

移动开发平台

社会化分享

产品文档



腾讯云

【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

社会化分享

Android 文档

Android 快速入门

Android 手动集成

分享内容

自定义分享界面

监听分享结果

分享效果追踪

iOS 文档

iOS 快速入门

分享内容

自定义分享界面

监听分享结果

分享效果追踪

开发者手册

名词解释

社会化分享

Android 文档

Android 快速入门

最近更新时间：2018-08-16 16:29:29

准备工作

首先您需要一个 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）

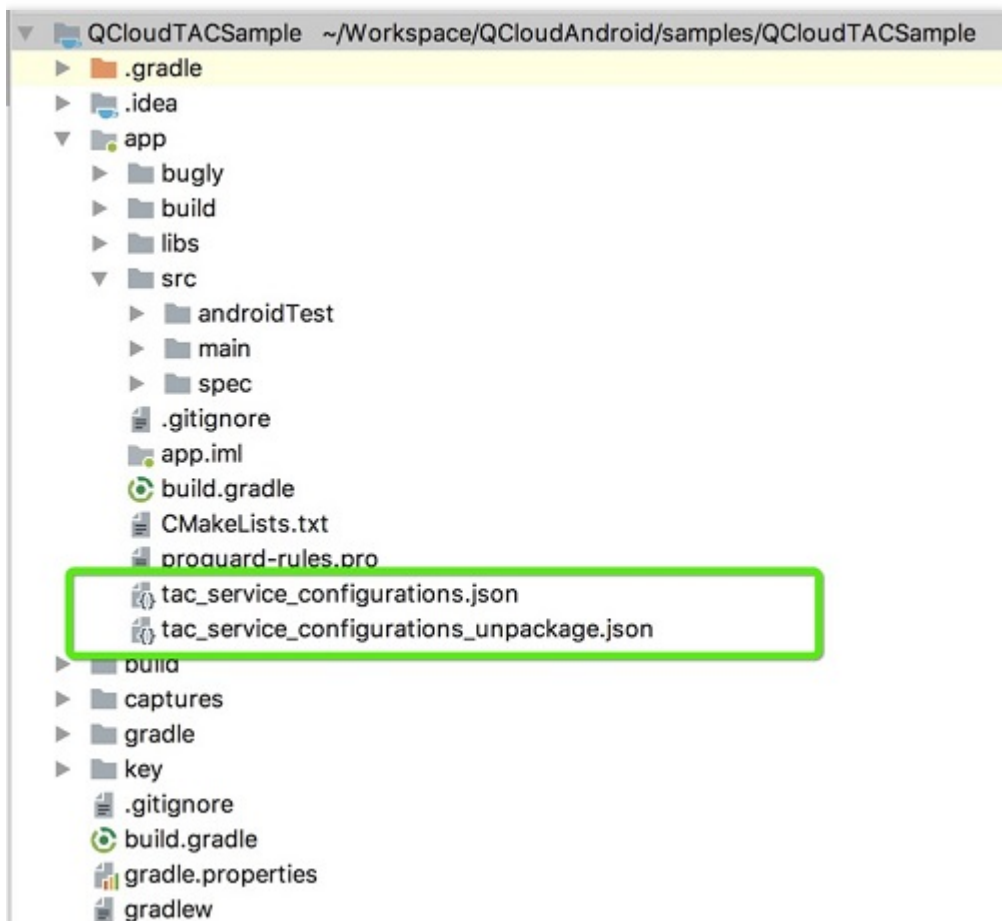
在使用我们的服务前，您必须先 MobileLine 控制台上 [创建项目和应用](#)。

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-social: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：配置第三方渠道

分享 SDK 需要配置社交渠道才能正常工作，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 分享服务。

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.** { *; }
```

Android 手动集成

最近更新时间：2018-07-10 11:20:37

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。

- 下载 [移动开发平台 \(MobileLine \) 核心框架资源包](#)，并解压。
- 下载 [移动开发平台 \(MobileLine \) Storage 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

分享 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 分享服务。

分享内容

最近更新时间：2018-06-11 17:22:49

分享 SDK 帮助您快速分享 app 内的内容，我们预设了分享的弹框界面，帮助您可以快速集成分享能力。下面的例子展示了如何调用 SDK，分享各种类型的内容，包括文字，图片，链接，文件等。

创建分享界面实例

首先，您需要创建一个分享界面实例，您可以在用户单击分享按钮时再创建：

```
TACShareDialog shareDialog = new TACShareDialog();
```

分享纯文字

```
shareDialog.share(activity, new PlainTextObject("this is a plain text"));
```

分享图片

```
Bitmap musicAlbum = ...;  
  
shareDialog.share(activity, new BitmapObject(musicAlbum));
```

分享网页链接

```
UrlObject urlObject = new UrlObject("http://carsonxu.com/tac/app_share_sample.html");  
urlObject.setMetadata(new ObjectMetadata.Builder()  
.title("WebPage title")  
.description("WebPage description")  
.build());  
  
shareDialog.share(activity, urlObject);
```

分享在线音乐链接

```
Bitmap musicAlbum = ...; // 音乐的封面缩略图
String musicAlbumThumbUrl = ...; // 音乐的封面缩略图url
String musicUrl = ...; // 音乐网页的地址
String musicDataUrl = ...; // 音乐的播放地址

JsonObject urlObject = new JsonObject(musicUrl);
urlObject.setMetadata(new ObjectMetadata.Builder().audio()
    .thumb(musicAlbum)
    .thumbUrl(musicAlbumThumbUrl)
    .dataUrl(musicDataUrl)
    .title("分享的标题")
    .description("分享的描述")
    .build());

shareDialog.share(activity, urlObject);
```

分享在线视频链接

```
Bitmap videoAlbum = ...; // 视频的封面缩略图
String videoThumbUrl = ...; // 视频的封面缩略图url
String videoUrl = ...; // 视频链接的地址

JsonObject urlObject = new JsonObject(videoUrl);
urlObject.setMetadata(new ObjectMetadata.Builder().video()
    .thumb(videoAlbum)
    .thumbUrl(videoThumbUrl)
    .title("分享的标题")
    .description("分享的描述")
    .build());

shareDialog.share(activity, urlObject);
```

分享文件

分享单个本地图片

```
File photoFile = ...;
FileObject fileObject = new FileObject(photoFile.getPath());
fileObject.setMetadata(new ObjectMetadata.Builder().image().build());

shareDialog.share(activity, fileObject);
```

分享单个本地视频

```
File videoFile = ...;
FileObject fileObject = new FileObject(videoFile.getPath());
fileObject.setMetadata(new ObjectMetadata.Builder().video().build());

shareDialog.share(activity, fileObject);
```

分享单个本地普通文件

```
File file = ...;

shareDialog.share(activity, new FileObject(file.getPath()));
```

分享多个本地图片

```
File file1 = ...;
File file2 = ...;

List<FileObject> files = new ArrayList<>(2);
files.add(new FileObject(file1.getPath()));
files.add(new FileObject(file2.getPath()));
ObjectMetadata metadata = new ObjectMetadata.Builder().image().build();

shareDialog.share(activity, files, metadata);
```

分享多个本地视频

```
File file1 = ...;
File file2 = ...;

List<FileObject> files = new ArrayList<>(2);
files.add(new FileObject(file1.getPath()));
files.add(new FileObject(file2.getPath()));
ObjectMetadata metadata = new ObjectMetadata.Builder().video().build();
```

```
shareDialog.share(activity, files, metadata);
```

分享多个本地普通文件

```
File file1 = ...;
File file2 = ...;

List<FileObject> files = new ArrayList<>(2);
files.add(new FileObject(file1.getPath()));
files.add(new FileObject(file2.getPath()));

shareDialog.share(activity, files);
```

分享微信小程序

```
String webpageUrl = "http://www.qq.com"; // 兼容低版本的网页链接
String userName = "gh_d43f693ca31f"; // 小程序原始id
String path = "/pages/media"; // 小程序页面路径
Bitmap thumb = ...; // 网页的缩略图

MiniProgramObject miniProgramObject = new MiniProgramObject(userName, path, webpageUrl);
miniProgramObject.setMetadata(new ObjectMetadata.Builder()
    .thumb(thumb)
    .title("分享的标题")
    .description("分享的描述")
    .build());

shareDialog.share(activity, miniProgramObject);
```

自定义分享界面

最近更新时间：2018-06-11 17:23:04

分享 SDK 预设了分享界面的样式，可以直接使用，但我们也支持您通过 SDK 定制界面。在定制界面时，您只需要关心渲染过程，而不需关心分享的流程。

默认分享界面样式

我们预设了一个轻快的分享界面，无需设置可以直接使用：



自定义分享界面

1. 实现 ShareUIRenderer 接口

您需要一个实现接口 `ShareUIRenderer` 的子类，请参考下面的例子：

```
public class CustomUIRenderer implements ShareUIRenderer {

    @Override
    public void showChooser(final Activity activity, final int[] channels, final
    TACShareDialog.Callback callback) {
        // 在这里弹出界面让用户选择渠道
        for (int c : channels) {
            ...
        }
        ...

        int userChooseChannel = ...; // 用户选择了想要分享的渠道

        // 回调 SDK 执行分享
        callback.onChannelSelected(activity, userChooseChannel);

    }
}
```

其中 `channels` 是当前分享内容支持的渠道列表，请使用渠道列表来渲染您的 UI 界面，避免显示不支持的渠道。您可以通过下面的类获取所有渠道的预设值：

```
public abstract class ShareChannel {

    /**
     * 微信朋友
     */
    public static final int WECHAT_SESSION = 1;

    /**
     * 微信朋友圈
     */
    public static final int WECHAT_TIMELINE = 2;

    /**
     * 微信收藏
     */
    public static final int WECHAT_FAVORITE = 3;

    /**
     * QQ
     */
    public static final int QQ = 4;

    /**
     * QQ空间
     */
    public static final int QZONE = 5;

    /**

```

```
* 微博  
*/  
public static final int WEIBO = 6;  
/**  
* 其他  
*/  
public static final int SYSTEM = 99;  
  
}
```

最后，在用户通过界面，选择了想要分享的渠道之后，请务必通过 `callback` 回调 SDK，执行真正的分享行为：

```
callback.onChannelSelected(activity, channel);
```

2. 实例化 TACShareDialog

在您实例化 `TACShareDialog` 时，传入一个 `CustomUIRenderer` 的实例：

```
TACShareDialog dialog = new TACShareDialog(new CustomUIRenderer());
```

这样，分享界面将完全由您自己渲染。

监听分享结果

最近更新时间：2018-06-11 17:22:54

分享 SDK 帮助您快速分享 app 内的内容，在分享行为之后，您可以查看下面的示例，监听分享结果。

分享结果

分享结果分为成功，失败和用户取消三种，下面的三种结果的定义：

```
public class ShareResult {  
    /**  
     * 分享成功  
     */  
    public static final int RESULT_SUCCESS = 1;  
    /**  
     * 分享失败  
     */  
    public static final int RESULT_ERROR = 2;  
    /**  
     * 用户取消了分享  
     */  
    public static final int RESULT_CANCEL = 3;  
}
```

您可以通过以下 API 拿到结果和错误信息：

```
// 获取结果  
int result = shareResult.getResult();  
  
// 获取错误码  
int errorCode = shareResult.getErrCode();  
  
// 获取错误信息  
String errorMessage = shareResult.getErrMsg();
```

获取 QQ 和 QQ 空间分享结果

您可以重载分享页所在 Activity 的 onActivityResult 方法，通过：

```
TACShareDialog shareDialog = ...;
```

```
ShareResult shareResult = shareDialog.getQQShareResult(requestCode, resultCode, data);
```

拿到分享结果。请参考下面的例子：

```
public class ShareActivity extends AppCompatActivity {  
    ...  
  
    @Override  
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
        ShareResult shareResult = shareDialog.getQQShareResult(requestCode, resultCode, data);  
        if (shareResult != null) {  
            Toast.makeText(this, "qq share result : " + shareResult.getResult(),  
                Toast.LENGTH_LONG).show();  
        }  
    }  
}
```

获取微信分享结果

首先您需要按照 [配置第三方渠道](#) 的指引，添加微信 SDK 的回调 Activity。

然后，在您接受回调的 `WXEntryActivity` 中重载 `onWeChatShareResult` 方法，可以获取微信分享结果的回调。请参考下面的例子：

```
package com.tencent.tac.sample.wxapi;  
  
import android.widget.Toast;  
  
import com.tencent.tac.social.WeChatBaseHandlerActivity;  
import com.tencent.tac.social.share.ShareResult;  
  
public class WXEntryActivity extends WeChatBaseHandlerActivity {  
  
    @Override  
    protected void onWeChatShareResult(ShareResult shareResult) {  
        super.onWeChatShareResult(shareResult);  
  
        Toast.makeText(this, "wechat share result : " + shareResult.getResult(),  
            Toast.LENGTH_LONG).show();  
    }  
}
```

```
}  
}
```

分享效果追踪

最近更新时间：2018-06-15 10:14:22

分享 SDK 帮助您快速分享 App 内的内容。不仅如此，我们还通过大数据和后台监控系统，为您统计和追踪分享效果，包括分享数、意向分享数、回流数、分享率等关键指标，帮助您快速决策，更好地运营您的应用。

为了更好地查看和追踪分享效果，您需要添加以下几步：

在 Android SDK 中统计落地页访问

落地页，通常是指您想要被分享的页面，例如活动页、推广页等，在 App 内对应的页面。我们需要统计落地页在 App 内的访问次数，以便更好的统计分享率。

请在页面访问开始的时候，通常是 Activity 的 onCreate 中调用：

```
TACShareDialog shareDialog = new TACShareDialog();  
  
shareDialog.onPageView();
```

在分享页的 H5 页面添加统计代码

分享出去的链接通常是一个 H5 页面，通过统计分享页面在社交平台被单击带来的访问次数，我们能更好地评估分享的效果。

请在您的分享页 H5 代码头部添加统计的 JS SDK：

```
<html>  
...  
  
<script>  
(function() {  
  var mta = document.createElement("script");  
  mta.src = "//pingjs.qq.com/h5/mobile_share_stats.js";  
  mta.setAttribute("name", "MTA_MS");  
  var s = document.getElementsByTagName("script")[0];  
  s.parentNode.insertBefore(mta, s);  
})();  
</script>
```

```
<body>
...
</body>
</html>
```

iOS 文档

iOS 快速入门

最近更新时间：2019-03-04 16:36:41

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分享服务。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，您首先需要有一个 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。

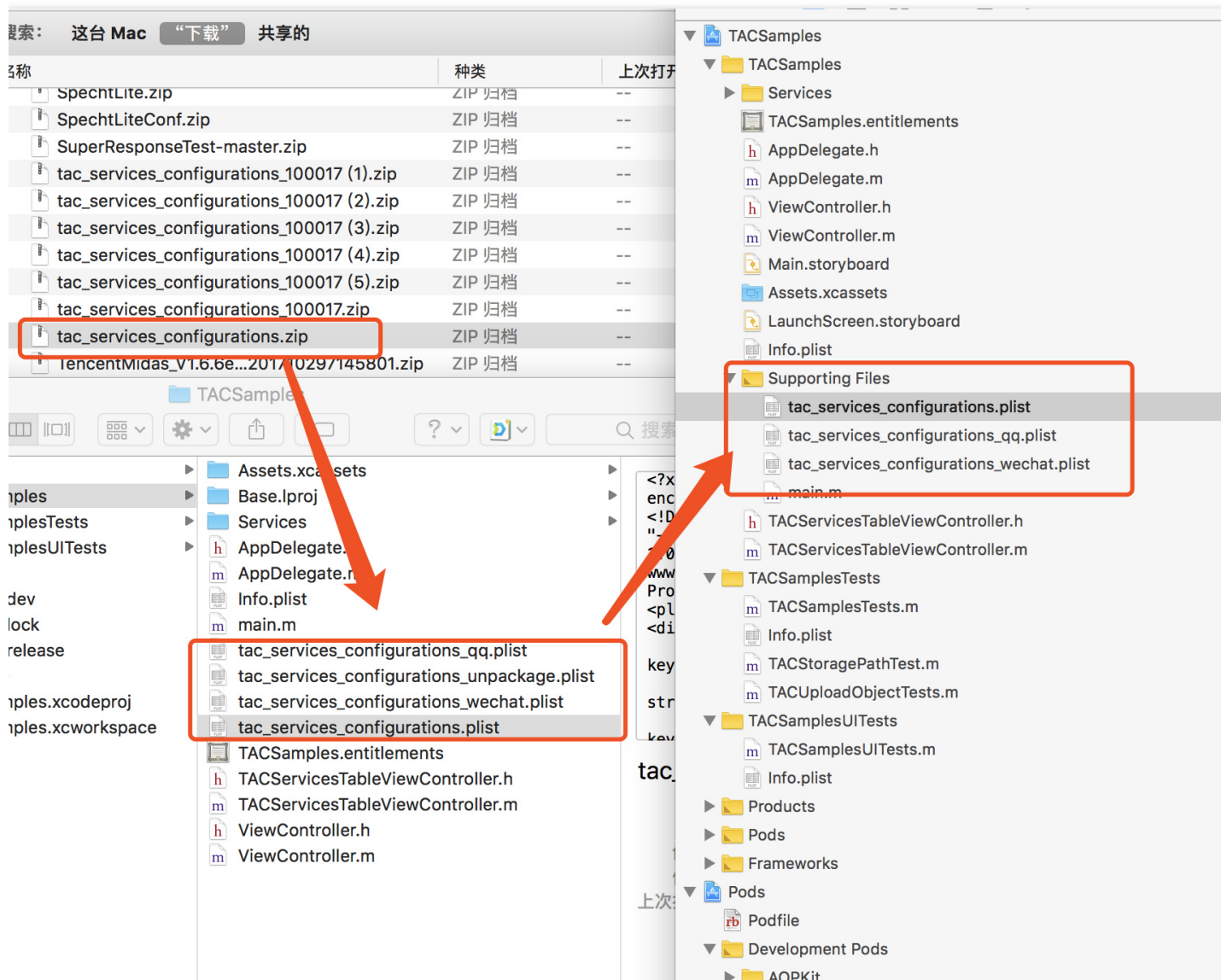
第二步：添加配置文件

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 'TACSocialShare'
```

TACSocialShare 模块依赖QQ和微信的第三方渠道，这要求您拥有 QQ支付和微信支付的

TACSocialShare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/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>services</key>
<dict>
<key>social</key>
<dict>
<key>qq</key>
<dict>
<key>appld</key>
<string>请填写您的 Appld</string>
<key>appKey</key>
<string>请填写您的 APPKey</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>
```


TACSocialShare 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>appld</key>
<string>wx256642f480c15e3e</string>
</dict>
</dict>
</dict>
</dict>
</plist>
```

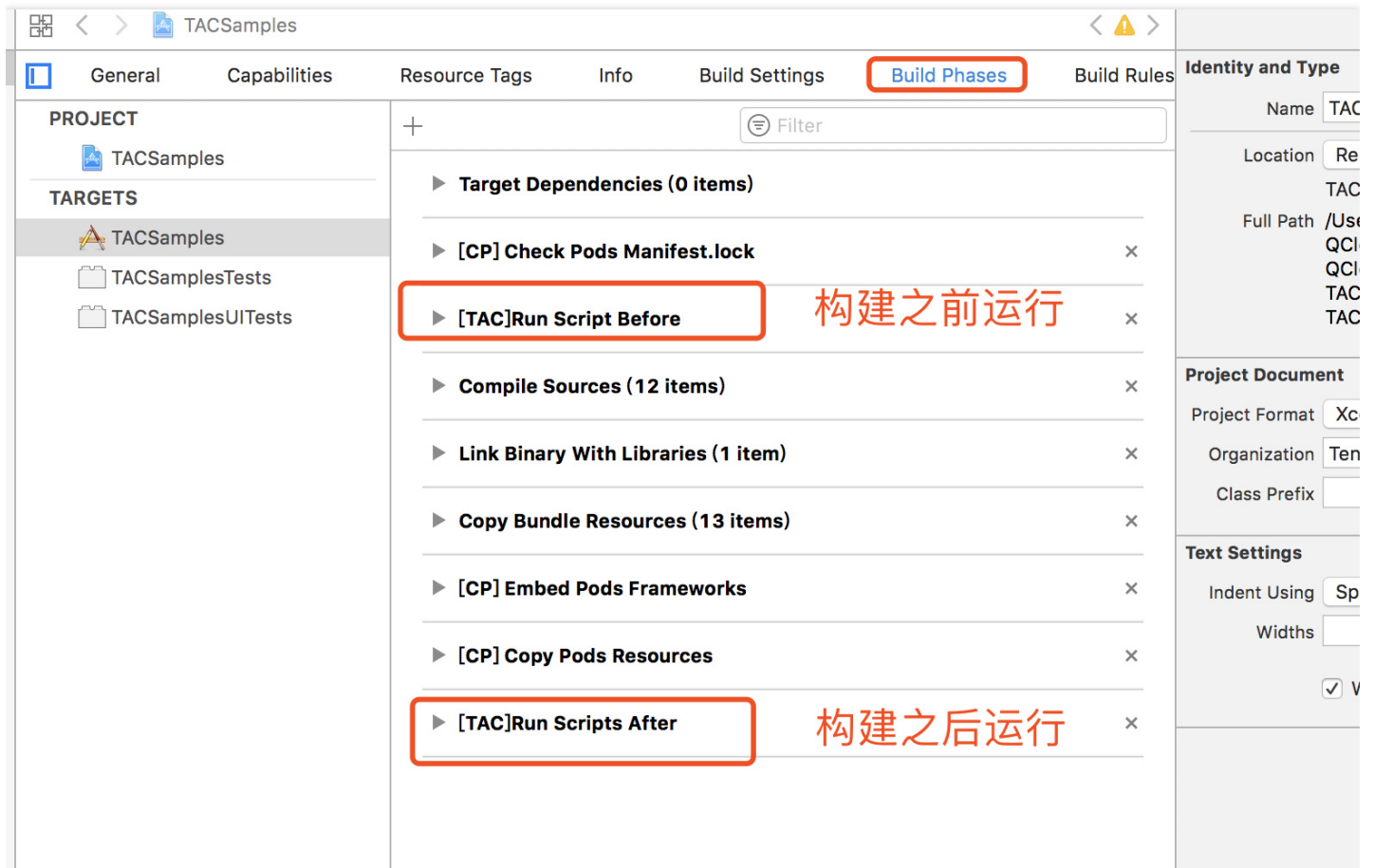
配置程序需要脚本

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}"
  end
end
```

```
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 existed"
  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。

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

Objective-C 代码示例：

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

Swift 代码示例：

```
import TACore
```

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 configure];
```

Swift 代码示例：

```
TACApplication.configure();
```

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 移动分享服务。

后续步骤

了解 MobileLine

- 查看 [MobileLine 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。

分享内容

最近更新时间：2018-06-11 17:22:06

分享 SDK 帮助您快速分享 App 内的内容，我们预设了分享的弹框界面，帮助您可以快速集成分享能力。下面的例子展示了如何调用 SDK，分享各种类型的内容，包括文字，图片，链接，文件等。

创建分享界面实例

首先，您需要创建一个分享界面实例，您可以在用户单击分享按钮时再创建：

```
// ViewController.m
TACShareDialog* shareDialog = [[TACShareDialog alloc] init];
self.shareDialog = shareDialog;
```

分享纯文字

```
// ViewController.m
TACSharePlainTextObject* obj = [TACSharePlainTextObject new];
obj.text = @"测试文本";
[self.shareDialog share:obj inViewController:self];
```

分享图片

```
// ViewController.m
UIImage* shareImage = /* image to be shared */;
TACBitmapObject* bitmapObject = [[TACBitmapObject alloc] initWithImage:shareImage];
bitmapObject.thumbImage = /* Thumb image */;
[self.shareDialog share:bitmapObject inViewController:self];
```

分享网页链接


```
// ViewController.h
```

```
TACShareURLObject* urlObject = [[TACShareURLObject alloc] init];  
urlObject.title = @"URL标题";  
urlObject.URLDescription = @"URL描述";  
urlObject.thumbImage = /*缩略图*/;  
urlObject.URL = @"需要分享的网页";  
[self.shareDialog share:urlObject inViewController:self];
```

分享在线音乐链接

与分享图片不同，这里分享的音视频都是指在互联网上的音视频，通过 URL 去访问，而不是分享存放在本地的音视频文件。如果需要分享存放在本地的音视频文件，那么可以通过分享文件的形式去实现。

遵循前面的假设的话，那么其实无论音频还是视频，其实都是通过在线 URL 去访问的资源，它们的分享都是通过 TACShareURLObject 去实现的。

```
TACShareURLObject* musicObject = [[TACShareURLObject alloc] init];  
musicObject.title = @"音乐分享标题";  
musicObject.URLDescription = @"音乐分享详细描述";  
musicObject.thumbImage = /* 封面URL */;  
musicObject.URL = @"http://music.163.com/#/share/2095551/3510133339";  
//这里是用户单击进去以后，打开的网页 URL  
musicObject.dataURL = @"http://123123123-1253653367.costj.myqcloud.com/谢安琪 - 绵绵.mp3";  
//这里是用户可以不单击进分享的具体内容，单击播放按钮就可以直接播放的音乐 URL  
musicObject.URLType = TAC_SHARE_URL_TYPE_MUSIC;  
[self.shareDialog share:musicObject inViewController:self];  
//should import header file TACShareDialog.h
```

分享在线视频链接

```
TACShareURLObject* videoObject = [[TACShareURLObject alloc] init];  
videoObject.URLType = TAC_SHARE_URL_TYPE_VIDEO;  
videoObject.title = @"分享视频的标题";  
videoObject.URLDescription = @"分享视频的详细描述";  
videoObject.URL = @"https://www.bilibili.com/video/av21061574/";  
//单击进去以后，展示视频的 URL  
[videoObject setThumbImage:/* 视频的封面图片 */];  
[self.shareDialog share:videoObject inViewController:self];
```

分享文件

分享文件实际上等同于“用其它应用打开”，事实上是将某个本地文件路径对应的数据传输至另一个 App 中进行处理。常见的有这几种场景：

- 传输单张或者数张图片至另一个 App 中，如微信等。
- 传输单个或者多个视频至另一个 App 中，如微博等。
- 调用其它应用打开某个文件，例如调用用户手机的 Word 来打开一个 docx 格式的文档等。

分享单个本地图片

```
TACShareFileObject* fileObject = [[TACShareFileObject alloc] init];
fileObject.metaData.type = TAC_SHARE_FILE_TYPE_PHOTO;
fileObject.data = [NSData dataWithContentsOfFile:[NSBundle mainBundle] pathForResource:@"test"
ofType:@"png"];//图片的二进制数据

[self.shareDialog shareMultiFiles:@[fileObject] inViewController:self];
```

分享单个本地视频

```
TACShareFileObject* fileObject = [[TACShareFileObject alloc] init];
fileObject.metaData.type = TAC_SHARE_FILE_TYPE_VIDEO;
fileObject.data = [NSData dataWithContentsOfFile:[NSBundle mainBundle] pathForResource:@"test"
ofType:@"png"];//图片的二进制数据

[self.shareDialog shareMultiFiles:@[fileObject] inViewController:self];
```

分享单个本地普通文件

```
TACShareFileObject* fileObject = [[TACShareFileObject alloc] init];
fileObject.metaData.type = TAC_SHARE_FILE_TYPE_VIDEO;
fileObject.data = /* NSData*格式的文件的二进制数据，或者指向文件的NSURL */;

[self.shareDialog shareMultiFiles:@[fileObject] inViewController:self];
```

分享多个本地图片

```
TACShareFileObject* fileObject0 = [[TACShareFileObject alloc] init];
fileObject0.metaData.type = TAC_SHARE_FILE_TYPE_PHOTO;
fileObject0.data = [NSData dataWithContentsOfFile:[NSBundle mainBundle] pathForResource:@"test"
ofType:@"png"];
```

```
TACShareFileObject* fileObject1 = [[TACShareFileObject alloc] init];
fileObject1.metaData.type = TAC_SHARE_FILE_TYPE_PHOTO;
fileObject1.data =
[NSData dataWithContentsOfFile:[[NSBundle mainBundle] pathForResource:@"TestThumb" ofType:
@"png"]];
[self.shareDialog shareMultiFiles:@[fileObject0,fileObject1] inViewController:self];
```

分享多个本地视频

```
TACShareFileObject* fileObject0 = [[TACShareFileObject alloc] init];
fileObject0.metaData.type = TAC_SHARE_FILE_TYPE_VIDEO;
fileObject0.data = /* NSData*格式的视频二进制数据，或者指向文件的NSURL* */;

TACShareFileObject* fileObject1 = [[TACShareFileObject alloc] init];
fileObject1.metaData.type = TAC_SHARE_FILE_TYPE_VIDEO;
fileObject1.data = /* NSData*格式的视频二进制数据，或者指向文件的NSURL* */;
[self.shareDialog shareMultiFiles:@[fileObject0,fileObject1] inViewController:self];
```

分享多个本地普通文件

```
TACShareFileObject* fileObject0 = [[TACShareFileObject alloc] init];
fileObject0.data = /* NSData*格式的文件二进制数据，或者指向文件的NSURL* */;

TACShareFileObject* fileObject1 = [[TACShareFileObject alloc] init];
fileObject1.data = /* NSData*格式的文件二进制数据，或者指向文件的NSURL* */;
[self.shareDialog shareMultiFiles:@[fileObject0,fileObject1] inViewController:self];
```

分享微信小程序

分享小程序目前只通过微信对话的渠道进行分享。并且小程序的开发者账号与当前进行分享使用的微信开发平台账号需要是同一个账号，否则无法进行分享。

```
TACShareMiniProgramObject* miniProgram = [[TACShareMiniProgramObject alloc] init];
miniProgram.title = @"小程序分享测试Title";
miniProgram.miniProgramDescription = @"小程序分享详细内容";
miniProgram.hdImage = /*缩略图*/
miniProgram.userName = @"小程序的原始id";
miniProgram.path = @"/pages/media";
miniProgram.webPageURL = @"兼容旧版本的小程序url";
[self.shareDialog share:miniProgram inViewController:self];
```

自定义分享界面

最近更新时间：2018-06-11 17:22:18

分享 SDK 预设了分享界面的样式，可以直接使用，但我们也支持您通过 SDK 定制界面。在定制界面时，您只需要关心渲染过程，而不需关心分享的流程。

默认分享界面样式

我们预设了一个轻快的分享界面，无需设置可以直接使用：



自定义分享界面

但如果有自定义分享界面的需求的话，那么可以指定 TACShareDialog 的 UIRenderer 属性来实现。

@interface TACShareDialog : NSObject

```
/**
渲染分享 UI 的委托对象。如果希望自定义分享的弹窗样式，那么可以设置该对象，然后渲染 UI 会通过该委托对象来实现。没有设置的话会通过默认的弹框样式来显示。
*/
@property (weak, nonatomic) id<TACShareUIRenderProtocol> UIRenderer;

//.....other properties, methods..etc
```

UIRenderer 负责将需要分享的渠道信息展示在屏幕上，它需要遵循 TACShareUIRenderProtocol 协议，实现协议中的方法。当用户选择了相应渠道之后，需要通过调用回调 Block 来让 SDK 进行下一步的分享流程。

@protocol TACShareUIRenderProtocol <NSObject>

```
/**
如果希望实现自定义的分享界面，那么需要遵循该协议并实现该接口。SDK 会调用该接口，传入必要的信息让用户自定义实现分享的界面
```

在这里需要自定义实现的是，通过传入的 *ShareObject* 和分享渠道 *channels* 来在对应的 *ViewController* 中弹出分享的界面，并且在用户单击选择具体的渠道以后，通过 *block* 将回调内容传回到 SDK 内部里。

@param *shareObject* 要分享的 *Object*，包含了分享的具体内容信息。注意 *ShareObject* 可能是单个对象（*TACBaseShareObject* 的子类），也可能是一个 *NSArray* 类型的变量，放置了需要分享的多个对象的数组（这种情况下，数组内的对象都是 *TACShareFileObject* 类型的）。

@param *channels* 可供分享的渠道列表，也就是实际上可以提供给用户进行选择的渠道选项。

@param *viewController* 展示弹窗的 *ViewController*

@param *block* 用户单击对应渠道的回调，将用户单击的具体渠道回调给 SDK 继续调用下一步分享逻辑。

```
*/
- (void) presentShareViewController:(NSObject*)shareObject
withAvailableChannels:(NSArray<TACSocialShareChannel*>*)channels
inViewController:(UIViewController*)viewController
ShareChannelClickedBlock:(TACShareChannelClickedBlock)block;
```

@end

监听分享结果

最近更新时间：2018-06-11 17:22:01

分享 SDK 帮助您快速分享 App 内的内容，在分享行为之后，您可以查看下面的示例，监听分享结果。

分享结果

您可以通过以下 API 拿到结果和错误信息：

```
[TACSocialShareService defaultService].delegeta = self;  
  
- (void) onHandleShareChannel:(TACShareChannel)channel error:(NSError*)error {  
    if (nil == error) {  
        //分享成功  
    } else {  
        error.code; //包含了具体的失败信息，例如-999：用户取消，-1000：分享失败,其它错误码对应具体的 QQ 或者微信等的内部错误  
    }  
}
```

分享效果追踪

最近更新时间：2018-06-15 10:14:57

分享 SDK 帮助您快速分享 App 内的内容。不仅如此，我们还通过大数据和后台监控系统，为您统计和追踪分享效果，包括分享数、意向分享数、回流数、分享率等关键指标，帮助您快速决策，更好分享 SDK 帮助您快速分享 App 内的内容。不仅如此，我们还通过大数据和后台监控系统，为您统计和追踪分享效果，包括分享数、意向分享数、回流数、分享率等关键指标，帮助您快速决策，更好地运营您的应用。

以下的步骤是为了更好的统计效果，即使略过，也不会影响分享功能的使用，并且默认情况下我们仍然会统计分享次数和渠道。

为了更好地查看和追踪分享效果，您需要添加以下几步：

在 iOS SDK 中统计落地页访问

落地页，通常是指您想要被分享的页面，例如活动页、推广页等，在 App 内对应的页面。我们需要统计落地页在 App 内的访问次数，以便更好的统计分享率。

下面的例子中展示了该如何统计落地页展示的次数，该接口统计的是落地页展示的次数。请在页面 **每次展示** 的时候，通常是 `UIViewController` 的 `viewDidAppear:` 中调用：

```
// #import <TACSocialShare/TACSocialShare.h>
[TACShareDataUploader trackShareViewOpen]
```

在分享页的 H5 页面添加统计代码

分享出去的链接通常是一个 H5 页面，通过统计分享页面在社交平台被单击带来的访问次数，我们能更好地评估分享的效果。如果分享页内集成了我们提供的 JS SDK，那么我们会统计该分享页被单击的情况，达到更好的统计效果。

请在您的分享页 H5 代码头部添加统计的 JS SDK：

```
<html>
...

<script>
```

```
(function() {  
var mta = document.createElement("script");  
mta.src = "http://imgcache.xg.qq.com/mta/h5/ad_channel_test/mobile_share_stats.js";  
mta.setAttribute("name", "MTA_MS");  
var s = document.getElementsByTagName("script")[0];  
s.parentNode.insertBefore(mta, s);  
})();  
</script>  
  
<body>  
...  
</body>  
</html>
```

在控制台查看分享统计

在控制台的移动分享栏目下可以单击查看具体的分享统计。

地运营您的应用。

以下的步骤是为了更好的统计效果，即使略过，也不会影响分享功能的使用，并且默认情况下我们仍然会统计分享次数和渠道。

为了更好地查看和追踪分享效果，您需要添加以下几步：

1. 在 iOS SDK 中统计落地页访问

落地页，通常是指您想要被分享的页面，例如活动页、推广页等，在 App 内对应的页面。我们需要统计落地页在 App 内的访问次数，以便更好的统计分享率。

下面的例子中展示了该如何统计落地页展示的次数，该接口统计的是落地页展示的次数。请在页面**每次展示**的时候，通常是 `UIViewController` 的 `viewDidAppear:` 中调用：

```
// #import <TACSocialShare/TACSocialShare.h>  
[TACShareDataUploader trackShareViewOpen]
```

2. 在分享页的 H5 页面添加统计代码

分享出去的链接通常是一个 H5 页面，通过统计分享页面在社交平台被单击带来的访问次数，我们能更好地评估分享的效果。如果分享页内集成了我们提供的 JS SDK，那么我们会统计该分享页被单击的情况，达到更好的统计效

果。

请在您的分享页 H5 代码头部添加统计的 JS SDK：

```
<html>
...

<script>
(function() {
var mta = document.createElement("script");
mta.src = "//pingjs.qq.com/h5/mobile_share_stats.js";
mta.setAttribute("name", "MTA_MS");
var s = document.getElementsByTagName("script")[0];
s.parentNode.insertBefore(mta, s);
})();
</script>

<body>
...
</body>
</html>
```

3. 在控制台查看分享统计

在控制台的移动分享栏目下可以单击查看具体的分享统计。

开发者手册

名词解释

最近更新时间：2018-06-11 17:21:03

原始页面打开

落地页，通常是指您想要被分享的页面，例如活动页、推广页等，在 App 内对应的页面。落地页在 App 中打开的次数，分别统计打开次数和打开人数。

意向分享

用户单击分享按钮，并在分享弹框中选择一个渠道，视为一次意向分享，分别统计意向分享次数和意向分享人数。

分享数

用户选择了一个渠道并分享成功，视为一次分享。分别统计分享次数和分享人数。

回流数

回流数也可以视为分享页面打开数。分享生成的链接在社交平台上被其他用户单击，并访问页面，视为一次回流。分别统计回流次数和回流人数。

分享率

分享率等于分享成功次数除以页面总访问数，可以度量用户分享的意愿。此处的页面总访问数包含原始页面和分享页面的访问次数。

有效分享率

有效分享率等于回流数除以页面总访问数，可以度量页面分享带来的回流效果。此处的页面总访问数包含原始页面和分享页面的访问次数。