


 Search or jump to...


/


Pull requestsIssuesCodespacesMarketplaceExplore










 fw h1990/**react-native-upload** Public


 Watch 3


 Fork 21

 Star 104

CodeIssues 10Pull requestsActionsProjectsSecurityInsights


 master


 1 branch


 21 tags


Go to file


Add file


 Code


 fw h1990 chore: Bump version 2.2.1358bcbcMay 19, 202177 commits


 srcfeactor: Set qrcode error level to LApril 24, 202122:50


 .gitignorefeat: Upload app to pgyer.comAugust 31, 2019 23:00


 LICENSEInitial commitAugust 30, 2019 17:30


 README.mdchore: Change the default install type to publish for Pyger platformDecember 25, 2020

 example.pngfix: Unable to bundle ios app in some case. It's a breaking changeSeptember 12, 2019

 package.jsonchore: Bump version 2.2.1May 19, 202115:45

 publish.shfeat: Upload app to pgyer.comAugust 31, 2019 23:00

 tsconfig.jsonfeat: Upload app to pgyer.comAugust 31, 2019 23:00

 yarn.lockfeat: Print qrcode which contains the download linkApril 24, 202122:01

## README.md

# react-native-upload

一键上传 android/ios APP到各个测试平台和 app store

## 支持系统

MacOs (使用了 bash 语法, 而且ios只能依赖xcodesoftware)

## 已集成平台

- 蒲公英 (android + ios)
- fir.im (android + ios)
- App Store (ios)
- Test Flight (ios)

# 安装

```
# Npm
npm install react-native-upload --save-dev

# Yarn
yarn add react-native-upload --dev
```

# 生成配置

先执行这个命令:

```
npx upload-init
```

执行命令后会在项目根目录中创建一个 upload.json 文件, 并生成以下内容:

```
// 未用到的配置, 可以置空不填写, 也可以直接删除
{

  // 上传到蒲公英
  "pgy": {
    // 上传凭证, 访问链接 https://www.pgyer.com/account/api , 复制Api Key
    "pgy_api_key": "",
    // App安装方式, 共有三种 1:公开, 2:密码安装, 3:邀请安装
    "pgy_install_type": 1,
    // App安装时的访问密码, 选择 "2密码安装" 时, 访问密码必填
    "pgy_install_password": "",
    "ios_export_plist": "./ios-export/ad-hoc.plist"
  },

  // 上传到fir.im
  "fir": {
    // 上传凭证, 访问链接 https://betaqr.com/apps/apitoken , 复制token
    "fir_api_token": "",
    "ios_export_plist": "./ios-export/ad-hoc.plist"
  },

  // 上传到App Store
  "app_store": {
    #####
    ## 注意:user_ 与 api_ 是互斥的, 只需要填写其中一组即可正常上传 ##
    #####

    // 用户 (APPLE_ID)必须拥有该APP的上传权限
  }
}
```

```

"user_name": "",
// 随机密码, 访问链接 https://appleid.apple.com/account/manage , 点击 App专用密码 生成密码
"user_password": "",

// 密钥ID, 访问链接 https://appstoreconnect.apple.com/access/api , 点击蓝色圆形+号图标即可生成密钥。
#####
## 注意:生成密钥后, 必须下载密钥文件, 并复制到以下随意一个文件夹中:      ##
##      ./private_keys      ##
##      ~/private_keys      ##
##      ~/.private_keys      ##
##      ~/.appstoreconnect/private_keys      ##
#####
"api_key": "",
// 生成密钥后, 密钥的列表上方有个 Issuer ID
"api_issuer": "",

"ios_export_plist": "./ios-export/app-store.plist"
},

// 上传到Test Flight
// 默认从App_store配置中拿 user_* 或者 api_*, 也可以在test_flight配置下覆盖这几个参数
"test_flight": {
    "ios_export_plist": "./ios-export/ad-hoc.plist"
}
}

```

## 准备工作

ios\_export\_plist 即ios打包参数的存放路径。因为ios的打包参数十分复杂, 每个项目都会有一些差异, 所以为了保证能准确无误地打包出符合要求的app, 您需要手动执行一次以下内容(只需一次)。

### 1、手动打包

点击 Xcode -> Product -> Archive , 等待打包完成

### 2、导出app

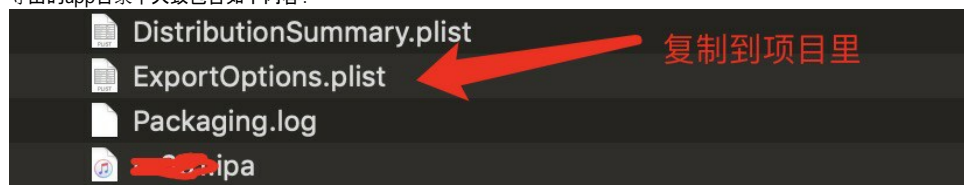
点击 Xcode -> Window -> Organizer , 选择刚才打的包, 点击右边按钮 Distribute App , 进行一系列选择之后, 最后点击 Export 按钮, 把文件下载到本地磁盘。

一般来说, 您可能需要手动导出两份app文件:

- 如果您想把app上传到测试平台, 请选择 Ad Hoc 的打包方式
- 如果您想把app上传到App Store, 请选择 ios App Store 的打包方式

### 3、复制plist文件

导出的app目录中大致包含如下内容:



请将文件 ExportOptions.plist 复制到项目中, 并保持与配置 ios\_export\_plist 所指向的路径一致。推荐您将文件重命名为打包方式的名称, 如 ad-hoc.plist 、 app-store.plist 等。

## 自动打包上传

### 蒲公英

```

npx upload-pgy

# 填写更新日志
npx upload-pgy --log "增加xxx功能"

# 忽略平台
npx upload-pgy --no-android
npx upload-pgy --no-ios

# android默认打包release版本, 可以改成debug版本
npx upload-pgy --variant=debug

# 多渠道打包时, 默认上传所有生成的android apk文件, 可以使用正则表达式指定文件名称
npx upload-pgy --apk=app-release.apk
npx upload-pgy --apk=x86_64
npx upload-pgy --apk=release-[0-9]

```

```
npx upload-fir

# 填写更新日志
npx upload-fir --log "增加xxx功能"

# 忽略平台
npx upload-fir --no-android
npx upload-fir --no-ios

# android默认打包release版本, 可以改成debug版本
npx upload-fir --variant=debug

# 多渠道打包时, 默认上传所有生成的android apk文件, 可以使用正则表达式指定文件名称
npx upload-fir --apk=app-release.apk
npx upload-fir --apk=x86_64
npx upload-fir --apk=release-[0-9]
```

## App Store

```
npx upload-appstore

# 或者缩写
npx upload-as
```

## Test Flight

```
npx upload-testflight

# 或者缩写
npx upload-tf
```

# 只打包不上传

由于某种原因, 您只想安安静静地打包出app而不上传到任何平台, 您可以用以下指令处理您的需求:

```
# 同时打包android和ios
npx upload-build --ios-export-plist path/to/xxx.plist

# 安卓默认打包release版本, 可以改成debug版本
npx upload-build --ios-export-plist path/to/xxx.plist --variant=debug

# 单独打包android
npx upload-build --no-ios


# 单独打包ios
npx upload-build --no-android --ios-export-plist path/to/xxx.plist
```


欢迎使用并给我提建议, 有任何通用平台需要集成也可以cue我


## About


( MacOS ) 一键打包部署 android/ios APP到各个测试平台和App Store. ci/cd神器, 拒绝繁琐


[react-native-deploy](#) [react-native-pgy](#) [react-native-fir](#) [react-native-appstore](#)

 [Readme](#)


 [MIT license](#)

 **104** stars

 **3** watching

 **21** forks

## Releases 21

 v2.2.0 Latest  
Apr 24, 2021  
[+ 20 releases](#)

## Packages

No packages published

## Languages

 Shell72.3%  JavaScript27.7%

 © 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)