移动开发平台

微信 QQ 登录

产品文档

腾讯云

文档目录

微信 QQ 登录
产品介绍

最近更新时间：2018-03-28 16:53:34

简介

Authorization 是 MobileLine 的身份验证服务，目前仅仅支持使用 QQ 和微信来进行身份验证。当调起 QQ 或者微信登录成功后，可以获取用户 QQ 或者 微信的用户昵称以及 ID 号等基本信息，您可以自己在后台存储这些信息用于用户身份校验。

优势

带来海量新用户

使用 QQ 和微信登录可减少用户登录交互操作，大大降低应用的登录成本，给您带来海量新用户。

丰富用户资料

应用可以通过用户资料接口获取用户的空间昵称和头像，来丰富用户注册资料。

客户案例

王者荣耀、欢乐斗地主、天天酷跑、保卫萝卜。

Android 文档
Android 快速入门

最近更新时间：2018-08-16 16:26:29

准备工作

您首先需要一个 Android 工程，这个工程可以是您现有的工程，也可以是您新建的一个空的工程。

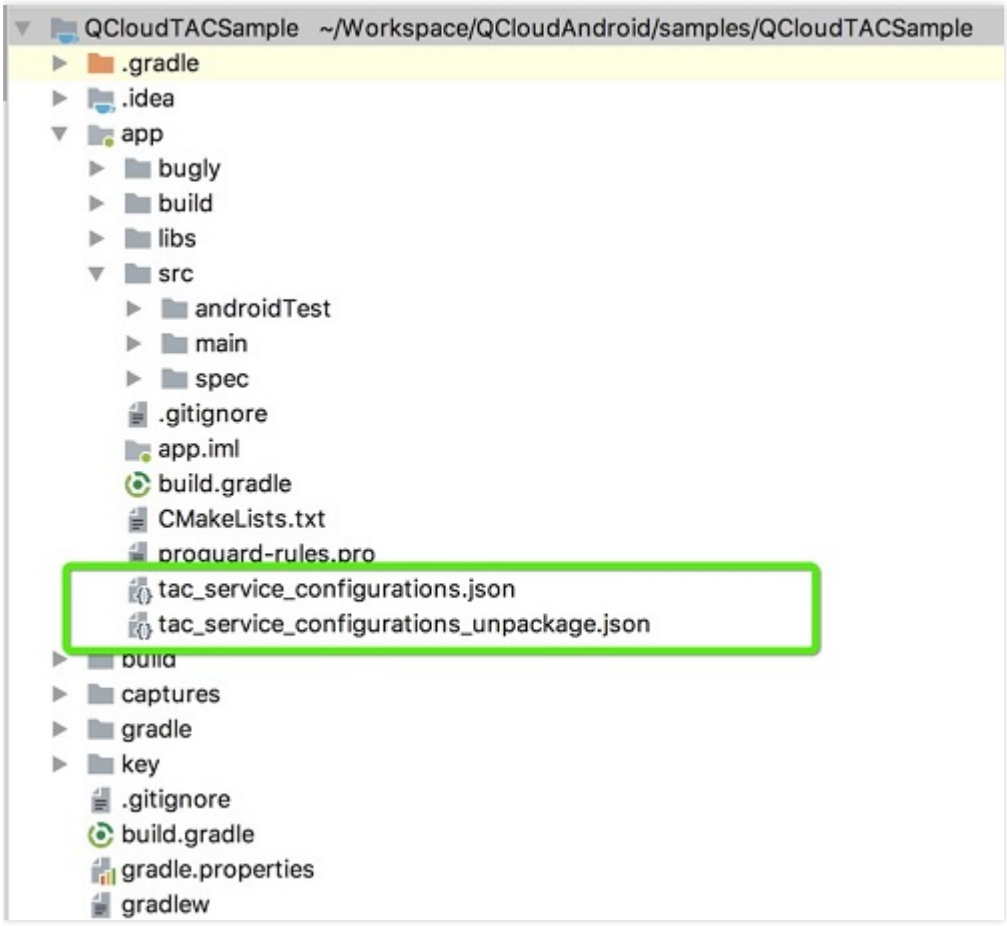
第一步：创建项目和应用（已完成请跳过）

在使用我们的服务前，您必须先在 MobileLine 控制台上 创建项目和应用。

第二步：添加配置文件（已完成请跳过）

在您创建好的应用上单击【下载配置】按钮来下载该应用的配置文件的压缩包：

解压该压缩包，您会得到 tac_service_configurations.json 和 tac_service_configurations_unpackage.json 两个文件，请您如图所示添加到您自己的工程中去。



注意：
请您按照图示来添加配置文件，`tac_service_configurations_unpackage.json` 文件中包含了敏感信息，请不要打包到 apk 文件中，MobileLine SDK 也会对此进行检查，防止由于您误打包造成的敏感信息泄露。

第三步：集成 SDK

1. gradle 集成 SDK

您需要在工程级 build.gradle 文件中添加 SDK 插件的依赖：

```
buildscript {
...
dependencies {
classpath 'com.android.tools.build:gradle:3.0.1'
// 添加这行
classpath 'com.tencent.tac:tac-services-plugin:1.3.+'
}
```

```
}

allprojects {
repositories {
...
maven { url "https://dl.bintray.com/thelasterstar/maven/" }
}
}
```

在您应用级 build.gradle 文件（通常是 app/build.gradle）中添加 social 服务依赖，并使用插件：

```
dependencies {
// 增加这行
compile 'com.tencent.tac:tac-core:1.3.+'
compile 'com.tencent.tac:tac-authorization:1.3.+'
}
...

// 在文件最后使用插件
apply plugin: 'com.tencent.tac.services'
```

2. 添加 QQ SDK

手动下载 QQ SDK ，并拷贝到应用模块的 `app/libs` 文件夹下，并在您应用级 build.gradle（通常是 app/build.gradle）文件中包含对 libs 目录的依赖：

```
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

第四步：配置QQ、微信渠道

登录 SDK 需要配置 QQ、微信渠道才能正常工作，关于如何配置，请参见 配置第三方渠道。

到此您已经成功接入了 MobileLine 登录与授权服务。

Proguard 配置

如果您的代码开启了混淆，为了 SDK 可以正常工作，请在 `proguard-rules.pro` 文件中添加如下配置：

```
# MobileLine Core

-keep class com.tencent.qcloud.core.** { *;}
-keep class bolts.** { *;}
-keep class com.tencent.tac.** { *;}
-keep class com.tencent.stat.*{*;}
-keep class com.tencent.mid.*{*;}
-dontwarn okhttp3.**
-dontwarn okio.**
-dontwarn javax.annotation.**
-dontwarn org.conscrypt.**

# Wechat

-keep class com.tencent.mm.opensdk.** {*;}
-keep class com.tencent.wxop.** {*;}
-keep class com.tencent.mm.sdk.** {*;}

# QQ
-keep class com.tencent.connect.** {*;}
-keep class com.tencent.open.** {*;}
-keep class com.tencent.tauth.** {*;}
-keep class com.tencent.mobileqq.openpay.** {*;}
```

QQ 渠道编程手册

最近更新时间：2018-06-11 18:03:32

准备工作

请确保您已经通过 gradle 配置或者手动将 QQ SDK 集成到工程中，并配置好 App ID。

使用 QQ 登录功能

1. 在登录 Activity 添加回调处理

在启动登录的 Activity 的 onActivityResult 中添加 QQ 登录回调的处理，否则在某些低端机上可能无法正确处理回调：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);
if (qqAuthProvider != null) {
qqAuthProvider.handleActivityResult(requestCode, resultCode, data);
}
}
```

2. 请求 QQ 登录

在用户登录界面，您可以调用以下方法让用户选择用 QQ 账号登录：

```
// 获取实例
TACAuthorizationService service = TACAuthorizationService.getInstance();
QQAuthProvider qqAuthProvider = service.getQQAuthProvider(context);

// 启动登录
qqAuthProvider.signIn(activity, new QCloudResultListener<OAuth2Credentials>() {
@Override
public void onSuccess(OAuth2Credentials credentials) {
// 登录成功，可以拿到QQ的用户凭证
mOAuth2Credentials = credentials;

String accessToken = credentials.getAccessToken();
String openId = credentials.getOpenId();
}
```

```
@Override
public void onFailure(QCloudClientException clientException, QCloudServiceException serviceExcepti
on) {
// 登录失败
}
});
```

QQ 获取的凭证有效期是 3 个月，之后需要用户重新登录授权。

3. 获取用户信息

登录成功后，您可以使用有效的用户凭证，调用 getUserInfo 方法获取 QQ 用户信息：

```
// 获取实例
TACAuthorizationService service = TACAuthorizationService.getInstance();
QQAuthProvider qqAuthProvider = service.getQQAuthProvider(context);

// 获取用户信息
qqAuthProvider.getUserInfo(mOAuth2Credentials, new QCloudResultListener<TACOpenUserInfo>() {
@Override
public void onSuccess(TACOpenUserInfo result) {
userInfoView.setText(result.toString());
}

@Override
public void onFailure(QCloudClientException clientException, QCloudServiceException serviceExcepti
on) {
// 获取出错
}
});
```

微信渠道编程手册

最近更新时间：2018-06-11 18:03:16

准备工作

请确保您已经通过 gradle 配置或者手动将微信 SDK 集成到工程中，并配置好 App ID。

使用微信登录功能

1. 请求微信登录

在用户登录界面，您可以调用以下方法让用户选择用微信账号登录：

```
// 获取实例
TACAuthorizationService service = TACAuthorizationService.getInstance();
WeChatAuthProvider weChatAuthProvider = service.getWeChatAuthProvider(context);

// 启动登录
weChatAuthProvider.signIn(activity, new QCloudResultListener<OAuth2Credentials>() {
@Override
public void onSuccess(OAuth2Credentials credentials) {
// 登录成功，可以拿到微信登录的authorization code
String authorizationCode = credentials.getAuthorizationCode();
}

@Override
public void onFailure(QCloudClientException clientException, QCloudServiceException serviceExcepti
on) {
// 登录失败
});
```

在用户登录成功之后，通过 authorization code 和 secret key 可以获取真正的 access token，出于安全的考虑，secret key 不建议明文存放在客户端，所以建议把这个请求的过程放到后端服务器中进行。详细的接口可以参考：微信接口说明

3. 获取用户信息

登录成功后，您可以使用有效的用户凭证，调用 getUserInfo 方法获取微信用户信息：

```
// 获取实例
TACAuthorizationService service = TACAuthorizationService.getInstance();
```

```
WeChatAuthProvider weChatAuthProvider = service.getWeChatAuthProvider(context);

// 获取用户信息
weChatAuthProvider.getUserInfo(mOAuth2Credentials, new QCloudResultListener<TACOpenUserInfo
>() {
@Override
public void onSuccess(TACOpenUserInfo result) {
userInfoView.setText(result.toString());
}

@Override
public void onFailure(QCloudClientException clientException, QCloudServiceException serviceExcepti
on) {
// 获取出错
}
});
```

4. 刷新 token

微信支持后台刷新 access token，access token 的生命周期通常只有 2 个小时，可以通过刷新的方式延长到一个月，之后需要用户重新登录授权：

```
// 获取实例
TACAuthorizationService service = TACAuthorizationService.getInstance();
WeChatAuthProvider weChatAuthProvider = service.getWeChatAuthProvider(context);

// 刷新token
weChatAuthProvider.refreshCredentialInBackground(mOAuth2Credentials,
new QCloudResultListener<OAuth2Credentials>() {
@Override
public void onSuccess(OAuth2Credentials result) {
//token刷新成功
}

@Override
public void onFailure(QCloudClientException clientException,
QCloudServiceException serviceException) {
if (WeChatAuthProvider.isUserNeedSignIn(serviceException)) {
// 刷新失败，需要用户重新登录微信授权
} else {
// 其他原因导致刷新失败，可以再次重试
}
}
});
```

Android 手动集成

最近更新时间：2018-07-10 13:15:28

如果您无法通过 gradle 远程依赖的方式来集成 SDK，我们提供了手动的方式来集成服务：

1. 下载服务资源压缩包

1. 下载 移动开发平台（MobileLine）核心框架资源包，并解压。
2. 下载 移动开发平台（MobileLine） Authorization 资源包，并解压。

2. 集成 QQ SDK

下载 QQ SDK ，并拷贝到应用模块的 libs 文件夹下。

3. 修改您工程的 AndroidManifest.xml 文件。

请把下载的 QQ open jar 包添加到 classpath 中，并在 AndroidManifest.xml 文件中添加以下权限和 Activity：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

<application>
...

<activity
android:name="com.tencent.tauth.AuthActivity"
android:launchMode="singleTask"
android:noHistory="true">
<intent-filter>
<action android:name="android.intent.action.VIEW" />

<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.BROWSABLE" />

<!--<data android:scheme="tencent${应用在QQ互联的app id}" />-->
</intent-filter>
</activity>

<activity
android:name="com.tencent.connect.common.AssistActivity"
android:screenOrientation="behind"
android:theme="@android:style/Theme.Translucent.NoTitleBar"
```

```
android:configChanges="orientation|keyboardHidden">
</activity>
</application>
```

4. 配置第三方渠道

登录 SDK 需要配置QQ、微信等第三方渠道才能正常工作，关于如何配置第三方渠道，请参见 配置第三方渠道。

到此您已经成功接入了 MobileLine 登录与授权服务。

配置第三方渠道

最近更新时间：2018-07-10 13:16:31

MobileLine 的分享和三方登录服务使用到了 QQ 和 微信等开放平台的能力，因此需要您将应用注册到对应的开放平台上，并获取一个应用 id。

配置 QQ 渠道

1. 注册应用

如果您还没有在 QQ 互联平台 注册应用，请先移步注册您的应用。

2. 配置应用

在您的应用模块（通常是app）目录下，新建一个或者在名为 `tac_service_configurations_social.json` 的文件中添加如下内容：

```
{
"services": {
"social": {
"qq": {
"appId": "应用在QQ互联平台的 app id"
}
}
}
}
```

配置 微信 渠道

1. 注册应用

如果您还没有在 微信开放平台 注册您的应用，请先移步注册您的应用，并且获取应用**登录**和**分享**能力。

2. 配置应用

在您的应用模块（通常是app）目录下，新建一个或者在名为 `tac_service_configurations_social.json` 的文件中添加如下内容：

```
{
"services": {
```

```
"social": {
"wechat": {
"appId": "应用在微信开放平台的 app id"
}
}
}
}
```

3. 添加微信 SDK 回调 Activity

微信 SDK 需要一个类名的 `WXEntryActivity` 的文件，用于拿到通过 SDK 登录或者分享后的结果回调。

我们已经帮您在 `AndroidManifest` 中注册了该 Activity，您只需要在您的应用包名（注意是最终发布的 App 的包名）下新建一个 `wxapi` 的包，然后新建一个名为 `WXEntryActivity` 的类，该类直接继承基类 `WeChatBaseSignInActivity` 即可。

例如下面的例子，App 的包名是 `com.tencent.tac.sample` ，因此 `WXEntryActivity` 的包名是 `com.tencent.tac.sample.wxapi` 。

```
package com.tencent.tac.sample.wxapi;

import com.tencent.tac.social.WeChatBaseHandlerActivity;

public class WXEntryActivity extends WeChatBaseHandlerActivity {
}
```

配置 微博 渠道

1. 注册应用

如果您还没有在 微博开放平台 注册您的应用，请先移步注册您的应用，并且获取应用**登录**和**分享**能力。（注意，若是应用没有被微博平台审核成功，则只有注册了此应用的微博账号才能使用分享功能)

2. 配置应用

在您的应用模块（通常是app）目录下，新建一个或者在名为 `tac_service_configurations_social.json` 的文件中添加如下内容：

```
{
"services": {
"social": {
"weibo": {
"appId": "微博开发平台获取的App Key",
```

```
"redirectUrl":"微博开发平台获取授权回调页地址url",
"scope":"微博开发平台scope, 可不填写"
}
}
}
}
```

iOS 文档
iOS 快速入门

最近更新时间：2019-03-04 16:25:37

移动开发平台（MobileLine）使用起来非常容易，只需要简单的 4 步，您便可快速接入移动崩溃监测。接入后，您即可获得我们提供的各项能力，减少您在开发应用时的重复工作，提升开发效率。

准备工作

为了使用移动开发平台（MobileLine）iOS 版本的 SDK，您首先需要一个 iOS 工程，这个工程可以是您现有的工程，也可以是您新建的一个空的工程。

第一步：创建项目和应用

在使用我们的服务前，您必须先在 MobileLine 控制台上 创建项目和应用。

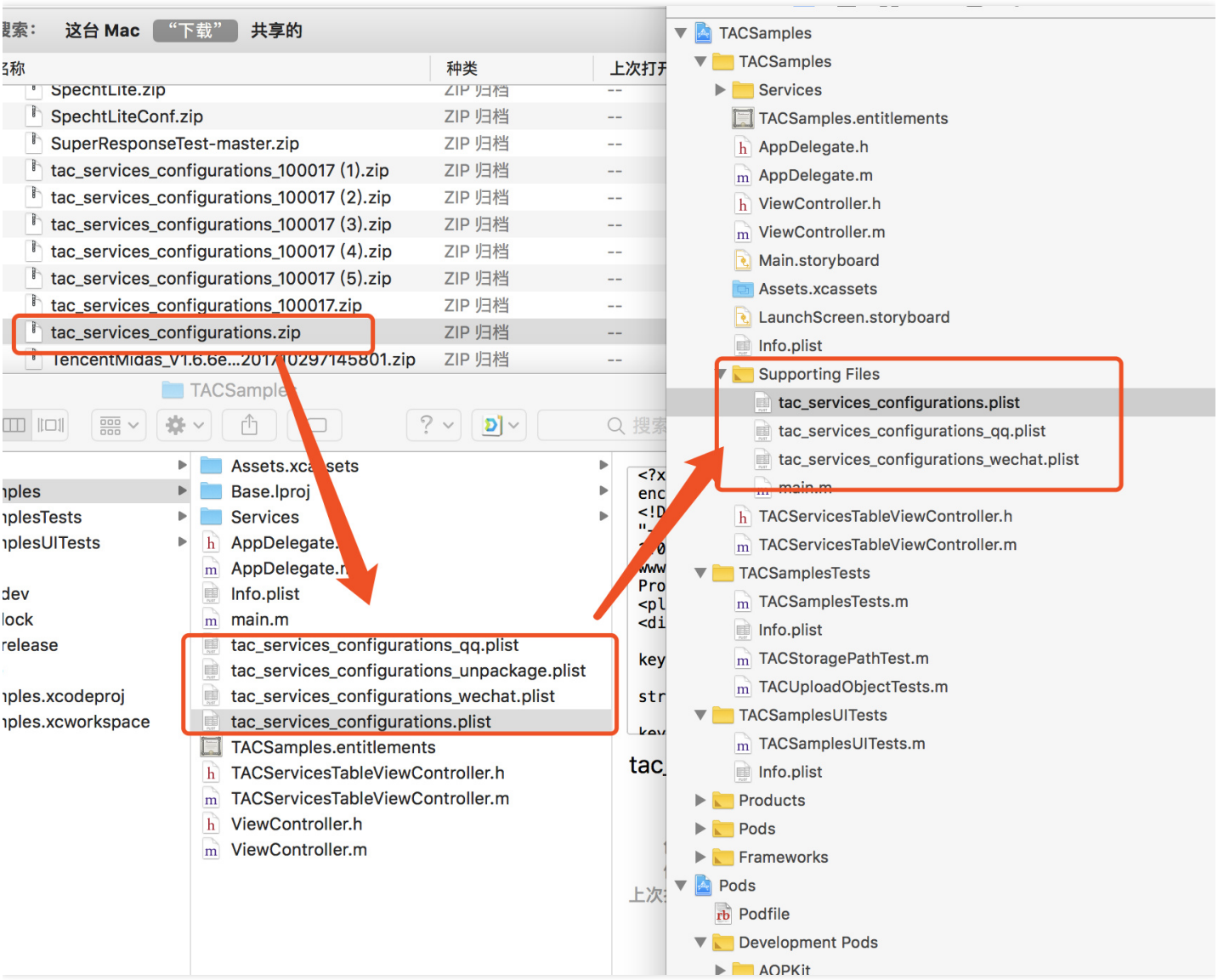
> 如果您已经在 MobileLine 控制台上创建过了项目和应用，请跳过此步。

第二步：添加配置文件

> 如果您已经添加过配置文件，请跳过此步。

创建好应用后，您可以单击红框中的【下载配置】来下载该应用的配置文件的压缩包：

解压后将 tac_services_configurations.plist 文件集成进项目中。其中有一个 tac_services_configurations_unpackage.plist 文件，请将该文件放到您工程的根目录下面(**切记不要将改文件添加进工程中**)。 添加好配置文件后，继续单击【下一步】。



注意：

请您按照图示来添加配置文件， `tac_service_configurations_unpackage.plist` 文件中包含了敏感信息，请不要打包到 apk 文件中，MobileLine SDK 也会对此进行检查，防止由于您误打包造成的敏感信息泄露。

第三步：集成 SDK

如果还没有 Podfile，请创建一个。

```
$ cd your-project directory
$ pod init
```

并在您的 Podfile 文件中添加移动开发平台（MobileLine）的私有源：

```
source "https://git.cloud.tencent.com/qcloud_u/cocopoads-repo"
source "https://github.com/CocoaPods/Specs"
```

在 Podfile 中添加依赖：

```
pod 'TACAuthorization'
```

目前我们支持两个渠道的支付能力：

模块名	渠道名称
TACAuthorizationQQ	QQ
TACAuthorizationWechat	微信

如果您需要两个渠道的能力则需要添加依赖

```
pod 'TACAuthorizationQQ'
pod 'TACAuthorizationWechat'
```

如果您使用了QQ支付，则您将引入 TACSocialQQ 模块，请参考该模块的配置，并添加配置文件 `tac_services_configurations_qq.plist` ，并将文件添加进XCode工程中。文件内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/Propert
yList-1.0.dtd">
<plist version="1.0">
<dict>
<key>services</key>
<dict>
<key>social</key>
<dict>
<key>qq</key>
<dict>
<key>appId</key>
<string>请填充您的 AppId</string>
<key>permissions</key>
<array>
```

```
<string>get_user_info</string>
<string>get_simple_userinfo</string>
<string>add_t</string>
</array>
</dict>
</dict>
</dict>
</dict>
</plist>
```

如果您使用微信支付，则您将引入 TACSocialWechat模块，请参考该模块的配置，并添加配置文件 `tac_services_configurations_wechat.plist` ，并将文件添加进XCode工程中。文件内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>services</key>
<dict>
<key>social</key>
<dict>
<key>wechat</key>
<dict>
<key>appId</key>
<string>wx256642f480c15e3e</string>

<!-- 注意，appKey 存放在本地，随 App 一起发布出去是危险的行为，存在着极大的泄露风险。调试时可以讲 appKey 写在配置文件中，但在生产环境里，需要从您的服务端去获取 appKey。 -->
<key>appKey</key>
<string>请填充您的 APPKey</string>
</dict>
</dict>
</dict>
</dict>
</plist>
```

> 注意，appKey 存放在本地，随 App 一起发布出去是危险的行为，存在着极大的泄露风险。调试时可以讲 appKey 写在配置文件中，但在生产环境里，需要从您的服务端去获取 appKey。

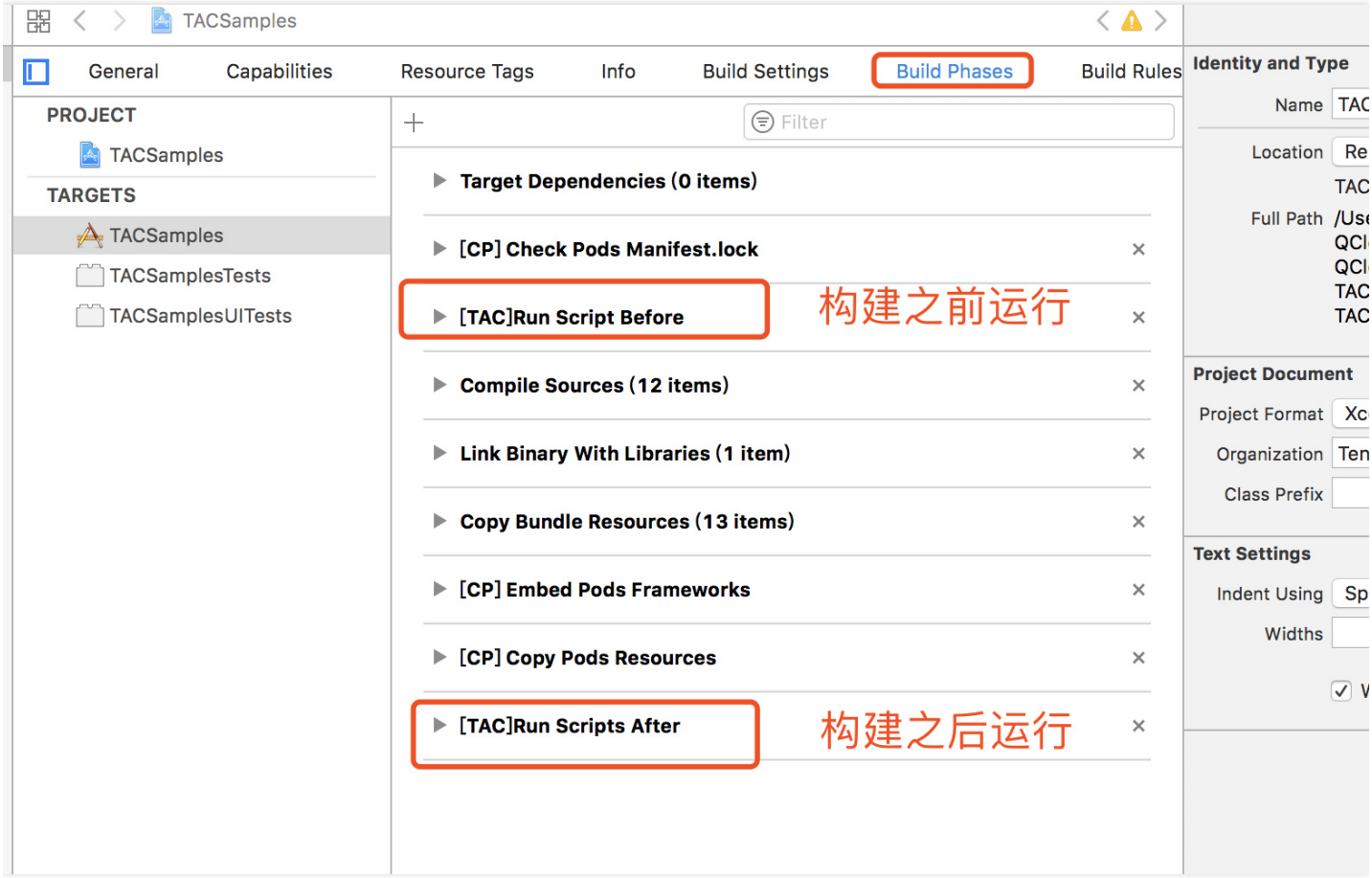
配置程序需要脚本

如果您在其他模块中完成了此步骤，请不要重复执行。

为了简化 SDK 的接入流程，我们使用 shell 脚本，帮助您自动化的去执行一些繁琐的操作，比如 crash 自动上报，在 Info.plist 里面注册各种第三方 SDK 的回调 scheme。因而，需要您添加以下脚本来使用我们自动化的加入流程。

脚本主要包括两个：

1. 在构建之前运行的脚本，该类型的脚本会修改一些程序的配置信息，比如在 Info.plist 里面增加 qqwallet 的 scheme 回调。
2. 在构建之后运行的脚本，该类型的脚本在执行结束后做一些动作，比如 Crash 符号表上报。



自动添加所有程序需要脚本

自动添加脚本目前仅支持通过 Cocoapods 方式进行集成的用户。如果使用 Cocoapods 集成的话，在 Podfile 的最后一行后面**新起一行**，并且将以下代码粘贴进去以后，运行 `pod install` 即可，就完成了配置程序需要脚本这一步。

```
pre_install do |installer|
puts "[TAC]-Running post installer"
xcodeproj_file_name = "placeholder"
Dir.foreach("./") do |file|
if file.include?("xcodeproj")
xcodeproj_file_name = file
end
end
puts "[TAC]-project file is #{xcodeproj_file_name}"
project = Xcodeproj::Project.open(xcodeproj_file_name)
project.targets.each do |target|
shell_script_after_build_phase_name = "[TAC] Run After Script"
shell_script_before_build_phase_name = "[TAC] Run Before Script"
puts "[TAC]-target.product_type is #{target.product_type}"
if target.product_type.include?("application")
should_insert_after_build_phases = 0
should_insert_before_build_phases=0
after_build_phase = nil
before_build_phase = nil
target.shell_script_build_phases.each do |bp|
if !bp.name.nil? and bp.name.include?(shell_script_after_build_phase_name)
should_insert_after_build_phases = 1
after_build_phase = bp
end
if !bp.name.nil? and bp.name.include?(shell_script_before_build_phase_name)
should_insert_before_build_phases = 1
before_build_phase = bp
end
end


if should_insert_after_build_phases == 1
puts "[TAC]-Build phases with the same name--#{shell_script_after_build_phase_name} has already ex
isted"
else
after_build_phase = target.new_shell_script_build_phase
puts "[TAC]-installing run afger build phases-- #{after_build_phase}"

end
after_build_phase.name = shell_script_after_build_phase_name
after_build_phase.shell_script = "
if [ -f \"${SRCROOT}/Pods/TACCore/Scripts/tac.run.all.after.sh\" ]; then
bash \"${SRCROOT}/Pods/TACCore/Scripts/tac.run.all.after.sh\"
fi
```

```
"
after_build_phase.shell_path = '/bin/sh'
if should_insert_before_build_phases == 1
puts "[TAC]-Build phases with the same name--#{shell_script_before_build_phase_name} has already existed"
else
before_build_phase = target.new_shell_script_build_phase
target.build_phases.insert(0,target.build_phases.pop)
puts "[TAC]-installing run before build phases-- #{before_build_phase}"

end
before_build_phase.name = shell_script_before_build_phase_name
before_build_phase.shell_script = "
if [ -f \"${SRCROOT}/Pods/TACCore/Scripts/tac.run.all.before.sh\" ]; then
bash \"${SRCROOT}/Pods/TACCore/Scripts/tac.run.all.before.sh\"
fi
"
before_build_phase.shell_path = '/bin/sh'
end
end
puts "[TAC]-Saving projects"
project.save()
end
```

注：运行 `pod install` 以后，可以按照上面的图片打开项目里的 Build Phases 确认是否有 [TAC] 开头，与图上类似的 Build phases 。如果没有的话，可再次运行 `pod install` 后检查即可。

手动添加程序需要脚本

请按照以下步骤来添加脚本：

添加构建之前运行的脚本

1. 在导航栏中打开您的工程。
2. 打开 Tab `Build Phases` 。
3. 单击 `Add a new build phase` ，并选择 `New Run Script Phase` ，您可以将改脚本命名 TAC Run Before

> **注意：**
> 请确保该脚本在 `Build Phases` 中排序为第二。

4. 根据自己集成的模块和集成方式将代码粘贴入 `Type a script...` 文本框:。

需要黏贴的代码

#export TAC_SCRIPTS_BASE_PATH=[自定义执行脚本查找路径，我们会在该路径下寻找所有以“tac.run.**all**.before.**sh**”命名的脚本，并执行，如果您不需要自定义不用动这里]
${TAC_CORE_FRAMEWORK_PATH}/Scripts/tac.run.**all**.before.**sh**

其中 THIRD_FRAMEWORK_PATH 变量的取值根据您的安装方式而不同：

- 如果您使用 Cocoapods 来集成的则为 ${PODS_ROOT}/TACCore ，您需要黏贴的代码实例如下：

  ${SRCROOT}/Pods/TACCore/Scripts/tac.run.**all**.before.**sh**

- 如果您使用手工集成的方式则为 您存储 TACCore 库的地址 ，即您 TACCore framework 的引入路径，您需要黏贴的代码实例如下：

  export TAC_SCRIPTS_BASE_PATH=[自定义执行脚本查找路径，我们会在该路径下寻找所有以“tac.run.**all**.after.**sh**”命名的脚本，并执行，如果您不需要自定义不用动这里]
  [您存储 TACCore 库的地址]/TACCore.framework/Scripts/tac.run.**all**.before.**sh**

添加构建之后运行的脚本

1. 在导航栏中打开您的工程。
2. 打开 Tab Build Phases 。
3. 单击 Add a new build phase ，并选择 New Run Script Phase ，您可以将改脚本命名 TAC Run Before。

   注意：
   请确保该脚本在 Build Phases 中排序需要放到最后。

4. 根据自己集成的模块和集成方式将代码粘贴入 Type a script... 文本框:。

需要黏贴的代码

#export TAC_SCRIPTS_BASE_PATH=[自定义执行脚本查找路径，我们会在该路径下寻找所有以“tac.run.**all**.after.**sh**”命名的脚本，并执行，如果您不需要自定义不用动这里]
${TAC_CORE_FRAMEWORK_PATH}/Scripts/tac.run.**all**.after.**sh**

其中 THIRD_FRAMEWORK_PATH 变量的取值根据您的安装方式而不同：

- 如果您使用 Cocoapods 来集成的则为 ${PODS_ROOT}/TACCore ，您需要黏贴的代码实例如下：

```
${SRCROOT}/Pods/TACCore/Scripts/tac.run.all.after.sh
```

- 如果您使用手工集成的方式则为 [您存储 TACCore 库的地址] ，即您 TACCore framework 的引入路径，您需要黏贴的代码实例如下：

```
#export TAC_SCRIPTS_BASE_PATH=[自定义执行脚本查找路径，我们会在该路径下寻找所有以“tac.run.all.after.sh”命名的脚本，并执行，如果您不需要自定义不用动这里]
[您存储 TACCore 库的地址]/TACCore.framework/Scripts/tac.run.all.after.sh
```

第四步：初始化

集成好我们提供的 SDK 后，您需要在您自己的工程中添加初始化代码，从而让 MobileLine 服务在您的应用中进行自动配置。整个初始化的过程很简单。

步骤 1 在 UIApplicationDelegate 子类中导入移动开发平台（MobileLine）模块。

Objective-C 代码示例：

```
#import <TACCore/TACCore.h>
```

Swift 代码示例：

```
import TACCore
```

步骤 2 配置一个 TACApplication 共享实例，通常是在应用的 application:didFinishLaunchingWithOptions: 方法中配置。

使用默认配置

通常对于移动开发平台（MobileLine）的项目他的配置信息都是通过读取 tac_services_configuration.plist 文件来获取的。

Objective-C 代码示例：

```
[TACApplication configurate];
```

Swift 代码示例：

```
TACApplication.configurate();
```

启动服务

移动授权 服务无需启动，到此您已经成功接入了 MobileLine 移动授权服务。

后续步骤

了解 MobileLine：

- 查看 MoblieLine 应用示例

向您的应用添加 MobileLine 功能：

- 借助 Analytics 深入分析用户行为。
- 借助 messaging 向用户发送通知。
- 借助 crash 确定应用崩溃的时间和原因。
- 借助 storage 存储和访问用户生成的内容（如照片或视频）。
- 借助 authorization 来进行用户身份验证。
- 借助 payment 获取微信和手 Q 支付能力

iOS 编程指南

最近更新时间：2018-07-27 13:29:07

准备工作

如果这是您首次向应用添加 Authorization，请完成以下步骤：

在 MobileLine 控制台中关联您的应用

1. 安装 应用云 SDK。
2. 在 MobileLine 控制台中，将您的应用添加到您的 应用云 项目中。
3. 参考 Authorization 快速入门，配置并初始化 Authorization。

获取第三方用户登录信息

请求 QQ 登录

> 请确保您集成并安装了 TACAuthorizationQQ 模块。

首先，您需要引入头文件：

Objective-C 代码示例：

```
#import <TACAuthorizationQQ/TACAuthorizationQQ.h>
```

Swift 代码示例：

```
import TACAuthorizationQQ
```

然后调用 SDK 提供的 requestCredential 方法调起登录即可：

Objective-C 代码示例：

```
TACQQAuthProvider* provider = [[TACAuthoriztionService defaultService] qqCredentialProvider];
[provider requestCredential:^(TACQQCredential*credential, NSError *error) {
//登录完成回调
if (error) {
TACLogError(@"ERROR %@", error);
```

```
} else {
//登录成功
}
}];
```

Swift 代码示例：

```
TACCrashService.share().setUserValue("内容", forKey:"key")
let provider = TACAuthoriztionService.default().qqCredentialProvider
provider?.requestCredential({ (credential:TACQQCredential?, error:Error?)->Void in
if error != nil{
print("ERROR",error as Any)
}else{
print("登录成功")
}
})
```

QQ 登录完成回调

登录完成回调以block的形式传入requestCredential:方法中，在请求登录后完成。

获取 QQ 登录用户信息

登录完成以后可以获取登录用户的信息，使用对应 Provider 的 requestUserInfoWithCredential 方法（可以在登录完成回调里去请求）：

Objective-C 代码示例：

```
TACQQAuthProvider* provider = [[TACAuthoriztionService defaultService] qqCredentialProvider];
[provider requestCredential:^(TACQQCredential*credential, NSError *error) {
//登录完成回调
if (error) {
TACLogError(@"ERROR %@", error);
} else {
[provider requestUserInfoWithCredential:credential completation:^(TACOpenUserInfo *userInfo, NSEr
ror *error) {
//获取到用户信息
}];
TACLogDebug(@"Credential %@", credential);
}
}];
```

Swift 代码示例：

```
TACCrashService.share().setUserValue("内容", forKey:"key")
let provider = TACAuthoriztionService.default().qqCredentialProvider
provider?.requestCredential({ (credential:TACQQCredential?, error:Error?)->Void in
if error != nil{
print("ERROR",error as Any)
}else{
provider?.requestUserInfo(with: credential, completation: { (userInfo:TACOpenUserInfo?, error:Error?)->Void in
print("credential",credential as Any)
})
}
})
```

然后您就可以使用 `TACQQAuthProvider` 来获取 QQ 的登录信息。

请求微信登录

> 请确保您集成并安装了 TACAuthorizationWechat 模块。
> 这里需要用到微信应用的 appKey 信息。**将 appkey 打包在程序中是个非常危险的操作，只建议您在测试阶段使用。**如果上线后，请部署自己的后端服务，通过后端服务的方式来获取相关的权限信息。
> 首先，您需要引入头文件：

Objective-C 代码示例：

```
#import <TACAuthorizationWechat/TACAuthorizationWechat.h>
```

Swift 代码示例：

```
import TACAuthorizationWechat
```

然后您就可以使用 `TACWechatAuthProvider` 来获取 微信 的登录信息

Objective-C 代码示例

```
TACWechatAuthProvider* provider = [[TACAuthoriztionService defaultService] wechatCredentialProvider];
[provider requestCredential:^(TACCredential *credential, NSError *error) {
//登录完成回调
if (error) {
TACLogError(@"ERROR %@", error);
} else {
//登录成功
```

```
}
}];
```

Swift 代码示例：

```
let provider = TACAuthoriztionService.default().wechatCredentialProvider
provider?.requestCredential({ (credential:TACCredential?, error:Error?)->Void in
if error != nil{
print("ERROR",error as Any)
}else{
//登录成功
}
})
```

微信登录完成回调

登录完成回调以 block 的形式传入requestCredential:方法中，在请求登录后完成。

获取微信登录用户信息

登录完成以后可以获取登录用户的信息，使用对应 Provider 的 requestUserInfoWithCredential 方法（可以在登录完成回调里去请求）：
Objective-C 代码示例

```
TACWechatAuthProvider* provider = [[TACAuthoriztionService defaultService] wechatCredentialProvi
der];
[provider requestCredential:^(TACCredential *credential, NSError *error) {
if (error) {
TACLogError(@"ERROR %@", error);
} else {
[provider requestUserInfoWithCredential:credential completation:^(TACOpenUserInfo *userInfo, NSEr
ror *error) {
//获取登录用户的信息
}];
TACLogDebug(@"Credential %@", credential);
}
}];
```

Swift 代码示例：

```
let provider = TACAuthoriztionService.default().wechatCredentialProvider
provider?.requestCredential({ (credential:TACCredential?, error:Error?)->Void in
if error != nil{
print("ERROR",error as Any)
}else{
```

```
provider?.requestUserInfo(with: credential as! TACWechatCredential, completation: { (userInfo:TACOpenUserInfo?, error:Error?)->Void in
//获取登录用户的信息
})
}
})
```

常见问题

最近更新时间：2018-03-28 17:10:06

通过 Authorization 是否可以分享应用数据到 QQ 空间或者朋友圈。

不支持，Authorization 只能获取 QQ 和微信用户的基本信息来简化用户登录您的应用。