DU Ad Platform_SDK for iOS接入手册

Version: DUAd_iOS_SDK_1.1.2

1. 概述

本文档描述如何在 iOS 应用中接入来自DAP开发者平台的 DU Ad Platform_SDK 产品。

DAP开发者平台可以为 iOS 应用提供广告服务。DU Ad Platform_SDK 是DAP开发者平台中用来提供原生广告的一款产品。

1.1 读者对象

本文档面向的读者是 iOS 应用的开发者。

1.2 前提

DU Ad Platform_SDK 目前支持 iOS8（含）以上的系统版本。本版本DU Ad Platform_SDK支持FacebookAudienceNetwork 4.99.0及以上版本。

2. 接入流程

DU Ad Platform_SDK 的接入流程如下：

- 原生广告接入流程
    1. 申请广告 ID
    2. 导入 DU Ad Platform_SDK 工程包
    3. 初始化 DU Ad Platform_SDK
    4. 原生广告接入
    5. 完成接入
- 插屏广告接入流程
    1. 申请广告 ID
    2. 导入 DU Ad Platform_SDK 工程包
    3. 初始化 DU Ad Platform_SDK
    4. 插屏广告接入
    5. 完成接入

3. 获取身份

本章描述 DU Ad Platform_SDK 接入过程中需要的四个身份：APP_ID，广告位 ID，Facebook Placement_ID，Admob Placement_ID。

3.1 APP_ID

1. 定义

   APP_ID 是开发者的应用在广告平台的唯一标识。

2. 获取方式

访问DAP开发者平台进行申请。

3. 代码

   `@License`

3.2 广告位 ID

1. 定义

   广告位 ID 是开发者平台上广告所在的广告位置的标识。开发者可以创建多个广告位。

2. 获取方式

   访问DAP开发者平台进行申请。

3. 代码

   `@pid`

3.3 Facebook Placement_ID（可选）

1. 定义

   Facebook Placement_ID 是 Facebook 广告所在广告位置的标识。使用 DAP 进行聚合 Facebook 广告时才需要 Facebook Placement_ID。

2. 获取方式

   访问 Facebook 开发者平台进行申请。

3. 代码

   `@fbids`

3.4 Admob Placement_ID（可选）

1. 定义

   Admob Placement_ID 是 Admob 广告所在广告位置的标识。使用 DAP 进行聚合 Admob 广告时才需要 Admob Placement_ID。

2. 获取方式

   访问 Admob 开发者平台进行申请。

3. 代码

   `@amid`

4. 加载与配置

本章描述在 iOS 应用中如何加载 DU Ad Platform_SDK 的压缩包。

请严格按照本章进行配置，否则有可能会出现运行异常。

4.1 下载 DU Ad Platform_SDK 的压缩包

4.2 解压 DU Ad Platform_SDK 的压缩包

解压后内容如下：

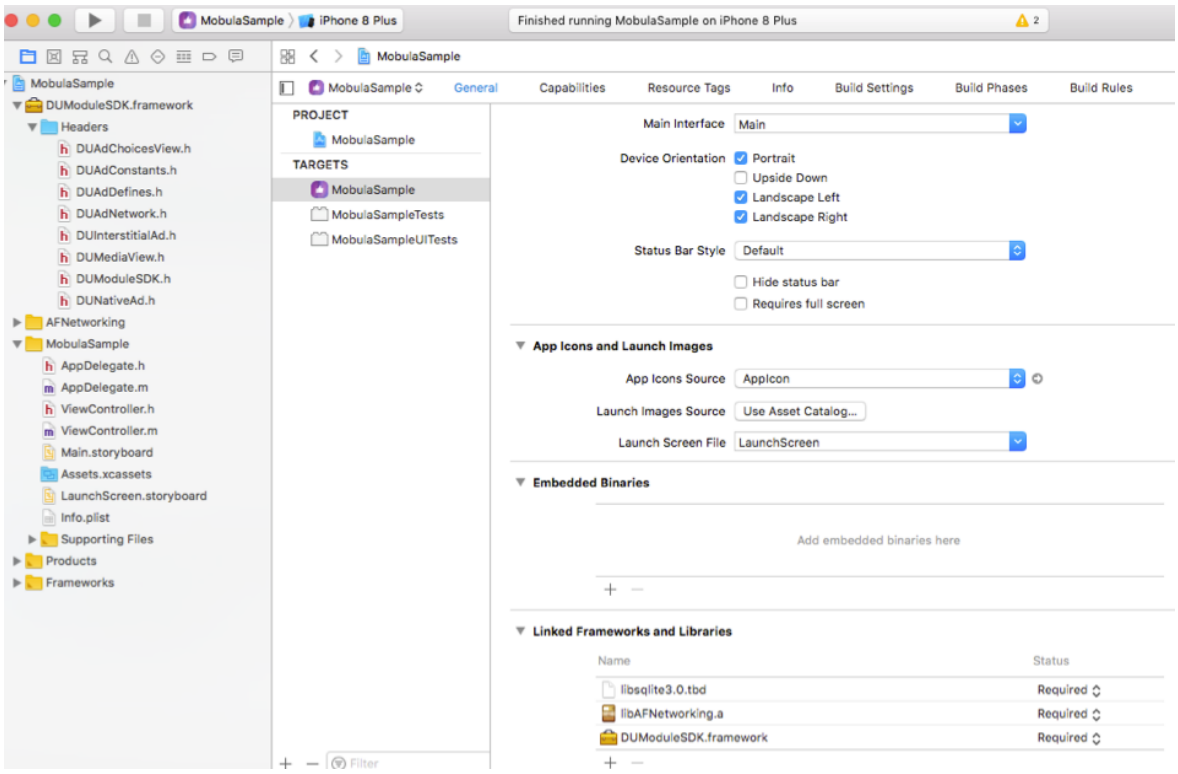
- 文件DUModuleSDK.framework:

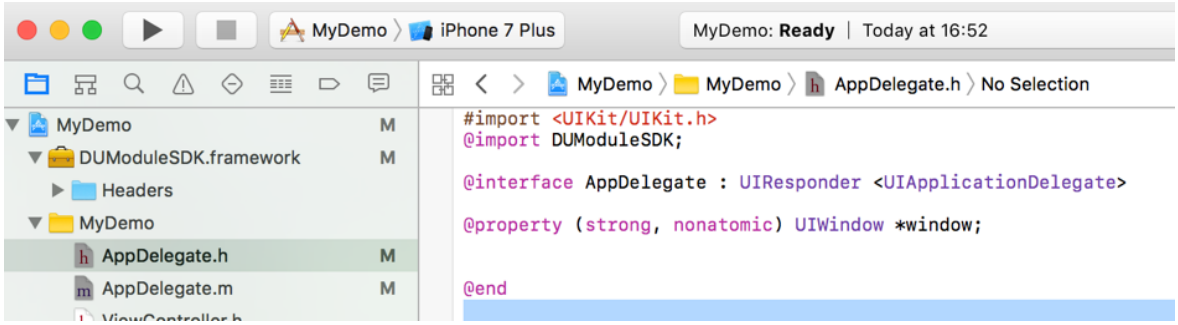
  DU Ad Platform_SDK 的framework 包

- 子文件夹MobulaSample:

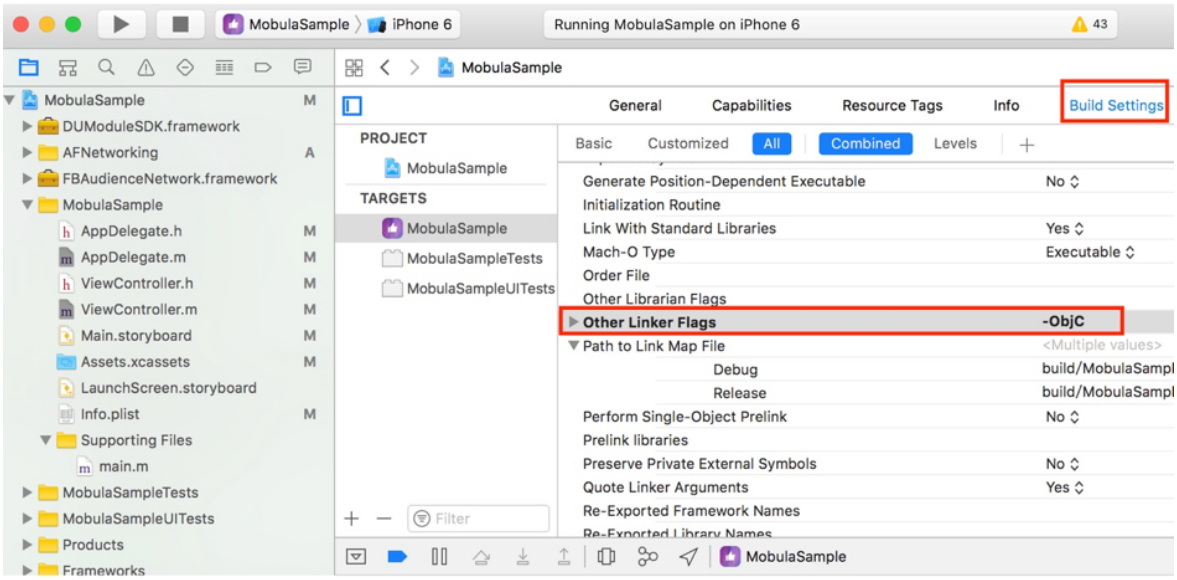
  该文件夹存放使用 DU Ad Platform_SDK 过程中的示例程序。本文档中所有接口都可以在 MobulaSample 中找到对应的使用示例。

4.3 使用 Xcode 导入DU Ad Platform_SDK

1. 将 libsqlite3.0.tbd 添加到到工程内
2. 将 DUModuleSDK_frameworks 文件夹下的所有内容拖拽到工程内. 选择 "Copy Items if needed" 并点击 "finish"。此时 "Linked Framework and Libraries" 中应显示所有需要的 framework。
3. 

4. 在需要接入广告的 .h 文件声明 DUModuleSDK


5. 在工程 “Build Setting” 处，将 “-ObjC” 标志加入 “Other Linker Flags”



5. 初始化

在完成 DU Ad Platform_SDK 接入操作之前，IOS应用首先需要对 DU Ad Platform_SDK 做初始化。

没有进行初始化的广告位 id 无法拉取广告。

1. 创建 Json 文件或字符串，将 Placement_ID 与广告位 ID 建立对应关系。具体格式见代码示例。

   注：如果开发者某广告位不需要 Facebook 广告, 请将该广告位对应的 "fbids" 部分删除。

2. 在 AppDelegate.m 文件，调用`[DUAdNetwork initWithConfigDic: withLicense:]`接口

   接口说明：

   > (void) initWithConfigDic: (NSDictionary*) aDic withLicense: (NSString*) aStr;

参数	说明
(NSDictionary*)aDic	Placement_ID 与广告位 ID 的对应关系
(NSString*)aStr	申请的 APP_ID

3. 设置 Log 等级，可输出更多详细信息。

   注：建议在正式版本移除该方法。

   接口说明：

   > (void) setLogLevel: (DUAdLogLevel) aLevel;

参数	说明
DUAdLogLevelDebug	调试模式，能够输出完整调试和错误信息

   代码示例：

```
NSDictionary *config=@{
  @"native" :
  @[
    @{
      @"pid" : @"YOUR_NATIVE_AD_PLACEMENT_ID" ,
      @"fbids" : @[@"YOUR_FBID"] ,
    },
    @{
      @"pid" : @"YOUR_INTERSTITIAL_AD_PLACEMENT_ID",
      @"fbids" : @[@"YOUR_FBID"] ,
    }
  ]
};

[DUAdNetwork initWithConfigDic:config
withLicense:@"YOUR_DAP_APP_LICENSE"];
[DUAdNetwork setLogLevel:DUAdLogLevelDebug];
```

6. 控制用户信息获取许可状态

此配置为针对GDPR做出的修改，适用于需要进行用户信息获取许可状态配置的地区，为可选配置。

6.1 用户信息获取许可状态的设置接口

建议在初始化时调用该接口。

接口说明：

> (void)setConsentStatus:(BOOL)userConsent;

参数	说明
True	获得了用户的许可时传入，正常使用广告功能。
False	未取得用户的许可时传入，然后停用所有广告功能。

6.2 用户信息获取许可状态的获取接口

接口说明：

> (BOOL)getConsentStatus;

获取当前用户信息获取许可状态，允许收集用户信息则返回 True，否则返回False。

7. 获取原生广告数据

本章描述如何获取广告数据。包括构造广告数据类接口，注册广告数据监听回调，和获取广告数据接口三个部分。

7.1 声明原生广告对象

在 View Controller头文件，加入 DUModuleSDK，声明 ViewController 实现 DUNativeAdDelegate 接口，并创建 DUNativeAd 实例对象。

步骤如下：

1. 构造原生广告类

   创建原生广告对象必须指定对应的广告位 ID 。不同的广告位会获取到不同的广告数据。

2. 设置广告缓存个数

   广告缓存个数可以设置 1-5 个。推荐不设置广告缓存个数。如果不设置或者设置无效值，会使用默认缓存：1个。

   注：此方法只在通过 DU Ad Platform 聚合其他渠道时生效。

接口说明：

> (nonnull instancetype)initWithPlacementID:(nonnull NSString*)placementID;
>
> (nonnull instancetype)initWithPlacementID:(nonnull NSString*)placementID cacheSize:(NSInteger)aSize;

参数	说明
(nonnull NSString*)placementID	广告位 ID，该 pid 注册在 Json 的 native 数组中
(NSInteger)aSize	缓存广告个数

代码示例：

```
#import <UIKit/UIKit.h>
@import DUModuleSDK; /*DU SDK*/
#import <FBAudienceNetwork/FBAudienceNetwork.h> /*FacebookSDK*/
@interface ViewController : UIViewController <DUNativeAdDelegate>
@property (strong, nonatomic) DUNativeAd *nativeAd;
@end

-(void)viewDidLoad
{
  [super viewDidLoad];
  DUNativeAd  _nativeAd = [[DUNativeAd alloc] initWithPlacementID: @"88888"
cacheSize:1];
  _nativeAd.delegate = self;
}
```

7.2 设置原生广告 delegate

请注册接收广告数据的回调，然后获取广告数据。

广告数据获取成功或失败，点击事件的响应是通过回调接口返回的。此过程与广告数据获取过程是异步的，不会阻塞开发者的线程。

接口说明：

> @protocol DUNativeAdDelegate <NSObject>

```
@protocol DUNativeAdDelegate <NSObject>
@optional

/*获取广告成功时回调*/
- (void)nativeAdDidLoad:(nonnull DUNativeAd *)nativeAd;

/*广告展示事件回调*/
- (void)nativeAdWillLogImpression:(nonnull DUNativeAd *)nativeAd;

/*获取广告失败回调*/
- (void)nativeAd:(nonnull DUNativeAd *)nativeAd didFailWithError:(nonnull
NSError *)error;

/*广告点击回调*/
- (void)nativeAdDidClick:(nonnull DUNativeAd *)nativeAd;

/*移交控制权时回调（广告被点击后，将展示一个modal view。用户点击modal view中的取消按钮时，发出该回调，并将应用控制权移交）*/
- (void)nativeAdDidFinishHandlingClick:(nonnull DUNativeAd *)nativeAd;

@end
```

获取广告数据失败的错误码及含义：

常量	错误码	说明
NETWORK_ERROR_CODE	1000	客户端网络错误
NO_FILL_ERROR_CODE	1001	没有获取到广告数据
LOAD_TOO_FREQUENTLY_ERROR_CODE	1002	请求接口过频繁
IMPRESSION_LIMIT_ERROR_CODE	1003	超出展示限制
SERVER_ERROR_CODE	2000	服务器错误
MISSING_PROPERTIES_CODE	2002	属性缺失，建议检查JSON
TIME_OUT_CODE	3000	获取广告数据等待时间超时
UNKNOW_ERROR_CODE	3001	未知错误
NO_CHANNEL_ERROR_CODE	3002	无可用渠道
NO_USER_CONSENT_ERROR_CODE	4000	用户信息获取未受到许可

7.3 获取原生广告数据接口

开发者可根据自己产品的需求，选择时机获取广告数据。

接口说明：

> (void) fillAd

调用`fillAd`接口可以提前缓存广告，在`loadAd`广告时可以更快获取。建议在广告展示的前置场景调用该方法。

注：广告数据会缓存到客户端内存中，不会缓存广告的图片数据，只会缓存图片的Url地址，缓存数据量小。

> (void) loadAd

异步获取广告对象数据，没有缓存时会进行广告请求。

建议在使用`loadAd`后再次调用`fillAd`接口进行广告缓存。

> (DUNativeAd*) getCacheAd

同步获取广告对象数据。可以循环拿取，一直到广告缓存为0。

在使用该接口展示广告时，请进行缓存非空判断，避免缓存池为空导致空指针。

建议在使用`getCacheAd`后再次调用`fillAd`接口进行广告缓存。

> (BOOL) isHasCached

获取当前是否有广告缓存，有缓存则返回 True。

代码示例：

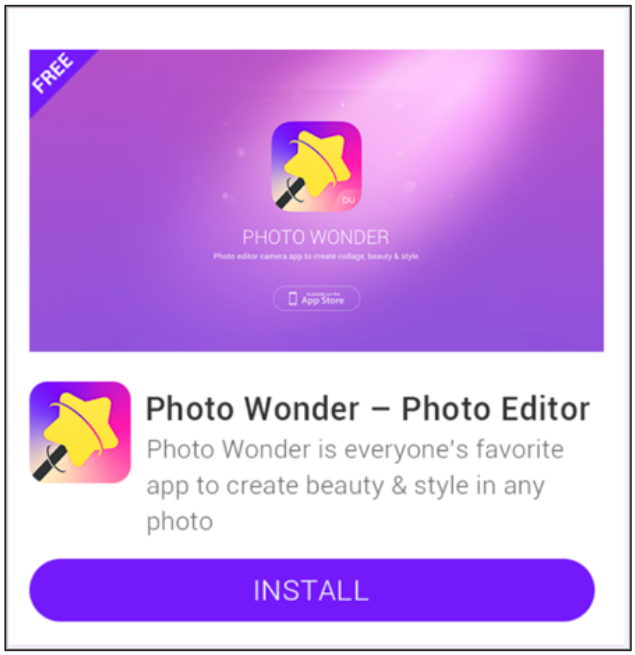
```
@interface ViewController ()<DUNativeAdDelegate>
{
  DUNativeAd *_nativeAd;
  DUMediaView *_mediaView;
  DUAdChoicesView *_chiocеView;
}

- (IBAction)loadAd:(id)sender
{
  NSLog(@"load Native ad.");
  [_nativeAd loadAd];
}

- (void)nativeAdDidLoad:(DUNativeAd *)nativeAd
{
  NSLog(@"Native ad was loaded.");
  [self nativeAdDisPlay:nativeAd];
}

- (void)nativeAdWillLogImpression:(DUNativeAd *)nativeAd
{
  NSLog(@"Native ad impression is being captured.");
}

- (void)nativeAd:(DUNativeAd *)nativeAd didFailWithError:(NSError *)error
{
  NSLog(@"Native ad failed to load with error: %@", error);
}

- (void)nativeAdDidClick:(DUNativeAd *)nativeAd
{
  NSLog(@"Native ad was clicked.");
}

- (void)nativeAdDidFinishHandlingClick:(DUNativeAd *)nativeAd
{
  NSLog(@"Native ad did finish click handling.");
}
```

8. 原生广告数据介绍

本章描述广告数据的构成元素及构成元素的获取接口。

8.1 构成元素

广告数据的构成元素包括图标，标题，CTA 按钮，宣传文案，评价和宣传图。



8.2 数据获取接口

- 图标获取接口

  > @property (nonatomic, strong, readonly, nullable) NSString *iconUrl

  返回广告图标的 Url 地址。

- 标题获取接口

  > @property (nonatomic, copy, readonly, nullable) NSString *title

  返回标题文案。广告中必须包含一个标题。 请保留至少 20 个字符的空间用来显示标题，可以用省略号代替超出的文本。

- CTA 按钮获取接口

  > @property (nonatomic, copy, readonly, nullable) NSString *callToAction

  返回 CTA 按钮文案。广告中必须包含一个触发按钮。

  请不要缩短或改变按钮文案。按钮文案的最大字符长度个数：25。

- 宣传文案获取接口

> @property (nonatomic, copy, readonly, nullable) NSString *shortDesc

返回广告的宣传文案。需确保有 72 个字符可以被显示。

如果广告区域不足以显示 72 个字符，建议不要在广告中添加宣传文案，或者使用滚动文本效果，让全部宣传文案能够被显示。

- 宣传图获取接口

> @property (nonatomic, strong, readonly, nullable) NSString *imgeUrl

返回广告宣传图的 Url 地址，当返回值为 NULL 时，当前广告数据中不含宣传图。

广告中可以添加宣传图片，促进用户点击广告的欲望。可以缩放和裁剪宣传图的一部分，但请不要扭曲和改变它。宣传图的大小通常是：796*416像素。

- DuAdChoicesView

该 View 是 Facebook 原生广告返回的 AdChoices 角标。使用 Facebook 原生广告时必须添加的元素，非 Facebook 原生广告不用添加。

构造代码示例:

```
DUAdChoicesView *choiceView = [[DUAdChoicesView alloc]
initWithNativeAd:nativeAd expandable:NO];
[self.adChoicesView addSubview:_chiocеView];
```

- 广告渠道类型

> @property (nonatomic, assign, readonly) DUAdChannelType adChannelType

返回值	说明
DUAdChannelTypeUnknow	未知广告渠道
DUAdChannelTypeDownload	DAP 广告
DUAdChannelTypeFacebook	Facebook广告

9. 注册原生广告 View 监听

DU Ad Platform_SDK会自动统计广告的展示和被点击次数，开发者必须注册广告可点击区域视图的监听。

接口说明：

> (void)registerViewForInteraction:(UIView *)view mediaView:(nonnull id)mediaView iconView:(nullable id)iconImageView viewController:(nullable UIViewController *)viewController;
>
> (void)registerViewForInteraction:(UIView *)view mediaView:(nonnull id)mediaView iconView:(nullable id)iconImageView viewController:(nullable UIViewController *)viewController clickableViews:(nullable NSArray<UIView *> *)clickableViews;

参数	说明
(UIView *)view mediaView	媒体view
(nullable id)iconImageView	图标view，可使用FACE BOOK定制图标view
(UIViewController *)viewController	更细致的子view
(NSArray<UIView *> *)clickableViews	广告内容中可点击的view

取消view注册接口：

> (void)unregisterView

10. 获取原生list广告数据

建议需要同时展示多条原生广告时使用此方法。

10.1 声明原生list广告对象

在 View Controller头文件，加入 DUModuleSDK，声明 ViewController 实现 DUNativeAdsManagerDelegate, 接口，并创建 DUNativeAdsManager实例对象。

步骤如下：

1. 构造原生list广告类

   创建原生list广告对象必须指定对应的广告位 ID 。不同的广告位会获取到不同的广告数据。

2. 设置广告缓存个数

   广告缓存个数可以设置 1-10 个。如果不设置或者设置无效值，会使用默认缓存：10个。

接口说明：

> (instancetype)initWithPlacementID:(NSString*)placementID;
>
> (instancetype)initWithPlacementID:(NSString *)placementID

```
                                                cacheSize:(NSInteger)aSize;
```

参数	说明
(NSString*)placementID	广告位ID，该placement 需要注册在 Json 的 list 数组中，参见第5章。
(NSInteger)aSize	缓存广告个数，默认值：10。

代码示例：

```
#import <UIKit/UIKit.h>
@import DUModuleSDK;

@interface ViewController : UIViewController <DUNativeAdsManagerDelegate,
DUNativeAdDelegate>
    @property (strong, nonatomic)DUNativeAdsManager *adsMgr;
@end


- (void)viewDidLoad {
    [super viewDidLoad];
/*!
 @method
 @abstract
 This is a method to initialize a DUNativeAdsManager object matching the
given placement id. This will use 10 as default cache size.
 @param placementID The id of the ad placement. You can create your
placement id from Mobula developers page.
 */

_adsMgr = [[DUNativeAdsManager alloc] initWithPlacementID:@"10035"];
    _adsMgr.delegate = self;
}
```

10.2 设置原生list广告 delegate

请注册接收广告数据的回调，然后获取广告数据。广告数据获取成功或失败是通过回调接口返回的。此过程与广告数据获取过程是异步的，不会阻塞开发者的线程。

接口说明：

```
@protocol DUNativeAdsManagerDelegate
```

```
@protocol DUNativeAdsManagerDelegate <NSObject>
@optional

/*获取广告成功时回调*/
- (void)nativeAdsLoaded:(NSArray<DUNativeAd *> *)nativeAds;

/*获取广告失败回调*/
- (void)nativeAdsFailedToLoadWithError:(NSError *)error;

@end
```

获取广告数据失败的错误码及含义：

常量	错误码	说明
NETWORK_ERROR_CODE	1000	客户端网络错误
NO_FILL_ERROR_CODE	1001	没有获取到广告数据
LOAD_TOO_FREQUENTLY_ERROR_CODE	1002	请求接口过频繁
IMPRESSION_LIMIT_ERROR_CODE	1003	超出展示限制
SERVER_ERROR_CODE	2000	服务器错误
MISSING_PROPERTIES_CODE	2002	属性缺失，建议检查JSON
TIME_OUT_CODE	3000	获取广告数据等待时间超时
UNKNOW_ERROR_CODE	3001	未知错误
NO_CHANNEL_ERROR_CODE	3002	无可用渠道
NO_USER_CONSENT_ERROR_CODE	4000	用户信息获取未受到许可

10.3 获取原生list广告数据接口

开发者可根据自己产品的需求，选择时机获取广告数据。

接口说明：

> (void) fillAds

调用`fillAds`接口可以提前缓存广告，在`loadAds`广告时可以更快获取。建议在广告展示的前置场景调用该方法。注：广告数据会缓存到客户端内存中，不会缓存广告的图片数据，只会缓存图片的Url地址，缓存数据量小。

> (void) loadAds

异步获取广告对象数据，没有缓存时会进行广告请求。

建议在使用`loadAds`后再次调用`fillAds`接口进行广告缓存。

> (NSArray<DUNativeAd *> *)getCacheAds

同步获取广告对象数据。可以循环拿取，一直到广告缓存为0。

在使用该接口展示广告时，请进行缓存非空判断，避免缓存池为空导致空指针。

建议在使用`getCacheAds`后再次调用`fillAds`接口进行广告缓存。

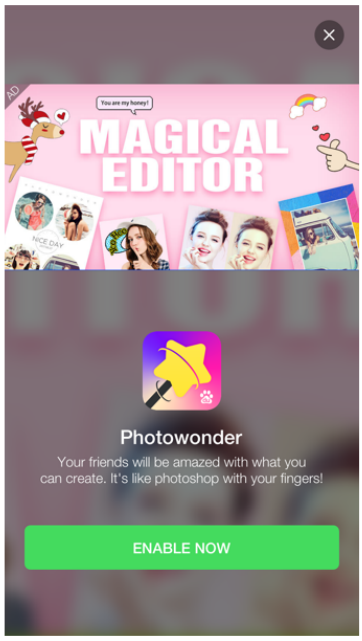
> (BOOL) isHasCached

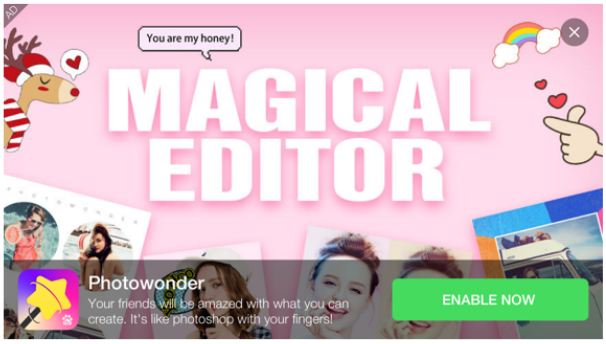
获取当前是否有广告缓存，有缓存则返回 True。

代码示例：

```
@interface NativeListAdViewController () < DUNativeAdsManagerDelegate,
DUNativeAdDelegate> {
    DUNativeAdsManager *adsMgr;
    NSMutableArray<DUNativeAd *> *m_adArray;
}

-(IBAction)load:(id)sender {
    NSLog(@"Native list ads were loaded. ");
    [adsMgr loadAds];
}

 - (void)nativeAdsFailedToLoadWithError:(nonnull NSError *)error {
    NSLog(@"Native list ads failed to load with error: %@", error);
}

- (void)nativeAdsLoaded:(nonnull NSArray<DUNativeAd *> *)nativeAds {
    NSInteger adCount = 0;
    if (nativeAds) {
        adCount = [nativeAds count];
    }
    if (adCount > 0) {
        [m_adArray addObjectsFromArray:nativeAds];
        [self showAds];
    }
    NSLog(@"nativeAdsLoaded  adCount : %ld, and cache : %@", adCount,
nativeAds);
}
```

11. 插屏广告使用





11.1 声明广告对象

在 View Controller头文件，加入 DUModuleSDK，声明 ViewController 实现 DUInterstitialAdDelegate 接口，并创建 DUInterstitialAd 实例对象。

步骤如下：

1. 构造插屏广告类

   创建插屏广告对象必须指定对应的广告位 ID 。不同的广告位会获取到不同的广告数据。

2. 设置广告缓存个数

   广告缓存个数可以设置 1-5 个。推荐不设置广告缓存个数。如果不设置或者设置无效值，会使用默认缓存：1个。

   注：此方法只在通过 DU Ad Platform 聚合其他渠道时生效。

接口说明：

> (nonnull instancetype)initWithPlacementID:(nonnull NSString*)placementID;
>
> (nonnull instancetype)initWithPlacementID:(nonnull NSString*)placementID cacheSize:(NSInteger)aSize;

参数	说明
(nonnull NSString*)placementID	广告位 ID，该 pid 注册在 Json 的 native 数组中
(NSInteger)aSize	缓存广告个数

如果需要聚合Admob插屏，请参考如下格式配置Json 文件或字符串。

```
{
  @"amaid" : "YOUR_ADMOB_APP_ID",
  @"native" :
      @[
         @{
            @"pid" :@"YOUR_DAP_INTERSTITIAL_PLACEMENT_ID",
            @"fbids" : @ [@"YOUR_FACEBOOK_INTERSTITIAL_PLACEMENT_ID"],
            @"amid" : @"YOUR_ADMOB_INTERSTITIAL_PLACEMENT_ID",
          }
       ]
}
```

代码示例：

```
#import <UIKit/UIKit.h>
@import DUModuleSDK; /*DU SDK*/
#import <FBAudienceNetwork/FBAudienceNetwork.h> /*FacebookSDK*/
@interface ViewController : UIViewController <DUInterstitialAdDelegate>
@property (strong, nonatomic) DUInterstitialAd *interstitialAd;
@end

-(void)viewDidLoad
{
  [super viewDidLoad];
  DUInterstitialAd _interstitialAd= [[DUInterstitialAd alloc]
initWithPlacementID: @"88888" cacheSize:1];
  _interstitialAd.delegate = self;
}
```

11.2 设置插屏广告 delegate

请注册接收广告数据的回调，然后获取广告数据。

广告数据获取成功或失败，点击事件的响应是通过回调接口返回的。此过程与广告数据获取过程是异步的，不会阻塞开发者的线程。

接口说明：

> @protocol DUInterstitialAdDelegate <NSObject>

```
@protocol DUInterstitialAdDelegate <NSObject>
@optional

/*获取广告成功时回调*/
- (void)interstitialAdDidLoad:(DUInterstitialAd *)interstitialAd;
```

```
/*广告展示事件回调*/
- (void)interstitialAdWillLogImpression:(DUInterstitialAd *)interstitialAd;

/*获取广告失败回调,错误码及含义见7.2*/
- (void)interstitialAd:(DUInterstitialAd *)interstitialAd didFailWithError:
(NSError *)error;

/*广告点击回调*/
- (void)interstitialAdDidClick:(DUInterstitialAd *)interstitialAd;

/*广告即将关闭回调*/
- (void)interstitialAdWillClose:(DUInterstitialAd *)interstitialAd;

/*广告关闭成功回调*/
- (void)interstitialAdDidClose:(DUInterstitialAd *)interstitialAd;
@end
```

11.3 获取广告数据接口

开发者可根据自己产品的需求，选择时机获取广告数据。

接口说明：

> (void) fillAd

调用`fill()`接口可以提前缓存广告，在`load()`广告时可以更快获取。建议在广告展示的前置场景调用该方法。

注：广告数据会缓存到客户端内存中，不会缓存广告的图片数据，只会缓存图片的Url地址，缓存数据量小。

> (void) loadAd

异步获取广告对象数据，没有缓存时会进行广告请求。

建议在使用`load()`后再次调用`fill()`接口进行广告缓存。

> (DUInterstitialAd*) getCacheAd

同步获取广告对象数据。可以循环拿取，一直到广告缓存为0。

在使用该接口展示广告时，请进行缓存非空判断，避免缓存池为空导致空指针。

建议在使用`get()`后再次调用`fill()`接口进行广告缓存。

> (BOOL) isHasCached

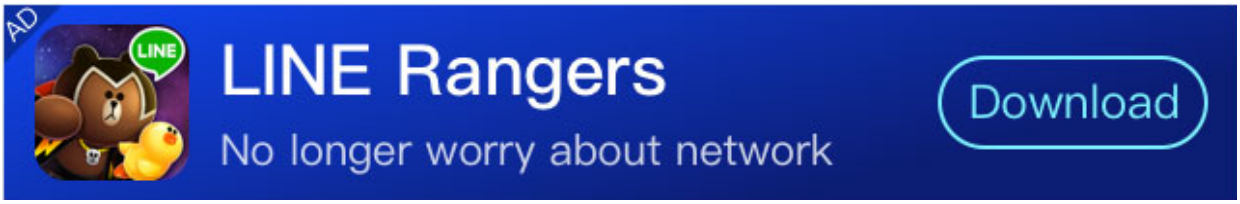
获取当前是否有广告缓存，有缓存则返回 True。

> (BOOL)showAdFromRootViewController:(nullable UIViewController *) rootViewController

展示插屏广告

12. 横幅广告使用

横幅广告示例：



12.1 声明广告对象

在 View Controller 头文件，加入 DUModuleSDK，声明 ViewController 实现 DUBannerAdViewDelegate 接口，并创建 DUBannerAdView 实例对象。

注：横幅广告目前不支持聚合其他渠道。

步骤如下：

1. 构造横幅广告对象

   在 ViewController 的 `viewDidLoad` 方法中初始化横幅广告对象。

2. 设置横幅广告位置

接口说明：

> (instancetype)initWithPlacementID:(NSString *)placementID adSize:
> (DUBannerAdSize)adSize rootViewController:(nullable UIViewController *)viewController;

参数	说明
(NSString *)placementID	广告位 ID，该 pid 注册在 Json 的 native 数组中
(DUBannerAdSize)adSize	广告尺寸。目前支持的值为：`kDUBannerAdSize320x50`：尺寸 320x50
(nullable UIViewController *)viewController	展示横幅广告的 View controller

代码示例：

```
#import <UIKit/UIKit.h>
@import DUModuleSDK; /*DU SDK*/

@interface ViewController : UIViewController <DUBannerAdViewDelegate>
    @property (strong, nonatomic) DUBannerAdView *bannerAd;
@end

- (void)viewDidLoad {
    [super viewDidLoad];
    _bannerAd=[[DUBannerAdView alloc] initWithPlacementID:PID_BANNER
adSize: kDUBannerAdSize320x50 rootViewController:self];
    _bannerAd.frame = CGRectMake(x-coordinate, y-coordinate,
adView.bounds.size.width, adView.bounds.size.height);
    _bannerAd.delegate=self;
    [self.view addSubview:_bannerAd];
}
```

12.2 设置横幅广告 delegate

请注册接收广告数据的回调，然后获取广告数据。

接口说明：

> @protocol DUBannerAdViewDelegate <NSObject>

```
@protocol DUBannerAdViewDelegate <NSObject>
@optional

/*获取广告成功时回调*/
- (void)adViewDidLoad:(DUBannerAdView *)bannerAdView;

/*获取广告失败回调,错误码及含义见7.2*/
- (void)adView:(DUBannerAdView *)bannerAdView didFailWithError:(NSError
*)error;
@end
```

12.3 获取广告接口

广告数据获取过程是异步的，不会阻塞开发者的线程。

接口说明：

> (void)loadAd;