



Adwo iOS SDK Programming Guide

Version 6.3

Confidential

Release Notes

- * 本次SDK将仅支持Xcode5.0或更高版本，当前版本仅支持32位。因此，在选择处理器架构时请只选用armv7与armv7s。而Valid Architectures选项里采用默认的，同时支持armv7、armv7s与arm64。Deployment为iOS5.0。
- * 增加了植入性广告相关接口。新增CoreMotion.framework、MessageUI.framework、MapKit.framework与CoreAudio.framework。
- * 需要第三方应用开发者在项目工程中添加-lstdc++这一连接器选项。
- * 增加了手工轮询Banner广告的设置接口。这个适用于聚合情况。
- * 新增了与ETA广告类型以及相关的错误码和相关的AWAdViewDelegate代理回调消息。
- * 新增了获取当前广告ID的接口。此接口可用于辅助排错。
- * 在开发包中新增了ifly框架，请把这个框架导入到项目工程中。另外res文件夹中也新增了若干资源文件，也请加入。
- * 注意！本文档以及附属的Demo属于本公司机密文档。未经许可不得擅自发布！
- * 关于Adwo广告SDK的进一步介绍可参见<http://wiki.adwo.com/index.php/IOS/home>

目录

- * 用户注册
- * 添加应用
- * 嵌入SDK的准备工作
- * Adwo广告SDK接口介绍
- * AWAdViewDelegate接口介绍
- * Adwo广告SDK隐藏接口
- * AWAdViewDelegate隐藏接口
- * 关于全屏广告的进一步介绍
- * 关于附赠Demo的介绍

用户注册

- * 进入安沃官方网站，点击注册按钮进入注册页面：
<http://www.adwo.com/adMemberB/register>
- * 填写好自己的邮箱（作为登录帐号）、密码以及联系方式后，提交。





















添加应用

- * 登陆安沃官网后，点击“会员登录”。
- * 登录后，在“程序操作”一栏点击“添加APP”。
- * 选择“iOS”，填写应用名称、Bundle ID、应用类型以及支持的屏幕方向。
- * 这里要注意，如果你的广告都嵌在仅支持竖屏的视图控制器中，那么仅选择支持竖屏即可，否则也要选择支持横屏；同样，如果你的广告都嵌在仅支持横屏的视图控制器中，那么选择支持横屏，否则也要选上支持竖屏。
- * 在应用中创建广告对象时所传入的Bundle ID必须与您申请广告时所填写的Bundle ID吻合，否则可能将无法收到广告。
- * 注册完毕后，你将获得一个唯一的Publish ID（即PID）号。

嵌入SDK的准备工作的准备工作

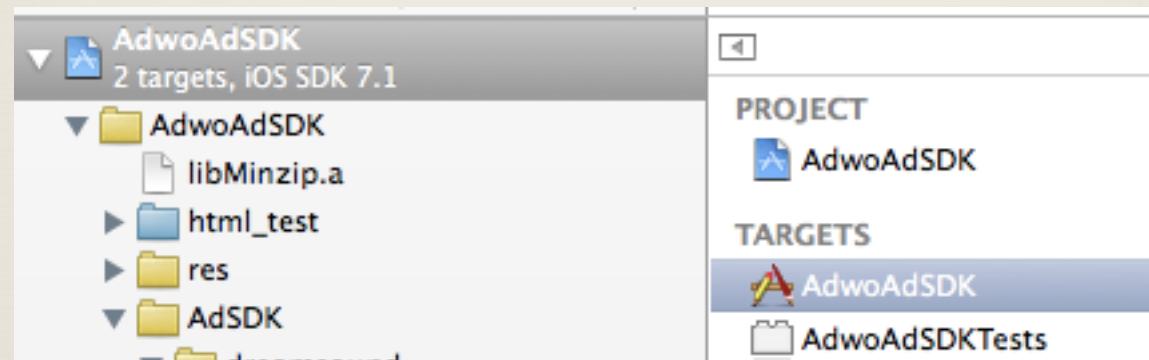
* 由于这次SDK仅支持XCode 5.0或更高版本，iOS 7.0 SDK，因此你需要在你的XCode工程中添加以下framework：

* AddressBook.framework
AdSupport.framework
AudioToolbox.framework
AVFoundation.framework
CoreAudio.framework
CoreLocation(optional)
CoreMedia.framework
CoreMotion.framework
CoreTelephony.framework
EventKit.framework
MapKit.framework
MessageUI.framework
PassKit.framework(optional)
QuartzCore.framework
StoreKit.framework
SystemConfiguration.framework
Social.framework(optional)
libz.1.2.5.dylib或libz.1.1.3.dylib

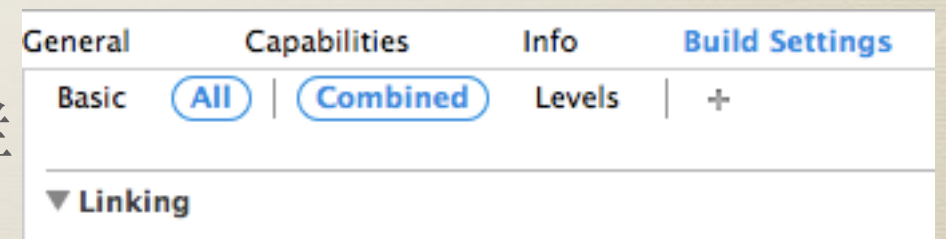
Name	Status
 MessageUI.framework	Required
 CoreMotion.framework	Required
 PassKit.framework	Optional
 libz.1.2.5.dylib	Required
 CoreLocation.framework	Optional
 Social.framework	Optional
 EventKit.framework	Required
 AddressBook.framework	Required
 StoreKit.framework	Required
 AdSupport.framework	Required
 minizip64.a	Required
 AudioToolbox.framework	Required
 AVFoundation.framework	Required
 CoreMedia.framework	Required
 CoreTelephony.framework	Required
 QuartzCore.framework	Required
 SystemConfiguration.framework	Required
 UIKit.framework	Required
 Foundation.framework	Required
 CoreGraphics.framework	Required

嵌入SDK的准备工作（续）

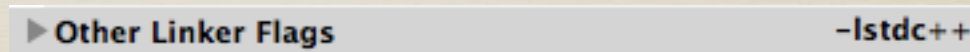
- * 随后，我们需要在项目设置中添加一个连接器选项，如果不加会导致连接失败。
- * 首先，点击项目工程，即蓝色Xcode图标。然后在TARGETS中选中你当前要构建的目标项目。



- * 然后，点击“Build Settings”找到“Linking”这一栏



- * 最后，找到“Other Linker Flags”这一项，在里面编辑，写上-lstdc++。

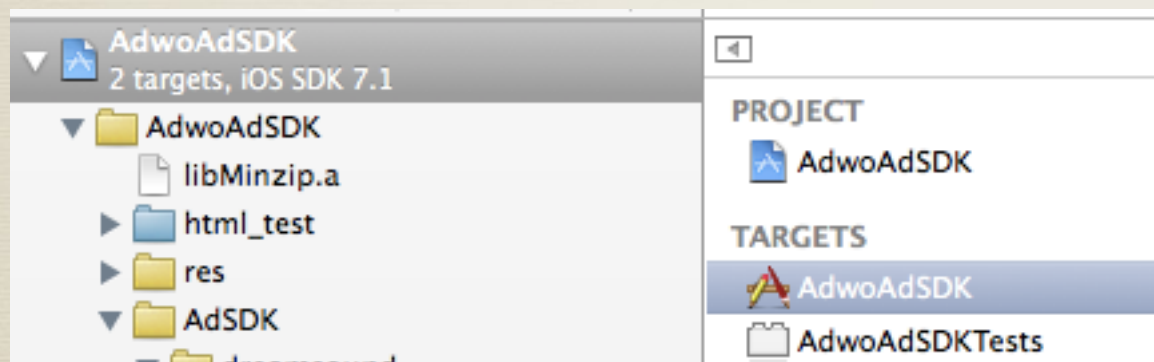


嵌入SDK的准备工作（续）

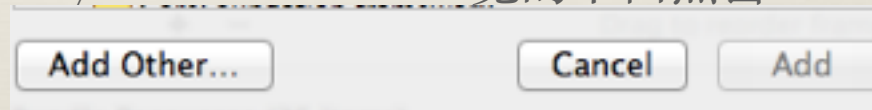
- * 由于本SDK支持Passbook，因此应用在Apple Developer网站上必须将App ID设置为允许Pass（**Enable for Passes**）。步骤为：登陆iOS开发者，进入iOS Provisioning Portal，然后选择App IDs，对你当前应用所用的App ID点击Configure，然后可以看到Enable Passes，把这个勾上就行。然后在“Pass Type IDs”这一览的右上角找到“New Pass Type ID”按钮，点击后填写pass标识即可。如果你实在不愿意植入PassKit.framework，那么将这个框架移除后，将 **ADWO_SDK_WITHOUT_PASSKIT_FRAMEWORK()** 这个宏加到你工程里的ViewController.m或AppDelegate.m源文件中。可以参考AdwoSDKBasic这个Demo。
- * 另外，本SDK可选地可使用CoreLocation.framework。若开发者的工程中没有包含CoreLocation.framework，那么必须要在你的AppDelegate.m或ViewController.m的类定义外添加一条宏语句——**ADWO_SDK_WITHOUT_CORE_LOCATION_FRAMEWORK()**。若已经包含了CoreLocation.framework，那么不要加这条语句。关于如何加这条宏语句可以参考AdwoSDKBasic这个Demo。

嵌入SDK的准备工作（续）

- * 本SDK加入了一个开源库minizip，被独立打包为libminizip.a。如果你的工程中已经使用了minizip开源代码，那么可以不将这个库文件加入到你的工程中；如果用了你自己的minizip库无法通过连接，那么尝试用本SDK所打包好的库文件。
- * 除此之外，本SDK还加入了第三方框架——iflyMSC.framework。添加第三方框架的方法是：先点击项目文件，然后在TARGETS中点击你当前的目标工程。



- * 然后，点击“Build Phases”，在“Link Binary with Libraries”一览的下面点击“+”号。然后点击左侧的“Add Others”，选择第三方库的路径即可。



- * 另外，本SDK加入了一些图片资源，文件夹名为res。请将这些图片资源也一同加入到你的工程中。
- * 工程的Deployment请设置为当前广告SDK支持的最低系统版本——iOS5.0

ADWO广告SDK接口介绍

下面介绍SDK接口。Adwo广告SDK从5.0版本起将采用纯C函数式样的接口，以提供更大的便利性和灵活性。

接口分为Banner、全屏以及植入性广告三种形式，以及Banner、全屏与植入性广告的公共接口。此外，5.0版本之前的代理protocol依然有。

Banner创建

* `UIView* AdwoAdCreateBanner(NSString *pid, BOOL showFormalAd, NSObject<AWAdViewDelegate> *delegate)`

创建一条Banner广告对象

参数pid: 申请一个应用后, 页面返回出来的广告发布ID (32个ASCII码字符)。

参数showFormalAd: 是否展示正式广告。如果传NO, 表示使用测试模式, SDK将给出测试广告; 如果传YES, 那么SDK将给出正式广告。

参数delegate: 设置AWAdViewDelegate代理对象。一般开发者在嵌入此banner的视图控制器来实现AWAdViewDelegate, 并且将此视图控制器对象传递给此参数。

返回: 如果返回为空, 表示广告初始化创建失败, 否则会返回一个有效的UIView的对象作为广告对象句柄(handle)。

初始化后, 可以对广告对象设置位置, 宽高可以都先设置为0。开发者若要移除banner广告, 必须调用AdwoAdRemoveAndDestroyBanner接口, 而不能直接调用removeFromSuperview或release。

另外, 如果要创建两条广告, 那么这两条广告的发布测试模式值必须相同, 即要么都是测试模式, 要么都是正式模式。当然, 本SDK并不推荐同时创建两个Banner, 不过开发者可以创建一个Banner, 一个全屏。

Banner移除

* **BOOL** AdwoAdRemoveAndDestroyBanner(**UIView** *adView)

此接口提供了移除banner广告的唯一接口。开发者不能直接使用removeFromSuperview或release来移除banner广告对象，只能通过此接口来完成。

参数adView: banner广告对象句柄

返回: 如果操作成功, 则返回YES, 否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。

* 开发者在使用此接口时不需要关心当前banner视图对象是否被加载到一个父视图上, SDK在移除时会自动判断。

加载Banner广告

* `BOOL AdwoAdLoadBannerAd(UIView *adView, enum ADWO_ADSDK_BANNER_SIZE bannerSize, NSTimeInterval *pRemainInterval)`

加载banner广告。这个接口先做的是对Banner广告进行请求，然后再加载广告资源予以展示。当广告真正加载完成时会发送代理的-

`(void)adwoAdViewDidLoadAd:(UIView*)adView`消息。若是加载失败，将会发送-
`(void)adwoAdViewDidFailToLoadAd:(UIView*)adView`消息。

* 参数adView: banner广告对象句柄

参数bannerSize: banner广告对象大小。当前的SDK在iPhone或iPod Touch上就一种规格的banner尺寸——`ADWO_ADSDK_BANNER_SIZE_NORMAL_BANNER`，表示320x50。在iPad上有两种规格——`ADWO_ADSDK_BANNER_SIZE_FOR_IPAD_320x50`，表示320x50；

`ADWO_ADSDK_BANNER_SIZE_FOR_IPAD_720x110`，表示720x110。

参数pRemainInterval: 指向剩余请求时间间隔。

返回: 如果操作成功，则返回YES，否则返回NO。开发者可以通过调用 `AdwoAdGetLatestErrorCode` 接口来获取错误码。

将Banner添加到父视图上

* **BOOL** AdwoAdAddBannerToSuperView(**UIView** *adView, **UIView** *superView)

将Banner广告对象添加到指定的父视图上。

* 参数：adView——Banner广告对象句柄
参数superView——用户指定的父视图

* 返回：如果操作成功，则返回YES，否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。

* 开发者应当使用这个接口将adView对象添加到superView对象上，尤其是在ARC环境下。

获取Banner最小请求时间间隔

* `NSTimeInterval` `AdwoAdGetBannerRequestInterval(void)`

获取Banner广告当前最小请求时间间隔。由于从iOS广告SDK5.1起，SDK本身不会自动对Banner进行刷新，因此开发者通过每次调用这个函数接口来获取当前最小需要间隔多长时间来获取下一次的Banner广告。

* 返回：当前Banner广告的最小请求时间间隔，单位是秒。

停止Banner广告自动刷新

* `void adwoAdStopBannerAutoRefresh(void)`

阻止所有Banner广告自动轮询刷新。这个接口一般适用于广告SDK的聚合。

* 调用这个函数的效果是全局的，并且必须在调用AdwoAdCreateBanner接口之前调用。一旦调用此接口之后，所有Banner广告的刷新都通过开发者自己进行，SDK将不会再自动轮询刷新Banner广告。

获取全屏广告对象句柄

* `UIView* AdwoAdGetFullScreenAdHandle(NSString *pid, BOOL showFormalAd, NSObject<AWAdViewDelegate> *delegate, enum ADWOSDK_FSAD_SHOW_FORM fsAdForm)`

获取全屏广告对象句柄。对于全屏广告，不需要像banner那样事先将广告对象视图加载到父视图上，而是在接收到- `(void)adwoAdViewDidLoadAd:(UIView*)adView`代理消息后调用加载展示接口。另一个与banner不同的地方是，全屏广告对象句柄获得之后，开发者不需要考虑如何释放。SDK会在全屏展示完成之后自动释放。另外，只有在开发者接收到- `(void)adwoFullScreenAdDismissed:(UIView*)adView`代理消息之后才能再次调用此接口，重新获得全屏广告对象。

另外，全屏广告的delegate最好能指向一个在程序运行当中不会被销毁的对象。

* 参数pid：申请一个应用后，页面返回出来的广告发布ID（32个ASCII码字符）。

参数showFormalAd：是否展示正式广告。如果传YES，则展示正式广告，传NO则展示测试广告。

参数delegate：设置AWAdViewDelegate代理对象。这里建议开发者使用一个在整个程序运行中不会被销毁的对象来实现AWAdViewDelegate，并且将此对象传递给此参数。

参数fsAdForm：全屏广告展示形式。ADWOSDK_FSAD_SHOW_FORM_APPFUN_WITH_BRAND表示展示App Fun插屏全屏以及品牌插屏，优先展示App Fun，而品牌插屏则用于补量；ADWOSDK_FSAD_SHOW_FORM_LAUNCHING表示展示应用启动时全屏；ADWOSDK_FSAD_SHOW_FORM_GROUND_SWITCH表示后台切换到前台全屏；ADWOSDK_FSAD_SHOW_FORM_APPFUN表示仅展示App Fun插屏；ADWOSDK_FSAD_SHOW_FORM_BRAND表示仅展示品牌插屏。

返回：若成功，则返回全屏广告对象句柄；否则，返回空。

加载全屏广告

* **BOOL** AdwoAdLoadFullScreenAd(**UIView** *fsAd, **BOOL** orientationLocked, **NSTimeInterval** *pRemainInterval)

加载全屏广告。开发者调用此接口后，SDK将会开始请求全屏广告资源，然后加载。

* 参数**fsAd**：全屏广告对象句柄

参数**orientationLocked**：方向锁定。如果当前应用在展示全屏广告的时候仅使用横屏或者竖屏，并且横竖屏在广告展示期间不会做切换，那么设置为YES。否则，设置为NO。当方向锁定时，SDK会向服务器仅请求适应于当前屏幕方向的素材，从而大大节省了流量，并且加快了网络加载广告素材的速度。

参数**pRemainInterval**：指向剩余请求时间间隔。

返回：如果操作成功，则返回YES，否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。另外，如果全屏广告类型为ADWOSDK_FSAD_SHOW_FORM_LAUNCHING时，若当前开屏全屏素材没加载好，则会返回NO，并且错误码为

ADWO_ADSDK_ERROR_CODE_FS_LAUNCHING_AD_REQUESTING。如果开屏全屏素材都已经加载好，那么将直接返回YES，此时，开发者接收到adwoAdViewDidLoadAd消息时可以直接做展示。

* 当全屏广告加载成功时，将会发送- (**void**)adwoAdViewDidFailToLoadAd:(**UIView***)adView代理消息；倘若加载失败，则会发送- (**void**)adwoAdViewDidLoadAd:(**UIView***)adView代理消息。

展示全屏广告

* **BOOL** AdwoAdShowFullScreenAd(**UIView** *fsAd)

展示全屏广告。当开发者接收到- (**void**)adwoAdViewDidLoadAd:(**UIView***)adView代理消息时，可以在此方法实现中调用此接口以展示全屏广告。当开发者加载的是开屏全屏广告（即类型为ADWOSDK_FSAD_SHOW_FORM_LAUNCHING）时，倘若在加载时返回的是NO，那么不予以展示；若在加载时返回的是YES，那么即可展示。

* 参数fsAd：全屏广告对象句柄

返回： 如果操作成功，则返回YES，否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。

设置后台切换到前台全屏自动展示

* **BOOL** AdwoAdSetGroundSwitchAdAutoToShow(**UIView** *fsAd, **BOOL** autoToShow)

设置是否自动展示后台切换到前台广告。

* 参数fsAd: 后台切换到前台全屏对象句柄

参数autoToShow: 是否自动展示。如果为YES, 那么当应用从后台切换到前台时, 且此时当后台切换到前台广告也准备好展示, 那么SDK将会自动展示此后台切换到前台广告。如果设置为NO, 那么在应用从后台切换到前台时, SDK将不会自动展示全屏广告, 开发者需要手工调用AdwoAdShowFullScreenAd接口来展示。默认情况下, 后台切换到前台广告是由SDK自动展示的。

* 返回: 如果操作成功, 则返回YES, 否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。

* 若当前后台切换到前台全屏是自动展示的, 那么当开发者自己调用AdwoAdShowFullScreenAd接口时, 将返回NO, 并且给出ADWO_ADSDK_ERROR_CODE_FS_ALREADY_AUTO_SHOW错误码。

获取全屏广告的最小请求时间间隔

* `NSTimeInterval` `AdwoAdGetFullScreenRequestInterval(enum ADWOSDK_FSAD_SHOW_FORM fsAdType)`

获取当前所指定类型的全屏广告的最小请求时间间隔。

* 参数 `fsAdType`: 全屏广告类型。

* 返回: 当前所指定类型的全屏广告的最小请求时间间隔。

创建植入性广告对象

* `UIView*` `AdwoAdCreateImplantAd(NSString *pid, BOOL showFormalAd, NSObject<AWAdViewDelegate> *delegate, NSString *adInfo)`

创建植入性广告对象

* 参数pid: 申请一个应用后, 页面返回出来的广告发布ID (32个ASCII码字符)。

参数showFormalAd: 是否展示正式广告。如果传NO, 表示使用测试模式, SDK将给出测试广告; 如果传YES, 那么SDK将给出正式广告。

参数delegate: AWAdViewDelegate代理。应用开发者应该将展示本SDK Banner的视图控制器实现AWAdViewDelegate代理, 并且将视图控制器对象传给此参数。此参数不能为空。注意, 此参数不会被retain。

参数adInfo: 广告信息。开发者可以设置指定广告信息来请求自己想要的广告植入性广告。如果设置为空或者非有效的字符串, 则由服务器来决定给出相应广告。注意, 这个参数会被retain, 因此如果字符串实参使用alloc分配的话, 在调用完这个接口之后需要release一次。

* 返回: 如果返回为空, 表示广告初始化创建失败, 开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。如果创建成功, 则返回一个UIView对象, 作为广告对象句柄。

移除并销毁植入性广告

* **BOOL** AdwoAdRemoveAndDestroyImplantAd(UIView *adView)

移除并销毁植入性广告

* 参数adView: 植入性广告对象句柄

* 返回: 如果销毁成功, 返回YES; 否则, 返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。

* 这里要注意的是, 当开发者要把植入性广告从父视图上移除时, 不能直接调用UIView类的removeFromSuperview方法, 而必须使用此接口。

加载植入性广告

- * `BOOL AdwoAdLoadImplantAd(UIView *adView, NSTimeInterval *pRemainInterval)`
加载植入性广告
- * 参数adView: 植入性广告对象句柄
参数pRemainInterval: 指向剩余请求时间间隔。
- * 返回: 若加载成功返回YES, 否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。
- * 这里要注意的是, 对广告对象做各种属性设置必须放在调用此接口之前。如果在调用此接口之后调用某些属性设置接口可能会无效。

展示植入性广告

- * **BOOL** AdwoAdShowImplantAd(**UIView** *adView, **UIView** *theSuperview)
展示植入性广告
- * 参数adView: 植入性广告对象句柄
参数theSuperview: 植入性广告视图将被添加到的父视图
- * 返回: 若加载成功返回YES, 否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。
- * 这个函数被调用后, 开发者无需自己调用release方法。

设置植入性广告互动信息

- * **BOOL** AdwoAdSetImplantAdInteractiveInfo(**UIView** *adView, **NSString** *info)
设置与植入性广告内容互动的应用信息。当开发者输入一些简短的字符串信息后，某些定制的植入性广告将会以某种方式将开发者传入的信息展示在广告页面上。
- * 参数adView: 植入性广告对象句柄
参数info: 指定的应用互动信息
- * 返回: 若加载成功返回YES，否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。
- * 这个接口可以在任意使用调用。广告当前植入性广告已经展示，那么重新设置的信息将会在页面上刷新显示。

激活植入性广告

* **BOOL** AdwoAdImplantAdActivate(**UIView** *adView)

激活当前植入性广告。开发者有时会先把植入性广告贴在某个大的Scrollview中，此时该植入性广告没被显示出来。当用户滚动scroll view之后，使得此植入性广告能够被看到，开发者可以调用此接口来激活植入性广告。植入性广告一旦被激活，可能会有动画、播放音频、播放视频等功能。

* 参数参数adView: 植入性广告对象句柄

* 注意，只有当这个接口被调用之后，展示才算有效果。

获取植入性广告的最小请求时间间隔

* `NSTimeInterval` `AdwoAdGetImplantRequestInterval(void)`

获取植入性广告当前最小请求时间间隔。

* 返回：当前植入性广告最小请求时间间隔，单位为秒。

启用事件触发型广告

* **BOOL** AdwoAdLaunchETA(**enum** ADWOSDK_ETA_TYPE etaType, **NSString** *pid, **BOOL** showFormalAd, **const struct** AdwoAdPreferenceSettings *settings)

启动事件触发型广告。事件触发型广告被启动后将会自动请求当前的指定的事件触发型广告。若指定的ETA广告能被触发，SDK会对应用发送adwoAdETARequestedAndActivated消息。

但是即便存在事件触发型广告，SDK也不会马上通知应用程序。而是等到SDK捕获到了相应事件触发特征码之后才会通知第三方应用的代理，发出adwoAdETAReceivedSpecifiedFeatureCode代理回调通知。

* 参数etaType: 指定的事件触发广告类型，ADWOSDK_ETA_TYPE_VOICEPRINT表示声纹广告；ADWOSDK_ETA_TYPE_IBEACON表示iBeacon广告。当前仅声纹广告有效。
参数pid: 申请一个应用后，页面返回出来的广告发布ID（32个ASCII码字符）
参数showFormalAd: 是否展示正式广告。如果传NO，表示使用测试模式，SDK将给出测试广告；如果传YES，那么SDK将给出正式广告。
参数settings: 指向广告属性设置结构体变量的指针。若为空，则启用默认设置

* 返回: 如果成功，则返回YES，否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。

设置事件触发型广告代理

- * `void AdwoAdSetETADelegate(enum ADWOSDK_ETA_TYPE etaType, NSObject<AWAdViewDelegate> *delegate)`
设置ETA广告的代理。这个接口在调用完AdwoAdLaunchETA接口之后仅跟着调用，且必须调用。而参数delegate不能为空。
 - * 参数etaType：指定的事件触发广告类型，
ADWOSDK_ETA_TYPE_VOICEPRINT表示声纹广告；
ADWOSDK_ETA_TYPE_IBEACON表示iBeacon广告。当前仅声纹广告有效。
- 参数delegate：AWAdViewDelegate代理。应用开发者应该将展示本SDK广告的视图控制器实现AWAdViewDelegate代理，并且将视图控制器对象传给此参数。注意，delegate参数不会被retain。因此，如果开发者要销毁delegate所对应的对象之前，必须调用此接口，并传空。

获取当前ETA广告的尺寸

- * `int AdwoAdGetETAViewSizes(enum ADWOSDK_ETA_TYPE etaType, CGSize adSizes[])`
获取指定ETA广告的尺寸。一个ETA广告可能含有多种尺寸。开发者可以根据当前界面环境来选择展示哪种广告尺寸，如果含有多个的话。
注意，此接口应该在收到adwoAdETAReceivedSpecifiedFeatureCode代理消息之后才能被调用，否则将会返回0。
- * 参数etaType：指定的事件触发广告类型，
ADWOSDK_ETA_TYPE_VOICEPRINT表示声纹广告；
ADWOSDK_ETA_TYPE_IBEACON表示iBeacon广告。当前仅声纹广告有效。

参数adSizes：存放SDK所返回的广告尺寸的数组，最大个数为ADWOSDK_MAX_ETA_AD_SIZES。
- * 返回：当前事件触发型广告所支持的尺寸个数。若是0，则说明当前事件触发型广告未被识别或尚未被激活。

添加ETA广告视图

* **BOOL** AdwoAdAddETAViewToSuperview(**enum** ADWOSDK_ETA_TYPE etaType, **UIView** *superview, **CGSize** adSize, **CGPoint** position)

将当前的ETA广告视图添加到当前的指定视图上，予以展示。

* 参数etaType: 指定的事件触发广告类型，
ADWOSDK_ETA_TYPE_VOICEPRINT表示声纹广告；
ADWOSDK_ETA_TYPE_IBEACON表示iBeacon广告。当前仅声纹广告有效。
参数superview: 将当前事件触发型广告所添加到的父视图
参数adSize: 指定当前广告页面尺寸。这个尺寸应该在
AdwoAdGetETAViewSizes接口所返回的广告尺寸数组中选择。
参数position: 指定当前事件触发型广告视图的位置

* 返回: 如果成功，则返回YES，否则返回NO。开发者可以通过调用
AdwoAdGetLatestErrorCode接口来获取错误码。

移除事件触发型广告视图

- * `void AdwoAdRemoveETAViewFromSuperview(enum ADWOSDK_ETA_TYPE etaType)`
将指定的事件触发型广告从当前父视图上移除
- * 参数etaType: 指定的事件触发广告类型,
ADWOSDK_ETA_TYPE_VOICEPRINT表示声纹广告;
ADWOSDK_ETA_TYPE_IBEACON表示iBeacon广告。当前仅声纹广告有效。

获取ETA广告信息

* `NSString *AdwoAdGetVoicePrintAdInfo(enum ADWOSDK_ETA_TYPE etaType, int *pAdID, BOOL *pIsReadyToShow)`
获取事件触发型广告相关信息。

* 参数etaType: 指定的事件触发广告类型,
ADWOSDK_ETA_TYPE_VOICEPRINT表示声纹广告;
ADWOSDK_ETA_TYPE_IBEACON表示iBeacon广告。当前仅声纹广告有效。

参数pAdID: 指向接受当前事件触发型广告ID的变量

参数pIsReadyToShow: 指向接受当前事件触发型广告是否准备好展示的变量

* 返回: 若当前事件触发型广告已经加载好, 则返回当前事件触发型广告信息字符串, 否则返回空。

获取最近错误码

* `enum ADWO_ADSDK_ERROR_CODE AdwoAdGetLatestErrorCode(void)`

返回最近一次的错误码。此接口对Banner与全屏广告均适用。

* 具体错误码及含义见下一页

错误码定义

- * ADWO_ADSDK_ERROR_CODE_SUCCESS: 操作成功
- ADWO_ADSDK_ERROR_CODE_INIT_FAILED: 广告对象初始化失败
- ADWO_ADSDK_ERROR_CODE_AD_HAS_BEEN_LOADED: 已经用当前的广告对象调用了加载接口
- ADWO_ADSDK_ERROR_CODE_NULL_PARAMS: 不该为空的参数却为空了
- ADWO_ADSDK_ERROR_CODE_ILLEGAL_PARAMETER: 参数值非法
- ADWO_ADSDK_ERROR_CODE_ILLEGAL_HANDLE: 非法的广告对象句柄
- ADWO_ADSDK_ERROR_CODE_ILLEGAL_DELEGATE: 代理为空或adwoGetBaseViewController代理方法没实现
- ADWO_ADSDK_ERROR_CODE_ILLEGAL_ADVIEW_RETAIN_COUNT: 非法的广告对象句柄引用计数
- ADWO_ADSDK_ERROR_CODE_UNEXPECTED_ERROR: 意料之外的错误
- ADWO_ADSDK_ERROR_CODE_AD_REQUEST_TOO_OFTEN: 广告请求过于频繁
- ADWO_ADSDK_ERROR_CODE_LOAD_AD_FAILED: 广告加载失败
- ADWO_ADSDK_ERROR_CODE_FS_AD_HAS_BEEN_SHOWN: 全屏广告已经被展示过
- ADWO_ADSDK_ERROR_CODE_FS_AD_NOT_READY_TO_SHOW: 全屏广告还没准备好展示
- ADWO_ADSDK_ERROR_CODE_FS_RESOURCE_DAMAGED: 全屏广告资源破损
- ADWO_ADSDK_ERROR_CODE_FS_LAUNCHING_AD_REQUESTING: 开屏全屏广告正在请求
- ADWO_ADSDK_ERROR_CODE_FS_ALREADY_AUTO_SHOW: 当前全屏已设置为自动展示
- ADWO_ADSDK_ERROR_CODE_ETA_DISABLED: 当前事件触发型广告已被禁用
- ADWO_ADSDK_ERROR_CODE_ETA_SIZE_INVALID: 没找到相应合法尺寸的事件触发型广告

错误码定义（续）

- * ADWO_ADSDK_ERROR_CODE_REQUEST_SERVER_BUSY: 服务器繁忙
- ADWO_ADSDK_ERROR_CODE_REQUEST_NO_AD: 当前没有广告
- ADWO_ADSDK_ERROR_CODE_REQUEST_UNKNOWN_ERROR: 未知请求错误
- ADWO_ADSDK_ERROR_CODE_REQUEST_INEXIST_PID: PID不存在
- ADWO_ADSDK_ERROR_CODE_REQUEST_INACTIVE_PID: PID未被激活
- ADWO_ADSDK_ERROR_CODE_REQUEST_REQUEST_DATA: 请求数据有问题
- ADWO_ADSDK_ERROR_CODE_REQUEST_RECEIVED_DATA: 接收到的数据有问题
- ADWO_ADSDK_ERROR_CODE_REQUEST_NO_AD_IP: 当前IP下，广告已投放完
- ADWO_ADSDK_ERROR_CODE_REQUEST_NO_AD_POOL: 当前广告都已投放完
- ADWO_ADSDK_ERROR_CODE_REQUEST_NO_AD_LOW_RANK: 没有低优先级的广告
- ADWO_ADSDK_ERROR_CODE_REQUEST_BUNDLE_ID: 开发者在Adwo官网所注册的Bundle ID与当前应用的Bundle ID不一致
- ADWO_ADSDK_ERROR_CODE_REQUEST_RESPONSE_ERROR: 服务器响应出错
- ADWO_ADSDK_ERROR_CODE_REQUEST_NETWORK_CONNECT: 当前网络没连接或网络信号不好
- ADWO_ADSDK_ERROR_CODE_REQUEST_INVALID_REQUEST_URL: 请求URL出错

设置广告属性接口

* 设置全屏、Banner广告以及植入性广告的公共属性：

```
BOOL AdwoAdSetAdAttributes(UIView *adView, const struct  
AdwoAdPreferenceSettings *settings)
```

* 参数：adView——Banner或全屏的广告对象句柄

参数：settings——AdwoAdPreferenceSettings结构体变量地址

返回：如果操作成功，则返回YES，否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。

* 开发者应该在加载广告之前调用此接口。此接口如果不调用，那么SDK将以默认方式工作。另外要注意的是，如果开发者先在函数内定义AdwoAdPreferenceSettings结构体变量然后再逐一赋值的话，应该在赋值前先用memset清零，避免未初始化的数据存在于此结构体变量中。建议定义此变量后直接对其做初始化。

AdwoAdPreferenceSettings 结构体介绍

- * 结构体AdwoAdPreferenceSettings的成员属性如下描述：
- * **adSlotID**——广告slot ID。此属性一般在Banner广告上设置，表示特定的广告位。一般使用此属性需与本公司做密切合作才有效。对于全屏广告，SDK将会根据不同的广告展示类型自动设置此属性。
- * **animationType**——动画类型。这里要注意的是，Banner和全屏动画用的是两个不同的枚举，后两章将会介绍。
- * **spreadChannel**——推广渠道。目前就两种：ADWOSDK_SPREAD_CHANNEL_APP_STORE表示App Store渠道；ADWOSDK_SPREAD_CHANNEL_91_STORE表示91渠道。
- * **disableGPS**——是否禁用GPS。如果为YES，则表示禁用，NO则表示开启。默认为开启GPS。
- * **unclickable**——该广告是否不可点击。如果为YES，说明所请求的广告不可被点击；若是NO，则说明请求的广告可以被点击，不过服务器可能也会返回不能被点击的广告。
- * **userSpecAdSize**——开发者自己指定的广告尺寸。如果当前没有此尺寸广告，服务器可能会返回“No ad”的错误码。

Banner动画类型介绍

- * Banner动画类型采用enum ADWO_ANIMATION_TYPE这个枚举。下面介绍各个值
- *
 - ADWO_ANIMATION_TYPE_AUTO——表示由服务器自动控制动画类型
 - ADWO_ANIMATION_TYPE_NONE——表示不做动画，直接切换页面
 - ADWO_ANIMATION_TYPE_PLAIN_MOVE_FROM_LEFT——从左到右的推移
 - ADWO_ANIMATION_TYPE_PLAIN_MOVE_FROM_RIGHT——从右到左的推移
 - ADWO_ANIMATION_TYPE_PLAIN_MOVE_FROM_BOTTOM——从下到上的推移
 - ADWO_ANIMATION_TYPE_PLAIN_MOVE_FROM_TOP——从上到下的推移
 - ADWO_ANIMATION_TYPE_PLAIN_COVER_FROM_LEFT——新广告从左到右推移，并覆盖在老广告条上
 - ADWO_ANIMATION_TYPE_PLAIN_COVER_FROM_RIGHT——新广告从右到左推移，并覆盖在老广告条上
 - ADWO_ANIMATION_TYPE_PLAIN_COVER_FROM_BOTTOM——新广告从下到上移动，并覆盖在老广告条上
 - ADWO_ANIMATION_TYPE_PLAIN_COVER_FROM_TOP——新广告从上到下移动，并覆盖在老广告条上
 - ADWO_ANIMATION_TYPE_CROSS DISSOLVE——淡入淡出切换
 - ADWO_ANIMATION_TYPE_CURL_UP——向上翻页
 - ADWO_ANIMATION_TYPE_CURL_DOWN——向下翻页
 - ADWO_ANIMATION_TYPE_FLIP_FROMLEFT——从左到右翻转
 - ADWO_ANIMATION_TYPE_FLIP_FROMRIGHT——从右到左翻页

全屏动画类型

- * 全屏动画类型采用enum `ADW0_SDK_FULLSCREEN_ANIMATION_TYPE`这个枚举。下面介绍各个值。
- * `ADW0_SDK_FULLSCREEN_ANIMATION_TYPE_AUTO`——表示由页面控制动画类型
`ADW0_SDK_FULLSCREEN_ANIMATION_TYPE_NONE`——不做任何动画，直接出现、消失
`ADW0_SDK_FULLSCREEN_ANIMATION_TYPE_MOVE_FROM_LEFT_TO_RIGHT`——从左到右出现/消失
`ADW0_SDK_FULLSCREEN_ANIMATION_TYPE_MOVE_FROM_RIGHT_TO_LEFT`——从右到左出现/消失
`ADW0_SDK_FULLSCREEN_ANIMATION_TYPE_MOVE_FROM_BOTTOM_TO_TOP`——从底到顶出现/消失
`ADW0_SDK_FULLSCREEN_ANIMATION_TYPE_MOVE_FROM_TOP_TO_BOTTOM`——从顶到底出现/消失
`ADW0_SDK_FULLSCREEN_ANIMATION_TYPE_SCALE_LEFT_RIGHT`——水平方向伸缩
`ADW0_SDK_FULLSCREEN_ANIMATION_TYPE_SCALE_TOP_BOTTOM`——垂直方向伸缩
`ADW0_SDK_FULLSCREEN_ANIMATION_TYPE_CROSS DISSOLVE`——淡入淡出

关键字

- * SDK提供了两个重载的接口用于设置关键字——`BOOL AdwoAdSetKeywords(UiView *adView, NSString *keywords)`以及`BOOL AdwoAdSetKeywords(UiView *adView, NSDictionary *keywords)`
- * 关键字由开发者传入，用于帮助SDK做更精准的广告投放。关键字中可以设置用户性别、年龄、广告类别等。关键字一般是第三方与本SDK有密切合作关系的方能使用。
- * 参数keywords的格式如下：


```
<keywords> => <keyword item>;<keywords>  
                <keyword item>
```



```
<keyword item> => <key>=<value list>
```



```
<value list> => <value>,<value list>  
                <value>
```
- * 例如：`AdwoAdSetKeywords(adView, @"subject=Adwo ad SDK demo;title=Adwo Basic Aggregation Demo;functions=basic embedding,aggregation function,other hidden message invocation,keywords usage");`
- * 而第二种字典参数方式提供了更方便的关键字设置操作，比如上面代码等价于：`AdwoAdSetKeywords(adView, @{@"subject": @"Adwo ad SDK demo", @"title": @"Adwo Basic Aggregation Demo", @"functions": @[@"basic embedding", @"aggregation function", @"other hidden message invocation", @"keywords usage"]});`

详细见附赠的Demo——AdwoSDKAGG。

设置代理对象

- * **BOOL** AdwoAdSetDelegate(UIView *adView, NSObject<AWAdViewDelegate> *delegate)
设置AWAdViewDelegate代理对象
- * 参数adView: 广告对象句柄
参数delegate: AWAdViewDelegate代理。应用开发者应该将展示本SDK Banner的视图控制器实现AWAdViewDelegate代理，并且将视图控制器对象传给此参数。此参数不能为空。注意，此参数不会被retain。
- * 返回: 如果成功，则返回YES，否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。
- * 开发者可以在任何时候调用此接口，来重新设置AWAdViewDelegate代理对象。另外，当开发者使用全屏广告的时候，若当前要销毁与一个或多个全屏广告相关联的视图控制器（即全屏广告的delegate属性指向了该视图控制器），那么必须调用此接口，将这些全屏广告对象的delegate全都置空。除非，全屏广告对象已被销毁。

获取当前广告信息

- * `NSString* AdwoAdGetAdInfo(UIView *adView)`
获取当前广告信息
- * 参数adView: 广告对象句柄
- * 返回: 如果成功, 则返回当前广告信息。如果失败, 则返回空, 开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。
- * 开发者可以在获得AWAdViewDelegate代理的adwoAdViewDidLoadAd通知之后来获取当前广告相关的广告信息。如果为空字符串, 说明当前广告并无特别信息。

获得当前广告的广告ID

* **BOOL** AdwoAdGetCurrentAdID(**UIView** *adView, **int** *pAdID)

获得当前广告的广告ID。

* 参数adView: 广告对象句柄

pAdID: 指向输出的存放广告ID变量的指针。若所得到的广告ID为-1, 说明当前广告请求尚未成功。

* 返回: 如果成功, 则返回YES, 否则返回NO。开发者可以通过调用AdwoAdGetLatestErrorCode接口来获取错误码。

注册社区分享代理

- * `void AdwoAdRegisterSocialShareDelegate(NSObject<AWSocialShareDelegate> *delegate, NSString *socialName)`
这个接口可用于注册多个代理，使得SDK能做制定社区分享。目前，近支持微信（WeChat）分享。
- * 参数delegate：接收负责分享指定信息到指定社区消息的对象代理
参数socialName：这个是本SDK定义好的字符串。@"WeChat"表示此代理用于负责接收分享到微信的消息通知
- * 注意，当调用这个接口之后，参数delegate对象会被retain一次。因此，当你要释放这个delegate的时候，请传空来调用此接口一次，如：
`AdwoAdRegisterSocialShareDelegate(nil, @"WeChat");`使得代理对象delegate能被最终释放掉

社区分享的结果响应

* `void AdwoAdReceiveSocialShareResult(UIView *adView, BOOL result, NSString *socialName);`

当社区分享完成之后，第三方应用程序调用此接口来通知SDK端分享的结果。

* 参数adView：当前广告对象句柄

参数result：分享的结果。YES表示分享成功；NO表示分享失败

参数socialName：社区名，比如"WeChat"表示微信社区

AWAdViewDelegate接口

* – (UIViewController*)adwoGetBaseViewController;

获取基视图控制器。大部分广告都是在用户点击之后弹出SDK自带的Browser，从而可以展示相应的mini site页面。开发者必须提供用于展示SDK所需的各类modal view的基础视图控制器。通常，这个视图控制器是用于展示广告Banner所属视图的控制器。

这个方法必须被实现，并且不能返回为空。[注：这个接口与原来SDK2.5.3版本中的– (UIViewController*)viewControllerForPresentingModalView方法用法一样。]

* – (void)adwoAdViewDidFailToLoadAd:(UIView*)adView;

捕获当前加载广告失败通知。当你所创建的广告视图对象请求广告失败后，SDK将会调用此接口来通知。参数adView指向当前请求广告对象。开发者可以通过错误码接口来查询失败原因。对于全屏广告，开发者可以在此消息方法中重新请求全屏广告，但至少得延迟3秒，详细可参考所有与全屏相关的Demo。

* – (void)adwoAdViewDidLoadAd:(UIView*)adView;

捕获广告加载成功通知。当你广告加载成功时，SDK将会调用此接口。参数adView指向当前请求广告对象。这个接口对于全屏广告展示而言，一般必须实现以捕获可以展示全屏广告的时机。

* – (void)adwoFullScreenAdDismissed:(UIView*)adView;

当全屏广告被关闭时，SDK将调用此接口。一般而言，当全屏广告被用户关闭后，开发者应当释放当前的AWAdView对象，因为它的展示区域很可能发生改变。如果再用此对象来请求广告的话，展示可能会成问题。参数adView指向当前请求广告对象。开发者可以在此消息方法中重新请求全屏广告，但至少得延迟3秒，详细可参考所有与全屏相关的Demo。

AWAdViewDelegate (续)

* – (void)adwoDidPresentModalViewForAd:(UIView*)adView;

当SDK弹出自带的全屏展示浏览器时，将会调用此接口。参数adView指向当前请求广告对象。这里需要注意的是，当adView弹出全屏展示浏览器时，此adView不允许被释放，否则会导致SDK崩溃。

* – (void)adwoDidDismissModalViewForAd:(UIView*)adView;

当SDK自带的全屏展示浏览器被用户关闭后，将会调用此接口。参数adView指向当前请求广告对象。这里允许释放adView对象。

* – (void)adwoUserClosedImplantAd:(UIView*)adView;

用户点击植入性广告的关闭按钮之后，SDK将会发出此消息。参数adView指向当前请求广告对象句柄。在此消息中，开发者可以调用AdwoAdRemoveAndDestroyImplantAd接口。

* – (void)adwoUserClosedBannerAd:(UIView*)adView;

当用户点击Banner广告关闭按钮之后，SDK将会发出此消息。参数adView指向当前请求广告对象句柄。在此消息中，开发者可以调用AdwoAdRemoveAndDestroyBanner接口。

* – (void)adwoAdRequestShouldPause:(AWAdView*)adView

当广告由于外部事件（比如变半屏或弹出Browser等）后需要暂停刷新广告时，此时会通过此接口发出通知。聚合SDK通过此接口通知来暂停刷新当前广告视图，从而不影响用户对广告的体验。

* – (void)adwoAdRequestMayResume:(AWAdView*)adView

当广告接收到外部事件处理完毕后会发出此通知。此时，聚合SDK可以重新开始刷新广告。

AWAdViewDelegate (续)

* – (void)adwoAdETARequestedAndActivated;

当前SDK请求事件触发型广告成功过，通知开发者相应的ETA广告已被开启并且开始做相应的事件真侦听。

* – (void)adwoAdETAReceivedSpecifiedFeatureCode:(int)etaType;

当特定的ETA广告接受到指定的事件触发特征码之后准备展示，SDK将会给应用发送此消息。应用可以在此消息回调中通过使用AdwoAdAddETAViewToSuperview或AdwoAdAddETAViewWithAnimation接口来展示ETA广告。

参数etaType: ETA广告类型，请参考enum ADWOSDK_ETA_TYPE。

* – (void)adwoAdETAAnimationComplete;

当使用AdwoAdAddETAViewWithAnimation接口来添加ETA广告视图，通过实现这个协议接口来处理动画结束后的定制处理。

* – (void)adwoUserClosedETA:(int)etaType;

当使用adwoUserClosedETA接口来添加ETA广告视图，通过实现这个协议接口来处理动画结束后的定制处理。

参数etaType: ETA广告类型，请参考enum ADWOSDK_ETA_TYPE。

AWSocialShareDelegate

* – (void)adwoSocialShareMessage:(NSDictionary*)shareInfo
social:(NSString*)socialName;

分享指定消息到指定社区的代理。当SDK捕获到用户点击分享内容到某个社区之后，将会给先前注册的delegate发送此消息。

* 参数shareInfo: 分享信息。这是一个字典类型，根据不同的分享社区，key的名称也有可能不同。
参数socialName: 社区名，比如"WeChat"等。



Adwo广告SDK隐藏接口

* **BOOL** AdwoAdSetAGGChannel(**UIView** *adView, **enum** ADWOSDK_AGGREGATION_CHANNEL channel)

设置聚合渠道号。目前的聚合渠道号有：ADWOSDK_AGGREGATION_CHANNEL_NONE、ADWOSDK_AGGREGATION_CHANNEL_GUOHEAD、ADWOSDK_AGGREGATION_CHANNEL_ADVIEW、ADWOSDK_AGGREGATION_CHANNEL_MOGO、ADWOSDK_AGGREGATION_CHANNEL_ADWHIRL、ADWOSDK_AGGREGATION_CHANNEL_ADSAGE、ADWOSDK_AGGREGATION_CHANNEL_ADMOB。此接口的使用可参见本文档所附带的DemoBasic。

* 此接口仅提供给聚合SDK使用。

AWAdViewDelegate隐藏接口

* – (void)adwoRequestAdAction:(AWAdView*)adView;

当SDK发出请求后并成功地接受到响应后，将调用此接口作为通知。详细使用请参考本文档附赠的DemoBasicAGG。

* – (void)adwoClickAdAction:(AWAdView*)adView;

当用户点击广告后，SDK将会调用此接口来作为通知。

* – (void)adwoShowAdAction:(AWAdView*)adView;

当广告成功展示后，SDK将会调用此接口来作为通知。

AWAdViewDelegate隐藏接口

(续)

* – (void)adwoAppQuitToBackground:(AWAdView*)adView;

当点击一个广告时导致应用退出后台，SDK会在退到后台时调用代理的此方法。聚合可以可以根据这个通知来处理广告计时或广告切换等操作。

* – (void)adwoAppResumeToForeground:(AWAdView*)adView;

当应用从后台切回到前台后，SDK会调用此方法。

对全屏广告的进一步介绍

- * Adwo SDK的全屏广告就广告展示类型而言分为三种：应用启动全屏广告（简称为开屏全屏广告），插屏广告，后台切换至前台广告。
- * 目前，对于插屏广告而言，有两种全屏广告种类：一类是App Fun，一类是品牌插屏广告。
- * 后面将介绍如何嵌这三种形式的全屏广告。

关于普通插屏广告

- *
 - 1、通过AdwoAdGetFullScreenAdHandle接口获得全屏广告对象，并将参数fsAdShowForm设置为ADWOSDK_FSAD_SHOW_FORM_APPFUN_WITH_BRAND；
 - 2、若有需要，调用AdwoAdSetAdAttributes接口设置属性；
 - 3、若有需要，调用AdwoAdSetKeywords接口设置关键字；
 - 4、调用AdwoAdLoadFullScreenAd接口加载全屏广告；
 - 5、必须实现adwoGetBaseViewController代理；
 - 6、实现adwoAdViewDidLoadAd代理，可以在此方法实现中调用AdwoAdShowFullScreenAd接口来展示全屏广告；
 - 7、实现adwoAdViewDidFailToLoadAd代理，用来处理全屏加载失败的情况。开发者可以在此消息方法中重新请求全屏广告，但至少得延迟3秒；
 - 8、实现adwoFullScreenAdDismissed代理，用来处理全屏广告被关闭的情况。开发者可以在此消息方法中重新请求全屏广告，但至少得延迟3秒。

* 关于插屏广告的代码示例，请参考AdwoSDKFullScreen_AppFun。

关于应用启动全屏广告

- *
 - 1、通过AdwoAdGetFullScreenAdHandle接口获得全屏广告对象，并将参数fsAdShowForm设置为ADWOSDK_FSAD_SHOW_FORM_LAUNCHING；
 - 2、若有需要，调用AdwoAdSetAdAttributes接口设置属性；
 - 3、若有需要，调用AdwoAdSetKeywords接口设置关键字；
 - 4、调用AdwoAdLoadFullScreenAd接口加载全屏广告；这里需要注意，在一开始请求时，如果返回值为ADWO_ADSDK_ERROR_CODE_FS_LAUNCHING_AD_REQUESTING，那么说明当前SDK正在请求加载开屏全屏广告，此时可以直接执行后续操作；如果返回值为ADWO_ADSDK_ERROR_CODE_SUCCESS，那么说明可直接展示全屏广告。
 - 5、必须实现adwoGetBaseViewController代理；
 - 6、实现adwoAdViewDidLoadAd代理，在这里面需要进行判断——如果先前调用的AdwoAdLoadFullScreenAd的返回值为ADWO_ADSDK_ERROR_CODE_SUCCESS，那么可直接调用AdwoAdShowFullScreenAd方法来展示开屏全屏广告；否则的话执行后续操作；
 - 7、实现adwoAdViewDidFailToLoadAd代理，用来处理全屏加载失败的情况；
 - 8、实现adwoFullScreenAdDismissed代理，用来处理全屏广告被关闭的情况。
- * 关于开屏全屏广告的代码示例，请参考AdwoSDKDemo_launchingAd。

关于后台切换至前台全屏广告

- * 1、通过AdwoAdGetFullScreenAdHandle接口获得全屏广告对象，并将参数fsAdShowForm设置为ADWOSDK_FSAD_SHOW_FORM_GROUND_SWITCH；
- 2、若有需要，调用AdwoAdSetAdAttributes接口设置属性；
- 3、若有需要，调用AdwoAdSetKeywords接口设置关键字；
- 4、调用AdwoAdLoadFullScreenAd接口加载全屏广告；
- 5、实现adwoAdViewDidFailToLoadAd代理，用来处理全屏加载失败的情况。开发者可以在此消息方法中重新请求全屏广告，但**至少得延迟3秒**；
- 6、必须实现adwoGetBaseViewController代理；若当前为自动展示模式，则不需要以下步骤
- 7、实现adwoAdViewDidLoadAd代理，在这里接收全屏广告已加载完的通知；
- 8、实现adwoFullScreenAdDismissed代理，用来处理全屏广告被关闭的情况。开发者可以在此消息方法中重新请求全屏广告，但**至少得延迟3秒**；
- 9、订阅UIApplicationDidBecomeActiveNotification通知，在这个通知回调方法中判断全屏广告先前是否已经加载好，如果加载好则调用AdwoAdLoadFullScreenAd接口展示全屏。

* 关于后台切换至前台的全屏广告的详细使用，请参考AdwoSDKFullScreen的Demo。

关于附赠Demo的介绍

- * 由于XCode4.5中对于相同Bundle ID的应用，系统无法直接覆盖。因此必须先删除原先的应用才能加载运行新的带有与之前相同Bundle ID的应用。这里所有Demo的Bundle ID都是相同的，因此运行完一个之后必须先删除才能运行另一个。
- * AdwoSDKAGG目录下的Demo面向聚合平台以及想自己做聚合的第三方开发者。它介绍了关于聚合所需要的隐藏接口的使用方法，从而开发者可以更精确地对安沃的广告做相关统计并能取得更好的展示效果。
- * AdwoSDKBasic目录下的Demo提供了本SDK最基本的使用方法。如果开发者使用一些比较复杂的引擎（如Cocos2D-X、Unity3D等），那么参考这个Demo将是非常合适的。
- * AdwoSDKFullScreen目录下的Demo提供了一种比较好的全屏展示策略。AdwoFSAdcontainer可以直接拿来使用，它提供了同时请求应用启动全屏、一般插屏以及后台切换到前台的全屏广告。由于在每请求一个全屏广告之间会有20秒的请求间隔，因此当看到调试日志中显示了插屏广告加载完之后再点击Show按钮。而如果应用启动全屏加载完之后，从后台关闭应用，再打开就能即可看到。

关于附赠Demo的介绍（续）

- * AdwoSDKFullScreen_Basic目录下的Demo介绍了如何去展示插屏全屏广告。这里详细地给出了竖屏状态下的对全屏广告的展示。同时，这个Demo也展示了对于同时支持横竖屏的应用如何适当地处理屏幕旋转。这个处理也适用于Banner。
- * AdwoSDKDemo_launchingAd介绍了如何去展示开屏全屏广告。这里在viewDidLoad方法中将会先请求加载广告。如果此时广告不存在，则继续执行后续操作；否则，直接进行展示。
- * AdwoSDKNavigation的Demo主要描述了在多视图控制器场合下的Banner广告以及全屏广告对象的加载、消除、展示等管理。如果要提升应用的稳定性，对广告对象的适当管理是非常重要的。
- * ImplantAdSample描述了对植入性广告嵌入的方法。
- * Cocos2DTest展示了如何在Cocos2DTest引擎中以横屏模式嵌入Adwo iOS SDK。
- * ARCDemo提供了开发者使用ARC工程项目时嵌入Adwo广告SDK的方法。