# 移动开发平台

# 开始使用

# 产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2019 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

# 文档目录



版权所有：腾讯云计算（北京）有限责任公司

# 开始使用
# 创建应用

最近更新时间：2018-04-04 18:27:09

在使用我们的服务前，您必须先在 MobileLine 控制台 上创建项目，每个项目下可以包含多个应用，如 Android 或者 iOS 应用，当然，您也可以在同一个项目下创建多个 Android 或者 iOS 应用。

## 创建项目

登录 MobileLine 控制台 ，单击【创建第一个项目】按钮来创建一个新的项目，例如下图这里创建了一个名为 MyGreatProject 的项目：


> **注意：**
> 如果您之前已经创建过项目了，您可以使用该现有的项目，也可以重新创建一个新的项目。

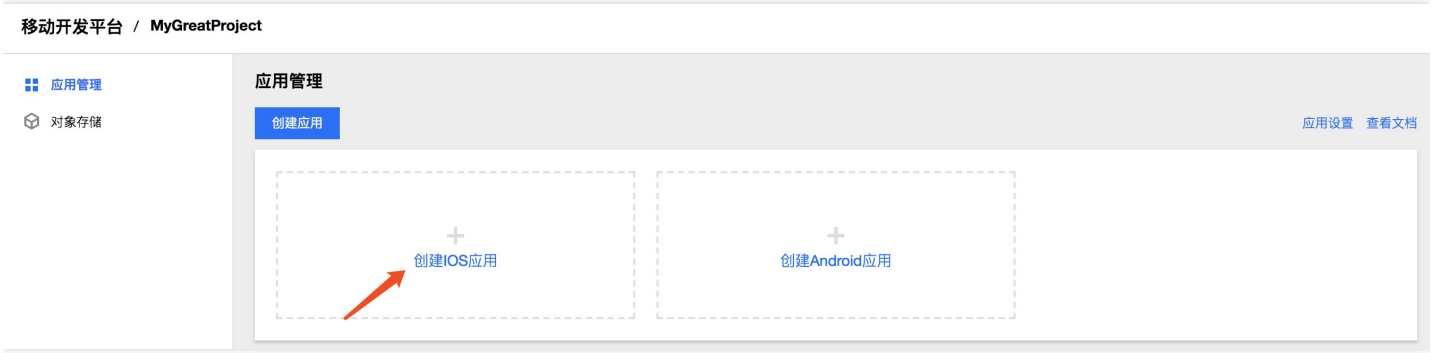
## 创建应用

创建好项目后，我们可以在 MyGreatProject 项目下创建应用，您可以选择创建 Android 和 iOS 应用。

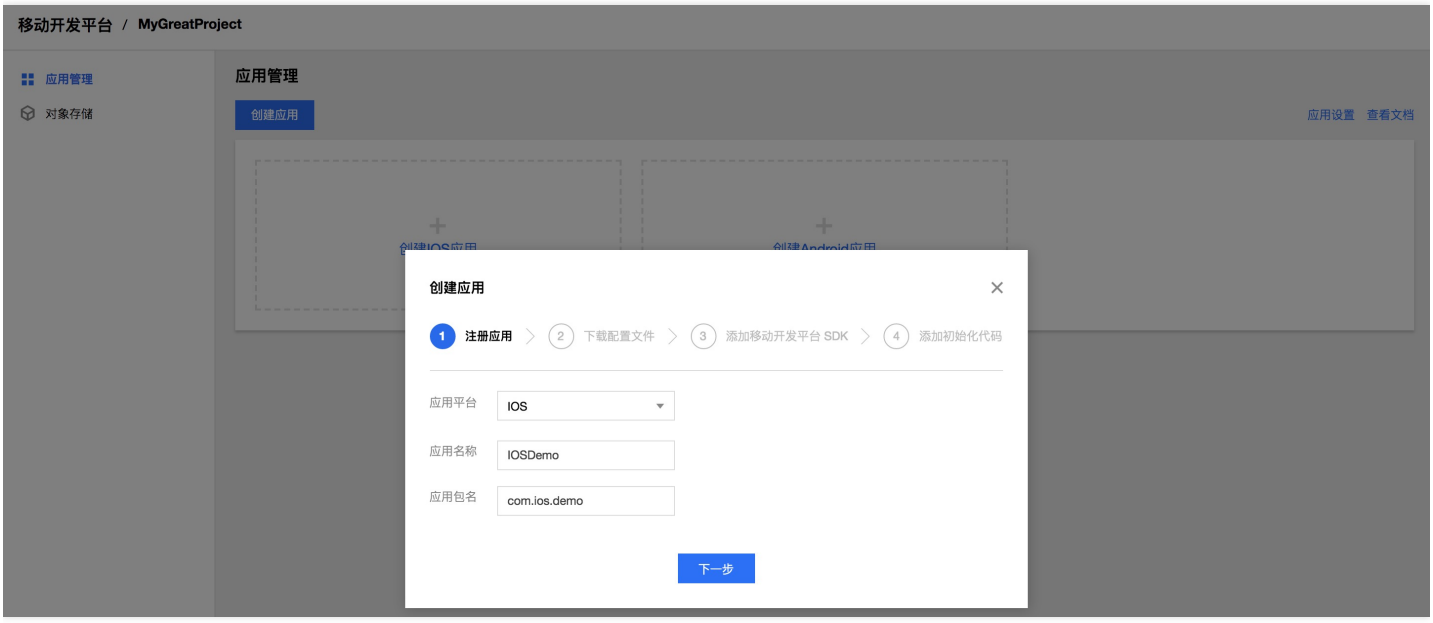
版权所有：腾讯云计算（北京）有限责任公司

## 创建 iOS 应用

单击【创建 iOS 应用】按钮来创建 iOS 应用，如图所示：




填写好 **应用名称** 和 **应用包名** 后，单击【下一步】。到此，您便已创建好了一个 MobileLine iOS 应用，此时，您可以单击关闭向导。

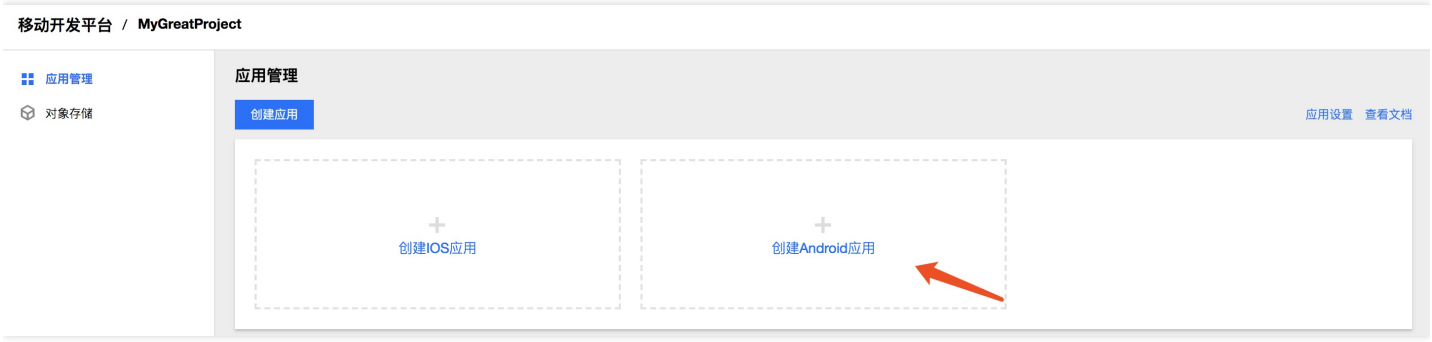



> **注意：**
> 对于 iOS 应用，应用包名即为 bundleId。

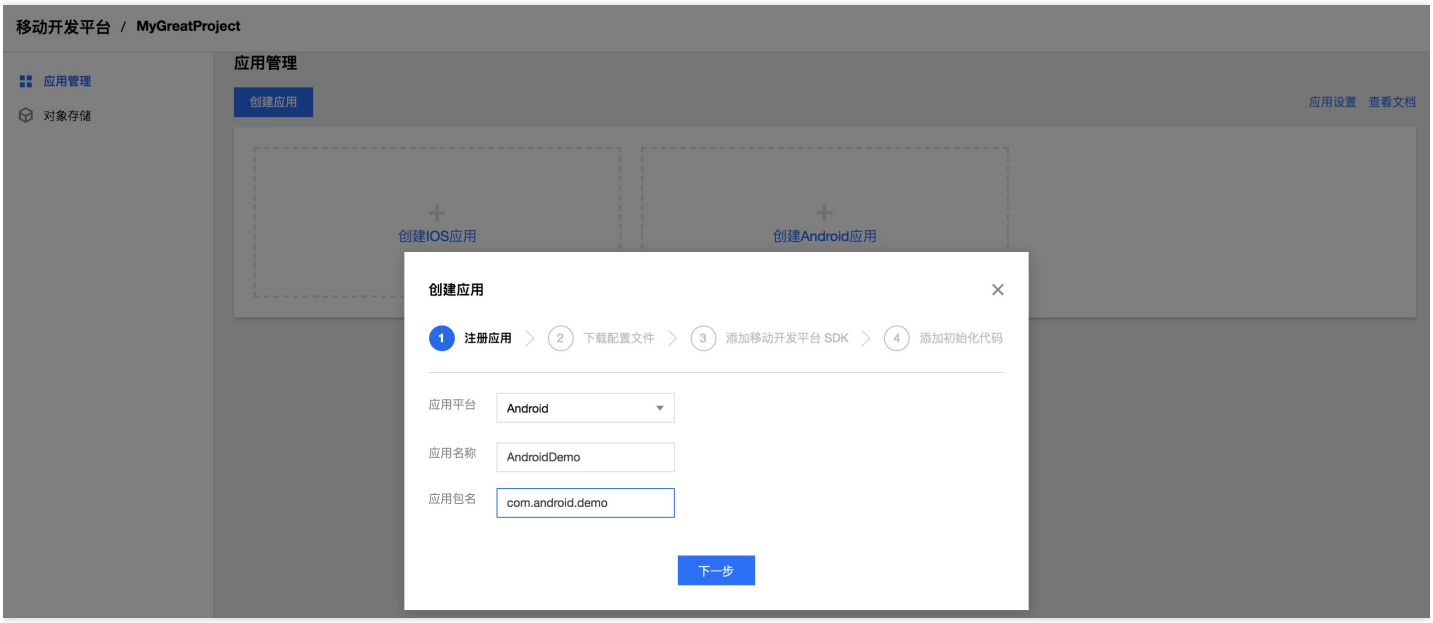
## 创建 Android 应用

您可以单击【创建 Android 应用】按钮来创建 Android 应用，同样您需要填写应用名称和应用包名：

版权所有：腾讯云计算（北京）有限责任公司

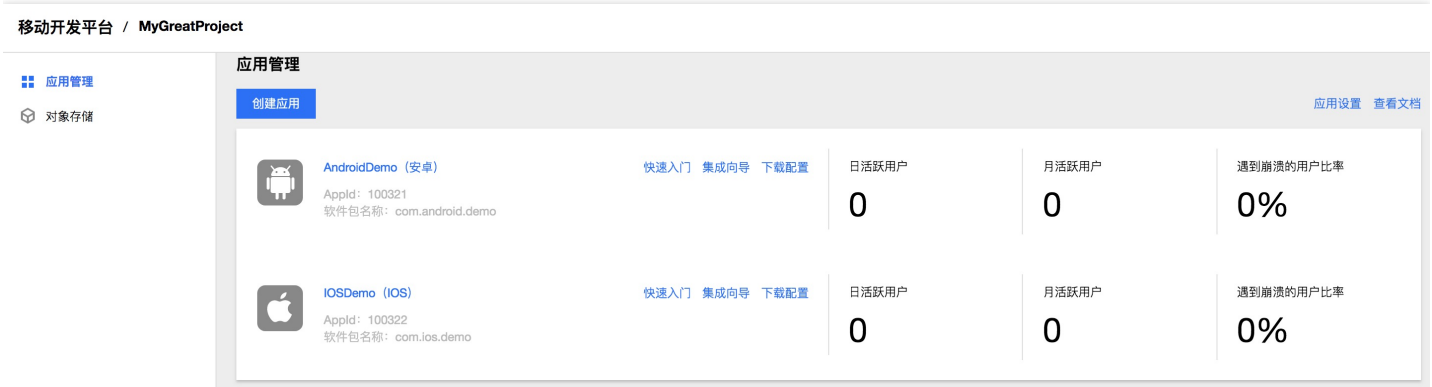



填写好应用信息后，单击【下一步】，您便创建好了一个 MobileLine Android 应用，此时，您可以关闭向导。




## 查看应用列表

创建好 Android/iOS 应用后，您可以在控制台上查看 MyGreatProject 项目的应用列表如下：




版权所有：腾讯云计算（北京）有限责任公司

# Android 文档
# Android 快速入门

最近更新时间：2018-08-16 16:30:57

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。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

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。

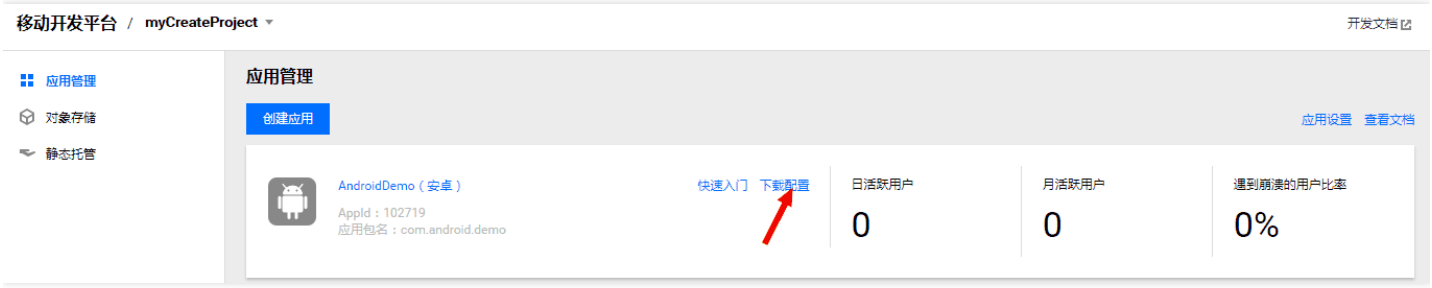
## 第一步：创建项目和应用

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。

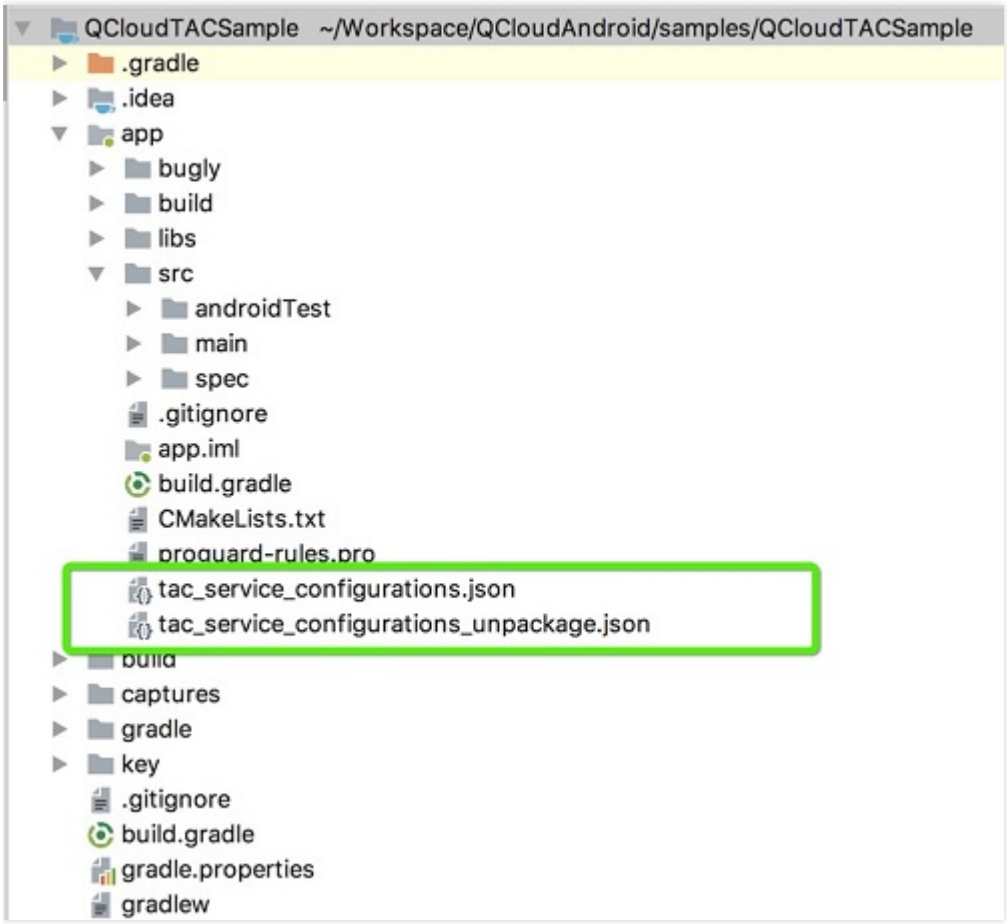
> 如果您已经在 MobileLine 控制台上创建过了项目和应用，请跳过此步。

## 第二步：添加配置文件

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

下表展示了 MobileLine 各种服务所对应的库

| 功能 | 服务名称 | Gradle 依赖项 |
| --- | --- | --- |
| 腾讯移动分析（MTA） | analytics | com.tencent.tac:tac-core |
| 腾讯移动推送（信鸽） | messaging | com.tencent.tac:tac-messaging |
| 腾讯崩溃服务（bugly） | crash | com.tencent.tac:tac-crash |
| 腾讯计费（米大师） | payment | com.tencent.tac:tac-payment |

版权所有：腾讯云计算（北京）有限责任公司

| 功能 | 服务名称 | Gradle 依赖项 |
| --- | --- | --- |
| 移动存储（Storage） | storage | com.tencent.tac:tac-storage |
| 登录与授权（Authorization） | authorization | com.tencent.tac:tac-authorization |

如果您想集成我们的各种服务，那么您只需要在您应用级 build.gradle 文件（通常是 app/build.gradle）中添加对应的服务依赖即可：

例如，您只想集成 `analytics` 服务，

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

如果您想集成 `messaging` 服务：

```
dependencies {
// 增加这两行，其中 core 是所有其他模块的基础
compile 'com.tencent.tac:tac-core:1.3.+'
compile 'com.tencent.tac:tac-messaging:1.3.+'
}
```

如果您想同时集成 `messaging` 和 `crash` 服务：

```
dependencies {
// 增加这三行，其中 core 是所有其他模块的基础
compile 'com.tencent.tac:tac-core:1.3.+'
compile 'com.tencent.tac:tac-messaging:1.3.+'
compile 'com.tencent.tac:tac-crash:1.3.+'
}
```

此外，我们提供了 gradle 插件，帮您进一步降低集成成本，

```
apply plugin: 'com.tencent.tac.services'
```

> **注意：**
> 使用 Payment 计费等服务时还需要额外的配置，详情请参见各自服务的快速入门。

# 第四步：参考各个服务的快速入门

一些子服务可能还有其他的集成步骤，请参考各个服务的快速入门文档。

| 功能 | 服务名称 | 入门指南 |
| --- | --- | --- |
| 腾讯移动分析（MTA） | analytics | Analytics 快速入门 |
| 腾讯移动推送（信鸽） | messaging | Messaging 快速入门 |
| 腾讯崩溃服务（bugly） | crash | Crash 快速入门 |
| 腾讯计费（米大师） | payment | Payment 快速入门 |
| 移动存储（Storage） | storage | Storage 快速入门 |
| 微信 QQ 登录（Authorization） | authorization | Authorization 快速入门 |

# 后续步骤

### 了解 MobileLine

- 查看 MoblieLine Android 应用示例

### 向应用中添加 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 获取微信和 QQ 登录能力。
- 借助 payment 获取微信和 QQ 支付能力。

版权所有：腾讯云计算（北京）有限责任公司

# 设备与账号标识

最近更新时间：2019-04-17 16:21:09

## 获取设备标识

MobileLine 会使用一个全局的 device id，它在所有服务中都唯一地标识了一个设备。如果您需要在代码中获取 device id，可以使用：

```
String deviceId = TACApplication.getDeviceId();
```

## 绑定用户标识

MobileLine 的多项服务（如 Analytics、Messaging 等）都可以在上报信息时带上用户标识，这样通过后台查看数据的时候，可以定位到具体的用户。

### 绑定您账号系统的 user id

您可在用户模块登录之后，调用 bindUserId 方法，绑定用户标识：

```
String userId = "your user id";
TACApplication.bindUserId(userId);
```

### 绑定三方登录系统的 open id

如果通过微信或者 QQ 等三方登录模块，可调用 useOpenId 方法，绑定用户标识：

```
String openId = "your open id";
TACApplication.useOpenId(openId);
```

绑定用户标识后，我们会自动通知所有服务，后续上报信息时，自动带上用户的 ID。

版权所有：腾讯云计算（北京）有限责任公司

# 自定义配置

最近更新时间：2019-04-17 16:23:51

通常我们的服务都是按照默认配置自动启动的。但您也可以根据需要，修改服务配置。您可以参考每个子服务的接入文档，以查看有哪些具体的配置项。

## 在 Application 子类中添加代码

如果您自己的应用中已经有了 Application 的子类，请重载它的 attachBaseContext(Context) 方法，在里面添加配置代码，如果没有，请自创建一个 Application 的子类。如：

```
public class MyCustomApp extends Application {
@Override
protected void attachBaseContext(Context base) {
super.attachBaseContext(base);
// 实例化一个新的配置
TACApplicationOptions applicationOptions = TACApplicationOptions.newDefaultOptions(this);

// 根据需要修改配置，这里是设置行为统计数据上报的策略
TACAnalyticsOptions analyticsOptions = applicationOptions.sub("analytics");
analyticsOptions.strategy(TACAnalyticsStrategy.INSTANT); // 立即发送

// 修改其他配置
...

// 让自定义设置生效
TACApplication.configureWithOptions(this, applicationOptions);
}
}
```

**注意：**
configureWithOptions 只能被调用一次，如果您有多个服务的配置需要修改，请全部修改好再调用 configureWithOptions，让配置生效。

## 在 AndroidManifest.xml 文件中注册

在创建好 Application 的子类并添加好代码后，您需要在工程的 AndroidManifest.xml 文件中注册该 Application 类：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.tac">
<application
<!-- 这里替换成你自己的 Application 子类 -->
android:name="com.example.tac.MyCustomApp"
...>
</application>
</manifest>
```

**注意：**

如果您的 `Application` 子类已经在 `AndroidManifest.xml` 文件中注册，请不要重复注册。

# 查看调试日志

最近更新时间：2019-04-17 16:22:24

如果您想查看 SDK 的一些调试日志，可以通过以下两种方式打开日志的 debug 开关。

### 命令行打开 debug 模式

在命令行执行：

```
adb shell setprop log.tag.tacApp DEBUG
```

### 代码打开日志输出开关

在代码中调用：

```
TACApplicationOptions applicationOptions = TACApplicationOptions.newDefaultOptions(this);
applicationOptions.setLoggingEnable(true);

...

TACApplication.configureWithOptions(this, applicationOptions);
```

# iOS 文档
# iOS 快速入门

最近更新时间：2019-04-03 17:43:25

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。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

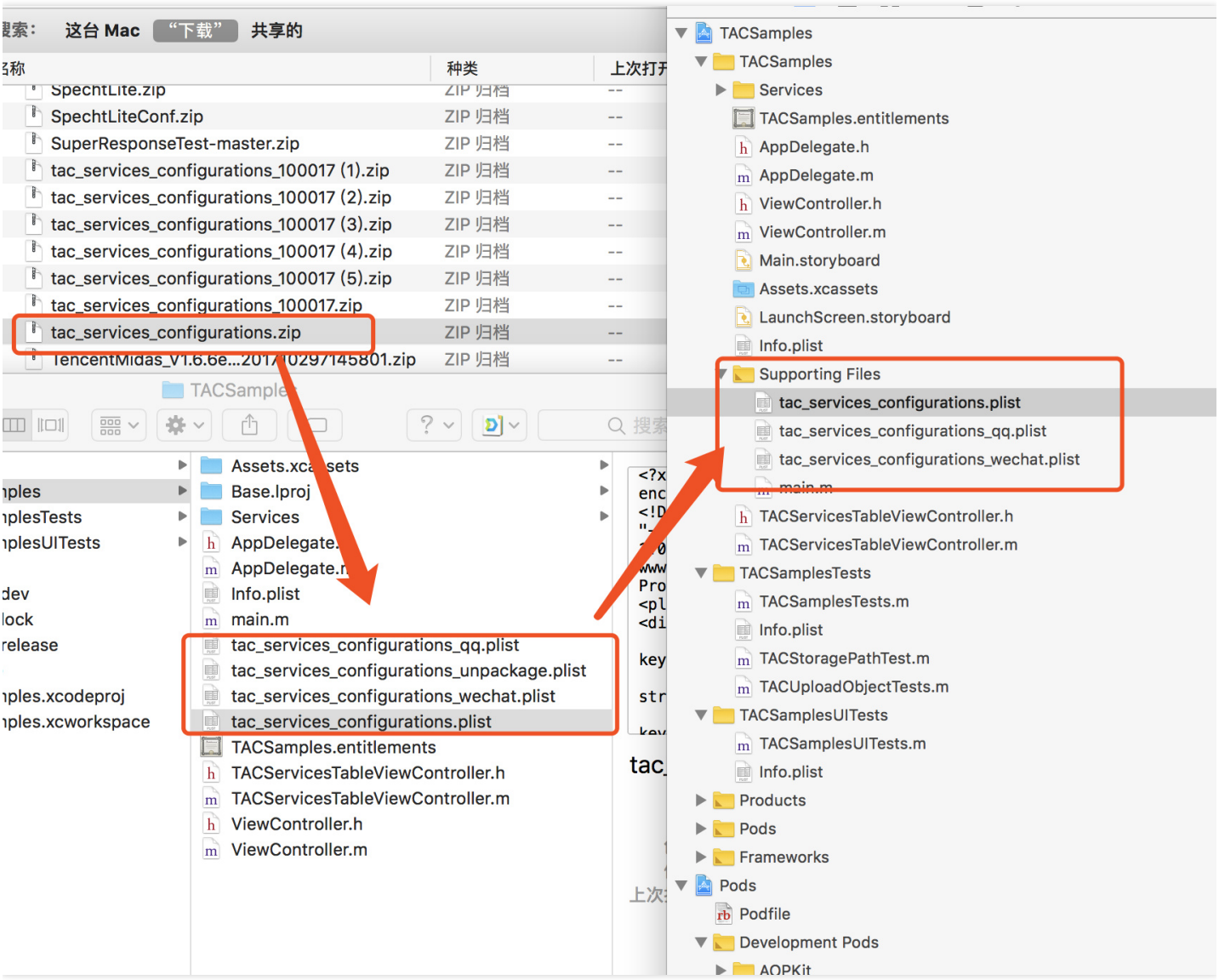
在使用我们的服务前，您必须先在 MobileLine 控制台上 创建项目和应用。

> ⚠ **注意：**
>
> 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

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(**切记不要将该文件添加进工程中**)。 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

**⚠ 注意：**

无论您使用哪种代码集成方式，都请**配置程序需要脚本**。如果您选择手工集成，则需要先从：下载地址,下载移动开发平台（MobileLine）所需要的 SDK 集合文件。并仔细阅读文件中的 Readme.md 文档。

每一个 MobileLine 服务都是一个单独的 SDK，其中 `TACCore` 是其他所有模块的基础模块，因此您必须至少将 `analytics` 模块集成到您的 App 中，下表展示了 MobileLine 各种服务所对应的库。

以下库分别对应各种移动开发平台（MobileLine）的功能。

| 功能 | cocoapods | 服务名称 |
| --- | --- | --- |
| 腾讯移动分析（MTA） | TACCore | analytics |
| 腾讯移动推送（信鸽） | TACMessaging | messaging |
| 腾讯崩溃服务（bugly） | TACCrash | crash |
| 移动存储（Storage） | TACStorage | storage |
| 授权（Authorization） | TACAuthorization | social |
| 腾讯计费（米大师） | TACPayment | payment |

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"
```

如果您想集成我们的各种服务，那么您只需要在 Podfile 中添加对应的服务依赖即可：

例如，您只想集成 analytics 服务

```
pod 'TACCore'
```

如果您想集成 `messaging` 服务：

```
pod 'TACMessaging'
```

如果您想同时集成 `messaging` 和 `crash` 服务：

```
pod 'TACMessaging'
pod 'TACCrash'
```

版权所有：腾讯云计算（北京）有限责任公司


**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
```

> ⚠ **注意：**
>
> 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**

**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();
```

**通过编程的方式自定义某些参数**

您可能也有需求在程序运行时，去改变一些特定的参数来改变程序的行为。为了支持您的这种需求，我们增加了修改程序配置的接口，您可以仿照如下形式来修改移动开发平台（MobileLine）的配置。

Objective-C 代码示例：

```
TACApplicationOptions* options = [TACApplicationOptions defaultApplicationOptions];
// 自定义配置
// opions.xxx= xxx
//
[TACApplication configurateWithOptions:options];
```

Swift 代码示例：

```
let options = TACApplicationOptions.default()
// 自定义配置
// opions.xxx= xxx
TACApplication.configurate(with: options);
```

### 启动服务

MobileLine iOS SDK 会自动帮您启动对应的服务，比如分析（Analytics）服务。有些服务比如Storage是按需启动，当您使用的时候，调用其接口即可。

| 功能 | 启动方式 | 服务名称 |
| --- | --- | --- |
| 腾讯移动分析（MTA） | 默认启动 | analytics |
| 腾讯移动推送（信鸽） | 默认启动 | messaging |
| 腾讯崩溃服务（bugly） | 默认启动 | crash |
| 移动存储（Storage） | 按需使用 | storage |
| 授权（Authorization） | 按需使用 | social |
| 腾讯计费（米大师） | 按需使用 | payment |

## 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

版权所有：腾讯云计算（北京）有限责任公司

# iOS 编程手册

最近更新时间：2018-03-16 15:35:12

## 绑定用户唯一标记

为了方便追踪用户信息，您可以绑定用户的唯一标记：

```
[[TACApplication defaultApplication] bindUserIdentifier:@"uuid-11"];
```

当您调用该功能后，我们会周知所有需要用户唯一 ID 的模块，绑定该 ID。

# iOS 统一配置服务介绍

最近更新时间：2018-06-21 16:35:45

通常我们在集成 SDK 的时候，需要对工程和 SDK 进行一些配置和初始化的操作。为了简化这两部分的工作，更加方便的集成配置 SDK，移动开发平台（MobileLine）实现了统一配置服务。

移动开发平台（MobileLine）支持通过脚本来配置工程需要的参数，同时也支持通过文件和程序配置 SDK。

一般情况下移动开发平台（MobileLine）的配置信息分成两类：

- 通过文件（满足正则条件：tac_services_configurations*.plist）来配置。
- 通过通过程序来配置。

### 通过引入脚本配置工程需要的参数

比如 QQ SDK 在集成的时候，需要您去设置回调的 scheme 之类的，这个我们通过脚本实现了。

### 通过文件进行配置 SDK

移动开发平台（MobileLine） SDK 统一配置服务，会读取所有位于您的 MainBundle 里面符合正则条件：`tac_services_configurations*.plist` ，并合并其里面的内容，然后传递给配置服务，进行对应的服务配置。

例如以下文件名都是合法的配置文件：

```
tac_services_configurations.plist
tac_services_configurations_custom.plist
tac_services_configurations_payment.plist
....
```

在读取配置文件的时候，会按照字典顺序依次读取（ z > a ），并对所有的配置进行合并。如果在多个配置文件中存在多个重复的 Key，则会以顺序靠后的配置文件中的内容为准。

例如文件 `tac_services_configurations.plist` 与 `tac_services_configurations_custom.plist` 都存在以 `payment` 为 Key 的内容，则会用 `tac_services_configurations_custom.plist` 中的覆盖掉 `tac_services_configurations.plist` 中的。

通过这种多文件配置的方式，您可以对移动开发平台（MobileLine）的配置文件进行拆分和自定义操作。例如在不改变从官网下载的配置文件的基础上，实现自定义的参数配置。

### 通过程序进行配置 SDK

在您引入了需要配置的模块的头文件的时候，您可以在 `TACApplicationOptions` 的实例中看到对应模块的 `options` 配置，只需要修改对应的参数即可。

# 服务配置修改指南

最近更新时间：2018-06-07 15:46:48

您可以通过控制台修改配置文件来定制服务，无需添加任何代码。请注意修改之后需要重新下载配置文件，集成到工程中。请参考下面列出了各个服务的配置项和含义。您可以通过 iOS 统一配置服务介绍，获取关于统一配置服务的使用说明。

如果您希望通过配置文件来改变移动研发平台相关服务的行为。您可以创建一个 plist 文件，名为： `tac_services_configurations_custom.plist` 。根据下属每个服务的配置内容来控制相关行为。

一个 plist 文件实例如下：

```
<?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>analytics</key> <!-- 服务关键字 -->
<dict> <!-- 服务内容通过字典的方式进行描述 -->
<key>appKey</key> <!-- 控制某个行为的配置项 -->
<string>xxxx</string> <!-- 控制某个行为的配置内容 -->
</dict>
</dict>
</plist>
```

## 腾讯移动分析（MTA）

服务关键字：analytics

| 配置项 | 配置说明 | 备注 |
| --- | --- | --- |
| appKey | appKey | 不可修改 。 |
| strategy | 上报策略，默认是批量上报 | 选项：实时，批量，App启动时，周期，在代码中主动发送，仅在 Wi-Fi 网络下发送并且发送失败以及非 Wi-Fi 网络情况下不缓存数据，不缓存数据，批量上报+间隔上报组合。 |

版权所有：腾讯云计算（北京）有限责任公司

| 配置项 | 配置说明 | 备注 |
| --- | --- | --- |
| minBatchReportCount | 最小批量发送消息个数，默认是 30 条 | 只在批量上报策略时有效。 |
| sendPeriodMillis | 发送时间间隔，单位毫秒，默认一天 | 只在周期上报策略时有效。 |
| smartReporting | 智能发送策略，wifi下立即发送，默认打开。 | |
| sessionTimeoutMillis | 会话超时时长，默认 30 秒，单位是 ms。 | |
| autoTrackPageEvents | 是否开启自动统计页面访问历史，默认开启。 | |
| multiProcess | 是否开启多进程的支持，默认开启。 | |
| channelUrl | 设置中间页的地址，例如投放的地址(如果是通过接入 js sdk 自定义的话，需要设置该属性)。 | |
| ifNeedCookie | 是否启用 Cookie。 | 如果和 App 的 UI 结构冲突，出现黑屏问题时，请设置为 NO。 |

# 腾讯移动推送（信鸽）

服务关键字：messaging

| 配置项 | 配置说明 | 备注 |
| --- | --- | --- |
| appId | appId | 推送模块的服务标志。 |
| appKey | appKey | 推送模块的服务 Key。 |
| autoStart | 是否默认启动推送服务 | 默认为开启。 |

# 崩溃监控服务（bugly）

服务关键字：crash

| 配置项 | 配置说明 |
| --- | --- |
| appId | 当前崩溃检测服务对应的 APPID。 |
| excludeModuleFilters | 崩溃数据过滤器，如果崩溃堆栈的模块名包含过滤器中设置的关键字，则崩溃数据不会进行上报。 |
| enable | 该服务是否启动。 |
| channel | 设置自定义渠道标识。 |
| version | 自定义版本号。 |
| deviceIdentifier | 自定义设备唯一标识。 |
| blockMonitorEnable | 卡顿监控开关。 |
| blockMonitorTimeout | 卡顿监控判断间隔，单位为秒。 |
| applicationGroupIdentifier | 设置 App Groups Id。 |
| symbolicateInProcessEnable | 进程内还原开关。 |
| unexpectedTerminatingDetectionEnable | 非正常退出事件记录开关，默认关闭。 |
| viewControllerTrackingEnable | 页面信息记录开关。 |
| consolelogEnable | 控制台日志上报开关。 |

## 腾讯计费（米大师）

服务关键字：payment

| 配置项 | 配置说明 | 备注 |
| --- | --- | --- |
| offerId | 腾讯计费服务的 appId | 不可修改。 |

# 移动存储（COS）

服务关键字：storage

| 配置项 | 配置说明 | 备注 |
| --- | --- | --- |
| bucket | 存储桶名称 | 不可修改。 |
| region | 存储桶所在的地域 | 不可修改。 |

# QQ 登录

服务关键字：social 服务下面 qq

| 配置项 | 配置说明 |
| --- | --- |
| appId | QQ 互联中程序的唯一标志。 |
| permissions | QQ 互联进行授权的时候。 |

# 微信登录

服务关键字：social 服务下面 wechat

| 配置项 | 配置说明 |
| --- | --- |
| appId | 当前程序在微信开放平台中 AppID。 |

版权所有：腾讯云计算（北京）有限责任公司

# 打印程序日志

最近更新时间：2018-07-04 17:22:54

## 日志上报

移动研发平台套件可以为您提供基础的日志打印服务。其中日志功能是基于 QCloudLogger 扩展而来，您可以通过配置环境 QCloudLogger 相关配置来改变日志的行为。整个移动研发平台与 QCloudCore 共享同一日志框架。日志统一缓存在 Cache 目录下面。会按照 7 天一次的规则进行清理，也就说本地会保留 7 天的日志。

## 设置日志上报级别

您需要在 QCloudLogger 里面设置日志级别：

```
[QCloudLogger sharedLogger].logLevel = QCloudLogLevelError;
```

## 调试模式下输出 Debug 日志

默认情况下，SDK 内部的日志并不会直接输出到控制台中。在 Debug 等情况下需要查看日志的话，可以设置对应的环境变量开启。开启的具体方式为：在 Xcode 左上角选择单击当前的 target-Edit Scheme-在 Enviriments Variables 中填入 QCloudLogLevel 这个环境变量，如果需要输出所有 debug 信息，那么将值设置为 6。

版权所有：腾讯云计算（北京）有限责任公司