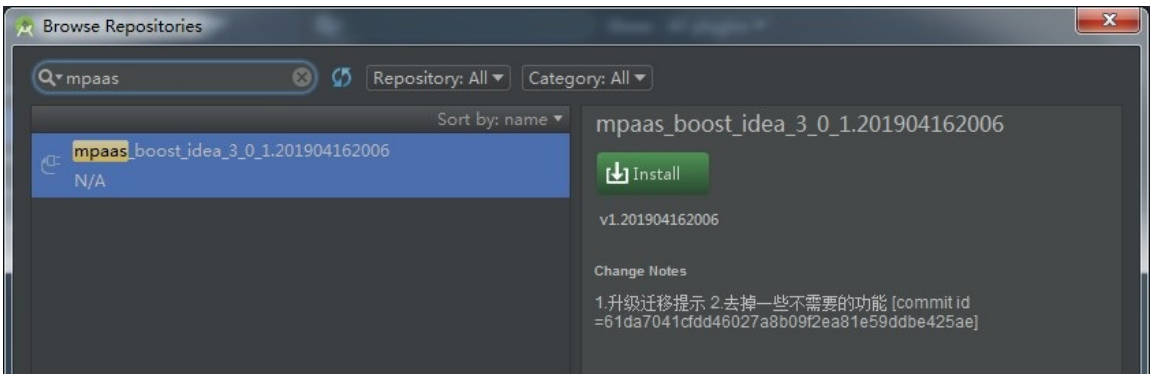
4. 点击 + 按钮。



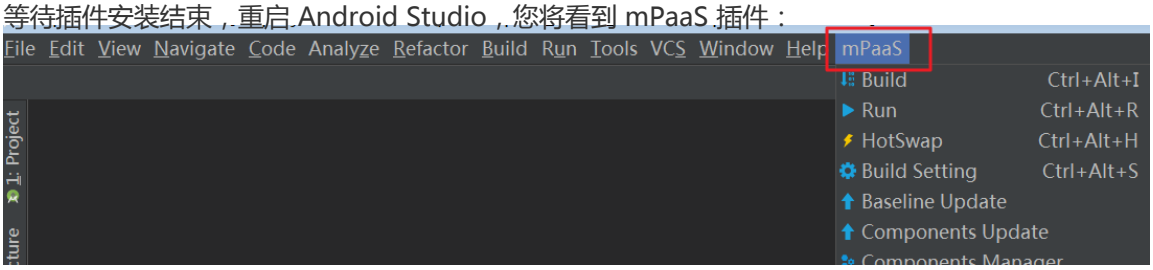
5. 在弹出框 **Repository URL** 中填入 https://mulei.oss-cn-hangzhou.aliyuncs.com/updatePlugins.xml，然后点击 **OK** 按钮。



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，然后点击插件右侧的 **Install** 安装该插件。



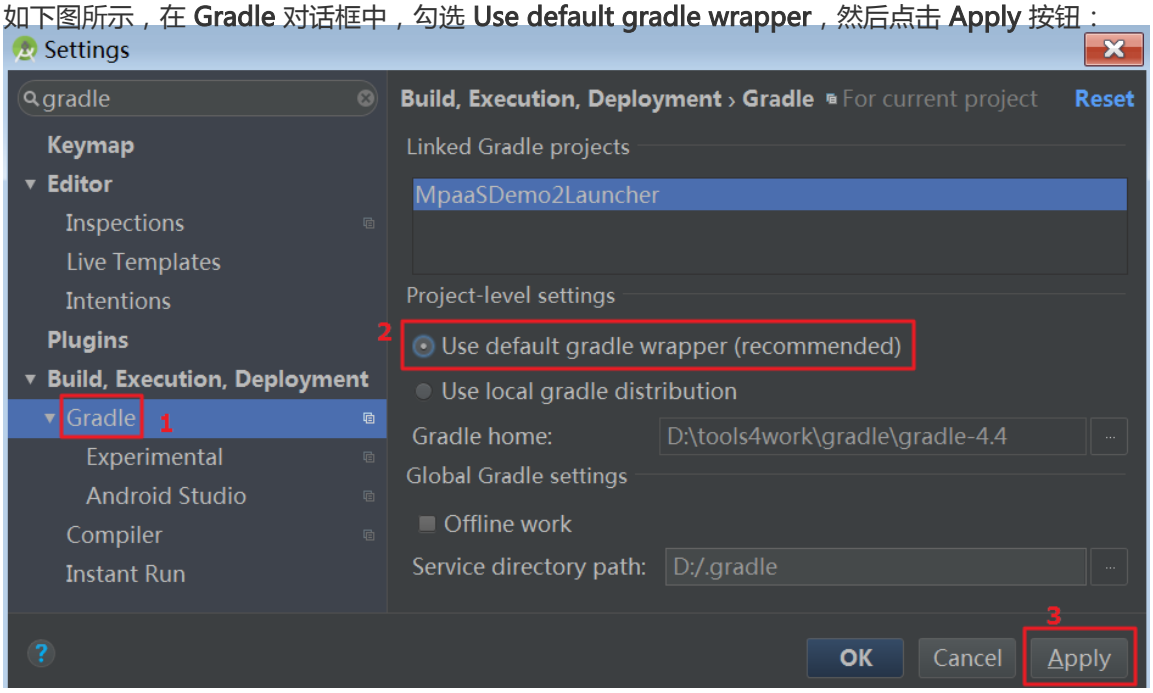
8. 等待插件安装结束，重启 Android Studio，您将看到 mPaaS 插件：



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：



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：

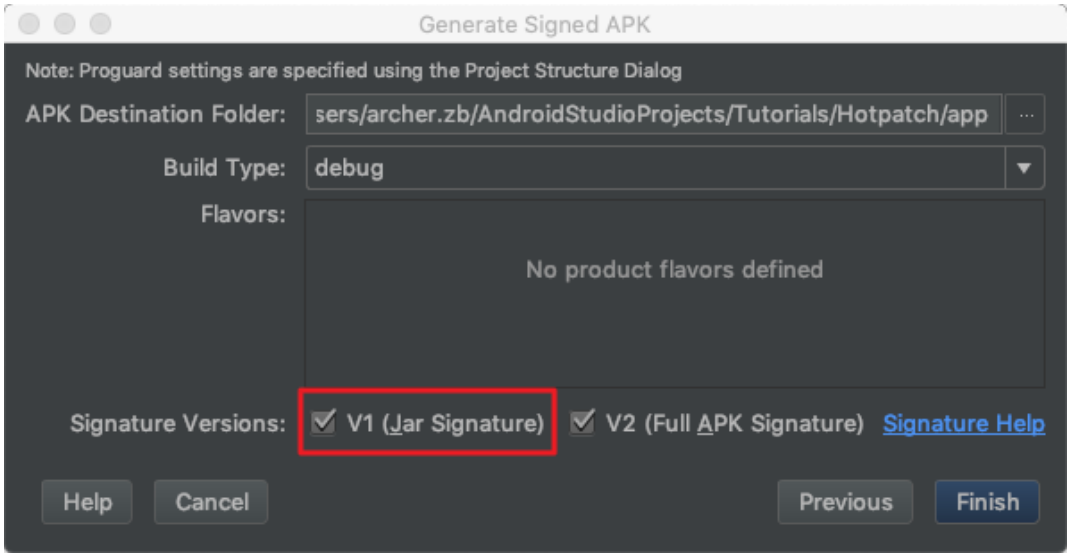
下载配置文件

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**。

重要：

- 此处上传的 APK 应和 **发布版本** 的 APK 使用相同的签名文件。
- 给应用签名时，Signature Versions 中 **V1(Jar Signature)** 必须勾选，**V2(Full APK Signature)** 可按需勾选：



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 。

4 组件化接入方式（Portal Bundle）

4.1 简介

组件化框架 是指 mPaaS基于 OSGi（Open Service Gateway Initiative，开放服务网关倡议）技术将把一个 App 划分成业务独立的一个或多个 Bundle 工程以及一个 Portal 工程的框架。mPaaS 会对每个 Bundle 工程的生命周期和依赖加以管理，使用 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，并且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

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'
```

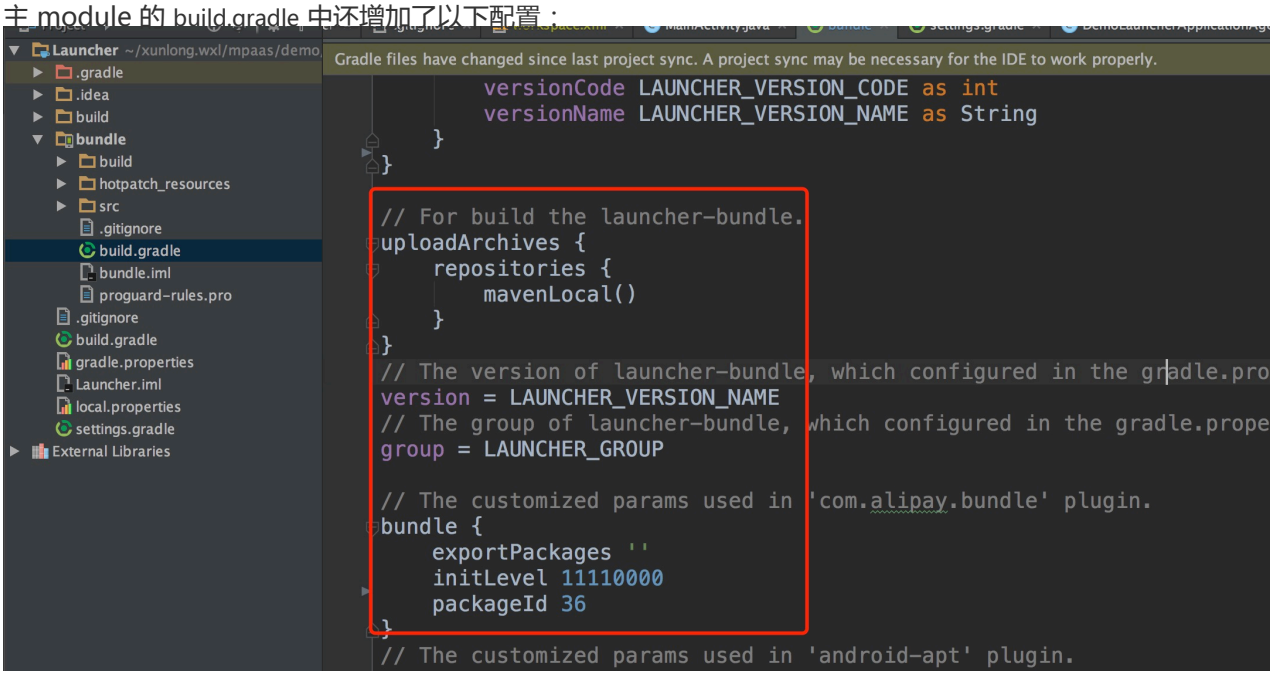
```
buildscript {
    repositories {
        mavenLocal()
        jcenter()
        // The Following repository is mPaaS address:
        maven {
            credentials {
                username "[redacted]"
                password "[redacted]"
            }
            url "http://mvn.cloud.alipay.com/nexus/content/repositories/r
        }
    }
    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:bundleB: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"

说明：

- 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。

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。

Portal 与传统工程区别

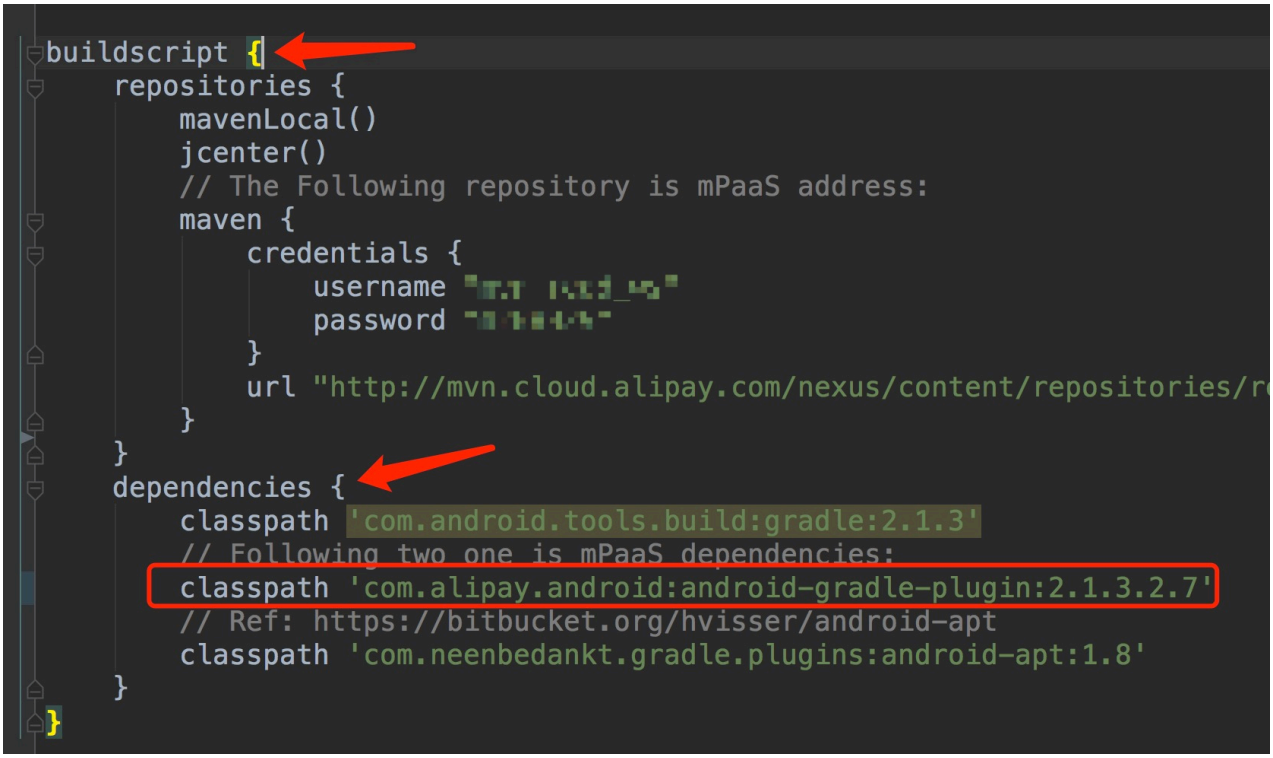
Portal 和传统开发中的工程的区别体现在 build.gradle:

- 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 "[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'
    }
}
```

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。

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

4.2 接入流程

4.2.1 接入流程简介

如果采用 **组件化接入方式**，您需要完成以下通用步骤以完成接入流程：

1. 配置开发环境 。
2. 在控制台创建应用 。
3. 客户端创建新工程 或 迁移原生工程至组件化框架 。
4. 管理组件依赖 。
5. 编译打包 。

4.2.2 迁移原生工程至组件化框架

如果已有基于 Android 原生框架的应用，您可以使用 mPaaS 插件的功能将其迁移至组件化框架。

前置条件

- 您已完成开发环境的配置。更多信息，请参考 配置本地开发环境 。
- 您已经在控制台创建了应用并下载到了配置文件。更多信息，请参考 创建应用 。

操作步骤

1. 打开 Android Studio。

2. 点击 **mPaaS** > **迁移至 mPaaS Portal**。
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。

说明：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，解决方法见下方的 查看及解决冲突 。

5. 接入 Application。
   - 如果您自己定义了 Application，则需要继承 com.alipay.mobile.quinox.LauncherApplication。如果没有自定义 Application，请忽略本步骤。
   - mPaaS 带有多 dex 的加载机制，所以需要在代码去掉 MultiDex 类的引用。

后续事项

查看及解决冲突

1. 查看冲突：请在 conflict.json 文件中查看冲突项目。
2. 解决冲突

开源库名	mPaaS 库名	冲突解决方案
fastjson	com.alipay.android.phone.thirdparty:fastjson-build	建议使用 mPaaS fastJson，该版本稳定且安全性高。
alipayInside	rpc、UTDID(com.alipay.android.phone.thirdparty:utdid-build)	升级到无 UTDID 版本 SDK 即可，点击这里查看详情。
supportv4	com.alipay.android.phone.thirdparty:androidsupport-build	如果仅需要 supportV4，建议使用 mPaaS 版本。如果是使用官方的 appcompat，则需要客户手动实现自动化埋点（在 fragment 中调用 mPaaS 在 supportv4 中定制的 fragment 的方法即可）。
appcompat	com.alipay.android.phone.thirdparty:androidsupport-build	使用官方 appcompat，则需要客户手动实现自动化埋点（在 fragment 中调用 mPaaS 在 supportv4 中定制的 fragment 的方法即可）。
libdatabase_sqlcrypto.so 统一存储的 so 库冲突	storage 模块	删掉依赖的三方包中的 so 文件：解压 zip 包，然后删掉 so 文件，再重新压缩即可。
okio	com.alipay.android.phone.thirdparty:wire-build	可以提供一个没有 okio 的 RPC 模块，需要回归测试 mPaaS RPC 报文解析功能。
高德地图、定位（Amap location or Amap map）	com.alipay.android.phone.mobilecommon:lbs-build	建议使用 mPaaS 的模块。如果使用高德版本请注释 lbs-build 模块，并且回归测试定位模块。

移除 mPaaS 依赖

如果采用移除 mPaaS 依赖的方法解决冲突，则需要在 build.gradle mpaascomponents 中排除依赖，示例如下：

```
mpaascomponents{
// when you want exclude mpaas dependencies ,you can add dependency ga to to excludeDependencies
```

```
// or set removed = true in mpaaspakcages.json
excludeDependencies=['com.alipay.android.phone.thirdparty:wire-build']
}
```

4.2.3 客户端创建新工程

关于此任务

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

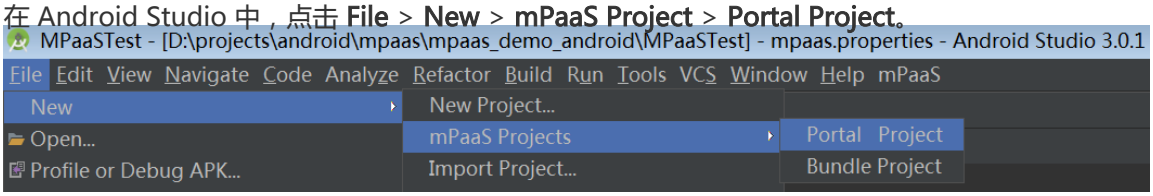
创建 Portal

如果您需要组件化方案，可以使用 Portal Bundle 方式，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，下同）、**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 > 组件管理 功能添加所需依赖。

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

说明：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**（MPaaSDemo）和 **Launcher**（MPaaSDemoLauncher）工程。

说明：在创建 Portal 工程的时候，Android Studio 会自动为其创建一个的 Bundle 工程。该 Bundle 工程的工程名即 Portal 工程的工程名加Launcher 后缀，如MPaaSDemoLauncher。

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。

关于 Bundle 开发的更多信息，请参见 Bundle 工程 。

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

组件化接入方式 > 简介 ：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**。

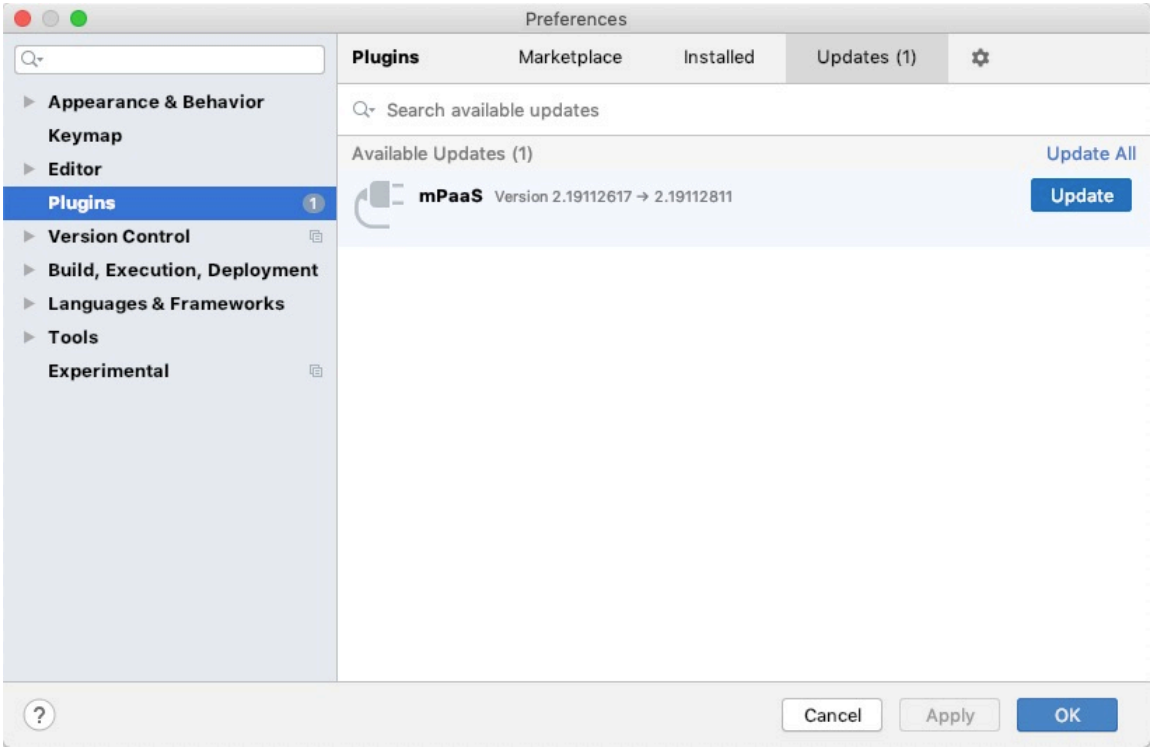
4.2.4 管理组件依赖

升级 IDEA 插件

说明：如果已是最新版本可跳过此升级。

为了更便捷地升级 mPaaS SDK 基线和组件，您需要先升级 mPaaS IDEA 插件至最新版本：

1. 在 Android Studio 中选择 **Preferences** > **Plugins**。
2. 在下方 **Updates** 结果中找到 **mPaaS**，并点击其右侧的 **Update** 进行升级。
3. 按照提示重启 Android Studio。



管理组件依赖

为了使用 mPaaS 组件 ，您需要分别在 Portal 和 Bundle 工程中添加对应组件的依赖：

- 在 Portal 工程中添加，将确保相应依赖在打包时打入您的 APK 。
- 在 Bundle 工程中添加，将确保您能在 Bundle 工程中调用对应组件的 API 。
- 对于单 Portal 工程模式，您只需在 Portal 工程中添加。
- 如果您在创建 mPaaS 工程时已选择过需要使用的组件，您仍可以按照下文步骤增删组件。

使用 IDEA 插件增删依赖步骤为：

1. 在 Android Studio 中选择 **mPaaS** > **组件管理**，如果您此前未使用过 IDEA 插件管理组件依赖，您会先进入 **选择 mPaaS 基线版本** 来选择版本：

2. 随后您将看到组件列表：

3. 您可以根据需要增删组件，install 为添加，uninstall 为删除，框架和必备组件无法被删除。

首次使用插件管理依赖

如果您此前未使用过 IDEA 插件管理组件依赖，当您首次使用 **组件管理** 功能添加完组件后，您还需要检查或修改以下配置：

1. 检查 Portal/Bundle 工程根目录 build.gradle 文件，确保包含以下依赖且不低于以下版本：

```
buildscript {
...
dependencies {
classpath 'com.android.boost.easyconfig:easyconfig:2.1.6'
}
}
```

2. 检查 Portal 工程主 module 下的 build.gradle 文件，确保包含以下内容：

```
apply plugin: 'com.alipay.portal'
portal {
allSlinks true
mergeAssets true
}
```

```
apply plugin: 'com.alipay.apollo.baseline.update'
mpaascomponents{
excludeDependencies=[]
}
```

3. 删除旧依赖：

   重要：强烈建议您在删除以下内容前先进行备份。

   - 对于 Portal + Bundle 模式，您需要在 Portal 工程主 module 下的 build.gradle 文件中删除 dependencies 节点下 mPaaS 组件相关依赖（mpaas-baseresjar 除外）。
   - 对于单 Portal 工程模式，您需要在主 module 下的 build.gradle 文件中删除以下内容：

     ```
     apply from: rootProject.getRootDir().getAbsolutePath() +"/mpaas_bundles.gradle"
     apply from: rootProject.getRootDir().getAbsolutePath() +"/mpaas_apis.gradle"
     ```

     并删除工程根目录下的 mpaas_bundles.gradle 和 mpaas_apis.gradle 文件，但需要注意，删除 mpaas_apis.gradle 文件可能导致编译失败，您需要按照下文在子 module 中修改配置。

4. 如果您需要在子 module 中调用 mPaaS 组件 API：
   - 对于 Portal + Bundle 模式的工程，您需要在 Bundle 工程子 module 下的 build.gradle 文件中添加：

     ```
     apply plugin: 'com.alipay.apollo.baseline.update'
     ```

   - 对于单 Portal 工程模式，您需要在子 module 下的 build.gradle 文件中删除：

     ```
     apply from: rootProject.getRootDir().getAbsolutePath() +"/mpaas_apis.gradle"
     ```

     并添加：

```
apply plugin: 'com.alipay.apollo.baseline.update'
```

5. 如果旧依赖中有为您定制的库，您还需要 添加定制依赖 。
6. 如果由于库冲突导致编译失败，您可以 移除 mPaaS 组件单个依赖 。

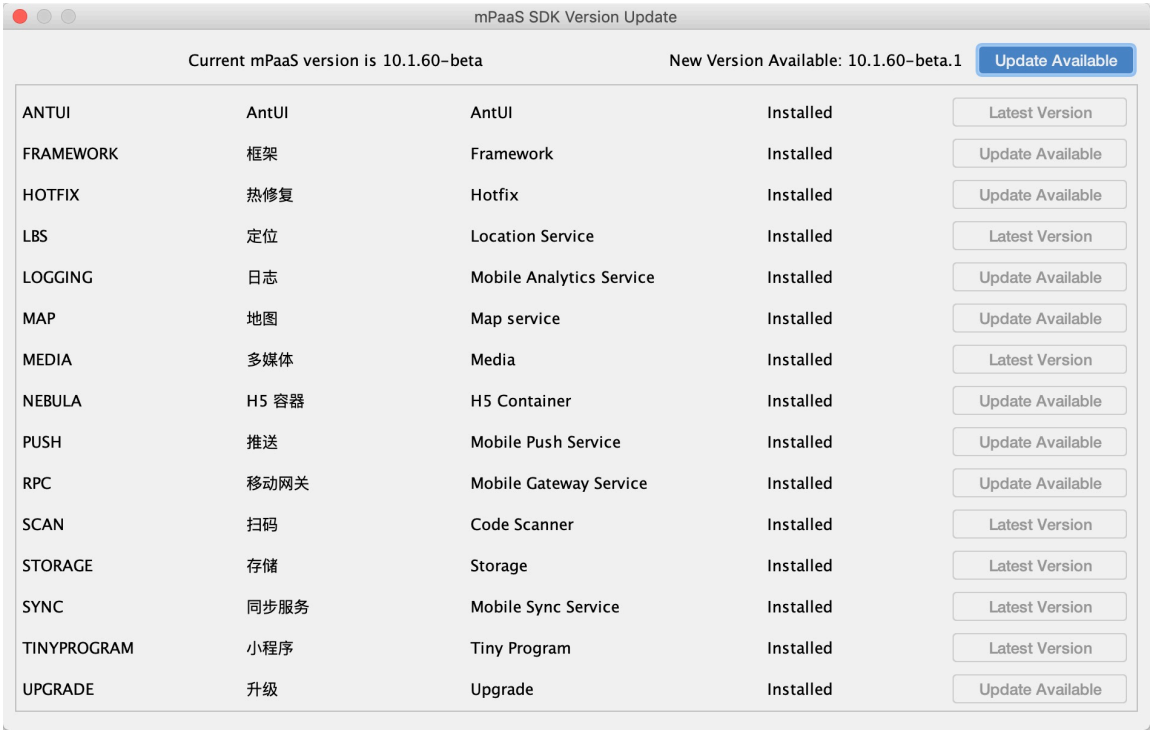
升级 SDK 版本

1. 在 Android Studio 中选择 **mPaaS** > **基线升级**，您将看到当前 SDK 版本信息。
2. 点击版本下拉框，选择一个新版本，然后点击 **OK** 按钮，即可升级 mPaaS SDK 版本。

升级单个组件

新版

1. 在 Android Studio 中选择 **mPaaS** > **组件升级**，您将看到组件列表。
2. 查看组件状态，进行升级操作，若右上角有提示可更新，那么点击之后就能更新了。



旧版

1. 在 Android Studio 中选择 **mPaaS** > **组件升级**，您将看到组件列表。
2. 查看组件状态，进行升级操作：
    - 若为 **最新版**，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。

添加定制依赖

- 如果您首次使用 **组件管理** 管理组件但未升级 SDK，您只需将定制库写在 Portal 工程主 module 下 build.gradle 文件中的 dependencies 节点下，例如：

```
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'
```

- 如果您首次使用 **组件管理** 管理组件且升级了 SDK，或使用 **基线升级** 升级了 SDK，您的定制库可能需要基于新版本重新定制，请 提交工单 或联系 mPaaS 支持人员确认，重新定制或确认无需重新定制后，您可按照上文添加定制依赖。

4.2.5 编译打包

本文介绍组件化工程的编译与打包功能。

操作方法

请使用 mPaaS Android Studio Plugin 提供的打包功能：

- 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，参见 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，**打 debug 包** 勾选框将失效。

Custom Params

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

4.3 注册通用组件

4.3.1 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>
```

4.3.2 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

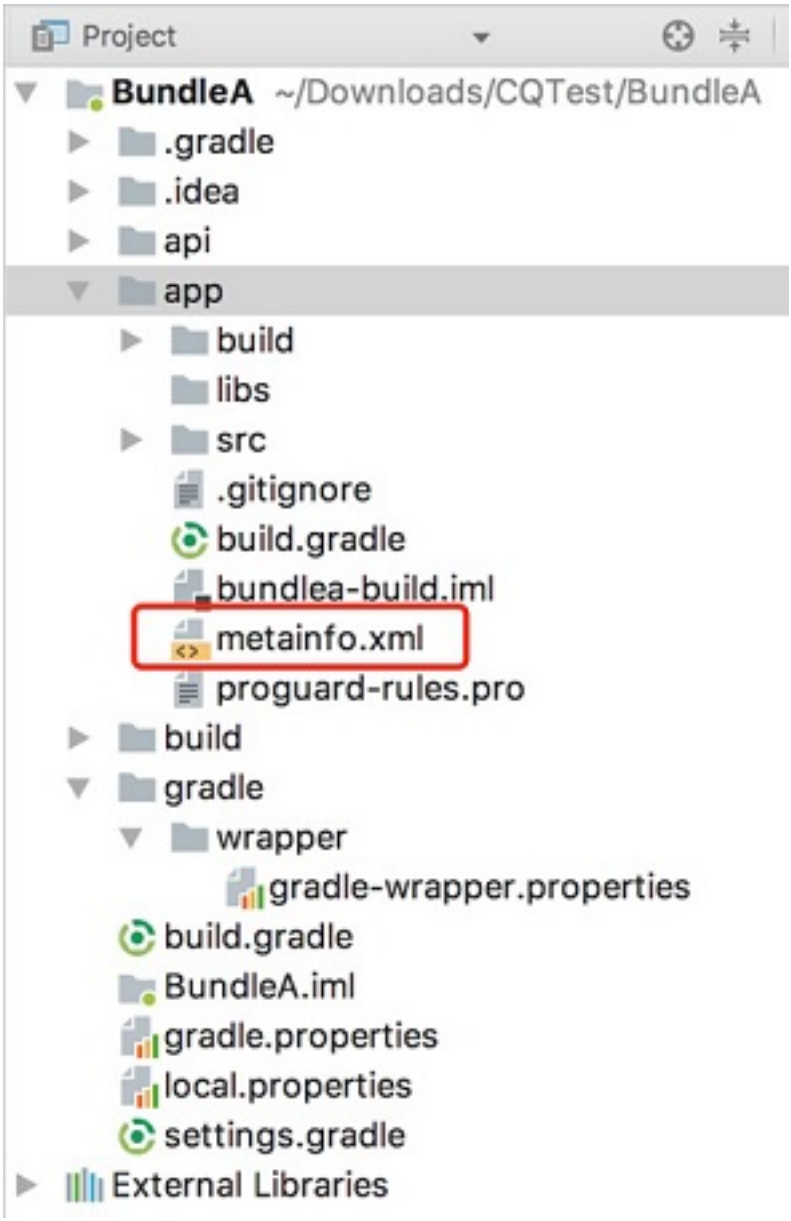
ActivityApplication 的创建、销毁等一系列逻辑完全由 mPaaS 框架来管理。业务方只需要处理其收到的参数并管理自己业务下的 Activity，这样业务方和调用方之间被有效的隔离开来。业务方和调用方只需要协调调用的参数，使得依赖更加轻量。

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：

```
// 若 SDK 版本 < 10.1.32，则如下获取 MicroApplicationContext 对象：
MicroApplicationContext context = LauncherApplicationAgent.getInstance().getMicroApplicationContext();
// 若 SDK 版本 ≥ 10.1.32，则如下获取 MicroApplicationContext 对象：
MicroApplicationContext context = MPFramework.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);
```

4.3.3 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。

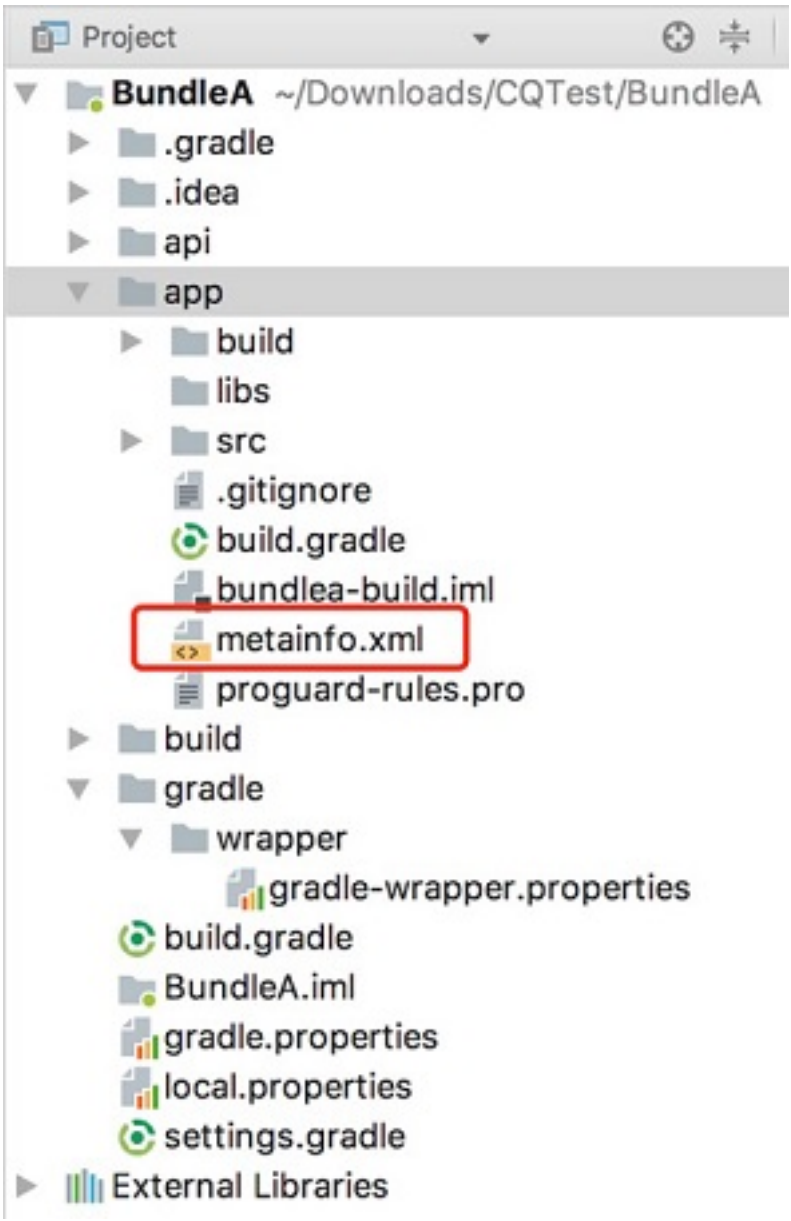
外部调用都通过 MicroApplicationContext 的 findServiceByInterface 接口，通过interfaceName 获取相应的服务。对于 Bundle 使用来说，只暴露服务抽象接口类，即 interfaceName 中定义的类，抽象接口类会定义在接口包中。

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();
```

4.3.4 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：

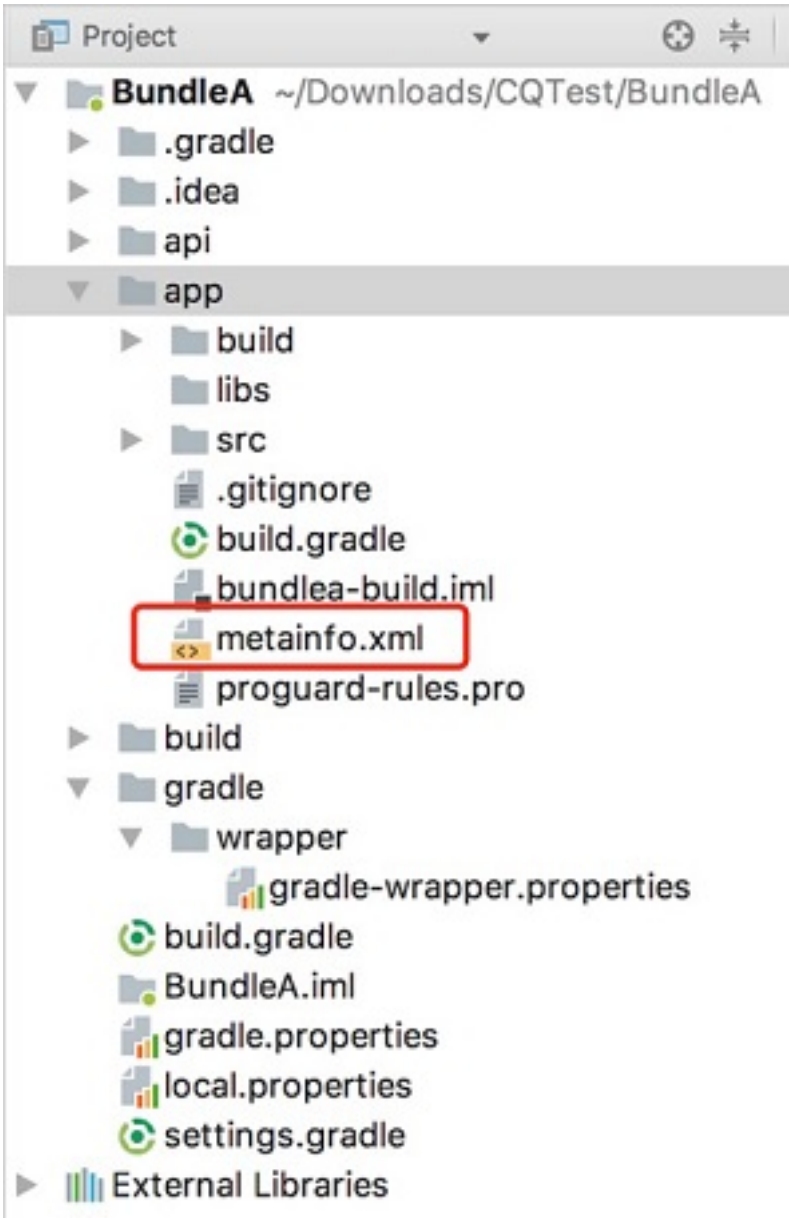
```
IntentFilter filter = new IntentFilter();
filter.addAction(MsgCodeConstants.FRAMEWORK_BROUGHT_TO_FOREGROUND);
filter.addAction(MsgCodeConstants.FRAMEWORK_ACTIVITY_USERLEAVEHINT);
LocalBroadcastManager.getInstance(this).registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context 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"));
```

4.3.5 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

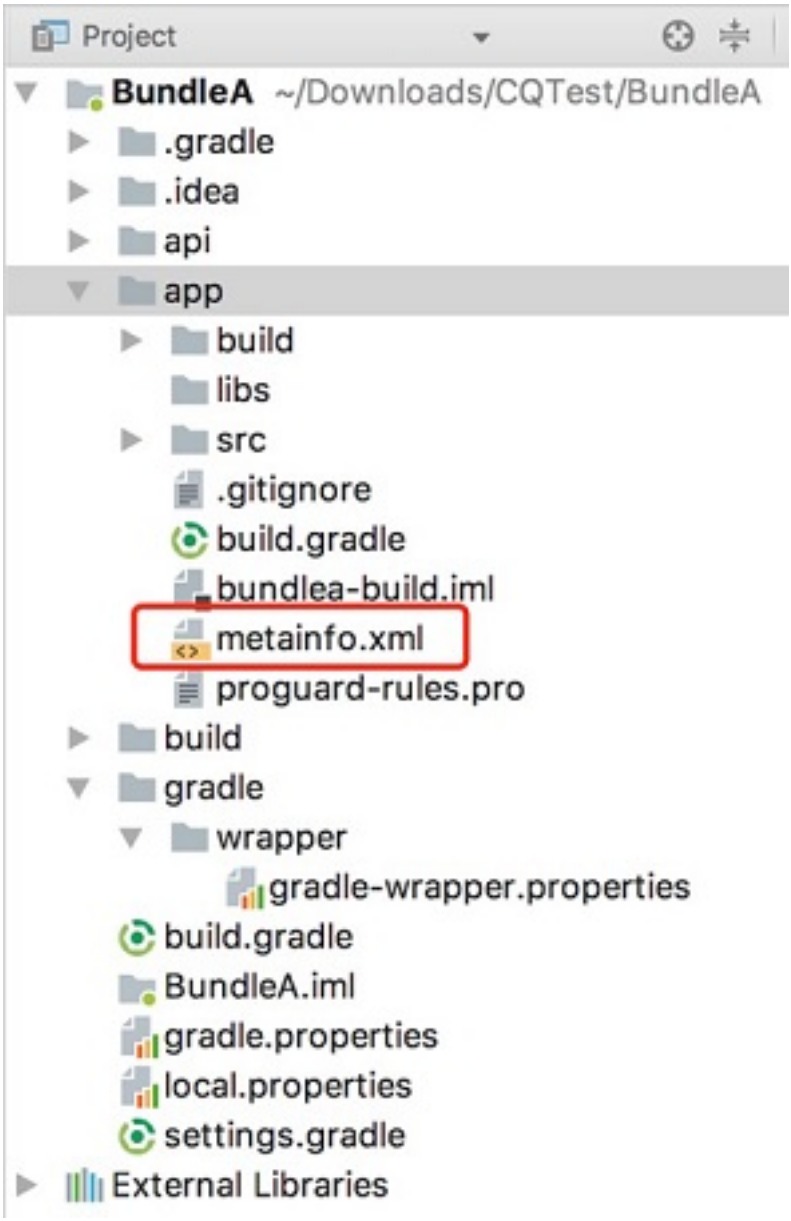
因为 Pipeline 的调用是由框架触发，使用者只需要在 metainfo 里指定相应的时机。

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();
}
}
```

4.4 使用 Material Design

4.4.1 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'
```

4.4.2 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 的版本号为 3.0.0.5.45 及以上。如：

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

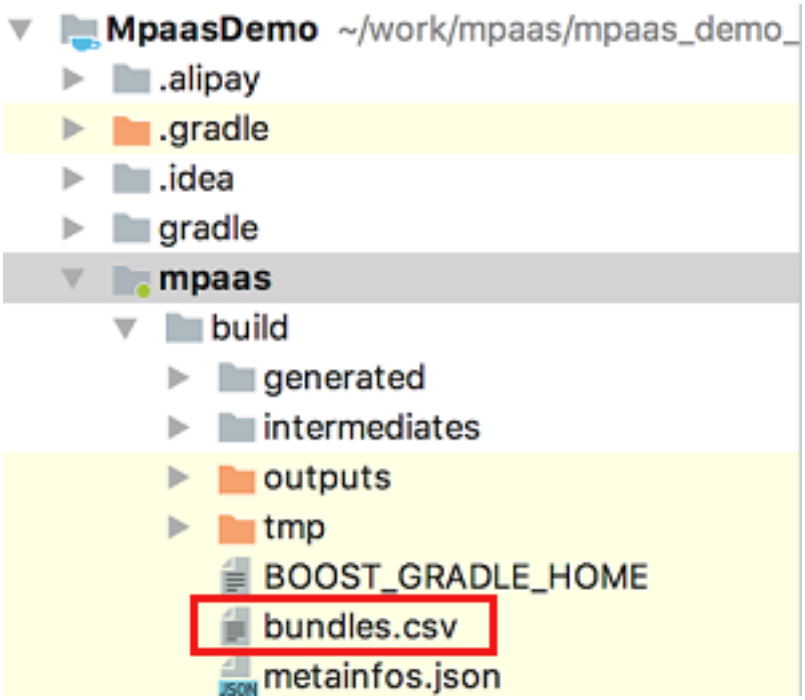
检测步骤

1. 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。
2. 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。

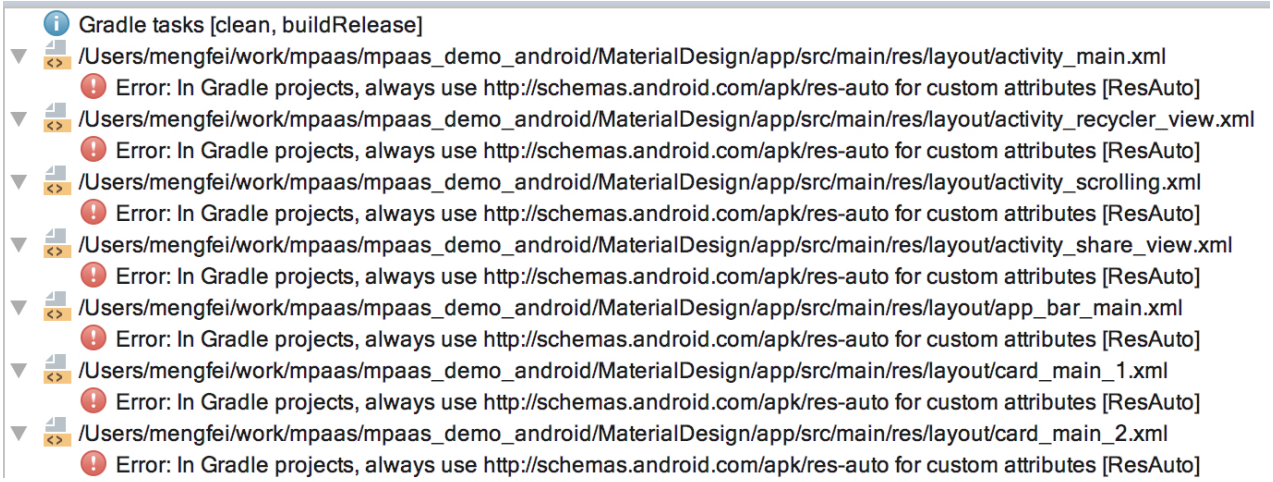


2. 将 PackageId 列进行排序，查看有无重复的 Package ID；确保没有重复的 Package ID 后再重新编译。

使用另一个 Bundle 中的资源

场景示例

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

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. 配置 Portal 工程。在 Portal 中引入资源 Bundle 的信息：

```
// 引入提供资源的 bundle
bundle"com.mpaas.demo.materialdesign:materialdesign-build:1.0-SNAPSHOT:raw@jar"
manifest"com.mpaas.demo.materialdesign:materialdesign-build:1.0-SNAPSHOT:AndroidManifest@xml"
// 为了在编译时能找到资源，还需要 provided 该 bundle 的 jar 包
provided '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，参见 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>
```

则您需要：

- 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'
```

4.5 迁移至 Gradle 3.0

如果您当前的 mPaaS 是基于 Android Plugin for Gradle V2.x 的工程，您可以将 V2.x 版本迁移至 V3.0.x。通过本文了解迁移步骤和迁移故障排查信息。此外，您还可以下载迁移后的示例工程预览迁移结果。

前置条件

- 确保 mPaaS 是基于 Gradle V2.x 的工程。
- 您已搭建 Python 2.7 运行环境。

操作步骤

以下迁移步骤假设将 Gradle 2.x 版本迁移到 3.0.0.4.5 版本：

1. 点击这里 下载 Python Gradle 2.x 迁移到 Gradle 3.x 版本的 transform_tools.py.zip 脚本工具。
2. 解压您下载的 transform_tools.py.zip 工具。
3. 打开命令行窗口，使用 cd 命令定位到解压后的工具文件夹下。
4. 执行 python transform_tools.py <yourpojectpath> 命令，其中 <yourpojectpath> 是您的工程根路径，如下图所示：

示例工程

参考 获取代码示例 ，下载迁移后的示例工程。

工具迁移失败故障排查

如果工具迁移失败，根据以下步骤依次排查失败原因并做调整：

1. 确认 Java Development Kit (JDK) 为 1.8 版本。
2. 查看是否已经设置 mPaaS 所需要编译环境，如 Python 2.7 等。
3. 检查是否已有 BOOST_GRADLE_HOME 和 ANDROID_HOME 环境变量，并且已经安装了 Android SDK build-tools 26.0.2。

5 mPaaS Inside 接入方式

5.1 简介

mPaaS 已经提供了 Portal 和 Bundle 的接入方式，并且提供了一套完整的组件化方案。组件化为我们提供了解耦的便利，但同时也产生引入第三方 SDK 不便捷的问题。针对此类问题，我们推出了新的接入方式 —— mPaaS Inside。mPaaS Inside 的接入方式与原生工程接入 aar 的方式几乎一致，而且可以继续使用 mPaaS 提供的能力，适合于对 Bundle 组件化方案没有特别需求却急需使用 mPaaS 能力的客户，降低了开发者的接入成本，让开发者更轻松地使用 mPaaS。

mPaaS Inside 工程和 Portal、Bundle 工程的关系

mPaaS Inside 工程和 Portal 工程并列，支持 Bundle 工程，但是在mPaaS Inside 工程中不再推荐将业务代码放入 Bundle 中。您可以使用bundle来引入 bundle 包，它会自动将所有的 bundle 都打到一个工程里去。您可以在 mPaaS Inside 工程中自由地使用资源，包括但不限于android-support-library、databinding。

功能	Inside	Portal Bundle
使用 support 等 aar	无需处理，支持	需要经过特殊处理
databinding	目前支持v1	不支持
xml 使用资源	直接使用	需要特殊命名空间
apt	无需处理，支持	需要把生成的类拷贝出来
组件化	轻量，基于 library module	基于 bundle 隔离
bundle 支持	支持	支持
mPaaS 通用组件	基于 apt 支持（60基线）	完整支持

mPaaS Inside 工程和原生工程的不同之处

mPaaS Inside 工程更加拥抱社区，可以使用大部分社区提供的能力，所以和原生工程的差异很少。此处仅列出和社区方式的细小差异：

1. **需要使用 mPaaS 提供的 android-gradle-plugin**
   因为原先 mPaaS 提供的组件均是使用 bundle 的方式，因此，在 mPaaS Inside 工程中引入 mPaaS 提供的组件的时候，您需要使用bundle的指令引入。如果您原先接入了 mpaas easyconfig 这个插件的话，您感知不到任何区别。
2. classpath 'com.android.boost.easyconfig:easyconfig:2.3.0'
3. **android-gradle-plugin 依然强依赖 Google 提供的插件版本**
   因为兼容性问题，该最新版本依然强依赖 Google 提供的 Android Gradle Plugin 3.0.1 版本和

Gradle 4.4 版本。

classpath 'com.android.tools.build:gradle:3.0.1'

接入最新版本的 Android Gradle Plugin 已经在计划当中，会在不久之后发布。

4. **在 android module 工程中使用 mPaaS 组件**

   如果您接入了 easyconfig，那么您只需在 module 工程中应用插件即可。easyconfig 插件会自动将我们的库使用compileOnly(provided)的方式引入到您的工程中去。

   apply plugin: 'com.alipay.apollo.baseline.update'

继续使用 Bundle 组件

在 mPaaS Inside 工程中您可以继续使用 bundle 组件，只需要在您的工程中添加相关的声明即可。

```
bundle"your_group:your_artifact:version@jar"
```

5.2 接入流程

5.2.1 接入流程简介

如果采用 **mPaaS inside 接入方式**，您需要完成以下通用步骤以完成接入流程：

1. 配置开发环境 。
2. 在控制台创建应用 。
3. 客户端创建新工程 或 迁移原生工程至 mPaaS Inside 。
4. 管理组件依赖 。
5. 引入 mPaaS Inside 框架.
6. 编译打包 。

5.2.2 客户端创建新工程

关于此任务

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
3. 确认您的 SDK 版本 为 10.1.60-beta 或更新。

操作步骤

具体创建步骤如下：

1. 在 Android Studio 中，点击 **File** > **New** > **mPaaS Project** > **Inside Project** 或者在欢迎页面点击 **新建 mPaaS Inside 工程**。
2. 填写 **Application name**（应用名，假定为 MPaaSDemo，下同）、**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 > 组件管理 功能添加所需依赖。

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 inside 接入方式 > 简介 ：介绍 **mPaaS inside 工程与 Portal、Bundel 工程的关系** 、**mPaaS inside 工程与原生工程的不同之处**以及如何 **在 mPaas inside 工程中继续使用 Bundle 组件**。

5.2.3 迁移原生工程至 mPaaS Inside

如果已有基于 Android 原生框架的应用，您可以使用 mPaaS 插件的功能将其迁移至 mPaaS Inside 框架。

前置条件

- 您已完成开发环境的配置。更多信息，请参考 配置本地开发环境 。
- 您已经在控制台创建了应用并下载到了配置文件。更多信息，请参考 创建应用 。

操作步骤

1. 打开 Android Studio。
2. 点击 **mPaaS** > **安装 mPaaS Inside** 菜单。
3. 在弹出的对话框中，配置 **mPaaS Config Location**，选择 mPaaS 配置文件的路径，选择完毕后，其他配置项将自动填写。
4. 点击 **Finish** 按钮，等待迁移完成。

   说明：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，解决方法见下方的 查看及解决冲突 。
5. 检查 gradle.properties 里面是否已经有quinoxless=true。

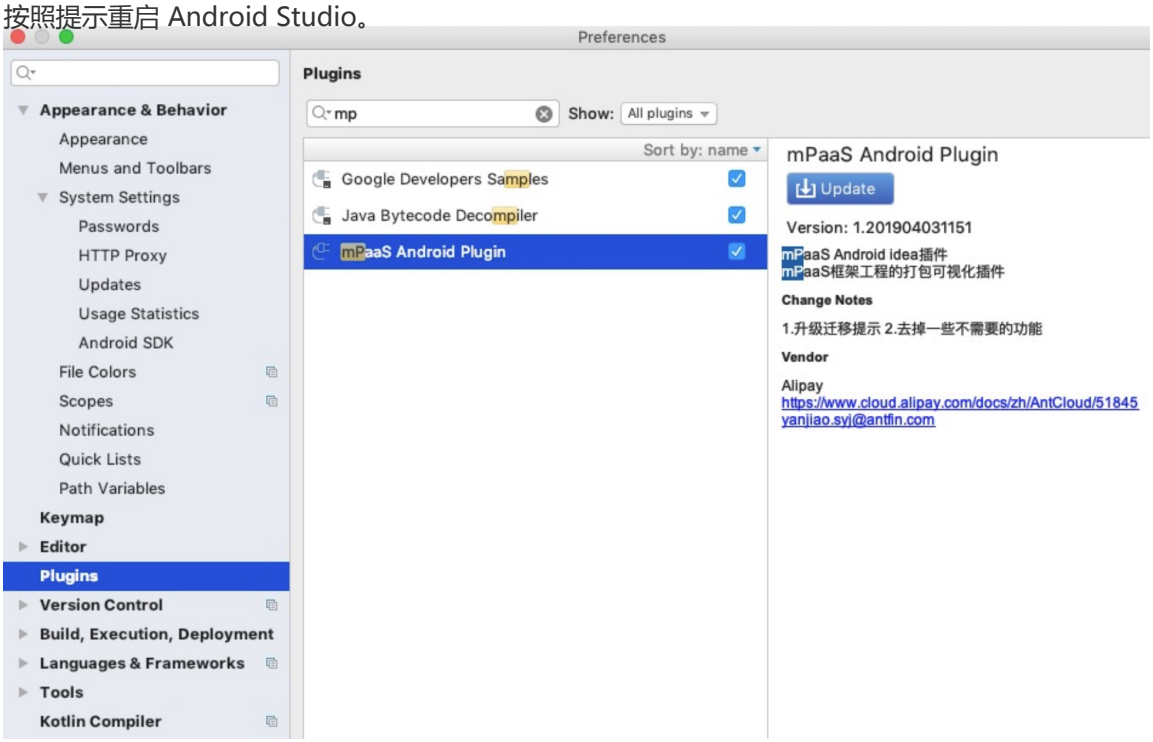
5.2.4 管理组件依赖

升级 IDEA 插件

说明：如果已是最新版本可跳过此升级。

为了更便捷地升级 mPaaS SDK 基线和组件，您需要先升级 mPaaS IDEA 插件至最新版本：

1. 在 Android Studio 中选择 **Preferences** > **Plugins**。
2. 搜索关键字 mPaaS，在下方结果中找到 **mPaaS Android Plugin**，并点击其右侧的 **Update** 进行升级。
3. 按照提示重启 Android Studio。



管理组件依赖

为了使用 mPaaS 组件 ，您需要在 mPaaS Inside 工程中添加对应组件的依赖：

说明：如果您在创建 mPaaS 工程时已选择过需要使用的组件，您仍可以按照下文步骤增删组件。

使用 IDEA 插件增删依赖步骤为：

1. 在 Android Studio 中选择 **mPaaS** > **组件管理**，如果您此前未使用过 IDEA 插件管理组件依赖，您会先进入 **选择 mPaaS 基线版本** 来选择版本：

2. 随后您将看到组件列表：

3. 您可以根据需要增删组件，install 为添加，uninstall 为删除，框架和必备组件无法被删除。

首次使用插件管理依赖

如果您此前未使用过 IDEA 插件管理组件依赖，当您首次使用 **组件管理** 功能添加完组件后，您还需要检查或修改以下配置：

1. 检查 mPaaS Inside 工程根目录 build.gradle 文件，确保包含以下依赖且不低于以下版本：

```
buildscript {
...
dependencies {
classpath 'com.android.boost.easyconfig:easyconfig:2.1.6'
}
}
```

检查工程主 module 下的 build.gradle 文件，确保包含以下内容：

```
apply plugin: 'com.alipay.portal'
portal {
allSlinks true
mergeAssets true
}
```

```
apply plugin: 'com.alipay.apollo.baseline.update'
mpaascomponents{
excludeDependencies=[]
}
```

若需要在子 module 中调用 mPaaS 组件 API，则在工程子 module 下的 build.gradle 文件中添加：

```
apply plugin: 'com.alipay.apollo.baseline.update'
```

4. 如果旧依赖中有为您定制的库，您还需要 添加定制依赖 。
5. 如果由于库冲突导致编译失败，您可以 移除 mPaaS 组件单个依赖 。

升级 SDK 版本

1. 在 Android Studio 中选择 **mPaaS** > **基线升级**，您将看到当前 SDK 版本信息。
2. 点击版本下拉框，选择一个新版本，然后点击 **OK** 按钮，即可升级 mPaaS SDK 版本。

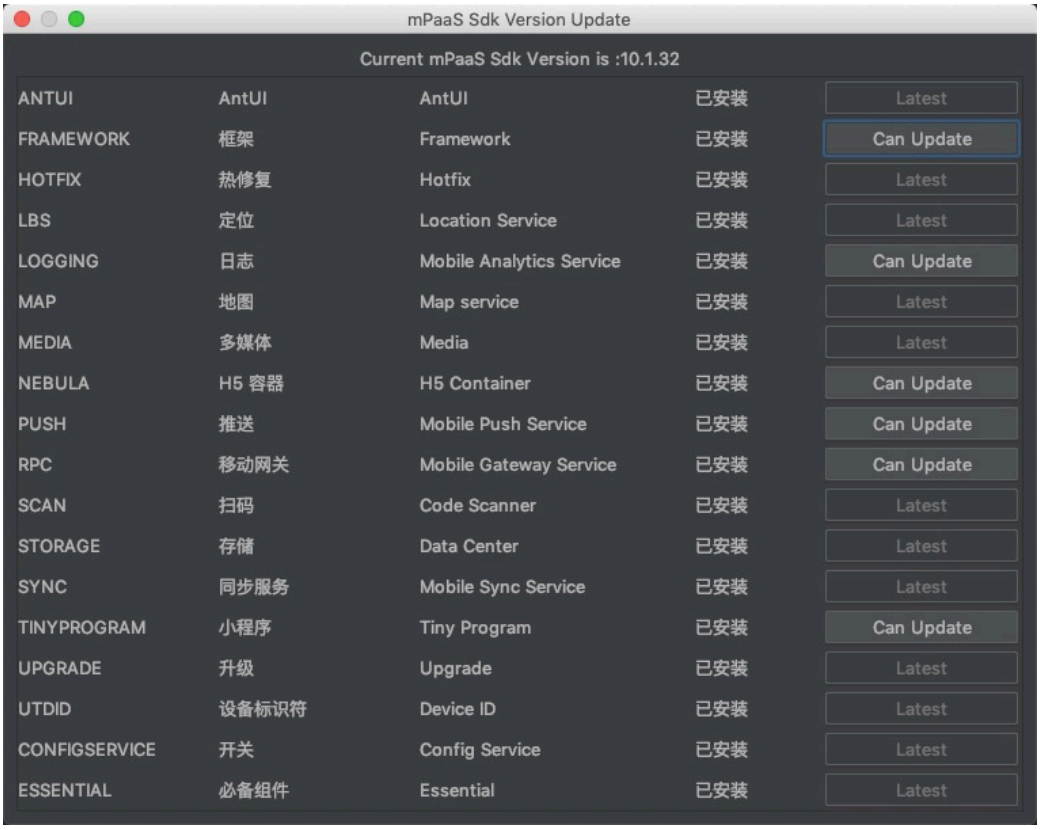
升级单个组件

新版

1. 在 Android Studio 中选择 **mPaaS** > **组件升级**，您将看到组件列表。
2. 查看组件状态，进行升级操作，若右上角有提示可更新，那么点击之后就能更新了。

旧版

1. 在 Android Studio 中选择 **mPaaS** > **组件升级**，您将看到组件列表。
2. 查看组件状态，进行升级操作：
    - 若为 **最新版**，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。



添加定制依赖

- 如果您首次使用 **组件管理** 管理组件但未升级 SDK，您只需将定制库写在工程主 module 下 build.gradle 文件中的 dependencies 节点下，例如：

```
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'
```

- 如果您首次使用 **组件管理** 管理组件且升级了 SDK，或使用 **基线升级** 升级了 SDK，您的定制库可能需要基于新版本重新定制，请 提交工单 或联系 mPaaS 支持人员确认，重新定制或确认无需重新定制后，您可按照上文添加定制依赖。

5.2.5 引入 mPaaS Inside 框架

初始化 Application 对象

mPaaS 框架需要进行一些初始化操作，因此需要对 Application 对象进行少许的改造，所以您需要根据是否使用热修复功能以采取不同的接入操作。

不使用热修复接入

如果不需要使用 **热修复** 功能，那么您需要在 Application 中添加如下代码:

```
@Override
protected void attachBaseContext(Context base) {
super.attachBaseContext(base);
QuinoxlessFramework.setup(this, new IInitCallback() {
@Override
public void onPostInit() {
// 在这里开始使用 mPaaS 功能
}
});
}

@Override
public void onCreate() {
super.onCreate();
QuinoxlessFramework.init();
}
```

使用热修复接入

如果需要使用 **热修复** 功能，您还需要完成以下两步操作。

需要将您 Application 对象重新继承为 QuinoxlessApplicationLike，并注意将这个类防混淆。此处以“MyApplication” 为例。

```
@Keep
public class MyApplication extends QuinoxlessApplicationLike implements
Application.ActivityLifecycleCallbacks {

private static final String TAG ="MyApplication";

@Override
protected void attachBaseContext(Context base) {
super.attachBaseContext(base);

Log.i(TAG,"attacheBaseContext");

}

@Override
public void onCreate() {
super.onCreate();
Log.i(TAG,"onCreate");
registerActivityLifecycleCallbacks(this);
}

@Override
public void onMPaaSFrameworkInitFinished() {
LoggerFactory.getTraceLogger().info(TAG, getProcessName());
}

@Override
public void onActivityCreated(Activity activity, Bundle savedInstanceState) {
Log.i(TAG,"onActivityCreated");
}
```

```
@Override
public void onActivityStarted(Activity activity) {

}

@Override
public void onActivityResumed(Activity activity) {

}

@Override
public void onActivityPaused(Activity activity) {

}

@Override
public void onActivityStopped(Activity activity) {

}

@Override
public void onActivitySaveInstanceState(Activity activity, Bundle outState) {

}

@Override
public void onActivityDestroyed(Activity activity) {

}}
```

2. 在 AndroidManifest.xml 文件中将 Application 对象指向 mPaaS 提供的Application对象。 将您刚刚生成的 “MyApplication” 类添加到 key 为 mpaas.quinoxless.extern.application 的 meta-data 中。示例如下：

```
<application
android:name="com.alipay.mobile.framework.quinoxless.QuinoxlessApplication">
<meta-data
android:name="mpaas.quinoxless.extern.application"
android:value="com.mpaas.demo.MyApplication"
/>
</application>
```

后续事项

查看及解决冲突

1. 查看冲突：请在 conflict.json 文件中查看冲突项目。
2. 解决冲突

开源库名	mPaaS 库名	冲突解决方案
	com.alipay.android.phone.thirdparty:fastjson-	

fastjson	build	建议使用 mPaaS fastJson，该版本稳定且安全性高。
alipayInside	rpc、UTDID(com.alipay.android.phone.thirdparty:utdid-build)	升级到无 UTDID 版本 SDK 即可，点击这里查看详情。
supportv4	com.alipay.android.phone.thirdparty:androidsupport-build	如果仅需要 supportV4，建议使用 mPaaS 版本。如果是使用官方的 appcompat，则需要客户手动实现自动化埋点（在 fragment 中调用 mPaaS 在 supportv4 中定制的 fragment 的方法即可）。
appcompat	com.alipay.android.phone.thirdparty:androidsupport-build	使用官方 appcompat，则需要客户手动实现自动化埋点（在 fragment 中调用 mPaaS 在 supportv4 中定制的 fragment 的方法即可）。
libdatabase_sqlcrypto.so 统一存储的 so 库冲突	storage 模块	删掉依赖的三方包中的 so 文件：解压 zip 包，然后删掉 so 文件，再重新压缩即可。
okio	com.alipay.android.phone.thirdparty:wire-build	可以提供一个没有 okio 的 RPC 模块，需要回归测试 mPaaS RPC 报文解析功能。
高德地图、定位（Amap location or Amap map）	com.alipay.android.phone.mobilecommon:lbs-build	建议使用 mPaaS 的模块。如果使用高德版本请注释 lbs-build 模块，并且回归测试定位模块。

引入 apache http client

您使用 RPC 或者**热修复**功能的时候，会调用到 Apache http client 相关的功能，您可以按照这个文档：https://developer.android.com/about/versions/pie/android-9.0-changes-28?hl=zh-cn#apache-p 来进行接入。具体来说，**不要使用** 导入 jar 包的方式，而是使用在 AndroidManifest.xml 加入

```
<uses-library android:name="org.apache.http.legacy"android:required="false"/>
```

以上组件的方式引入依赖。

移除 mPaaS 部分组件依赖

因为有些组件可能会与您目前接入的组件冲突，如果您需要采用移除 mPaaS 依赖的方法解决冲突，则需要在 build.gradle mpaascomponents 中排除依赖，示例如下：

```
mpaascomponents{
// when you want exclude mpaas dependencies ,you can add dependency ga to to excludeDependencies
// or set removed = true in mpaaspakcages.json
excludeDependencies=['com.alipay.android.phone.thirdparty:wire-build']
}
```

5.2.6 编译打包

本文介绍 mPaaS Inside 工程的编译与打包。

配置打包插件

mPaaS 通过 mPaaS Gradle 插件提供了接近原生的打包方式。mPaaS Gradle 插件有稳定版和测试版

（beta）两种版本。

基于 Android Gradle Plugin 3.0.1 版本的 mPaaS Gradle 插件是稳定版，但要求使用 4.4 版本的 Gradle。

```
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.8.3
```

基于 Android Gradle Plugin 3.5.3 版本，我们提供了 3.5.x 的版本的mPaaS Gradle 插件，目前最新的版本是 3.5.0-beta8。

```
classpath 'com.alipay.android:android-gradle-plugin:3.5.0-beta8'
```

说明：请仅在必要的时候使用 beta 版本，并且去除 apply plugin:'com.android.application'。

插件接入

如果您使用的是 Android Studio mPaaS 插件创建的新工程，或者是由原生工程转换的工程，那么应该已经执行了以下步骤，在此只需要进行检查确认。

1. 在项目根目录的 build.gradle 中引入这几个插件。

```
classpath 'com.android.boost.easyconfig:easyconfig:2.3.0'
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.8.3' //或者beta版本
```

2. 在主工程中应用插件。

```
apply plugin: 'com.alipay.portal'
apply plugin: 'com.alipay.apollo.baseline.update'
portal {
allSlinks true
mergeAssets true
}
mpaascomponents{
// 如果有不需要的 mpaas bundle，把他们加入到这个数组里
excludeDependencies=[
"com.alipay.android.phone.thirdparty:androidsupport-build"
]
}
```

在 gradle.properties 文件中有quinoxless=true。

确保根目录下已经有 mpaas_packages.json 文件。如果没有的话，请使用 Android Studio mPaaS 插件中的 **mPaaS** > **基线升级** 功能安装基线。

打包

您可以直接使用 Android Studio 提供的 Build 按钮直接进行打包，也可以使用 Gradle Wrapper 执行以下命

令采用脚本方式进行打包。

```
./gradlew clean assembleDebug
```

5.3 mPaaS Inside 使用外部资源（aar）

使用原生外部资源

在 mPaaS Inside 功能提供整合的情况下，您使用资源非常方便，像其他原生工程一样，只用在依赖中引入您需要的组件。

```
dependencies {
implementation"com.android.support:appcompat-v7:$support_version"
}
```

使用的时候，按照原生工程一样即可。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:aui="http://schemas.android.com/apk/res/com.alipay.mobile.antui"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<TextView
android:id="@+id/title_atb"
android:layout_width="match_parent"
android:layout_height="wrap_content"/>
</LinearLayout>
```

使用 mPaaS 资源（例如 antui）

使用 antui 的时候，您还是需要依照原先提供的方式引用，例如，在xml中使用，需要为 antui 增加前缀。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:aui="http://schemas.android.com/apk/res/com.alipay.mobile.antui"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<com.alipay.mobile.antui.basic.AUTitleBar
android:id="@+id/title_atb"
aui:rightIconResid="@com.mpaas.demo.sharedres:drawable/titlebar_right"
android:layout_width="match_parent"
android:layout_height="wrap_content"/>
</LinearLayout>
```

5.4 在 mPaaS Inside 中使用 mPaaS 的通用组件

为了兼容原来提供的组件化（路由）的方案，在 60 基线上，我们支持使用 apt 的方式使用以下 4 个组件：

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

说明：这 4 个组件的使用方式与在组件化接入中的使用方式相同，您可点击上方组件名查看详情。

使用组件

1. 在您的 library 和 application 工程中引入相关的依赖：

```
implementation 'com.mpaas.mobile:metainfo-annotations:1.0.0'
kapt 'com.mpaas.mobile:metainfo-compiler:1.0.1'
// 或者
annotationProcessor 'com.mpaas.mobile:metainfo-compiler:1.0.1'
```

2. 在您定义上述四个组件的地方，分别加上特定的 Annotation，Annotation 有如下四种:
   - @Application
   - @Service
   - @BroadcastReceiver
   - @Pipeline
     Annotation 中的参数与 metainfo.xml 定义的相同。例如，当您想使用 @Application 时，只需要如下的方式：

```
@Application(appId ="123123")
public class MicroApplication extends ActivityApplication {
}
```

没有 library module

如果您没有 library module，那么您只需要在您的 app module 工程中的任意一个类加上 @MetaInfoApplication 即可。

使用了 library module

如果您在 library module 工程中定义了上述 4 个组件：

1. 在 **任意** 类中声明一个 @MetaInfoLibrary，其中的参数传入 library module 的 packageName，例如：

```
@MetaInfoLibrary(applicationId=BuildConfig.APPLICATION_ID)
```

2. 在您的 app module 工程中的任意一个类加上 @MetaInfoApplication，依赖传入您在 library module 中

生成的 MetaInfoConfig.java，例如：

```
@MetaInfoApplication(dependencies={com.mylibrary.MetaInfoConfig.class})
```

混淆相关

需要把相关的类加入混淆的白名单，特别是 com.alipay.mobile.core.impl.MetaInfoConfig，可以使用以下命令：

```
-keep public class com.alipay.mobile.core.impl.MetaInfoConfig
```

6 原生接入方式

6.1 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

6.2 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

7 开发者工具

7.1 Android Studio mPaaS 插件

7.1.1 关于 mPaaS 插件

如图，mPaaS 插件提供多种开发辅助功能，包括：

功能	mPaaS 插件	
组件化	Inside	
编译打包	编译打包	Build、Build Setting
管理组件依赖	管理组件依赖	基线升级、组件升级、组件管理
迁移原生工程至 mPaaS 框架	迁移原生工程至 mPaaS Inside	安装 mPaaS Portal、安装 mPaaS Inside
热修复包		生成热修复补丁、合并补丁
加密图片		Generate YWJpg(Private) Generate YWJpg(Private Config File) Generate YWJpg RSA Generate YWJpg(With Config)

相关链接

- 配置开发环境 ：了解配置开发环境的方法，其中包含了 mPaaS 插件的安装方法。

- 更新及卸载 mPaaS 插件 ：对 mPaaS 插件进行更新或卸载。

7.1.2 加密图片

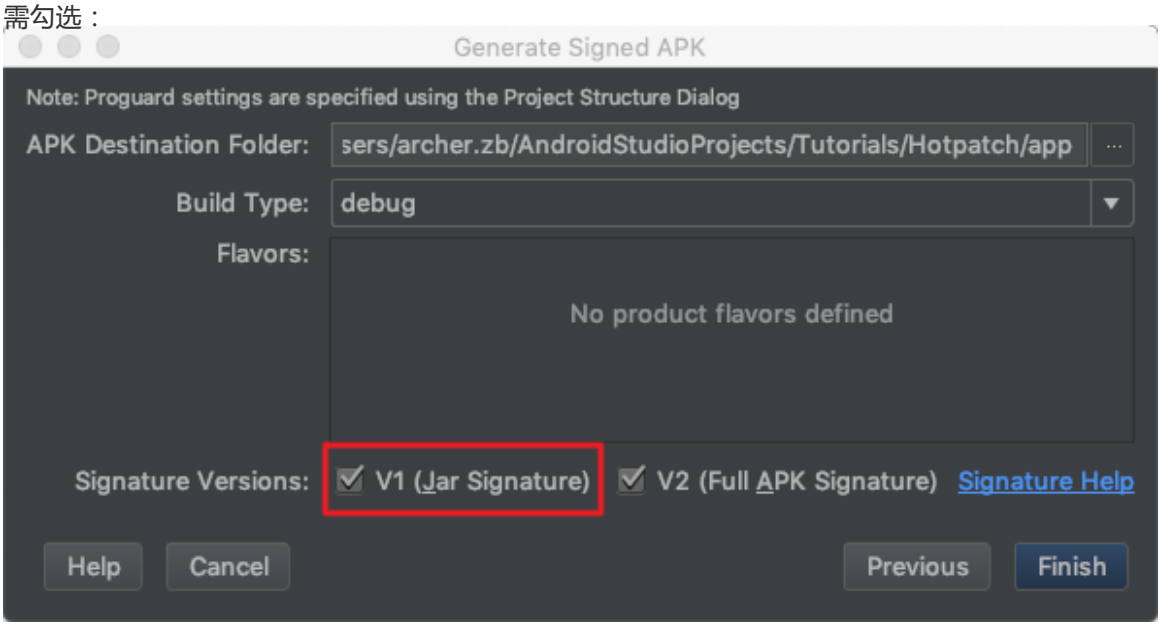
为了安全，mPaaS 某些组件（如热修复、消息推送）访问网络时需要对内容进行加密。具有特殊名称 yw_1222.jpg 的图片为加解密提供密钥信息。mPaaS 组件自动使用该图片进行加解密，您无需额外操作。

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 Portal 工程签名之后的 APK。具体的签名步骤，请参考 Android 官网。注意：

- 此处的 APK 应和 **发布版本** 的 APK 使用相同的签名文件。
- 给应用签名时，Signature Versions 中 **V1(Jar Signature)** 必须勾选，**V2(Full APK Signature)** 可按需勾选：



- 生成的加密图片只能用在该 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。关于签名 APK 的要求，请参见 准备 。

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。

专有云

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"-->
```

7.1.3 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 **生成热修复补丁** 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：接入 Android——基于 mPaaS 框架 。

生成热修复包

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

针对不同的 mPaaS 集成方式，选择对应的包，通过 mPaaS 插件的 **生成热修复补丁** 生成热修复包：

- **原生工程和 mPaaS Inside 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。

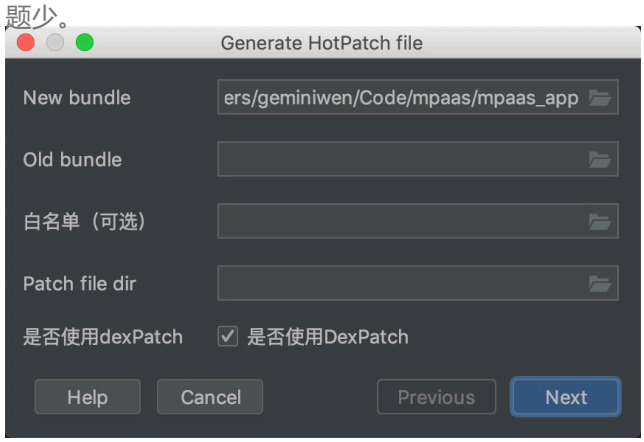
      提示：在 **New bundle** 栏，填写修复后的 apk 包的本地地址。在 **Old bundle** 栏， 填写有 bug 的 apk 包的本地地址。在 **白名单** 一栏输入白名单。

- **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。

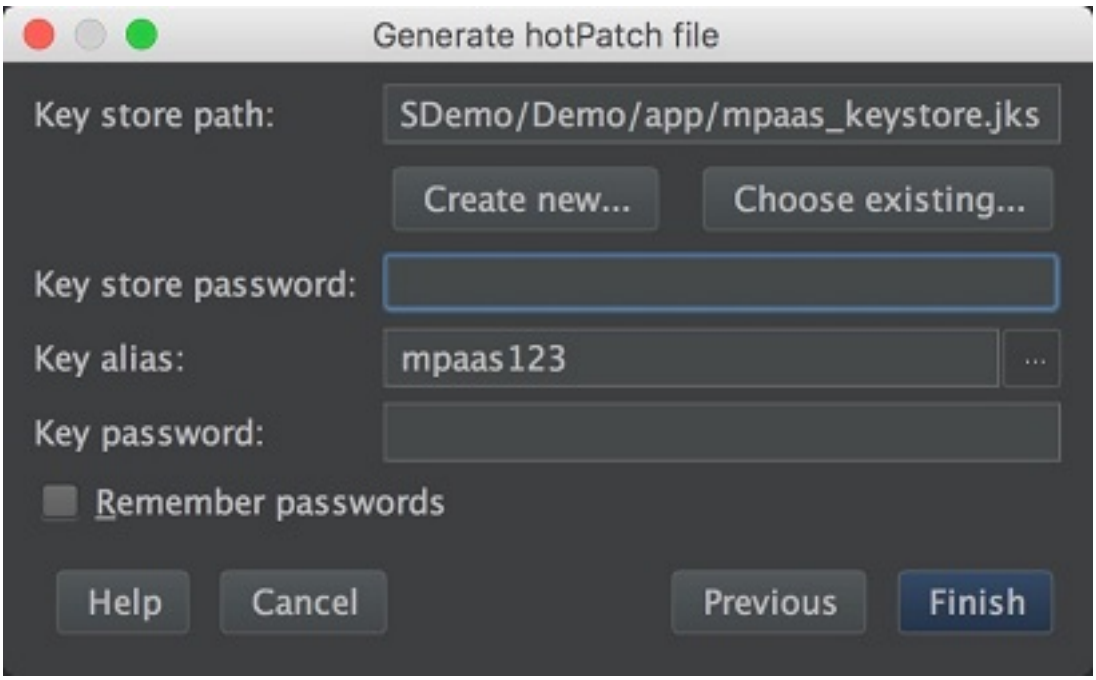
  提示：bundle 的输出路径为 bundle 的主 module 目录下的 build/intermediates/bundle/xxxx-raw.jar。如果是 release 包则没有 -raw。其中：

  - **New bundle**：选择修复 bug 的包路径。
  - **Old bundle**：选择有 bug 的包路径。
  - **白名单**：指定修复的类，使用原生工程和 mPaaS Inside 时强烈推荐这个功能，后文介绍白名单功能。
  - **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 **生成热修复补丁** 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。



热修复白名单功能

因为 apk 不像 bundle 一样只存在一个 dex，如果 apk 中有多个 dex，那么必须使用白名单功能指定需要修复的类，首先白名单文件是一个**文本文件**，可以存成任意格式（比如 txt）白名单文件的内容如下：

```
Lcom.mpaas.demo.MainActivity
HostDex: true
PatchType: dexpatch
```

以L开头，指定需要修复的类，每一个类一行，比如如果我需要修复多个类，就要像下面一样：

```
Lcom.mpaas.demo.MainActivity1
Lcom.mpaas.demo.MainActivity2
HostDex: true
PatchType: dexpatch
```

默认使用 dexpatch 的方式进行修复。把这个内容存成一个文件，比如filter.txt，然后在上面对话框中指定路径即可。

多 Bundle 补丁，需要合并热修复包

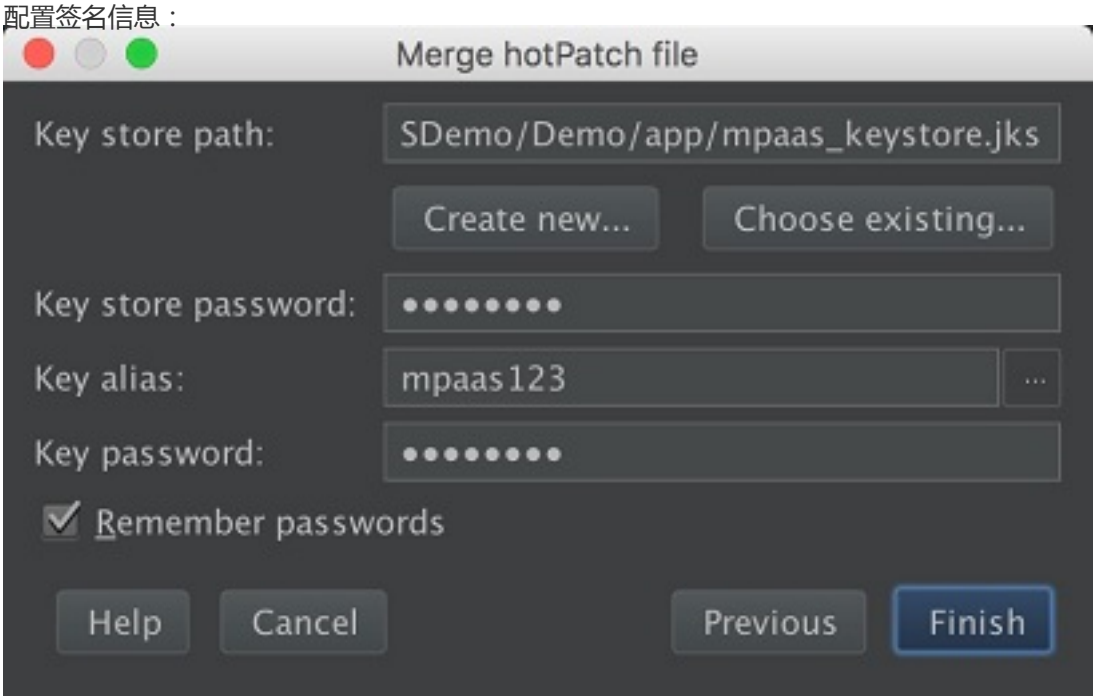
本节指针对使用 Bundle 工程的用户，mPaaS Inside 和原生工程用户可以在原先修复的基础上，相对于最原始未修复的包再打出一个补丁发布即可，可以ui

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。

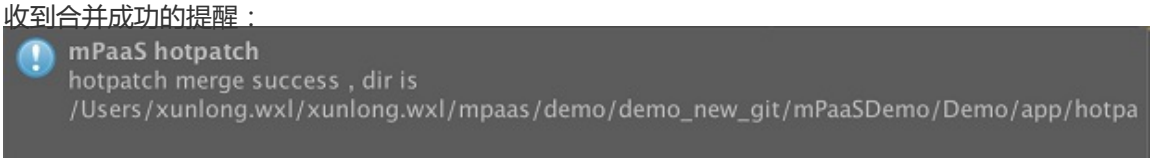
使用 mPaaS 插件的 **Merge Hotpatch**，通过以下步骤合并热修复包：

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。

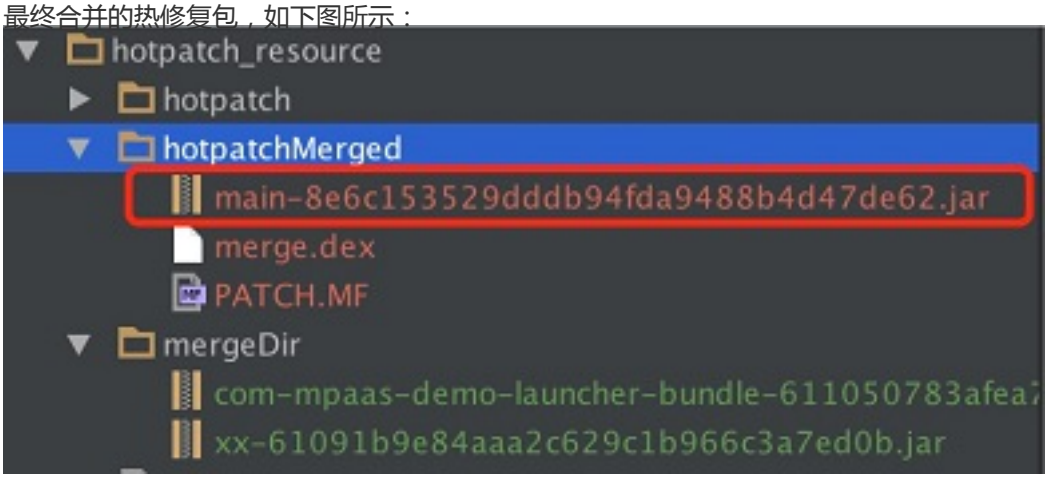
配置签名信息：



收到合并成功的提醒：



4. 最终合并的热修复包，如下图所示：



7.1.4 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"
```

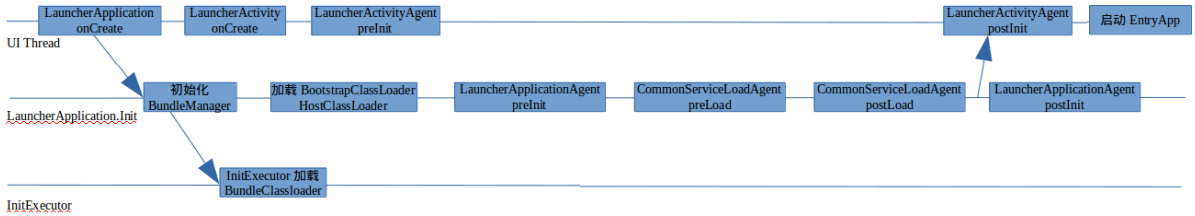
```
android:screenOrientation="portrait"
android:windowSoftInputMode="stateAlwaysHidden">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

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。当您在 bundle 工程中定义了这两个类时，在使用 ProGuard 进行代码混淆的时候则需要对这两个类进行防混淆设置，详情请参见 混淆 Android 文件 。

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

7.1.5 更新及卸载 mPaaS 插件

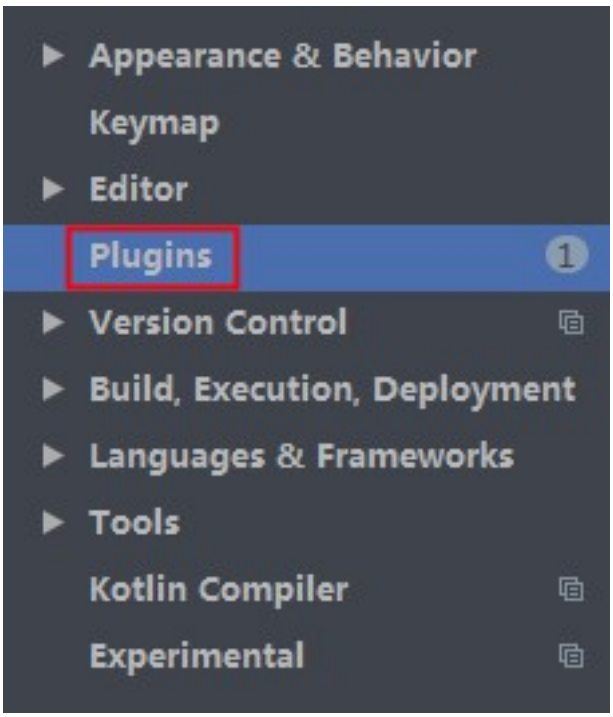
本文向您介绍如何更新及卸载 mPaaS 插件。

说明：本文档中的截图是从 Windows 环境下获得，MacOS 和 Linux 环境下操作流程与 Windows 环境下相似，此处不再赘述。

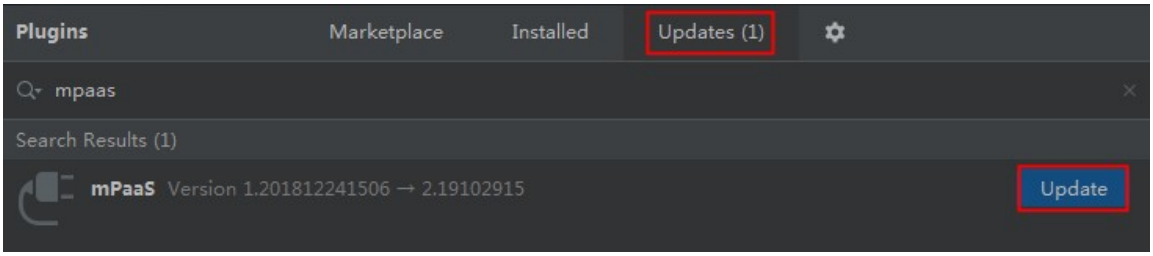
更新 mPaaS 插件

1. 打开 Android Studio，点击 **File** > **Settings**。

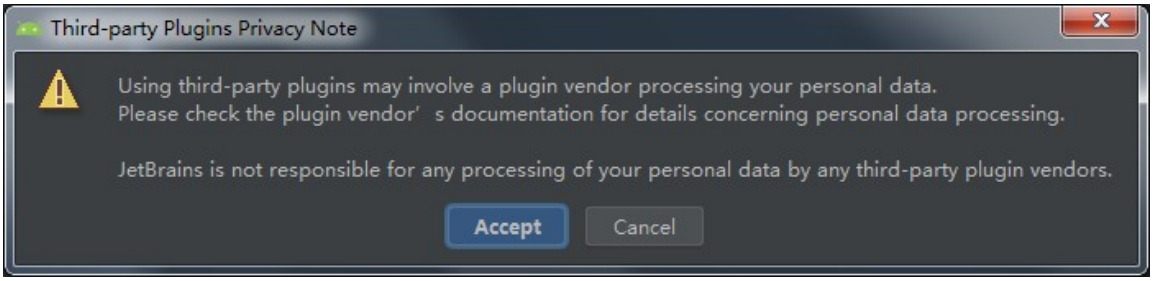
   在打开的 **Settings** 窗口的左侧导航栏，选择 **Plugins**。



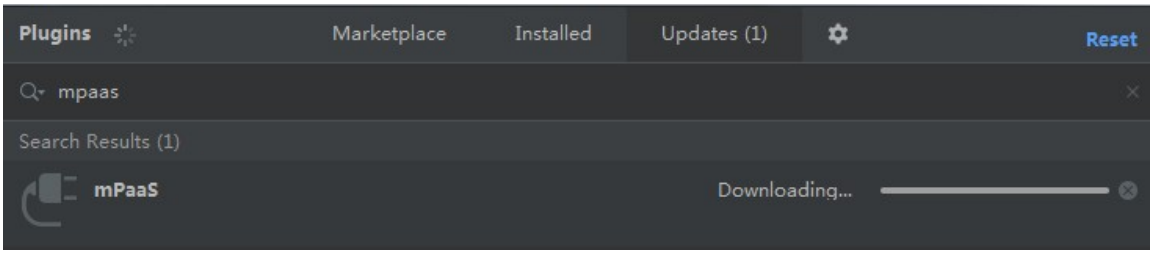
在窗口右侧，选择 **Updates** 选项卡，搜索并找到 mPaaS 插件，点击其右侧的 **Update**。



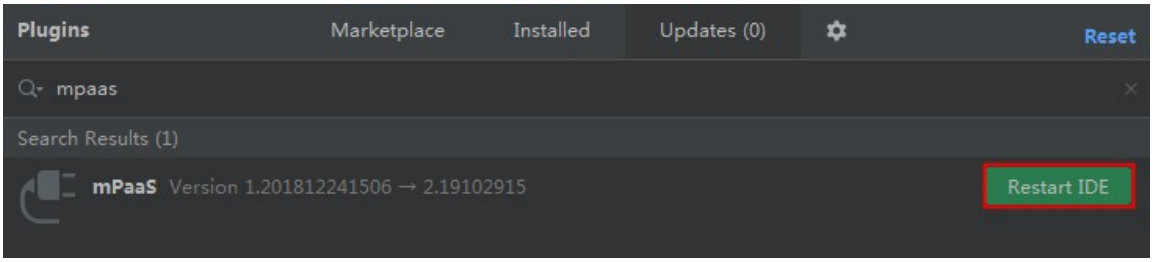
在弹出的第三方插件隐私说明中，点击 **Accept**。



随后，Android Studio 会自动下载 mPaaS 插件。



更新完毕后，点击 **Restart IDE**。并在弹出的确认窗口中点击 **Restart** 重启 Android Studio。

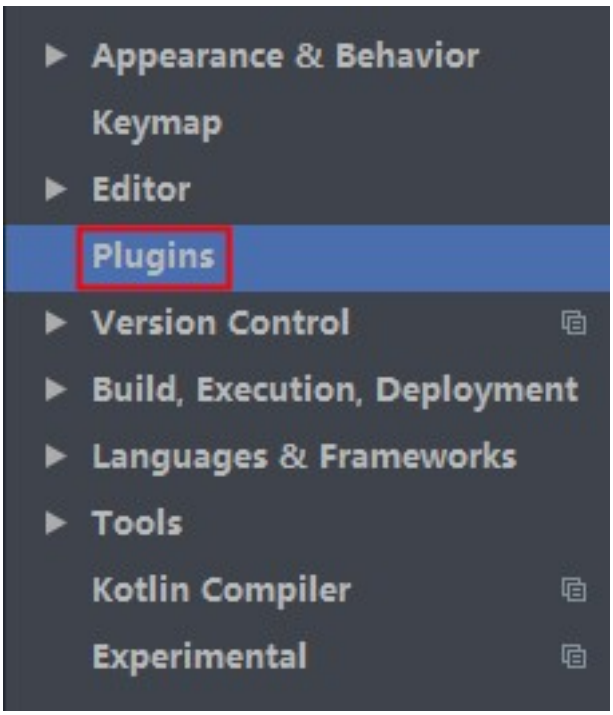


重启 Android Studio 后，重新进入 **File** > **Settings** > **Plugins**，您即可看到 mPaaS 插件已更新至最新版本。

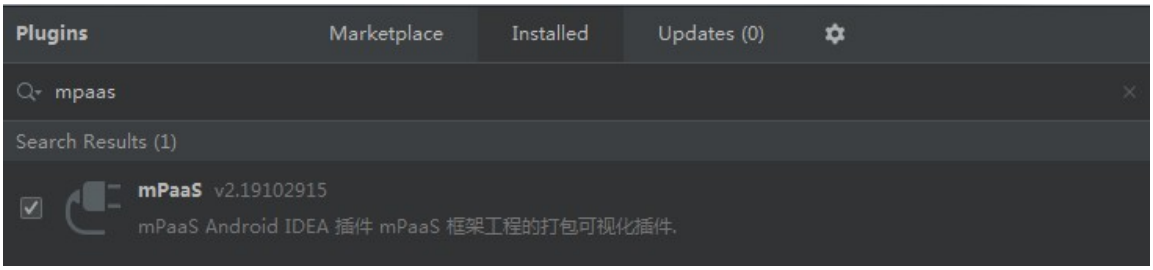
卸载 mPaaS 插件

1. 打开 Android Studio，点击 **File** > **Settings**。

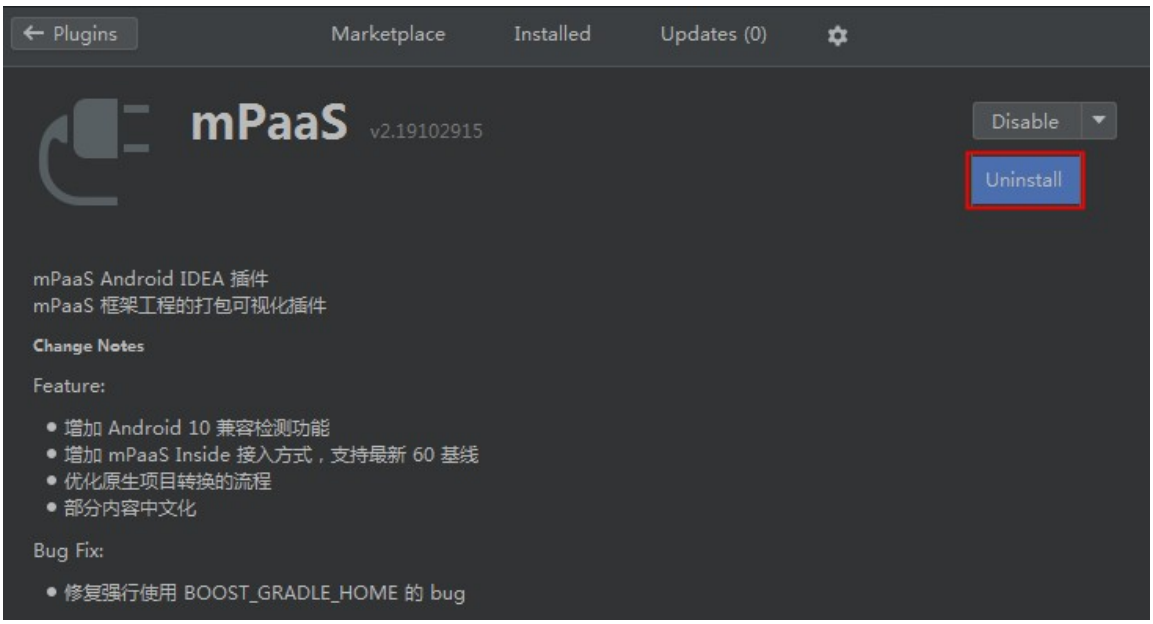
   在打开的 **Settings** 窗口的左侧导航栏，选择 **Plugins**。



在窗口右侧的 **Installed** 选项卡中，搜索并找到 mPaaS 插件，点击插件名，进入插件详情页。



在 mPaaS 插件详情页，点击窗口右上方 **Disable** 右侧的下拉箭头，选择 **Uninstall**。



在弹出的确认窗口中，点击 **Yes** 即可卸载 mPaaS 插件。

7.1.6 定制基线

通常情况下，我们提供的基线是面向所有客户的，如 10.1.20、10.1.32、10.1.60 。如果您需要定制一些 mPaaS 的功能，您需要向和您对接的人员提出需求，我们会按照您的需求为您定制基线。在交付的时候，mPaaS 的工作人员会向您提供定制基线的 ID，您只需要在 mPaaS 插件中填写该 ID， 即可获得此定制基线。

为 Android 开发工程引入定制基线

前提条件

确认您的 Android Studio mPaaS 版本为 V2.19111217 或以上。您可以参考 更新 mPaaS 插件 以了解当前的 mPaaS 插件版本和如何升级 mPaaS 插件。

操作步骤

1. 删除您 Android Studio 工程里已经存在的 mpaas_package.json 文件。
2. 点击 Android Studio 的菜单 **mPaaS** > **基线升级**。

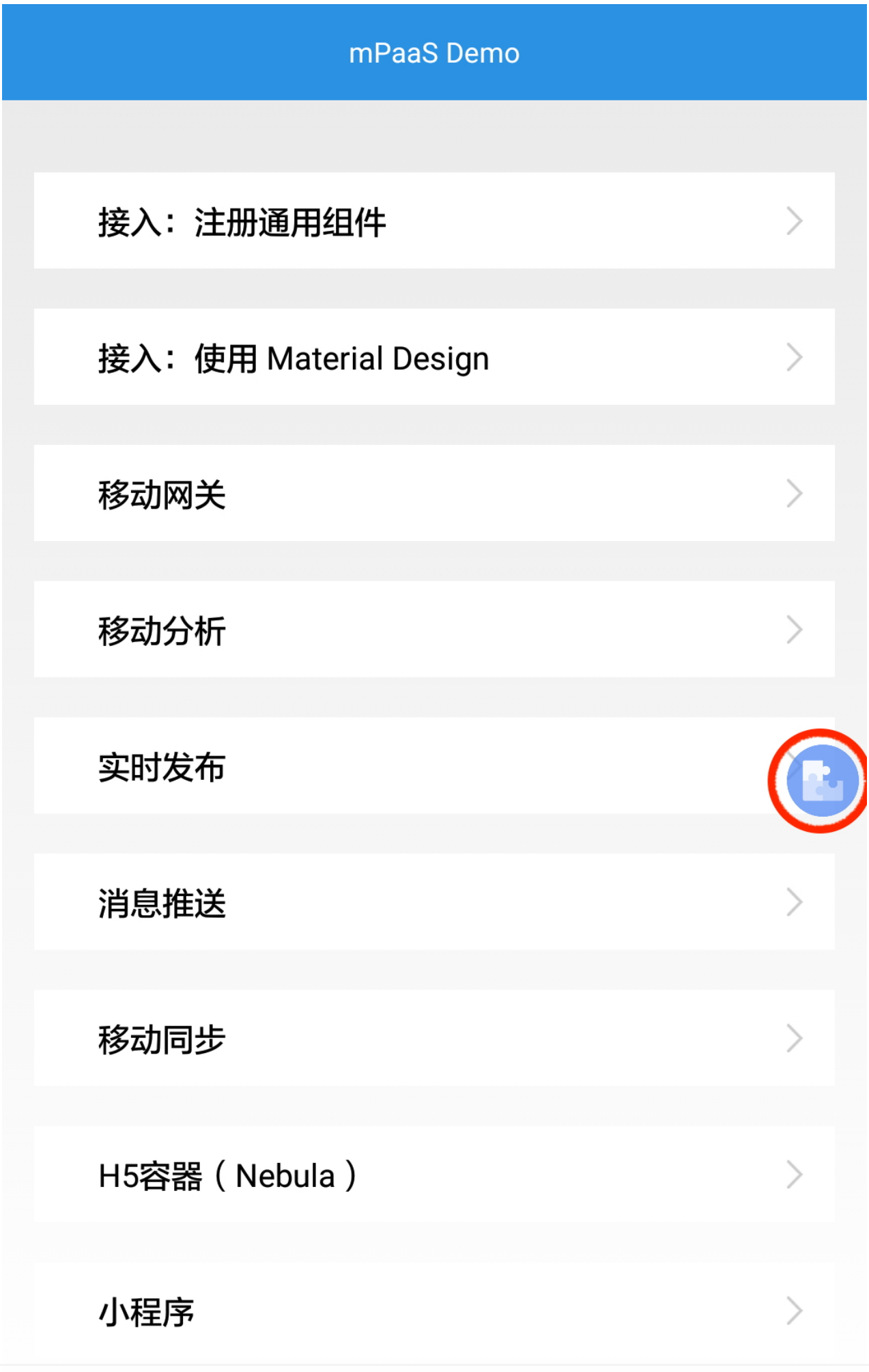
3. 在 **基线升级** 对话框中勾选 **定制基线** 并输入您得到的定制基线 ID。

4. 点击 **OK**，即完成定制基线的引入。

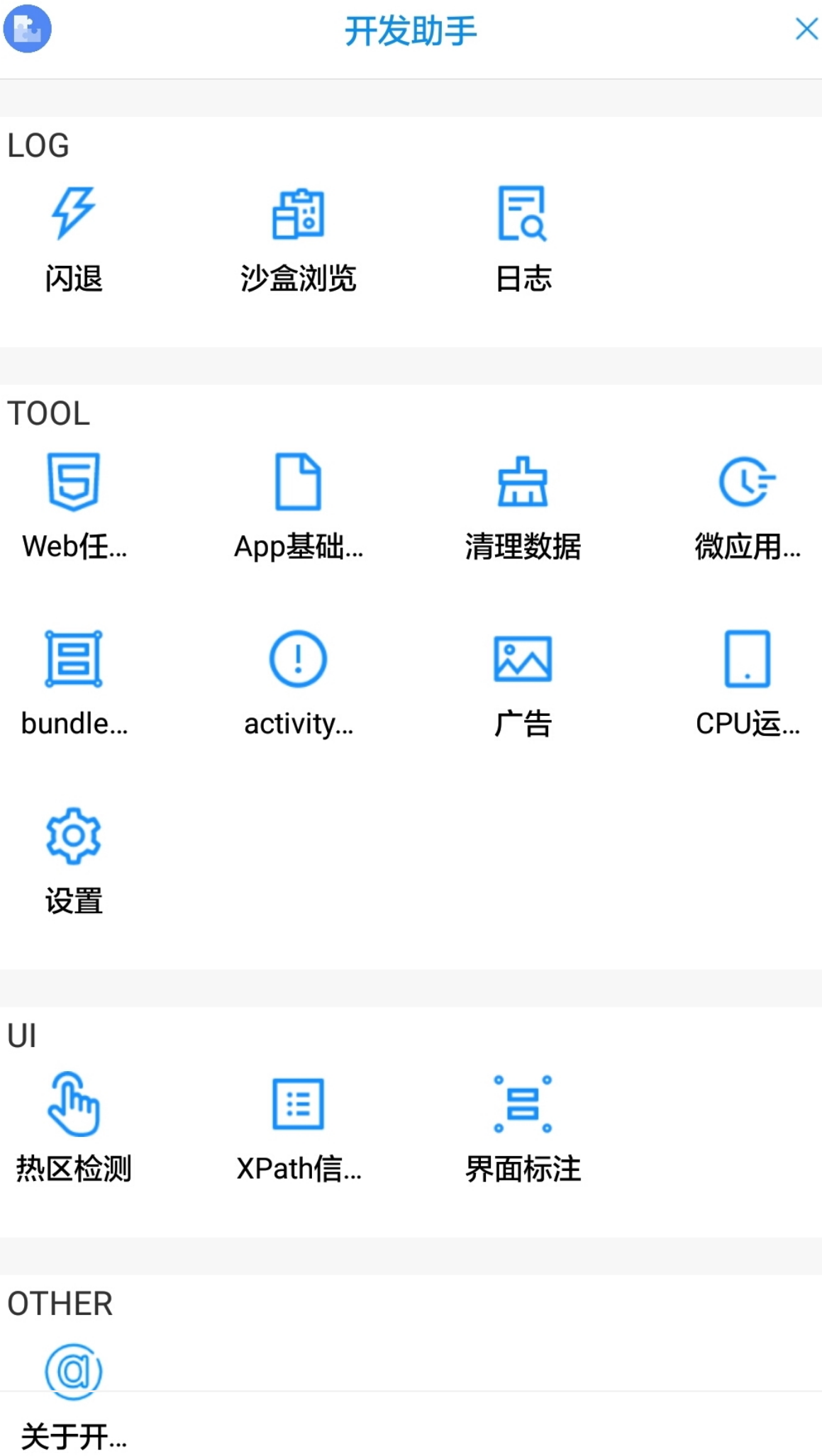
7.2 开发小助手

7.2.1 关于开发小助手

开发小助手 DevHelper 是基于 mPaaS 框架，整合了解决用户常见问题的开发工具集，该工具集随着 mPaaS 插件一起提供给开发者（ 点击快速了解开发小助手的功能 ）。在将开发工程接入 mPaaS 后或基于 mPaaS 插件创建工程后，即可将开发小助手接入开发工程（ 点击快速了解如何使用开发小助手 ），使用开发小助手进行调试、帮助开发。

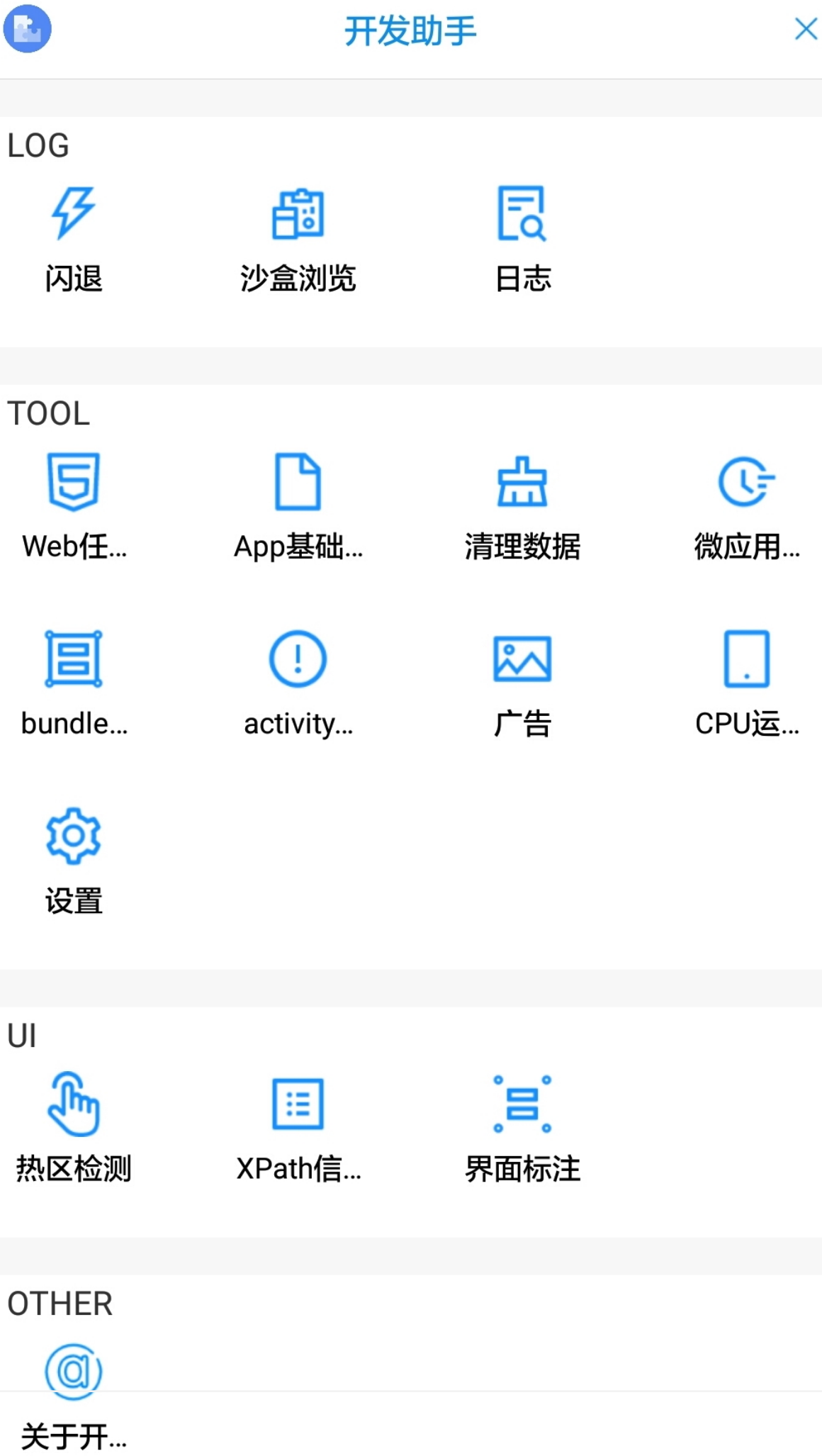

mPaaS Demo
接入：注册通用组件
接入：使用 Material Design
移动网关
移动分析
实时发布
消息推送
移动同步
H5容器（Nebula）
小程序

在应用中，开发小助手以悬浮窗的形式出现，在应用中的任何界面都可以点击悬浮窗，唤出开发小助手主页。


开发助手
LOG
闪退
沙盒浏览
日志
TOOL
Web任...
App基础...
清理数据
微应用...
bundle...
activity...
广告
CPU运...
设置
UI
热区检测
XPath信...
界面标注
OTHER
关于开...

7.2.2 开发小助手的功能

开发小助手提供的功能可分为四大类：LOG 日志类、TOOL 工具类、UI 界面类和OTHER 其他类。


开发助手
LOG
闪退
沙盒浏览
日志
TOOL
Web任...
App基础...
清理数据
微应用...
bundle...
activity...
广告
CPU运...
设置
UI
热区检测
XPath信...
界面标注
OTHER
关于开...

LOG 日志类

日志类包含闪退、沙盒浏览和日志三个功能。

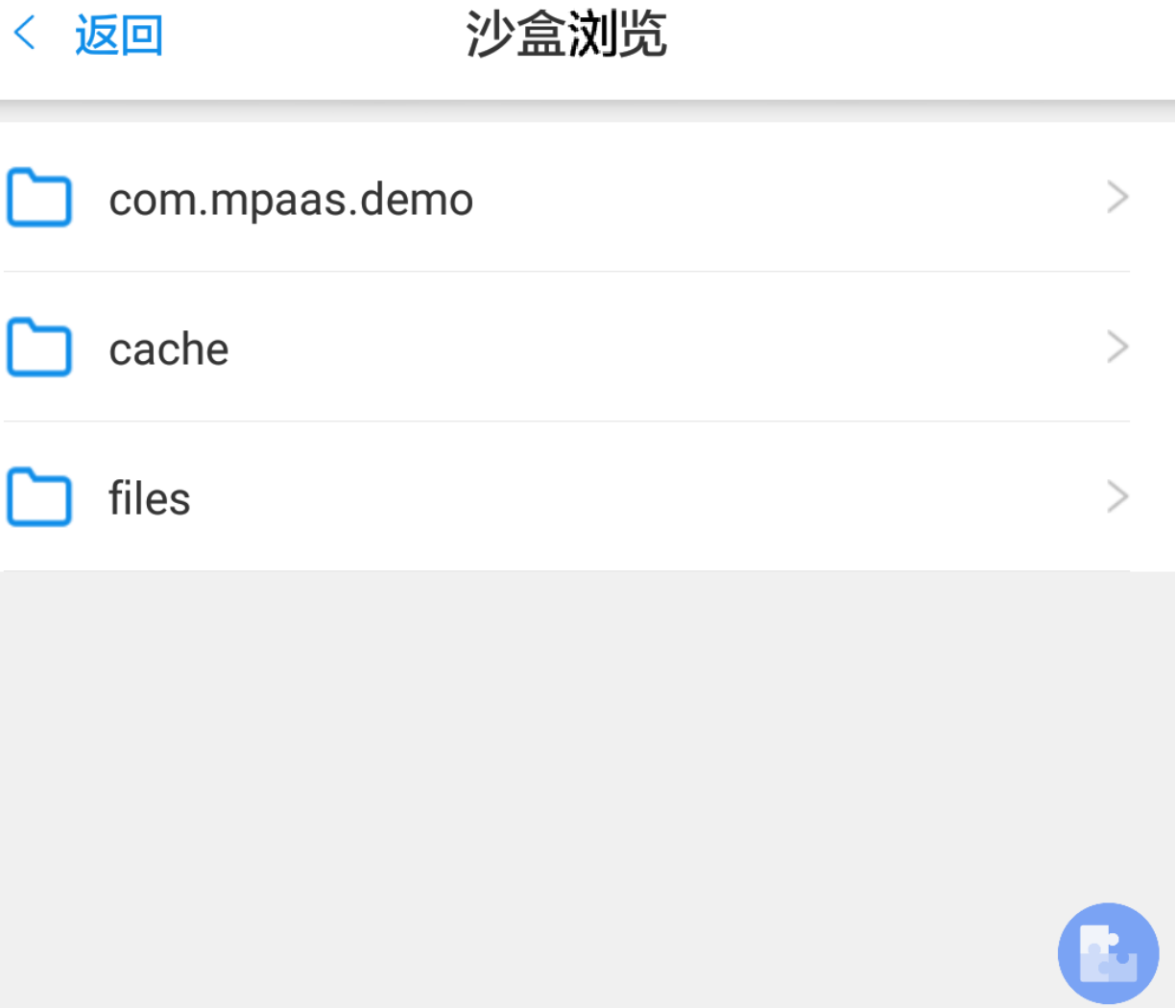
闪退

闪退功能支持快速查看闪退日志。


闪退日志
10-10 14:40:35
主线程crash
10-10 14:07:18
主线程crash

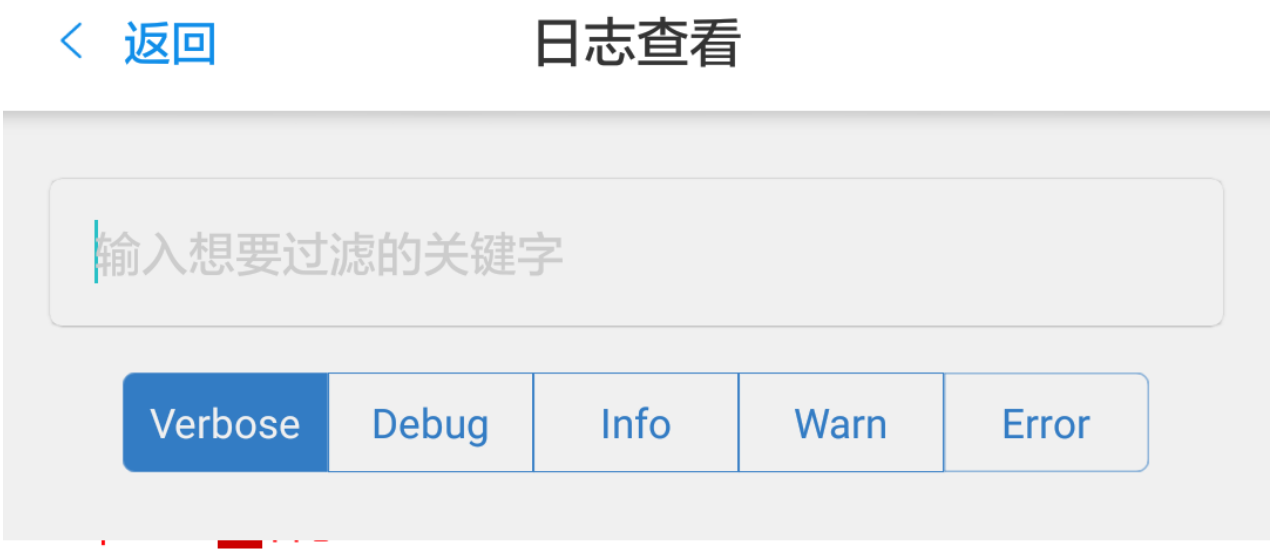
沙盒浏览

沙盒浏览功能支持查看 App 的所有文件。


返回
沙盒浏览
com.mpaas.demo
cache
files

日志

日志功能支持查看输出的客户端 Logcat日志。



dev_mdap	I	D-VM,2019-10-10 14:02:48.992,9C60AA6021124_ANDROID-d...
System	I	core_booster, getBoosterConfig = false
System.out	I	[logcat, -v, time, -d]
System.out	I	null
System.out	I	null
System.out	I	Calling by::className:com.mpaas.android.ex.helper.tools.log...
System	I	core_booster, getBoosterConfig = false
MemoryM...	I	[Monitor-apm-loop] got memoryInfo:84359
MemorySt...	I	[APMTimer] healthScore:1.0 info:524288,163840,360448,1.0,...
MemoryM...	I	[Monitor-apm-loop] updateMax for process:com.mpaas.demo
MemoryM...	I	[Monitor-apm-loop] new max memoryInfo:89679
MemoryM...	W	[APMTimer] runDetecting: GC occured !
BundleRes...	I	[main] Streamline Mode
Quinox_Bu...	W	Bundle=[com-mpaas-android-dev-helper-devhelper], founded...
System.out	I	[logcat, -v, time]
System.out	I	null
System.out	I	null
System.out	I	Calling by::className:com.mpaas.android.ex.helper.tools.log...
MemoryM...	I	[Monitor-apm-loop] got memoryInfo:13737
MemoryM...	I	[Monitor-apm-loop] updateMax for process:com.mpaas.demo...
MemoryM...	I	[Monitor-apm-loop] new max memoryInfo:13737
dev_mdap	E	{"code":200}
MemoryM...	I	[Monitor-apm-loop] got memoryInfo:9826
MemoryM...	I	[Monitor-apm-loop] updateMax for process:com.mpaas.demo...
MemoryM...	I	[Monitor-apm-loop] new max memoryInfo:10089
MemoryM...	I	[Monitor-apm-loop] updateMax for totalMemory.

TOOL 工具类

TOOL 工具类提供的功能有 **Web 任意门**、**App 基础信息**、**清理数据**、**微应用启动时长**、**Bundle 版本信息**、**Activity 辅助信息**、**广告**、和 **CPU 运行信息**。在 TOOL 工具类，您还可以通过 **设置** 对开发小助手的部分功能进行配置。

Web任意门

Web任意门功能支持通过开发小助手打开离线包或者在线页面。在输入网址、或者扫描二维码后，即可打开离线包或者在线网页。

返回

H5任意门

输入地址，点击按钮跳转

点击跳转

清空搜索歷史

App 基础信息

显示App的基本信息。

返回

App基本信息

App信息

包名	com.mpaas.demo
应用版本名	3.0.0.0
应用版本号	4
目标系统版本号	26

手机信息

手机型号	HUAWEI HUAWEI MT7-CL00
系统版本	6.0 (23)
sd卡剩余空间	2.34 GB/11.55 GB
系统剩余空间	2.36 GB/11.57 GB
ROOT	false
DENSITY	2.5
分辨率	1080x1920

清理数据

清理App的所有缓存以及数据信息。在清除数据后，App会自动退出。


开发助手
LOG
闪退
沙盒浏览
日志
TOOL
Web任...
App基础...
清理数据
微应用...
确认清除数据?
确定
取消
设置
UI
热区检测
XPath信...
界面标注
OTHER
本客户端包版本号:3.0.0.0

微应用启动时长

输入微应用的AppId，就可以查看和统计微应用的启动耗时。


微应用启动耗时
AppId
请输入AppId20000032
查询
清除数据

bundle 版本信息

显示mPaaS框架下Bundle的版本号。

输入bundle名称搜索

1.Bundle名:android-phone-mobilesdk-socketcraft
版本号1.1.2.190318171311

2.Bundle名:android-phone-wallet-basicstl
版本号1.0.0.190214150143

3.Bundle名:android-phone-mobilesdk-transport
版本号1.8.5.190729124755

4.Bundle名:android-phone-mobilesdk-framework
版本号2.6.2.190802104526

5.Bundle名:android-phone-mobilesdk-common
版本号1.8.2.190715164745

6.Bundle名:android-phone-mobilecommon-lbs
版本号1.9.0.190307124849

7.Bundle名:android-phone-mobilesdk-netsdkextdepend
版本号1.0.2.190620164823

8.Bundle名:android-phone-mobilesdk-netsdkextdependapi
版本号1.0.1.190620165649

9.Bundle名:android-phone-mobilecommon-dynamicrelease
版本号2.5.7.190801153211

10.Bundle名:android-phone-mobilesdk-commonbizservice
版本号1.7.0.190717171102

11.Bundle名:android-phone-wallet-h5worker
版本号1.0.0.190322133029

12.Bundle名:android-phone-wallet-nebula

activity 辅助信息

提供微应用启动Activity的参数信息。

activity辅助

appId:

viewId(activity名称):
com.mpaas.android.ex.helper.ui.UniversalActivity

intent入参:

```
{
   "fragment_index": 2
}
```

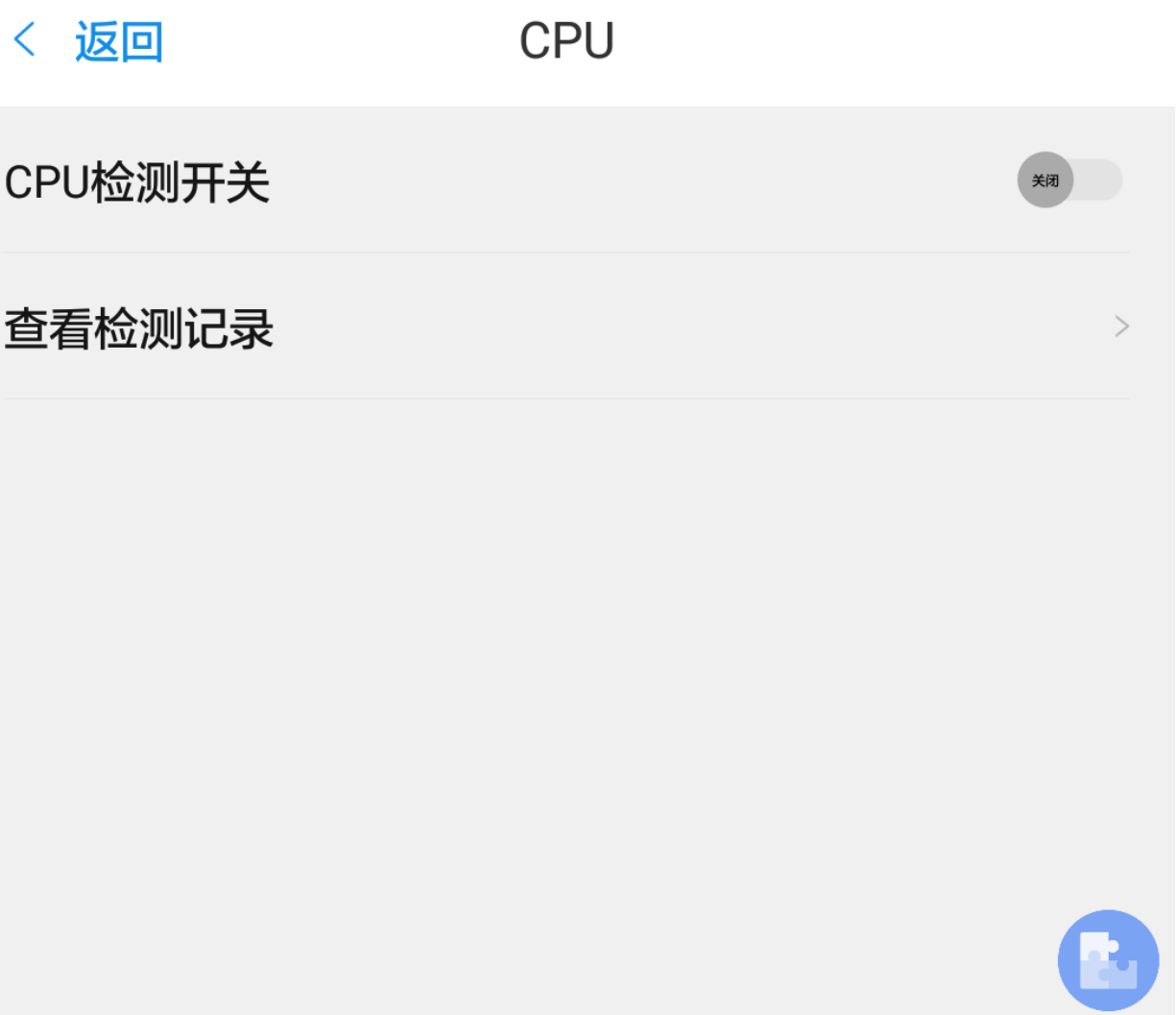
广告

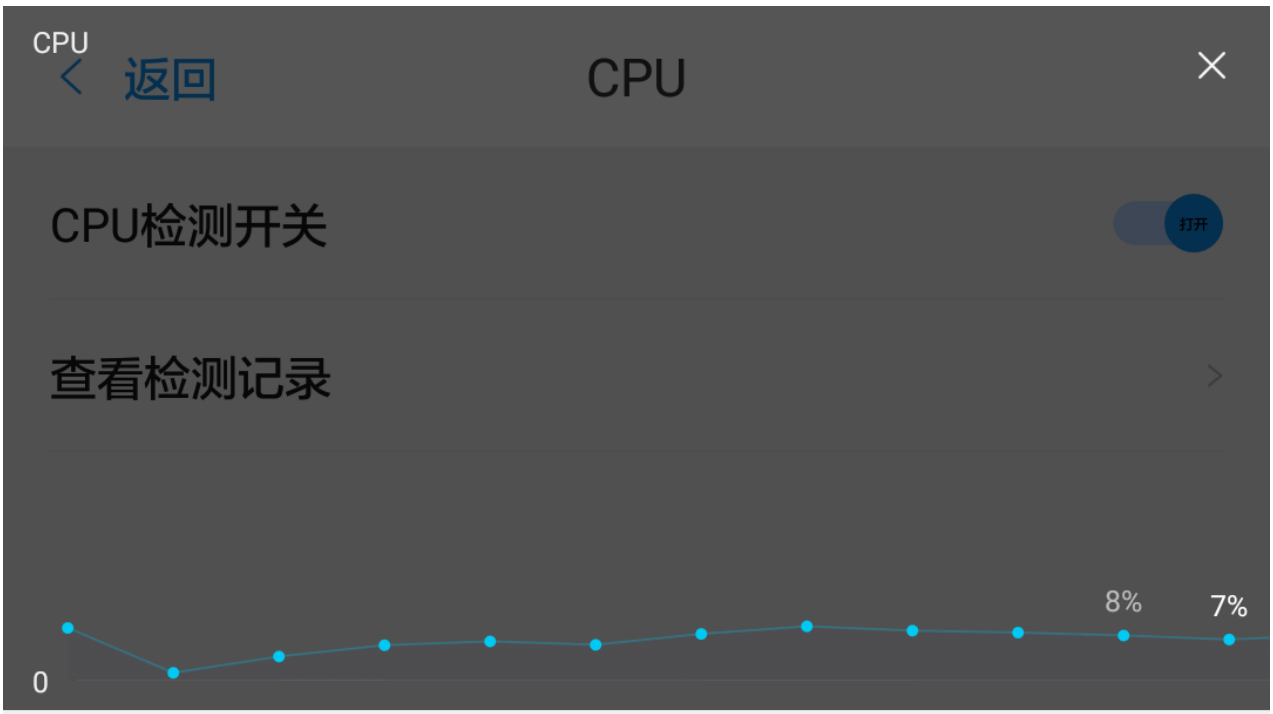
智能投放广告的动态化工具。在接入智能投放并预置展位后，可以动态给某一个位置查一个广告，并且可以提取页面上的广告。


广告
查看当前页面展位
QUERY
查看指定展位码展位
homepage_toptips
QUERY
动态插入top公告
SHOW
动态插入float公告
SHOW
动态插入半透明float公告
SHOW
动态插入H5弹屏
SHOW
动态插入native弹屏
SHOW
所有页面动态插入top公告
*点击query无结果说明没有符合条件的展位
*动态插入广告暂不支持首页四个tab,因为是埋坑接入的
*如果插入top公告失败,请先查询当前页面展位,如果该页面本身存在location=top的展位,我们无法mock
*使用疑问请参照文档

CPU 运行信息

打开 **CPU检测开关** 后，即可显示 CPU 使用率信息。该信息以浮层的形式在屏幕上方显示，此时您可以切换到应用的的其他页面使用应用的不同功能，对 CPU 的使用率信息进行实时监控。


返回
CPU
CPU检测开关
关闭
查看检测记录


CPU
返回
CPU
CPU检测开关
打开
查看检测记录
8%
7%
0

通过 **检测记录** 菜单，您还可以查看 CPU 运行信息的检测记录。

返回

检测记录

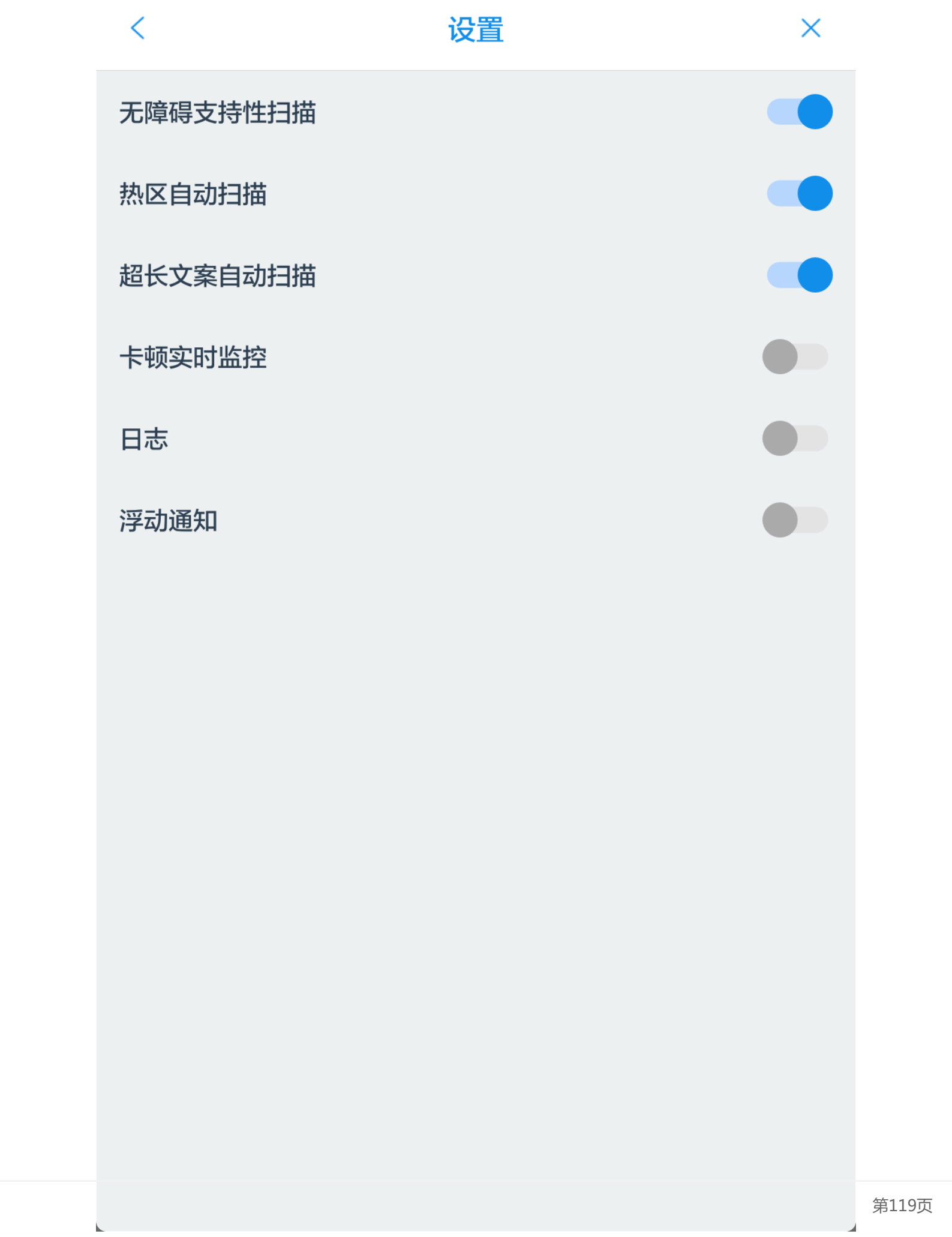
2019-10-11 13:49:57

CPU 4.31566

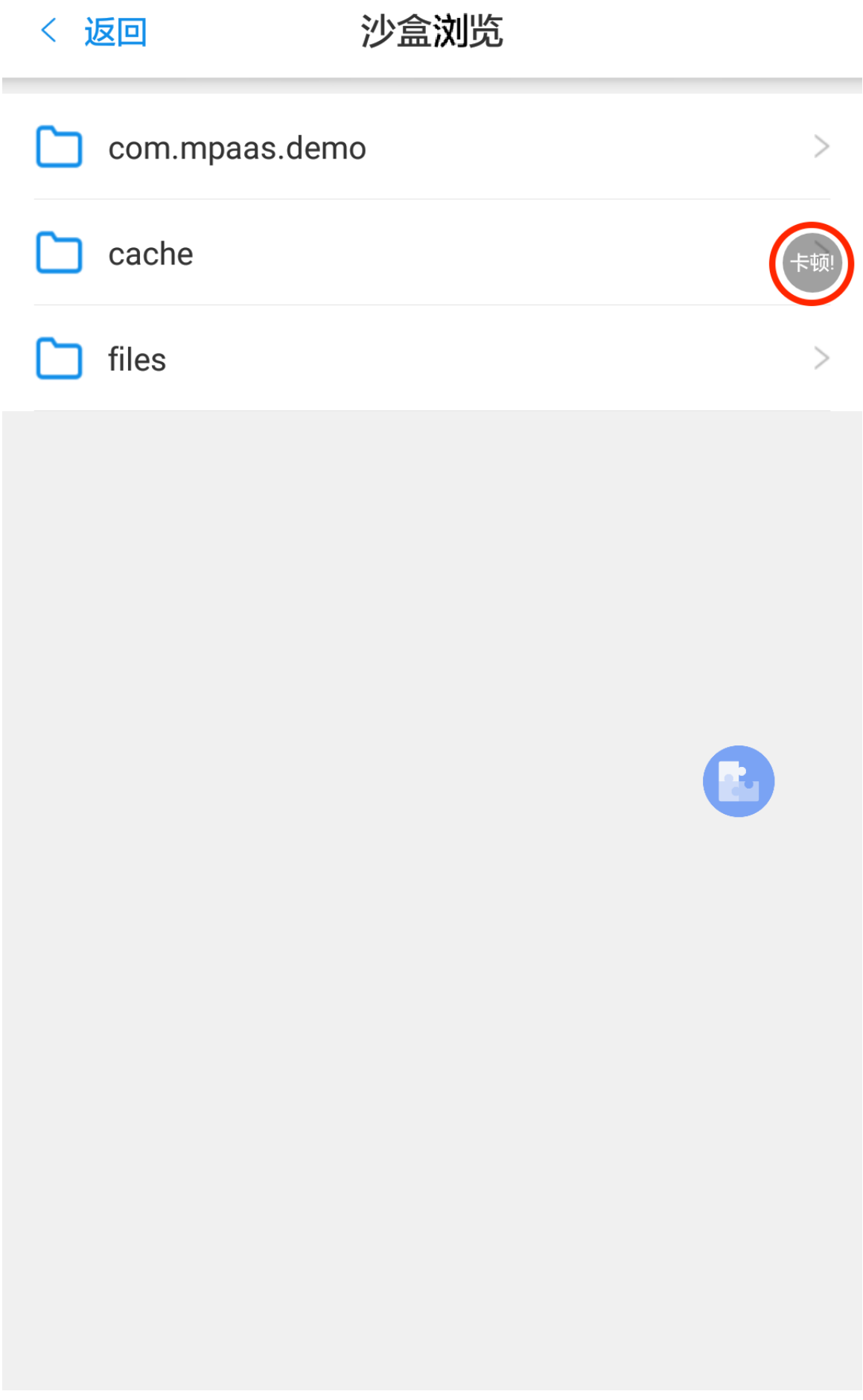
2019-10-11	13:49:56	↑0.0
2019-10-11	13:49:57	↑4.31566
2019-10-11	13:49:58	↑4.74407
2019-10-11	13:49:59	↑7.052897
2019-10-11	13:50:00	↑10.532994
2019-10-11	13:50:01	↑5.315204
2019-10-11	13:50:02	↑4.2767296
2019-10-11	13:50:03	↑5.5207024

设置

在 **设置** 中，您可以查看对开发小助手的功能进行配置。这些功能包括无障碍支持性扫描、热区自动扫描、超长文案自动扫描、卡顿实时监控、日志和浮动通知。


设置
无障碍支持性扫描
热区自动扫描
超长文案自动扫描
卡顿实时监控
日志
浮动通知

- 无障碍支持性扫描
  支持对 android 视障人群功能的界面检查。
- 热区自动扫描
  热区检测可以检测页面中点击区域的宽或高小于 30dp 的控件，并将检测结果实时显示在当前页面顶部。如果开启此开关，则会对扫描到的热区中宽或高小于 30dp 的控件自动标红。
- 超长文案自动扫描
  可以实现对重叠文本的检查。因为文本超长之后文字会不显示，通过开发小助手，就可以通过文本控件的 api 检查出来。
- 卡顿实时监控
  开启后，开发小助手会实时监控 App 的运行状态，一旦发生卡顿，就会以悬浮窗的形式进行提醒。
  点击悬浮球，即可查看卡顿信息。
  查看卡顿信息后返回 App，悬浮窗会自动消失。


返回
沙盒浏览
com.mpaas.demo
cache
卡顿!
files
卡顿查看
10月-11 14:05:09.731
查看调用栈

- 日志
  开启此开关后，即可阅读日志类型的文件。
- 浮动通知
  关闭此开关时，发生卡顿时将不会再弹出悬浮窗通知。

UI 界面类

UI 界面类提供的功能有 **热区检测**、**XPath 信息** 和 **界面标注**。

热区检测

热区检测可以检测页面中点击区域的宽或高小于 30dp 的控件，并将检测结果实时显示在当前页面顶部。

接入：注册通用组件

接入：使用 Material Design

移动网关

移动分析

实时发布

消息推送

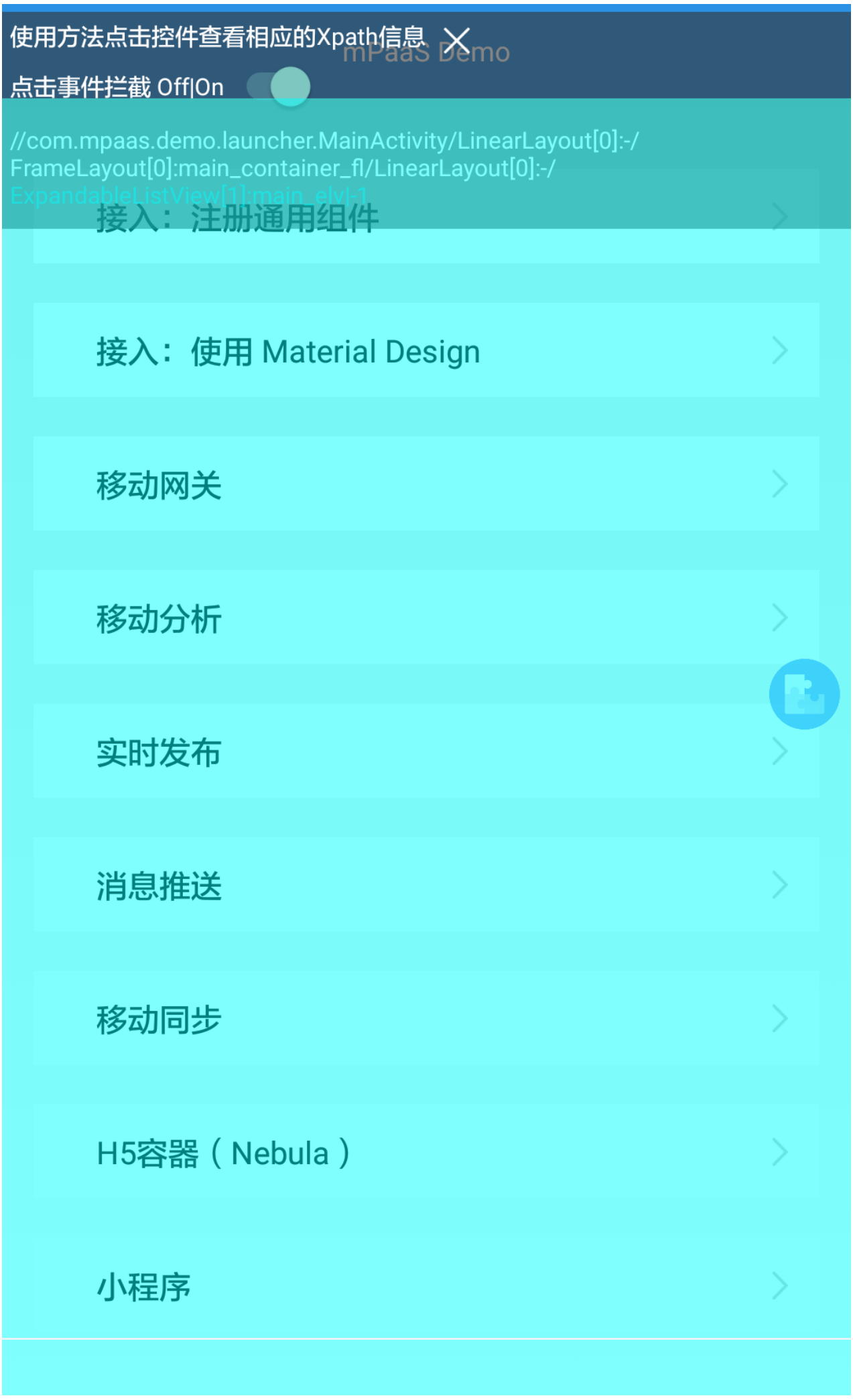
移动同步

H5容器（Nebula）

小程序

XPath 信息

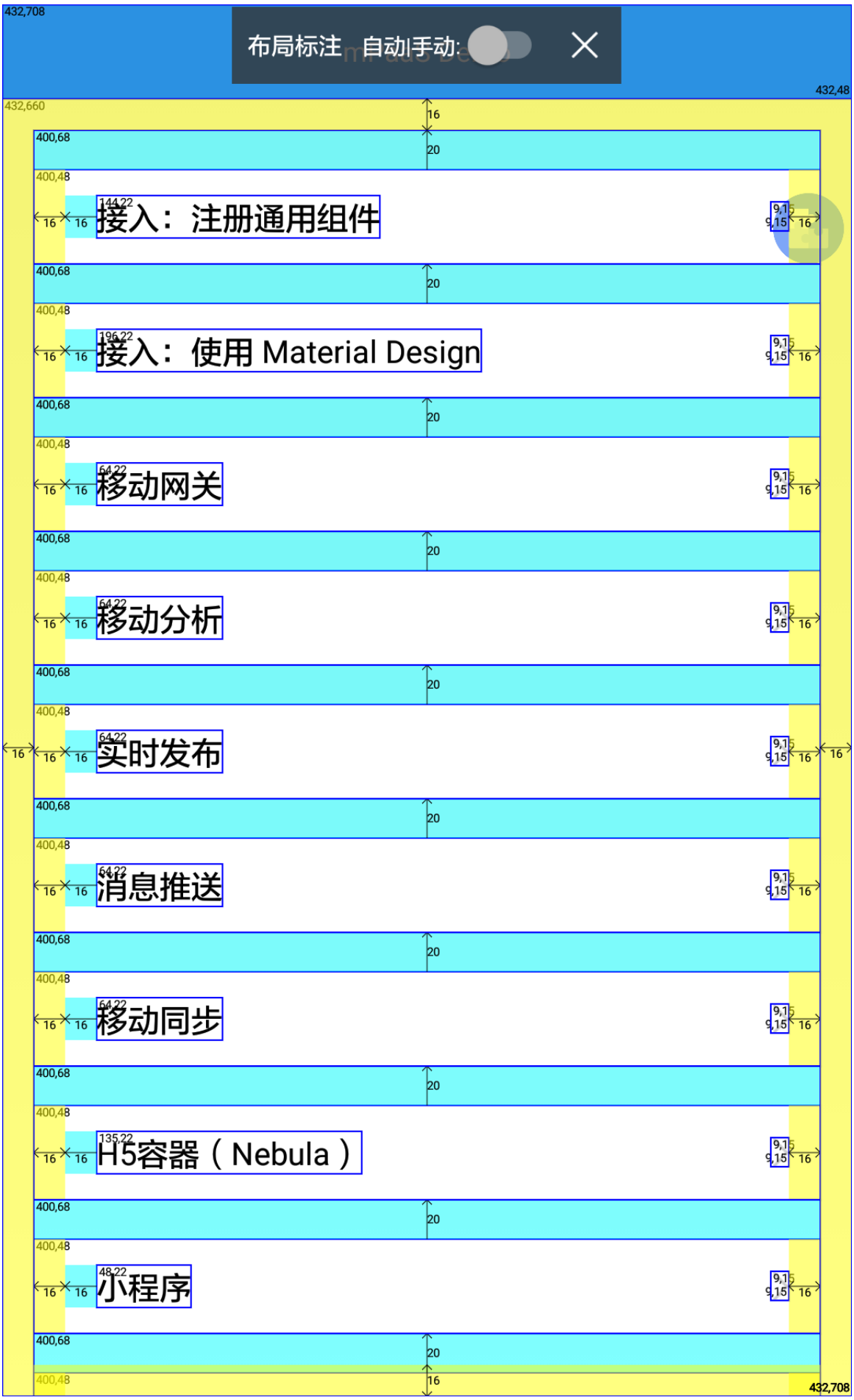
热区检测可以检测页面中显示控件的 XPath 信息。打开 **事件拦截** 开关后，开发小助手会将 XPath 信息记性显示。


使用方法点击控件查看相应的Xpath信息
mPaaS Demo
点击事件拦截 Off|On
//com.mpaas.demo.launcher.MainActivity/LinearLayout[0]:-/
FrameLayout[0]:main_container_fl/LinearLayout[0]:-/
ExpandableListView[1]:main_elv|-1
接入：注册通用组件
接入：使用 Material Design
移动网关
移动分析
实时发布
消息推送
移动同步
H5容器（Nebula）
小程序

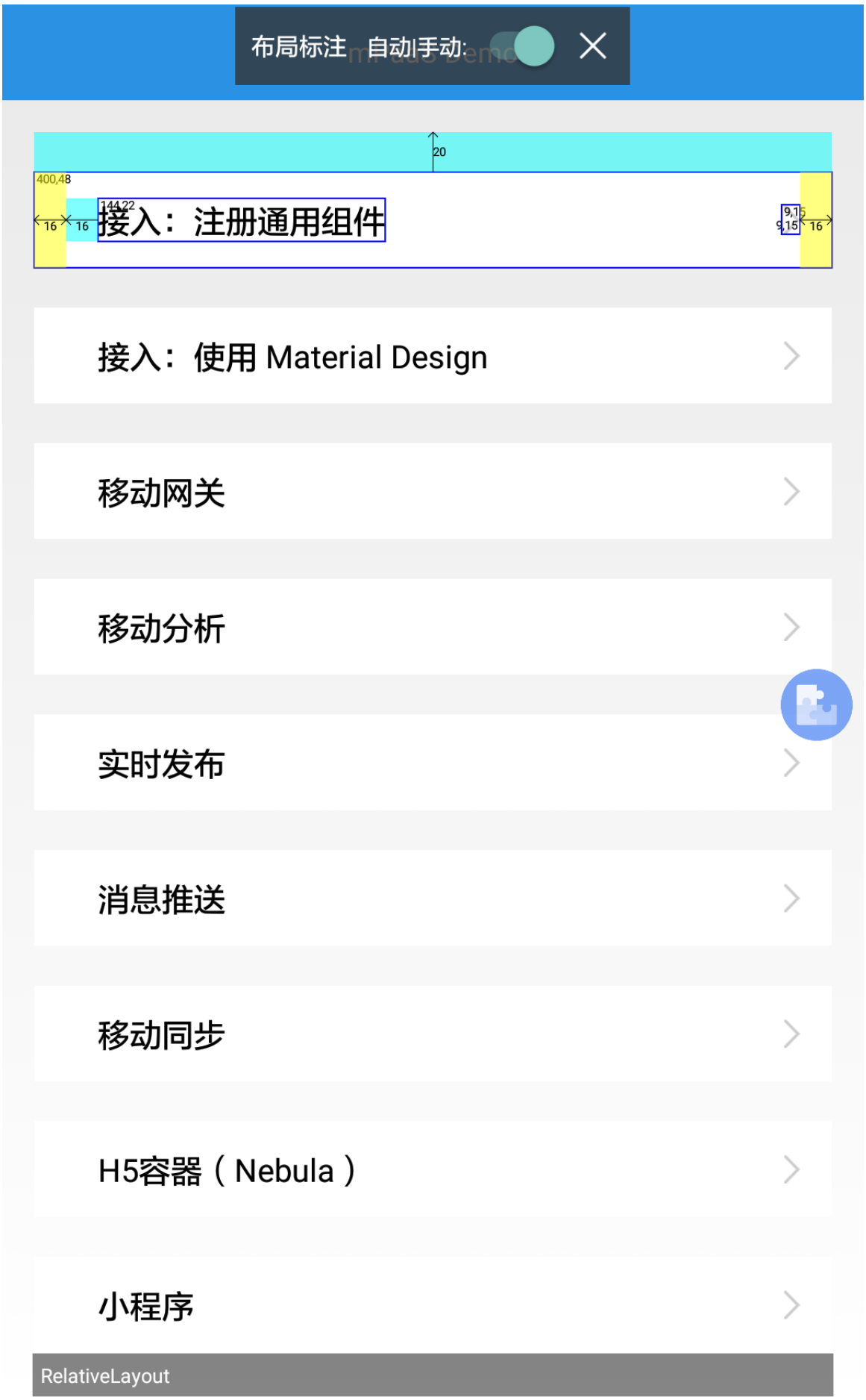
界面标注

界面标注可以将页面中的控件大小标注出来，标注分为自动和手动两种方式。

- 自动标注：开发小助手会自动将当前页面中的所有控件进行标注。


布局标注
自动|手动:
接入：注册通用组件
接入：使用 Material Design
移动网关
移动分析
实时发布
消息推送
移动同步
H5容器（Nebula）
小程序

- 手动标注：开发小助手会只标注被点击了的控件。


布局标注 自动|手动:
20
400,48
144,22
16
16
接入：注册通用组件
9,15
9,15
16
接入：使用 Material Design
移动网关
移动分析
实时发布
消息推送
移动同步
H5容器（Nebula）
小程序
RelativeLayout

OTHER 其他类

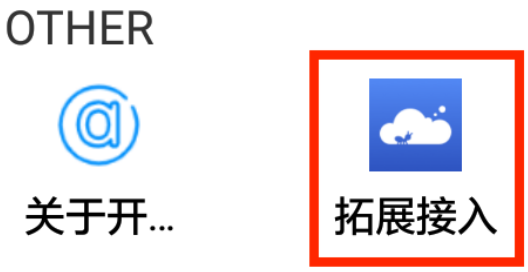
OTHER 其他类提供了跳转到开发小助手帮助文档的入口。同时，对开发小助手拓展的功能也会显示在 OTHER 其他类。

关于开发小助手

跳转到开发小助手的帮助文档。

拓展的功能

用户自己拓展的功能都会显示在 OTHER 其他类 中。如下图所示，“拓展接入”即为一个拓展的开发小助手的功能。



7.2.3 使用开发小助手

本文将从接入、启动和拓展三方面向您介绍如何使用开发小助手。

接入开发小助手

前置条件

已接入mPaaS 框架。

接入步骤

您只需在 Portal 工程的 build.gradle 文件中加入如下代码，即可完成开发小助手的接入。

```
devbundle"com.mpaas.android.dev.helper:devhelper-build:1.0.0.191012132031@jar"
devmanifest"com.mpaas.android.dev.helper:devhelper-build:1.0.0.191012132031:AndroidManifest@xml"
```

启动开发小助手

开发小助手随 mPaaS 框架启动，无需额外代码启动。但在以下情况下，在启动开发小助手时需要特别注意：

1. 当安卓手机运行在monkey模式下时，开发小助手自动不启动。
2. 当开发模式为 release mode 时，即 android:debuggable="false" 时，开发小助手默认不启动。 如果需要在 release mode 下启动开发小助手，可以通过在 AndroidManifest.xml 文件中设置如下属性强制启动开发小助手。但是在正式发布时一定要去掉该功能。代码配置如下：

```
<meta-data
android:name="devhelper_start"
android:value="true"/>
```

并且将开发小助手的依赖按照如下代码所示进行更换：

```
bundle"com.mpaas.android.dev.helper:devhelper-build:1.0.0.191012132031@jar"
mainfest"com.mpaas.android.dev.helper:devhelper-build:1.0.0.191012132031:AndroidManifest@xml"
```

拓展开发小助手

在实际使用中，您还可以根据实际需求，对开发小助手进行功能拓展，如查看用户的加密数据。拓展的功能依赖于增加新的实现类。

拓展步骤

1. 在 assets 文件夹中添加加一个 .devhelper.json 结尾的文件，并确保文件名唯一。该json文件需包含以下代码：

```
[
{
"className":"com.alipay.android.phone.devtool.devhelper.woodpecker.panel.items.SpmItem",
"bundleName":"com-mpaas-android-dev-helper-devhelper"
}
]
```

className：类名。该类名即步骤2中新建的实现类的类名。bundleName：bundle 名。bundle 名可在主 module 的 /build/intermediates/bundle/META-INF/BUNDLE.MF 文件中查看。如果该拓展功能是加在 Portal 工程中，则 bundleName 需填空字符串 “” 。

新建一个类并实现如下方法，需要确保该类有且只有一个无参构造方法。

```
/**
* @param context
* @return 小助手面板入口图片
*/
```

```
Drawable icon(Context context);

/**
* @param context
* @return 小助手面板显示的名称
*/
String title(Context context);

/**
* @param context
* @return 是否处理相应，如已处理，请返回True
*/
boolean onClick(Context context);

/**
* @param context
* @return 该功能是否可用，如可用，请返回True
*/
boolean enabled(Context context);

/**
* 返回如下分组
* {@link #GROUP_OTHER} = 'OTHER'
* {@link #GROUP_TOOL} = 'TOOL'
* {@link #GROUP_UI} = 'UI'
* {@link #GROUP_LOG} = 'LOG'
*
* @return
*/
String group();
```

代码示例

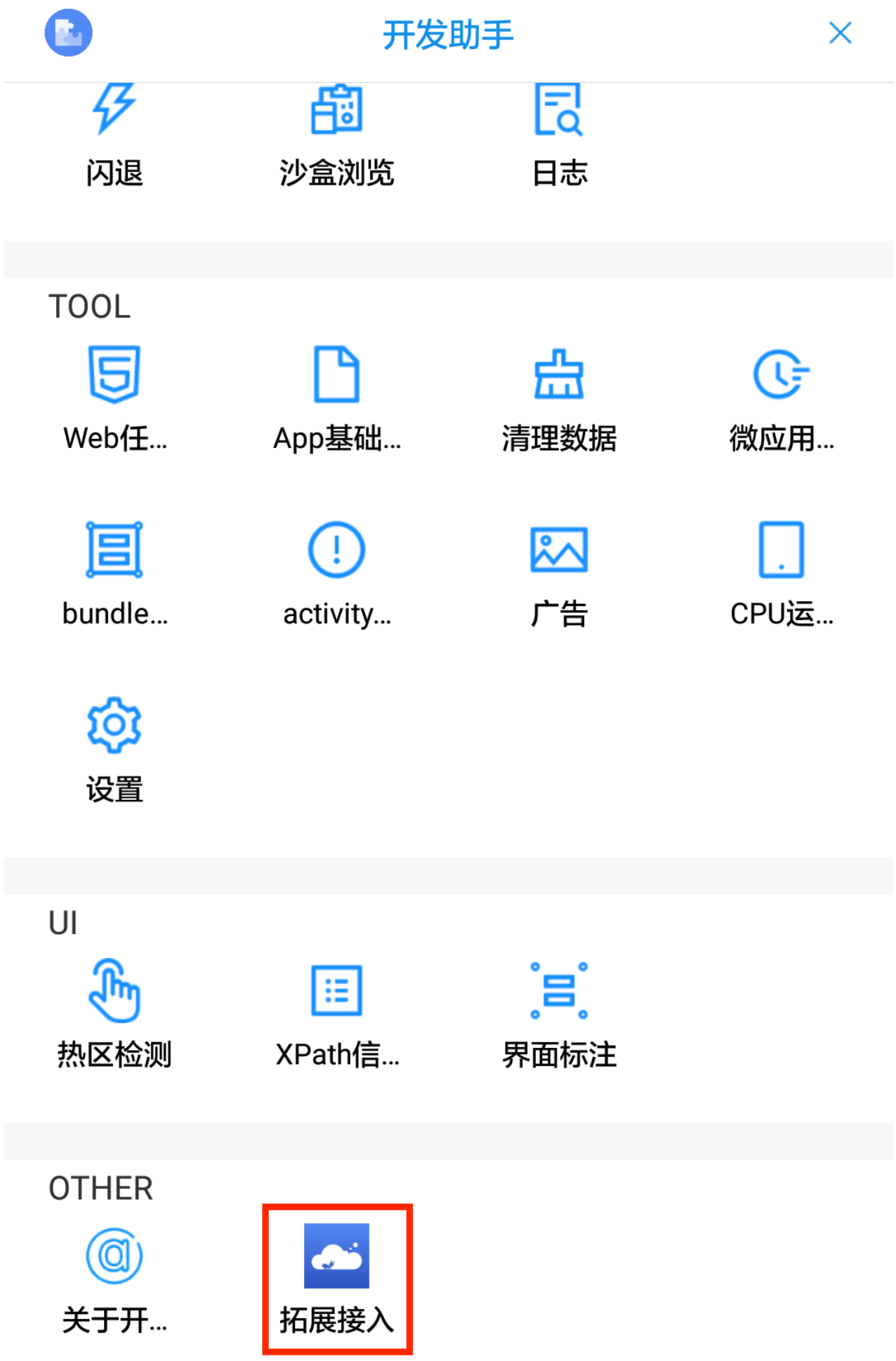
以下为一个拓展开发小助手的简单示例。该示例增加了拓展功能 **拓展接入**，实现点击图标后返回字符串“onclick”的功能。

```
package com.alipay.mpaas.test.helper;

import android.annotation.TargetApi;
import android.content.Context;
import android.graphics.drawable.Drawable;
import android.os.Build;
import android.widget.Toast;
import com.mpaas.demo.R;

public class SpmItem {
private final static String TAG = SpmItem.class.getName();

@TargetApi(Build.VERSION_CODES.LOLLIPOP)
public Drawable icon(Context context) {
return context.getDrawable(R.drawable.appicon);
}

public String title(Context context) {
```

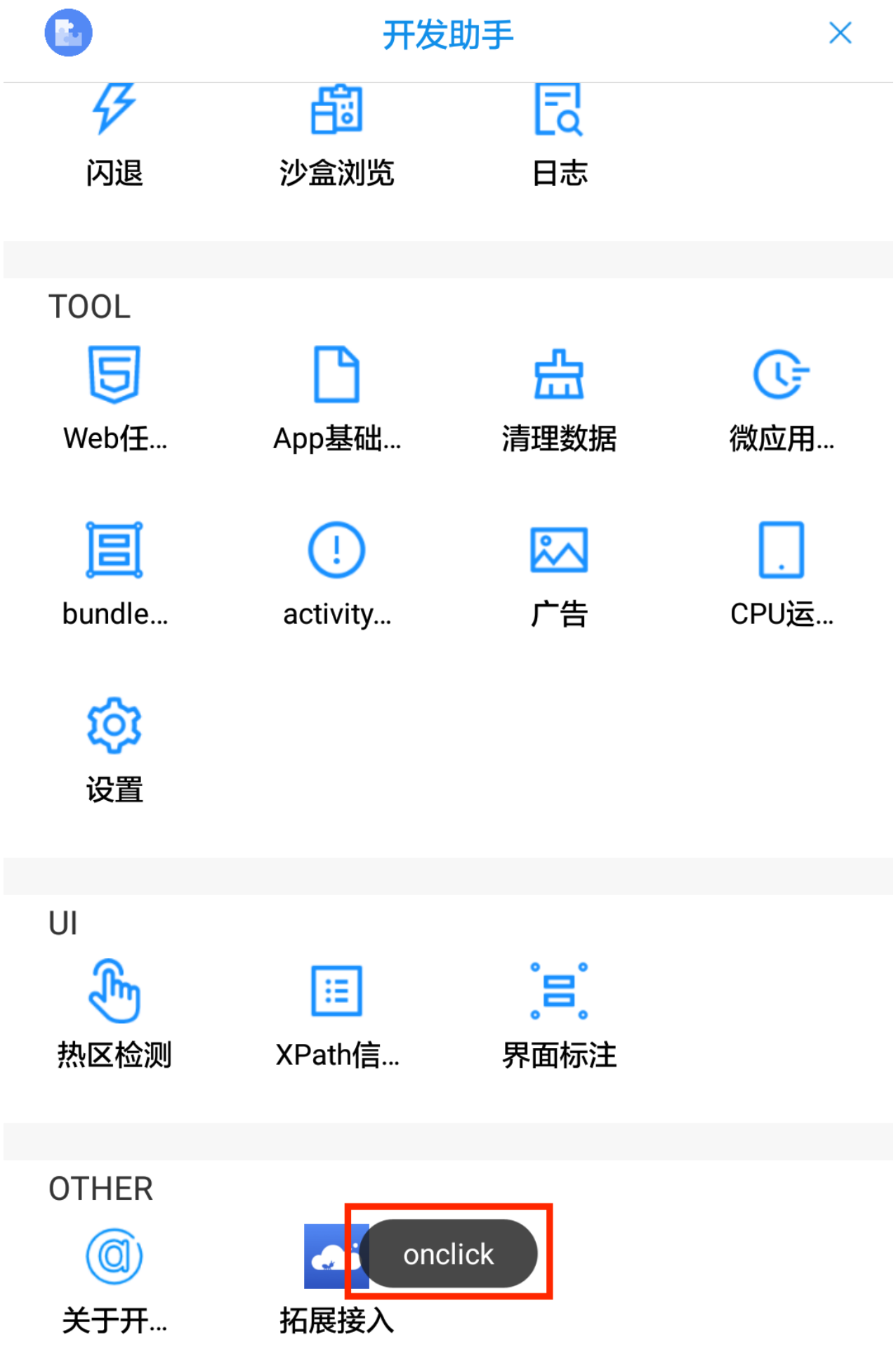
```
return"拓展接入";
}

public boolean onClick(Context context) {
Toast.makeText(context,"onclick",Toast.LENGTH_LONG).show();
return true;
}

public boolean enabled(Context context) {
return true;
}

public String group() {
return"OTHER";
}
}
```

运行结果

如下图所示，在开发小助手的 OTHER 其他类 中已经增加了 **拓展接入**，点击图标后，即可显示字符串“onclick”。


开发助手
闪退
沙盒浏览
日志
TOOL
Web任...
App基础...
清理数据
微应用...
bundle...
activity...
广告
CPU运...
设置
UI
热区检测
XPath信...
界面标注
OTHER
关于开...
拓展接入
本客户端包版本号:3.0.0.0

)


开发助手
闪退
沙盒浏览
日志
TOOL
Web任...
App基础...
清理数据
微应用...
bundle...
activity...
广告
CPU运...
设置
UI
热区检测
XPath信...
界面标注
OTHER
关于开...
onclick
拓展接入
本客户端包版本号:3.0.0.0

)

8 Android 适配说明

8.1 mPaaS 10.1.60 正式版升级指南

关于 mPaaS 10.1.60 正式版

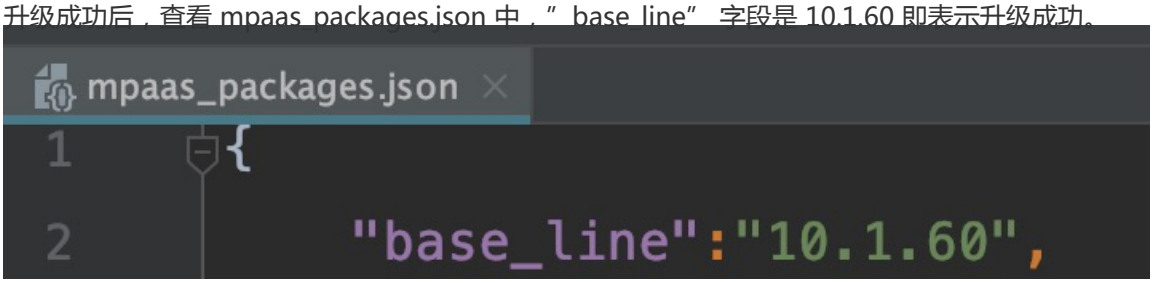
1. 10.1.60 基线已适配 **Android 10**，详情请参考 mPaaS 适配 Android 10 指南 。
2. 10.1.60 基线新增加了 **小程序组件** 正式版。小程序正式版拥有更加完善的 API，且在稳定性、兼容性等方面有了大幅提高。关于小程序升级请参见 小程序升级说明 ，关于小程序 IDE 新增调试、预览、发布等功能的详情请参见 小程序 IDE 。
3. 10.1.60 基线对 **H5 容器** 整体进行大幅优化，提供了更加简化的接入流程，持续补强能力，在兼容性、稳定性等方面有显著提高。关于 H5 容器和离线包升级，请参见 H5 容器升级说明 。
4. 10.1.60 基线新增加了对 **社交分享** 组件的管理支持，提供了简化的接入流程。关于社交分享的升级，请参见 迁移到 10.1.60 基线 。
5. 10.1.60 基线新增加 **智能投放** 组件。智能投放提供了在应用内个性化投放广告的能力，支持针对定向人群进行个性化广告投放，帮助 APP 运营人员精准、及时触达用户，详情请参见 智能投放 。
6. 10.1.60 基线的整体组件的兼容性、稳定性都有了大幅提高，功能也有着显著提升，具体的发布说明请参见 Android SDK 发布说明 。

mPaaS 10.1.60 正式版升级指南

操作步骤

1. 升级 Android Studio mPaaS 插件到 v2.19123015 或以上。
   关于更新 mPaaS 插件，请参见 更新 mPaaS 插件 。
2. 在 Android Studio 中的当前工程下，点击菜单 **mPaaS** > **基线升级**，选择 **10.1.60**，并点击 **OK**。

3. 升级成功后，查看 mpaas_packages.json 中，”base_line” 字段是 10.1.60 即表示升级成功。



注意：10.1.60-beta 基线转为正式版也需要按上述操作。

组件使用升级指南

10.1.60 基线中的 H5 容器、小程序和社交分享组件在接入、使用等方面做了大幅调整。如您接入了上述组件，请详细阅读下列说明：

- 请阅读 H5 容器升级说明 了解 H5 容器和离线包升级的更多信息。
- 请阅读 小程序升级说明 了解小程序升级的更多信息。】
- 社交分享 SDK 接入方式升级:

- 请阅读 分享 SDK 迁移到 10.1.60 基线 了解 **社交分享** 组件升级的更多信息。

注意：

- 从 10.1.60 开始分享 SDK 使用 mPaaS 插件进行管理。如果需要安装分享组件，请参照 分享 SDK 迁移到 10.1.60 基线 进行操作。
- 如未使用插件进行分享 SDK 的接入，则会导致分享 SDK 的升级与问题修复不能得到及时更新。

组件 API 变更

mPaaS 组件从 10.1.32 基线开始起添加了适配层，如您使用的基线版本低于 10.1.32 或未使用适配层 API，请先行阅读 mPaaS 10.1.32 基线升级指引 。

建议您在升级 SDK 后使用适配层的 API，具体可参考以下各组件文档中的旧版本升级注意事项：

- 移动分析：
  - 新增适配器，简化使用，参见 自定义事件日志 。
  - 部分 API 废弃，您需使用新的 API，参见 移动分析 SDK 发布说明 ，否则可能导致编译失败。
- 移动推送：新增适配器，简化使用，参见 移动推送 SDK 10.1.32 。
- 移动同步：新增适配器，简化使用，参见 移动同步 SDK 10.1.32 。
- 热修复：新增适配器，简化使用，参见 热修复 SDK 10.1.32 。
- 版本升级：新增适配器，简化使用，参见 版本升级 SDK 10.1.32 。
- 开关配置：新增适配器，简化使用，参见 开关配置 SDK 10.1.32 。
- H5 容器：
  - 新增适配器，简化使用，参见 H5 容器 SDK 10.1.32 。
  - 容器配置方法变更，如果升级前为 10.0.18 版本，您需使用新的容器配置方法，参见 容器配置 10.1.32 ，否则您的容器配置将无法生效。
  - 10.1.60基线变更参考 升级说明 。
- 小程序：
  - 请先进行H5容器升级。
  - 升级变更信息 升级说明 。

注意：强烈建议您修改代码，使用中间层（适配器）方法而非直接使用底层方法，因为某些底层方法可能会在将来的版本中发生变更或废弃。如果您继续使用，在将来的更新中可能需要花费更多的时间进行适配。

定制依赖处理

请查看所有 build.gradle 中 dependencies 的依赖配置，确认是否配置有 mpaas 组件的bundle 依赖，若有依赖且是从低版本 SDK（例如 10.1.20、10.1.32）升级至 10.1.60 版本，您的定制库可能需要基于新版本重新定制，否则可能会出现不兼容等问题，请 提交工单 或联系 mPaaS 支持人员确认。

8.2 mPaaS 适配 Android 10 指南

背景

谷歌已于 9 月 4 日正式发布 Android 10（Q）系统，华为、小米等国内厂商也将于 9 月开始陆续在部分机型上推出基于 Android 10 的新版系统。在 Android 10 设备上，应用的某些行为将受到限制，其中影响到 mPaaS SDK 的有：

- 系统部分 API 被禁用，可能导致使用 UC 内核打开 H5 页面的应用在 Android 10 设备上崩溃。
- 旧版中使用 file:// 为前缀调用系统安装界面的申请方式被废弃，将导致使用了版本升级功能并使用 mPaaS 默认升级弹窗的应用，在 Android 10 设备上无法调起系统安装界面。
- 不再允许获取手机 IMEI/IMSI/SN，这可能导致 Android 10 设备无法收到基于设备维度推送的消息。
- 热修复在 Android 10 设备上无法生效。

说明：

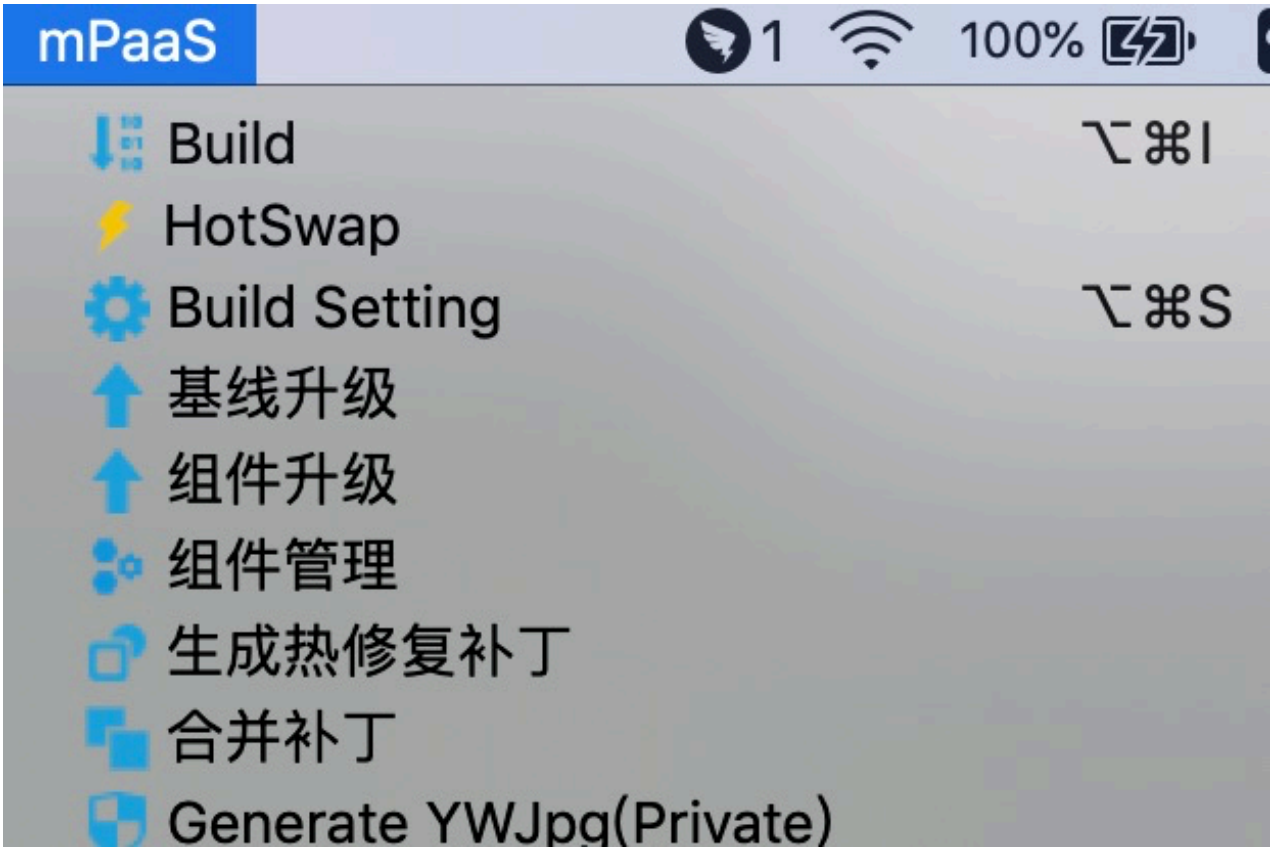
- mPaaS 已经在 10.1.32 版本上进行了 Android 10 适配工作，以上问题均已修复。**您需要升级 mPaaS SDK 或部分组件至 10.1.32 版本**，以确保您的应用中使用的 mPaaS SDK 在 Android 10 设备上能正常工作。
- 同时，我们也建议您阅读谷歌官方文档，检查您应用中的其他功能是否需要进行 Android 10 适配工作。

升级 SDK/组件

重要：请确保您的 mPaaS IDEA 插件已升级至最新版本。

使用 mPaaS IDEA 插件升级 SDK/组件，您可以选择以下两种方式：

- 组件管理
- 基线升级



您需要根据自身情况选择升级方式。如果您：

- 从未使用插件管理过 mPaaS 组件依赖（即 portal 工程根目录下不存在 mpaas_packages.json 文件）。请使用 **组件管理** 功能。升级完成后，您可能还需要 处理定制依赖 。
- 已经使用插件管理组件依赖，但当前使用的 SDK 版本低于 10.1.32。请使用 **基线升级** 功能。
- 已经使用插件管理组件依赖，且当前使用的 SDK 版本为 10.1.32。请先删除 portal 工程根目录下 mpaas_packages.json 文件，再使用 **组件管理** 功能。

具体操作步骤参见 mPaaS 插件管理组件依赖 。

添加配置

如果您使用了 mPaaS 升级 SDK ，您需要添加适配 Android 10 相关配置，否则在 Android 10 设备上将无法调起系统安装界面。

在 AndroidManifest.xml 文件中添加以下权限：

```
<uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES"/>
```

在 portal 工程主 module 的 src/main/res/xml 目录下创建文件 file_paths.xml，文件内容为：

```
<?xml version="1.0"encoding="utf-8"?>
<resources>
```

```
<paths>
<external-files-path
name="download"
path="com.alipay.android.phone.aliupgrade/downloads"/>
<external-path
name="download_sdcard"
path="ExtDataTunnel/files/com.alipay.android.phone.aliupgrade/downloads"/>
</paths>
</resources>
```

在 AndroidManifest.xml 文件中添加以下配置：

```
<provider
android:name="android.support.v4.content.FileProvider"
android:authorities="${applicationId}.fileprovider"
android:exported="false"
android:grantUriPermissions="true">
<meta-data
android:name="android.support.FILE_PROVIDER_PATHS"
android:resource="@xml/file_paths"/>
</provider>
```

API 变更

10.1.32 版本中，以下组件的 API 发生变更：

- 移动分析：
  - 新增适配器，简化使用，参见 自定义事件日志 。
  - 部分 API 废弃，您需使用新的 API，参见 移动分析 SDK 发布说明 ，否则可能导致编译失败。
- 移动推送：新增适配器，简化使用，参见 移动推送 SDK 10.1.32 。
- 移动同步：新增适配器，简化使用，参见 移动同步 SDK 10.1.32 。
- 热修复：新增适配器，简化使用，参见 热修复 SDK 10.1.32 。
- 版本升级：新增适配器，简化使用，参见 版本升级 SDK 10.1.32 。
- 开关配置：新增适配器，简化使用，参见 开关配置 SDK 10.1.32 。
- H5 容器：
  - 新增适配器，简化使用，参见 H5 容器 SDK 10.1.32 。
  - 容器配置方法变更，如果升级前为 10.0.18 版本，您需使用新的容器配置方法，参见 容器配置 10.1.32 ，否则您的容器配置将无法生效。

说明：强烈建议您修改代码，使用适配器方法而非直接使用底层方法。因为某些底层方法可能会在将来的版本中发生变更或废弃，如果您继续使用，在将来的更新中可能需要花费更多的时间进行适配。

处理定制依赖

如果此前您的依赖中包含定制库，您需要：

- 如果您是从低版本 SDK（例如 10.1.20）升级到 10.1.32 版本，您的定制库可能需要基于新版本重新

定制，请 提交工单 或联系 mPaaS 支持人员确认。

- 如果您已使用 10.1.32 版本，则只需更新部分组件。参见下文的 适配 Android 10 更新的库清单 ，检查您的定制库是否包含在其中。如果包含，您的定制库可能需要重新定制，请 提交工单 或联系 mPaaS 支持人员。如果不包含，您可继续使用该定制 bundle 库。
- 如果确认您的定制 bundle 库可继续使用，或已重新定制，您可以 添加定制依赖 。

适配 Android 10 更新的库清单

- quinox-build
- logging-build
- pushsdk-build
- nebulauc-build
- mpaasnebulaadapter-build
- tinyappcommon-build
- aliupgrade-build
- cloudfix-build
- androidsupport-build
- mpaasadapter-build
- nebulaucsdk-build（删除）

检测 SDK 是否适配 Android 10

重要：请确保您的 mPaaS IDEA 插件已升级至最新版本。

要检测当前使用的 SDK 版本是否已经适配 Android 10，您可按以下步骤进行：

1. 在 portal 工程中使用 mPaaS IDEA 插件的 **Android 10 兼容性检测** 功能。

   说明：使用该功能的前提是您已使用插件管理组件依赖（即 portal 工程根目录下已存在 mpaas_packages.json 文件）。

2. 根据提示进行不同处理：
   - 如果您看到如下提示，说明当前使用的 SDK 版本已经适配 Android 10。

- 如果您看到如下提示，点击 **OK** 按照向导升级组件后再次检测。

- 如果您看到如下提示，需要删除 mpaas_packages.json 文件，使用 **组件管理** 重新安装组件后再次检测。

8.3 增加对 64 位 CPU 的支持

背景

为应对谷歌在 Google Play 上发布的应用必须支持 64 位 CPU 架构、且 targetSdkVersion 不小于 28 的要求，mPaaS 在特定基线上对这两项要求进行了支持。如果您的应用需要在 Google Play 上发布，请联系技术支持人员或提交工单获取支持 64 位 CPU 的基线号，并参照本文中的 **更新 SDK**、**生成 64 位/ 32 位 APK** 和 **适配 targetSdkVersion 28**完成适配，并按照 **回归测试** 的内容对适配进行确认。

更新 SDK

前提条件

已使用 mPaaS IDEA 插件管理依赖（即 portal 工程根目录下有 mpaas_packages.json 文件）。

操作步骤

1. 更新 IDEA 插件 至最新版本。
2. 为确保更新成功，请在备份后删除 portal 工程根目录下已有的 mpaas_packages.json 文件。
3. 使用 IDEA 插件的 **组件管理** 功能，勾选 **自定义基线**，填入您获取到的基线号，点击 **OK** 。您也可以参考 定制基线 以获得更多信息。
4. 按照引导安装所需的组件。
5. 如果您需要使用分享 SDK，请在 portal 工程主 module 下的 build.gradle 中添加依赖：

```
bundle 'com.alipay.android.phone.mobilecommon:share-build:1.3.0.190929144835@jar'
manifest 'com.alipay.android.phone.mobilecommon:share-
build:1.3.0.190929144835:AndroidManifest@xml'
```

6. mPaaS SDK 中已移除内置的高德定位和搜索 SDK，如果您需要使用，可自行添加高德官方提供的 Google Play 版本 SDK。

生成 64 位/ 32 位 APK

操作步骤

1. 更新打包插件版本，确保 portal 工程根目录 build.gradle 中打包插件版本不低于：

```
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.8.3'
```

2. 配置打包属性，portal 工程根目录 gradle.properties 中添加：

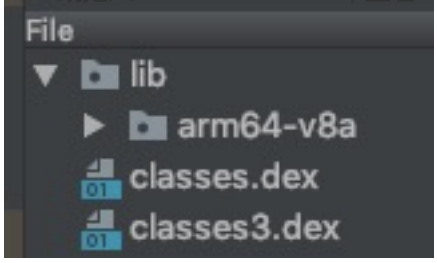
```
// 打32位 APK 就改成32
supportBit=64
```

3. 点击 Android Studio 原生按钮或 mPaaS IDEA 插件打包按钮均可。也可以使用 Gradle 命令打包：

```
gradle clean assembleRelease -PsupportBit=64
```

后续事项

检查 APK 是否为 64 / 32 位，解压 APK 或拖入 Android Studio 中查看 lib 目录（arm64-v8a 代表 64 位，armeabi 代表 32 位）：



适配 targetSdkVersion 28

在 portal 工程主 module 下的 build.gradle 文件中修改属性 targetSdkVersion 28。

通用配置

修改 portal 工程 AndroidManifest.xml：

- 添加权限：

```
<uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
```

- application 节点下添加：

```
<uses-library android:name="org.apache.http.legacy"android:required="false"/>
```

- LauncherActivity 删除以下属性（sdk 已改为通过代码设置）：

```
android:screenOrientation="portrait"
```

其他配置

允许 http 请求

Android 9.0 的网络配置默认禁止了 http 请求，只允许 https 请求，设置 targetSdkVersion 28 将在9.0+设备上启用 9.0 的网络配置。如果您仍然需要发送 http 请求，可通过配置 networkSecurityConfig 开启。

- 在 portal 工程 res/xml 下创建 network_security_config.xml 文件，内容如下：

```
<?xml version="1.0"encoding="utf-8"?>
<network-security-config>
<base-config cleartextTrafficPermitted="true">
<trust-anchors>
<certificates src="system"/>
</trust-anchors>
</base-config>
</network-security-config>
```

- 在 portal 工程 AndroidManifest.xml 中 application 节点添加属性：

```
android:networkSecurityConfig="@xml/network_security_config"
```

更多配置可参考 谷歌官方文档 。

透明背景 Activity 设置屏幕方向 crash

该适配点为 Android 8.0 系统 bug。在 8.0 设备上，当应用 targetSdkVersion > 26 时，透明背景的 Activity 如果设置了屏幕方向，打开该 Activity 就会触发 crash。触发具体条件为：

- Activity 使用的 theme 中 windowIsTranslucent 或 windowIsFloating 属性为 true。
- 在 AndroidManifest.xml 中设置了 screenOrientation 属性，或调用了 setRequestedOrientation 方法。

您需要检查所有 Activity 是否满足触发条件，同时除了您自定义的 style 外，请注意部分常用的系统 theme 也满足条件，例如：

```
@android:style/Theme.Translucent.NoTitleBar
@android:style/Theme.Dialog
```

推荐适配方式：

1. 对于 theme 满足条件的 Activity，删除 AndroidManifest.xml 中 screenOrientation 属性，改为调用 setRequestedOrientation 方法。

   在对应 Activity 或父类中重写 setRequestedOrientation 方法，try catch super.setRequestedOrientation() 兜底：

   ```
   @Override
   public void setRequestedOrientation(int requestedOrientation) {
   try {
   super.setRequestedOrientation(requestedOrientation);
   } catch (Exception ignore) {

   }
   }
   ```

3. mPaaS 提供的 BaseActivity、BaseFragmentActivity、BaseAppCompatActivtiy 均已重写 setRequestedOrientation 方法兜底。
4. 完成上述适配后，虽可避免 crash，但仍可能出现在8.0设备上锁定方向失效的情况，请确保您的 Activity 不会因旋转屏幕发生异常（例如重走生命周期导致某些成员变量为空）。

Android 8.0 系统相关源码：

```
@MainThread
@CallSuper
protected void onCreate(@Nullable Bundle savedInstanceState) {
    if (DEBUG_LIFECYCLE) Slog.v(TAG, "onCreate " + this + ": " + savedInstanceState);

    if (getApplicationInfo().targetSdkVersion > O && mActivityInfo.isFixedOrientation()) {
        final TypedArray ta = obtainStyledAttributes(com.android.internal.R.styleable.Window);
        final boolean isTranslucentOrFloating = ActivityInfo.isTranslucentOrFloating(ta);
        ta.recycle();

        if (isTranslucentOrFloating) {
            throw new IllegalStateException(
                    "Only fullscreen opaque activities can request orientation");
        }
    }

    if (mLastNonConfigurationInstances != null) {
        mFragments.restoreLoaderNonConfig(mLastNonConfigurationInstances.loaders);
    }
    if (mActivityInfo.parentActivityName != null) {
        if (mActionBar == null) {
            mEnableDefaultActionBarUp = true;
        } else {
            mActionBar.setDefaultDisplayHomeAsUpEnabled(true);
        }
    }
```

回归测试

- 您需要分别对32位和64位 apk 做全量回归测试。
- 如果设置 targetSdkVersion 28，全量回归测试的设备中必须包含 Android 9.0+设备，同时对于透明背景 Activity 设置屏幕方向 crash问题，请在 Android 8.0 机型上专项测试。
- 回归测试中您需要重点关注以下组件功能（如果使用）：

组件	验证项目
移动网关	- 开启 签名校验 后，RPC调用是否成功。 - 开启 数据加密 后，RPC调用是否成功。
热修复	- 热修复 是否能够生效。
UC内核	- H5容器使用UC内核 后，各项功能是否正常。 - 小程序（必须使用UC内核），各项功能是否正常。 - 小程序打开手机相册、拍照及预览 是否正常。 - 小程序发起http请求 是否成功（如果强制开启了http）。
扫一扫	- 标准 UI 扫码是否成功。 - 标准 UI 打开手机相册、拍照及预览是否正常。 - 自定义 UI 扫码是否成功，如自定义 UI，需要 适配部分新接口 。
统一存储	- 数据库加密存储 是否正常。 - 文件加密存储 是否正常。
分享	- 分享到新浪微博、QQ 是否成功。

9 参考

9.1 切换工作空间（Workspace）

应用开发过程中，常会有更换应用环境信息或多套环境（即工作空间，Workspace）并行研发的需求。mPaaS 提供的工具可帮助您在开发过程中方便地进行环境切换。根据切换环境的需求不同，分为以下两种方式：

- 静态环境切换
- 动态环境切换

静态环境切换

前置条件

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

公有云

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：

说明：因功能迭代，以下依赖的版本可能会不断增大。

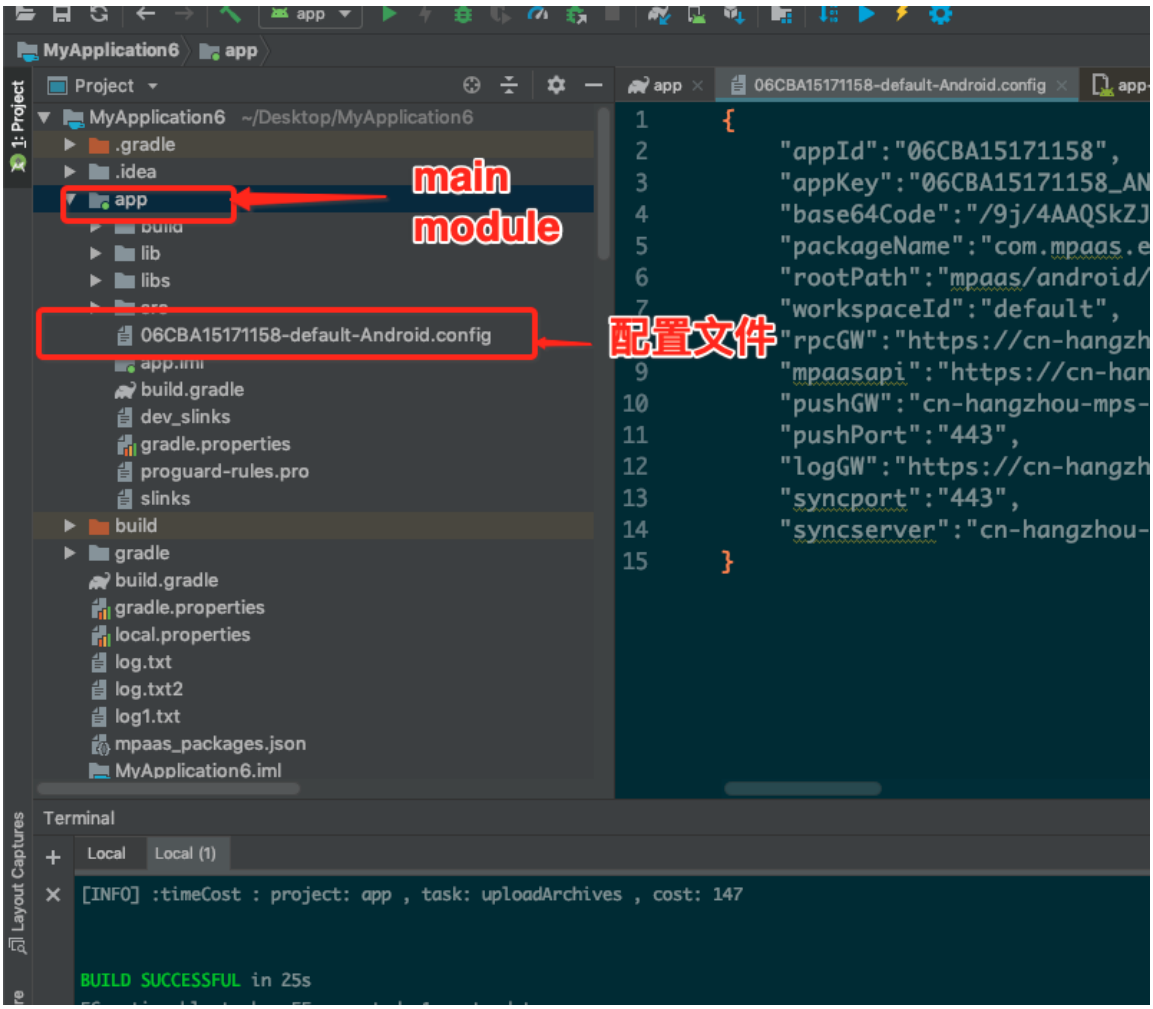
```
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.6.1'
// 版本号必须大于 2.1.0
classpath 'com.android.boost.easyconfig:easyconfig:2.1.0'
```

2. 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'
```

3. 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**。如下图所所示：



专有云

在专有环境中，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：

说明：因功能迭代，以下依赖的版本可能会不断增大。

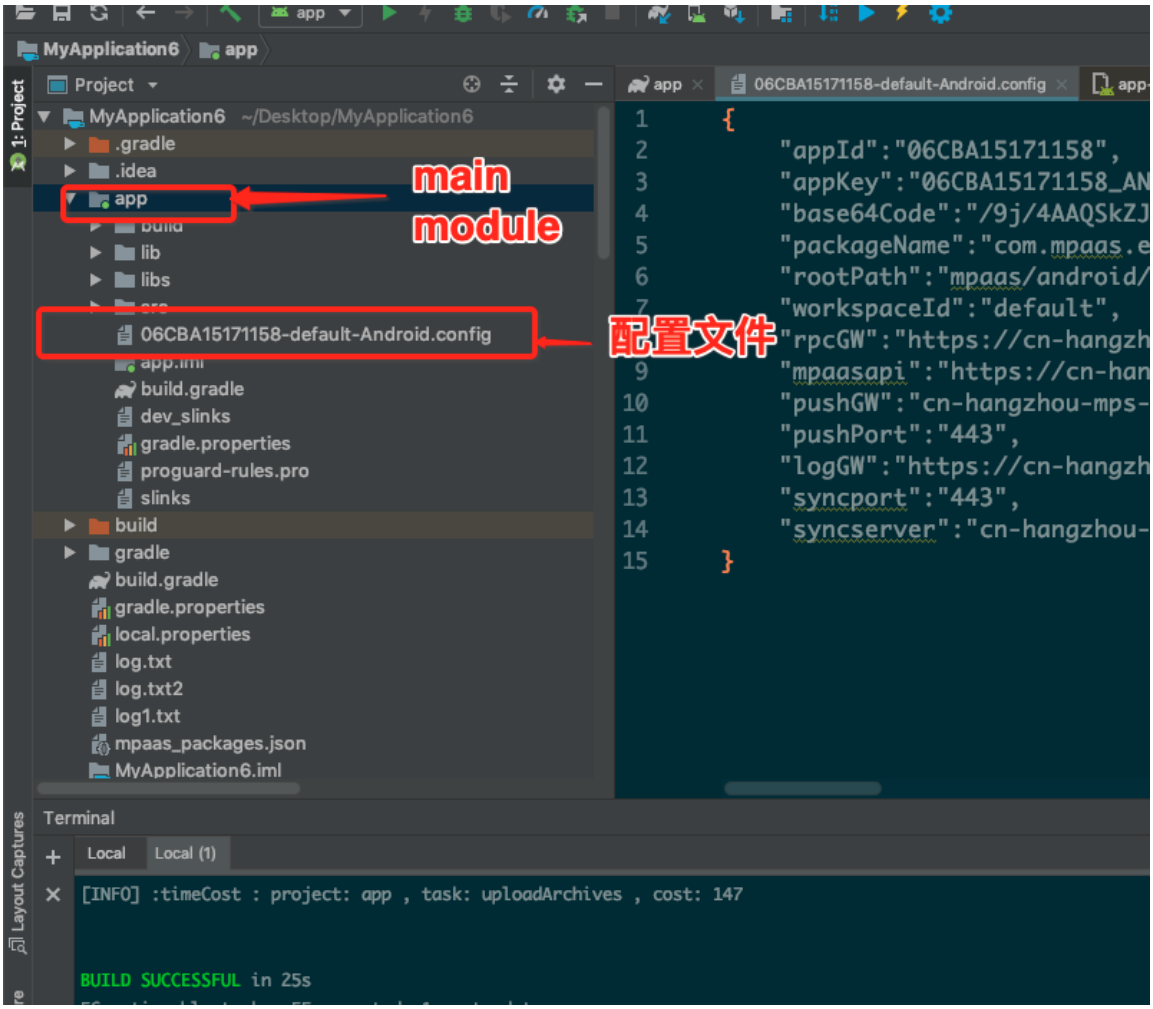
```
classpath 'com.alipay.android:android-gradle-plugin:3.0.0.6.1'
// 版本号必须大于 2.1.0
classpath 'com.android.boost.easyconfig:easyconfig:2.1.0'
```

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

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

3. 从控制台下载对应工作空间的 .config 配置文件。更多信息参考 在控制台创建应用 > 下载配置文件。

将下载的 .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**。如下图所所示：



5. 使用 mPaaS 插件生成 yw_1222.jpg 加密图片。更多信息参见 加密图片（专有云）。

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 下的 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 。

动态环境切换

动态环境切换指客户端不重新打包的情况下，通过修改手机设置中环境选项，动态修改应用的环境信息。

说明：

- 要使用动态切换，**版本必须 ≥ 10.1.32**。
- 动态切换适用于开发阶段多套环境并存且频繁切换的场景。

由于 mPaaS 安全验签机制的限制，更新环境配置信息会修改无线保镖验签 yw_1222.jpg 图片，因此动态切换环境有两个限制：

- **仅适用于开发阶段**：动态切换环境仅适用于开发阶段，上线前请注意删除对应的配置（Release 包会报 RuntimeException 异常）。

  mPaaS 控制台需关闭网络请求验签开关，否则会因验签图片信息不对导致请求失败。

添加动态环境切换 SDK

1. 在 portal 工程主 module 下面的 build.gradle 配置文件中的 dependencies 中添加以下依赖：

```
dependencies {
...
bundle 'com.mpaas.mocksettings:mocksettings-build:1.0.0.191122153217@jar'
manifest 'com.mpaas.mocksettings:mocksettings-build:1.0.0.191122153217:AndroidManifest@xml'
....
}
```

2. 将 portal 工程主 module 下面的 AndroidManifest.xml 中的 application 修改为如下配置：

```
<application
android:name="com.alipay.mobile.quinox.MockSettingsLauncherApplication"
android:debuggable="true"
...
>
...
</application>
```

动态切换

扫描二维码下载 **mPaaS 设置** App。

安装完成后，显示 **mPaaS 设置** App 的图标如下：

2. 将 mPaaS 控制台下载的 config 文件放入手机 SD 卡中。

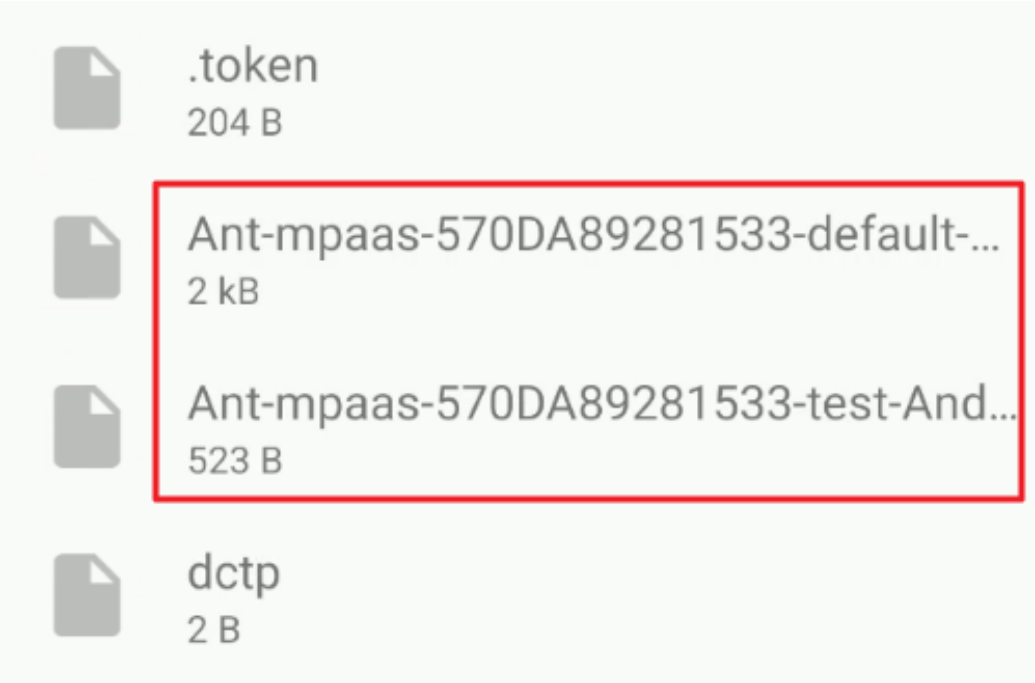
   添加环境。通过 **mPaaS 设置** App 将 config 文件添加到列表中。

   - 打开 **mPaaS 设置** App。

     点击 **环境列表** 页面底部的 **添加环境配置文件**。

     找到要添加的环境配置文件。



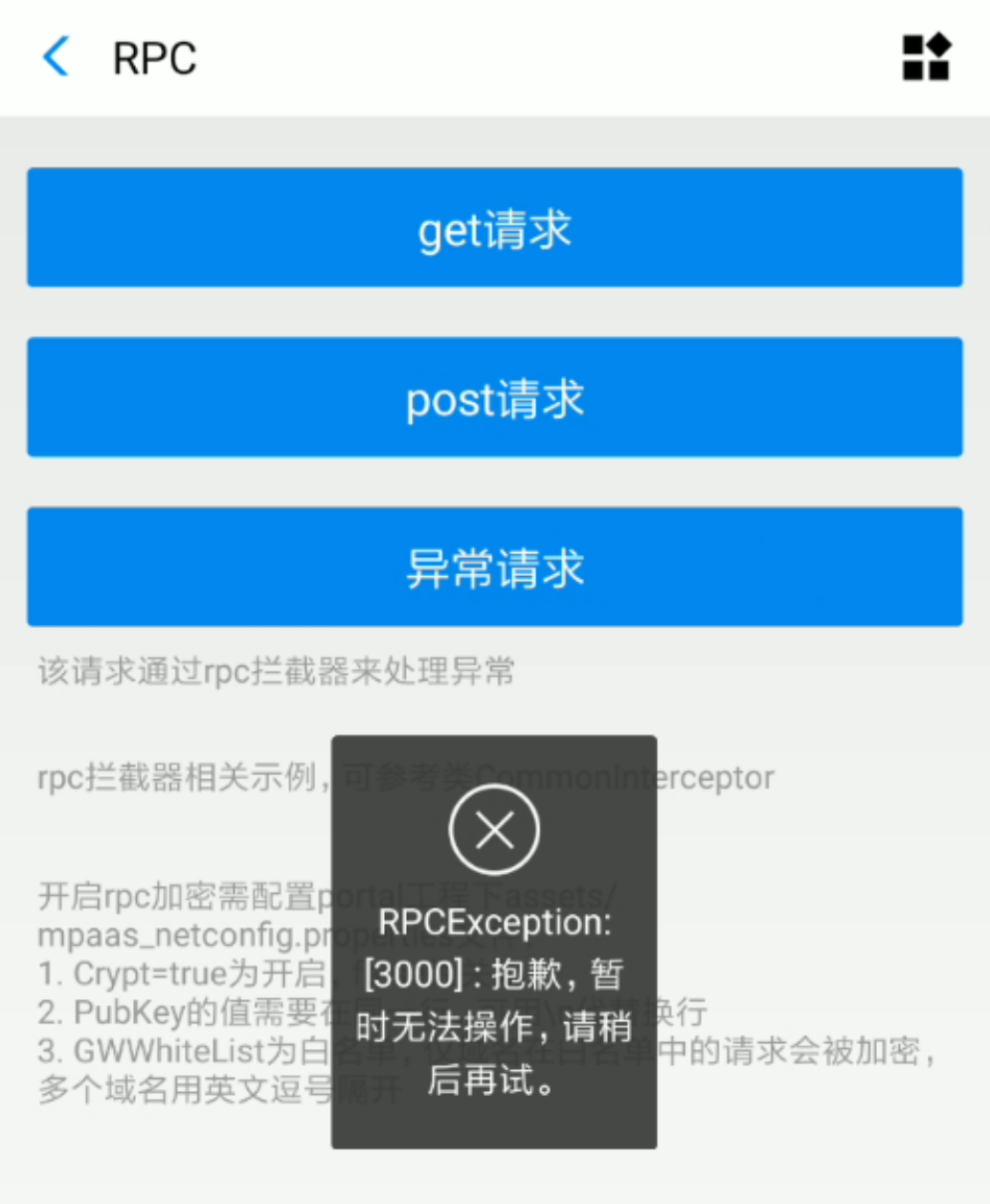
依次将两个文件（正式环境和测试环境）添加至环境列表。

切换环境。

选择上图中的一个环境，点击 **切换**，将其选中为当前环境。

然后启动该环境所对应的 App，测试请求可正常发送，说明环境切换成功。

此时若切换到另一个环境，再重启前一个环境所对应的 App，由于新的环境内没有对应的 operationType，所以系统会报 3000 异常，这是已成功切换到另一个环境后的正常结果。



9.2 管理 Gradle 依赖

9.2.1 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。

9.2.2 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: '*****')
}
}
}
```

9.3 基线列表

9.3.1 基线 10.1.32

10.1.32 版本 SDK 已经发布，具体的变更信息，请参见 发布说明 。

9.3.2 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'
```

9.3.3 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'
```

9.3.4 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

说明：基线 10.0.18 仅支持目标 API 级别 23。

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'
```

9.4 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 ${sdk.dir}/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*

#-keep public class * extends com.alipay.mobile.framework.LauncherApplicationAgent {
# *;
#}

#-keep public class * extends com.alipay.mobile.framework.LauncherActivityAgent {
# *;
#}

-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>;
}
```

说明：如果在您的 bundle 工程中定义了框架类LauncherApplicationAgent 和 LauncherActivityAgent，请注意进行防混淆设置。

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

9.5 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 。

10 常见问题

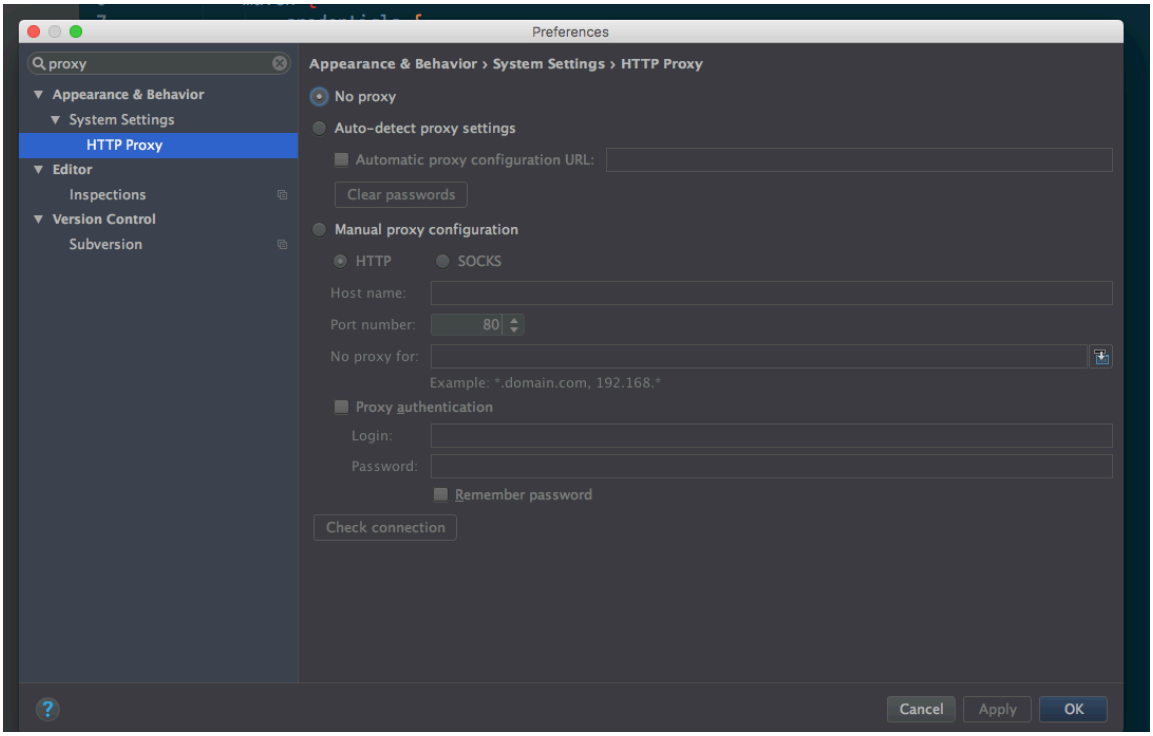
10.1 编译失败或卡顿

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

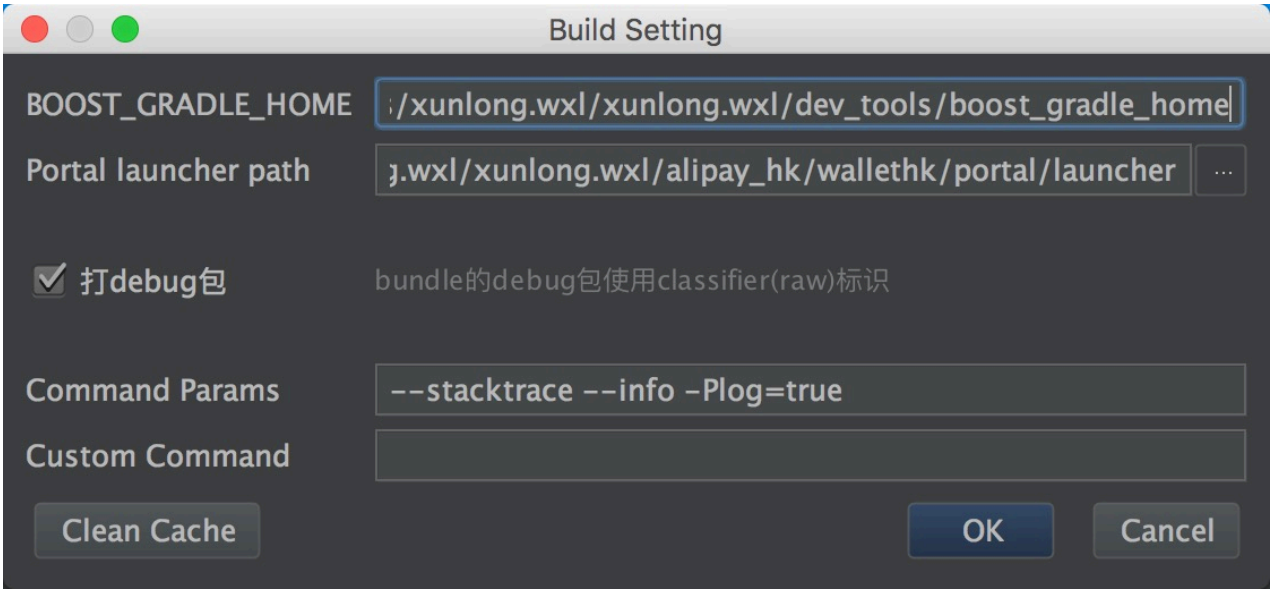
如果编译过程卡顿（等待超过 20 分钟），您可以通过以下步骤提高编译效率：

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/'}
}
}
```

10.2 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。



10.3 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**

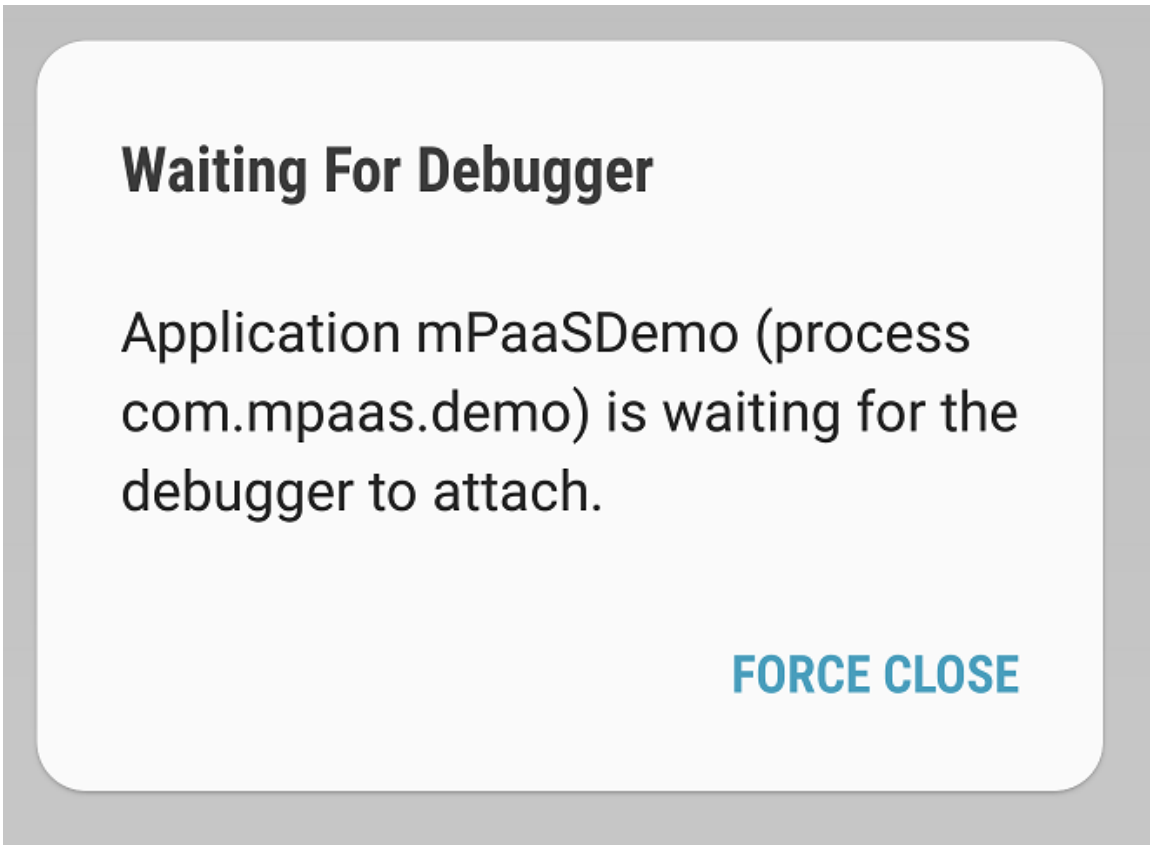
希望调试应用启动时的最初代码，比如在 application init 时初始化代码。

- **操作步骤：**

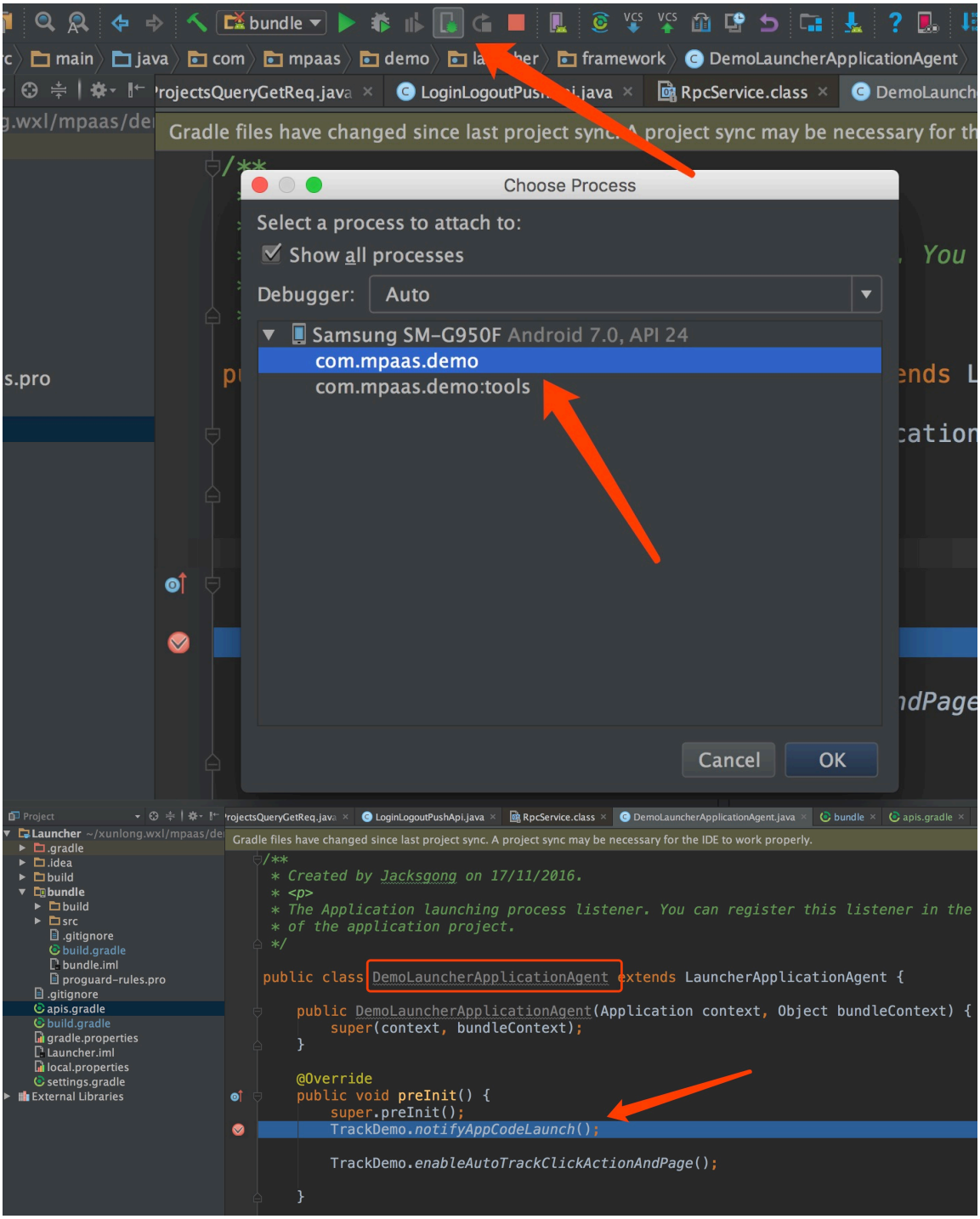
  执行命令 adb shell am start -W -S -D 应用包名/应用第一个启动的页面类名。例如，mPaaS Demo 的包名是 com.mpaas.demo，应用第一个启动的页面类名是com.alipay.mobile.quinox.LauncherActivity，那么可以使用命令行 adb shell am start -W -S -D com.mpaas.demo/com.alipay.mobile.quinox.LauncherActivity 以调试模式启动应用。第一个启动的类名如下图所示：

```
<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。

10.4 使用 MultiDex 的注意事项

mPaaS Inside 工程使用 MultiDex 的注意事项

接入 mPaaS Inside 的时候，我们已经提供了 multiDex，因此您可以把官方提供的 multidex 从 implementation 中删除。

```
dependencies{
implementation 'com.android.support:multidex:1.0.3' //删除此行
}
```

同时，建议您在 gradle 中，android 这个模块下，把 multiDexEnabled true 加上。

```
android {
defaultConfig {
multiDexEnabled true
}
}
```

如果您使用了 mPaaS Inside，同时**不接入热修复**，并且您需要 multidex 支持的话，您依旧要在Application中调用MultiDex.install(this)。

```
public class App extends Application() {
public void attachBaseContext(Context context) {
super.attachBaseContext(context);
MultiDex.install(this);
}
}
```

3. 如果您使用了热修复，也就是使用了 QuinoxlessApplication，您不需要在代码中显式调用。

mPaaS Portal、Bundle 工程使用 MultiDex 的注意事项

Portal 和 Bundle 不建议自行介入 MultiDex，除非您是单 portal 工程，需要使用multiDexEnabled true。如果您的 Bundle 过大，目前只能使用拆分 bundle 的方式进行，**不要在 bundle 中开启 multidex 支持。**

10.5 专有云接入问题

下载配置，接入 mPaaS 后，编译不通过

NullPointerException

```
Caused by: java.lang.NullPointerException
        at com.alipay.mpaas.codegen.modify.bean.QNameBean.containsAttributeValue(QNameBean.java:38)
        at com.alipay.mpaas.codegen.modify.AddedInterceptor.findElementEquals(AddedInterceptor.java:95)
        at com.alipay.mpaas.codegen.modify.ModifyInterceptor.canHandle(ModifyInterceptor.java:33)
        at com.alipay.mpaas.codegen.modify.ModfiedOrAddedInterceptor.canHandle(ModfiedOrAddedInterceptor.java:26)
        at com.alipay.mpaas.codegen.request.AbsInterceptor.handle(AbsInterceptor.java:26)
        at com.alipay.mpaas.codegen.fastconvert.ModifyAndroidManifestConverter.handleModifyElement(ModifyAndroidManifestConverter.java:134)
        at com.alipay.mpaas.easy.config.GradleModifyAndroidManifestConverter.initManifestObjects(GradleModifyAndroidManifestConverter.java:51)
        at com.alipay.mpaas.easy.config.GradleModifyAndroidManifestConverter.convert(GradleModifyAndroidManifestConverter.java:67)
        at com.alipay.mpaas.easy.config.MPaaSManifestTask.convert(MPaaSManifestTask.java:39)
        at com.alipay.mpaas.easy.config.MPaaSManifestTask.executeTask(MPaaSManifestTask.java:24)
        at org.gradle.internal.reflect.JavaMethod.invoke(JavaMethod.java:73)
        at org.gradle.api.internal.project.taskfactory.StandardTaskAction.doExecute(StandardTaskAction.java:46)
        at org.gradle.api.internal.project.taskfactory.StandardTaskAction.execute(StandardTaskAction.java:39)
        at org.gradle.api.internal.project.taskfactory.StandardTaskAction.execute(StandardTaskAction.java:26)
        at org.gradle.api.internal.AbstractTask$TaskActionWrapper.execute(AbstractTask.java:780)
        at org.gradle.api.internal.AbstractTask$TaskActionWrapper.execute(AbstractTask.java:747)
        at org.gradle.api.internal.tasks.execution.ExecuteActionsTaskExecuter$1.run(ExecuteActionsTaskExecuter.java:121)
        at org.gradle.internal.progress.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:336)
        at org.gradle.internal.progress.DefaultBuildOperationExecutor$RunnableBuildOperationWorker.execute(DefaultBuildOperationExecutor.java:328)
        at org.gradle.internal.progress.DefaultBuildOperationExecutor.execute(DefaultBuildOperationExecutor.java:199)
        at org.gradle.internal.progress.DefaultBuildOperationExecutor.run(DefaultBuildOperationExecutor.java:110)
        at org.gradle.api.internal.tasks.execution.ExecuteActionsTaskExecuter.executeAction(ExecuteActionsTaskExecuter.java:110)
        at org.gradle.api.internal.tasks.execution.ExecuteActionsTaskExecuter.executeActions(ExecuteActionsTaskExecuter.java:92)
        ... 103 more
```

一般都是配置文件（conf 文件）的问题，需要对字段进行检查。检查13个字段是否有缺少；和公有云下载过来的文件进行对比，确认字段名是否正确。


蚂蚁金服
ANT FINANCIAL
金融科技
FINANCIAL TECHNOLOGY