

nendSDK for iOS

ver 2.5.7

設定ガイド

2015年1月22日

更新履歴

| バージョン 公開日付 | 更新内容 |
|--------------------------|---|
| iOS_ver 1.2.0 2011/08/16 | - |
| iOS_ver 1.2.1 2011/08/24 | シミュレータ向け不具合の修正 |
| iOS_ver 1.2.2 2012/03/27 | デバイス ID (UDID) 取得の停止 サイズ変更 (320×48) から(320×50)の View を生成するように変更 |
| iOS_ver 1.2.3 2012/04/11 | 広告取得時に不必要的メモリをリリースしない不具合の修正 |
| iOS_ver 1.3.0 2012/06/25 | 広告受信成功通知の追加 広告受信エラー通知の追加 定期ロード中断の追加 定期ロード再開の追加 |
| iOS_ver 1.3.1 2012/08/31 | WebView タイプの場合ローテーションが管理されない不具合修正 |
| iOS_ver 1.3.2 2012/09/20 | iOS SDK 6 & iPhone 5 (armv7s) 対応 |
| iOS_ver 2.0.0 2013/04/02 | ターゲティング広告配信及びオプトアウト機能の実装 広告識別子 Advertising Identifier (IDFA) 利用開始 ログ出力設定プロパティ追加 その他不具合修正 |
| iOS_ver 2.0.1 2013/04/09 | 特定のライブラリ使用時に重複エラーが起きる問題に対応 |
| iOS_ver 2.0.2 2013/04/11 | 特定のライブラリ使用時に重複エラーが起きる問題に追加対応 AdSupport.framework の Link 設定に関する注意事項を追記 |
| iOS_ver 2.1.0 2013/05/29 | Click イベントの通知を追加 (メディエーション対応として) 受信エラー通知メソッド内で、NADView を release した後にクラッシュする問題 修正 release に関するサンプルコードを修正 |
| iOS_ver 2.2.0 2013/07/22 | 広告サイズ追加対応 NSError プロパティの追加、広告サイズごとのテスト ID の追加 ◆広告サイズについて を追加 ◆よくある質問 を WEB へ移動 |
| iOS_ver 2.2.1 2013/09/26 | iOS7 対応 arm64 アーキテクチャに対応 |
| iOS_ver 2.3.0 2013/11/05 | アイコン型広告対応 |
| iOS_ver 2.3.1 2013/12/02 | アイコン型広告サイズ変更対応 アイコン型広告余白部分を非表示にする設定を追加 |
| iOS_ver 2.3.2 2014/01/20 | 不具合修正 |
| iOS_ver 2.3.3 2014/03/06 | Interface Builder での実装に対応 delegate 通知 nadViewDidFinishLoad メソッドを任意に変更 NadView, NadIconLoader の delegate 処理見直し |
| iOS_ver 2.4.0 2014/06/18 | アニメーション GIF 対応 |

| | |
|--------------------------|--|
| iOS_ver 2.4.1 2014/07/15 | 不具合修正 |
| iOS_ver 2.5.0 2014/08/5 | インタースティシャル広告実装 |
| iOS_ver 2.5.1 2014/08/21 | 不具合修正 |
| iOS_ver 2.5.2 2014/09/10 | iOS8 でインタースティシャル広告の表示位置がずれる場合がある問題の修正 不具合修正 |
| iOS_ver 2.5.3 2014/09/19 | 不具合修正 |
| iOS_ver 2.5.4 2014/09/29 | 不具合修正 |
| iOS_ver 2.5.5 2014/10/06 | Swift 用 Bridging Header の利用方法と Swift での実装例を追加 不具合修正 |
| iOS_ver 2.5.6 2014/11/12 | iOS8.1 でのインタースティシャル広告の不具合修正 インタースティシャル広告の端末の向き設定条件 を変更 インタースティシャル広告の複数枠管理機能の追加 サンプルソースの設置場所を GitHub に変更 |
| iOS_ver 2.5.7 2015/01/22 | 不具合修正 |

目次

| | |
|---|----|
| ◆nendSDK iOSについて..... | 6 |
| 1. nendSDK導入のおおまかな流れ..... | 6 |
| 2. ファイル構成..... | 6 |
| 3. 対応環境..... | 6 |
| 4. nendSDK取得項目について..... | 7 |
| 5. インフォメーションボタンについて..... | 8 |
| ◆SDKの組み込み..... | 9 |
| 1. プロジェクトへのnendSDK追加..... | 9 |
| 2. 必須フレームワークの追加..... | 12 |
| 3. ビルド..... | 14 |
| 4. 広告ビューの設置..... | 15 |
| 4-1. バナー型広告..... | 15 |
| 4-1-1. 広告サイズについて..... | 15 |
| (1) 広告ビューサイズの指定..... | 15 |
| (2) 端末のディスプレイサイズよりも大きい広告サイズを指定した場合..... | 15 |
| 4-1-2. バナー型広告の実装手順..... | 16 |
| ○ヘッダファイル..... | 16 |
| ○実装ファイル..... | 17 |
| ○Interface Builderを使用した実装手順..... | 28 |
| 4-1-3. NADViewの内容..... | 31 |
| ○メソッド..... | 31 |
| ○プロパティ..... | 31 |
| ○Delegate..... | 32 |
| 4-2. アイコン型広告..... | 33 |
| 4-2-1. アイコン型広告のサイズについて..... | 33 |
| (1) アイコン型広告ビューのサイズ指定..... | 33 |
| (2) アイコン型広告ビュー内の配置..... | 33 |
| (3) アイコン型広告サイズと各種設定について..... | 34 |
| 4-2-2. アイコン型広告の実装手順..... | 35 |
| ○ヘッダファイル..... | 35 |
| ○実装ファイル..... | 37 |
| ○Interface Builderを使用した実装手順..... | 48 |
| 4-2-3. NADIIconLoaderの内容..... | 53 |
| ○メソッド..... | 53 |
| ○プロパティ..... | 53 |
| ○Delegate..... | 53 |
| 4-2-4. NADIIconViewの内容..... | 54 |
| ○メソッド..... | 54 |

| | |
|--|----|
| ○プロパティ | 54 |
| 4 – 2 – 5 . NADIconArrayView の内容 | 54 |
| ○メソッド | 54 |
| ○プロパティ | 55 |
| 4 – 3 . インタースティシャル広告 | 56 |
| 4 – 3 – 1 . インタースティシャル広告の実装手順 | 56 |
| (1) 広告のロード | 56 |
| (2) 広告の表示 | 58 |
| (3) 広告の非表示 | 61 |
| 4 – 3 – 2 . NADInterstitial の内容 | 62 |
| ○インターフェイスの端末の向き設定条件について | 62 |
| ○プロパティ | 63 |
| ○メソッド | 65 |
| ○NADInterstitialDelegate | 65 |
| ◆検証 | 70 |
| iOS アプリ向け表示テスト用 ID | 70 |
| ◆よくある質問 | 71 |

◆nendSDK iOSについて

1. nendSDK導入のおおまかな流れ

① nend 管理画面でアプリ登録と広告枠登録を行います。

※ 本マニュアルは広告枠登録後、「apiKey」「spotID」を発行し SDK を入手している前提で説明を行います。

※ 広告枠を登録していない場合には、別紙「管理画面マニュアル」をご参照の上、ご申請ください。

※ 広告枠を申請後、広告枠の管理>広告枠>SDK>「SDKをダウンロード」で SDK を入手できます。

② 本マニュアルに従って nendSDK をアプリに組み込みます。

2. ファイル構成

NendSDK_iOS.zip

| | |
|--------------------------|---|
| NendAd/ | SDK フォルダ |
| libNendAd.a | ライブラリ |
| NADView.h | バナー広告用ヘッダファイル |
| NADIconView.h | アイコン広告ビュー用ヘッダファイル |
| NADIconLoader.h | アイコンローダー用ヘッダファイル |
| NADIconArrayView.h | アイコン広告ビュー用ヘッダファイル(InterfaceBuilder 用) |
| NADInterstitial.h | インターミディシャル広告用ヘッダファイル |
| NADView_readme.txt | ライセンス文等 |
| nendSDK2.5.7_manual.pdf | 本マニュアル |
| NendAd-Bridging-Header.h | Swift 用の Objective-C Bridging Header ファイル |

サンプルソースの設置場所が下記 URL に変更になりました。

<https://github.com/fan-ADN/nendSDK-iOS>

3. 対応環境

デバイスは以下の環境にて動作確認を行っています。

| デバイス | モデル |
|------------|---|
| iPhone | iPhone3GS、iPhone4、iPhone4S、iPhone5、iPhone5S、iPhone5C、iPhone6、iPhone6 Plus |
| iPad | iPad、iPad2、iPad（第3世代以降）、iPad mini |
| iPod Touch | iPodTouch（第3世代以降） |

OS バージョンは、iOS 5.0 以上が動作保障対象となります。

それ以外の端末では正常に動作しない場合があります。

開発環境

Xcode 5.0 以上が必要です。

4. nendSDK 取得項目について

平成 24 年より、総務省においてスマートフォンのプライバシーに関する議論がなされ、「スマートフォンプライバシーイニシアティブ」「スマートフォンプライバシーイニシアティブⅡ」として取りまとめがされております。

スマートフォン プライバシー イニシアティブ 参考URL

http://www.soumu.go.jp/main_content/000171225.pdf

スマートフォン プライバシー イニシアティブⅡ 参考URL

http://www.soumu.go.jp/main_content/000247654.pdf

アプリ提供者、情報収集モジュール提供者、広告配信事業者の自主的かつ積極的な取り組みを期待されておりまますので、メディアパートナー様におかれましてもご確認頂きますようお願ひいたします。また、弊社においても広告配信システム提供事業者として、より一層透明性の高い取り組みを実施していく所存でございます。アプリ提供者（nend メディアパートナー様）がより安心して nendSDK をご利用いただけるよう、またスマートフォンプライバシーイニシアティブの内容にそって、nendSDK が取得している項目について以下に記載します。ご確認いただき、プライバシーポリシー作成時にお役立てください。

■ nendSDK がサーバに送信する情報と利用目的

- | | |
|----------------|---|
| ・apikey/spotID | nend がメディアごとに広告配信を行うために発行し取得するもの |
| ・OS/OSバージョン | nend が広告配信時に利用するために取得するもの |
| ・言語設定 | nend が広告配信時に利用するために取得するもの |
| ・機種名/デバイス名 | nend が広告配信時に利用するために取得するもの |
| ・SDKバージョン | nend が広告配信時に利用するために取得するもの |
| ・独自 ID | nend が広告効果測定、不正対策や一部メニューにおいて広告配信時のターゲティングのため、またその可否のため（オプトアウトしているかどうか）に利用するもの (UUID やデバイス推定 ID) |
| ・IDFA | nend や nex8（ファンコミュニケーションズ提供の DSP）が広告効果測定や不正対策や一部メニューにおいて広告配信時のターゲティングのため、またその可否のため（オプトアウトしているかどうか）に利用するもので Apple 社が発行する広告識別子。 |

■ その他サーバに送信せずに nendSDK が取得している情報と利用目的

- | | |
|----------|---|
| ・URLスキーム | nend が広告効果測定や不正対策や一部メニューにおいて広告配信時のターゲティングのために利用するもの |
|----------|---|

■ プライバシーポリシー

nend 運営会社 株式会社ファンコミュニケーションズ

「個人情報の取り扱いについて」は[こちら](#)

「個人情報保護方針」については[こちら](#)

「nendSDK プライバシーポリシー」については[こちら](#)

5. インフォメーションボタンについて



アプリに配信される広告バーに「インフォメーションボタン」が表示されます。

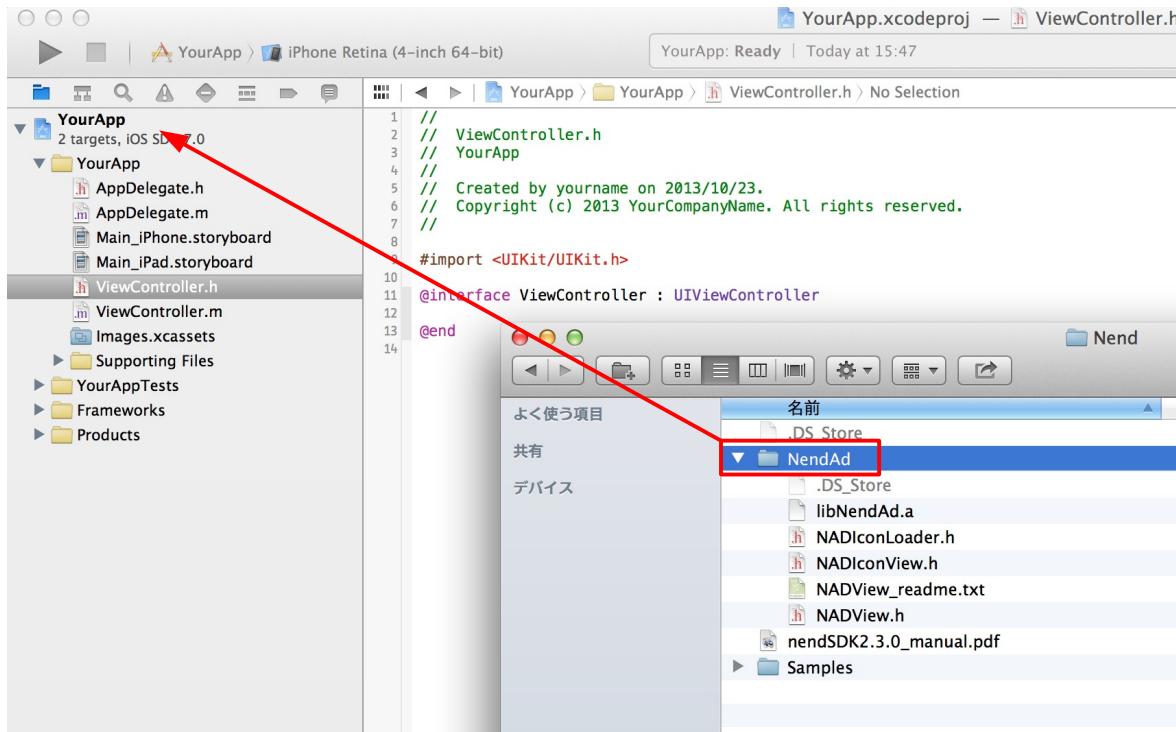
この機能が追加された nendSDK を利用することにより、nend が提供する一部ターゲティング広告に対して、よりユーザ（アプリ利用者）自身が容易にオプトアウトの設定をすることが可能になります。

尚、nend SDK における取得情報は、前項に記載の通りであり、個人情報に該当するものは取得しておりません。また端末のローカルストレージへの無断書き込み、読み込みも一切行っておりませんので安心してお使いください。

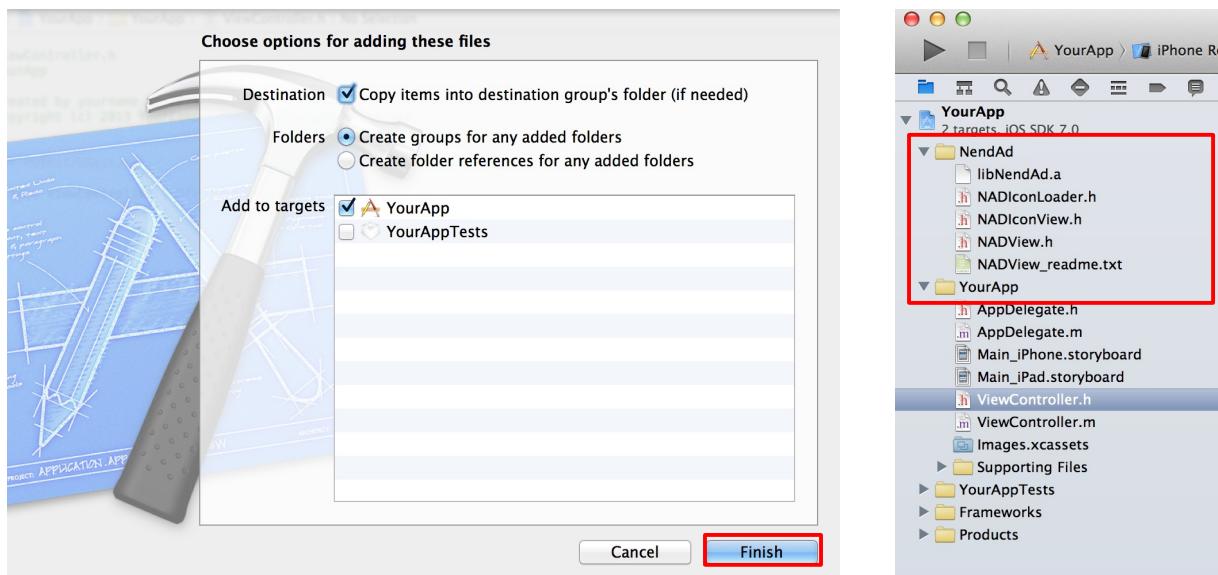
◆SDKの組み込み

1. プロジェクトへのnendSDK追加

Xcode上で対象プロジェクトに「NendAd」フォルダごとドラッグ&ドロップします。



下記、左図ダイアログの表示で必要に応じて任意の設定(※)を選び「Finish」をクリックします。



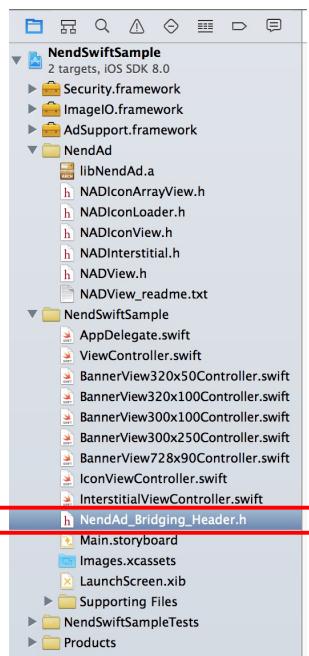
追加が完了するとXcodeのファイルリストに右図のように表示されます。

※既にプロジェクトフォルダ配下にNendAdフォルダを移動させたものに対して参照設定を追加する場合には、ファイルをコピーする必要がないため、“Copy items into destination group's folder (if needed)”にチェックを入れる必要はありません。

NendAd-Bridging-Header の追加 (Swift のプロジェクトで利用する場合)

○新規で Objective-C Bridging Header を利用する場合

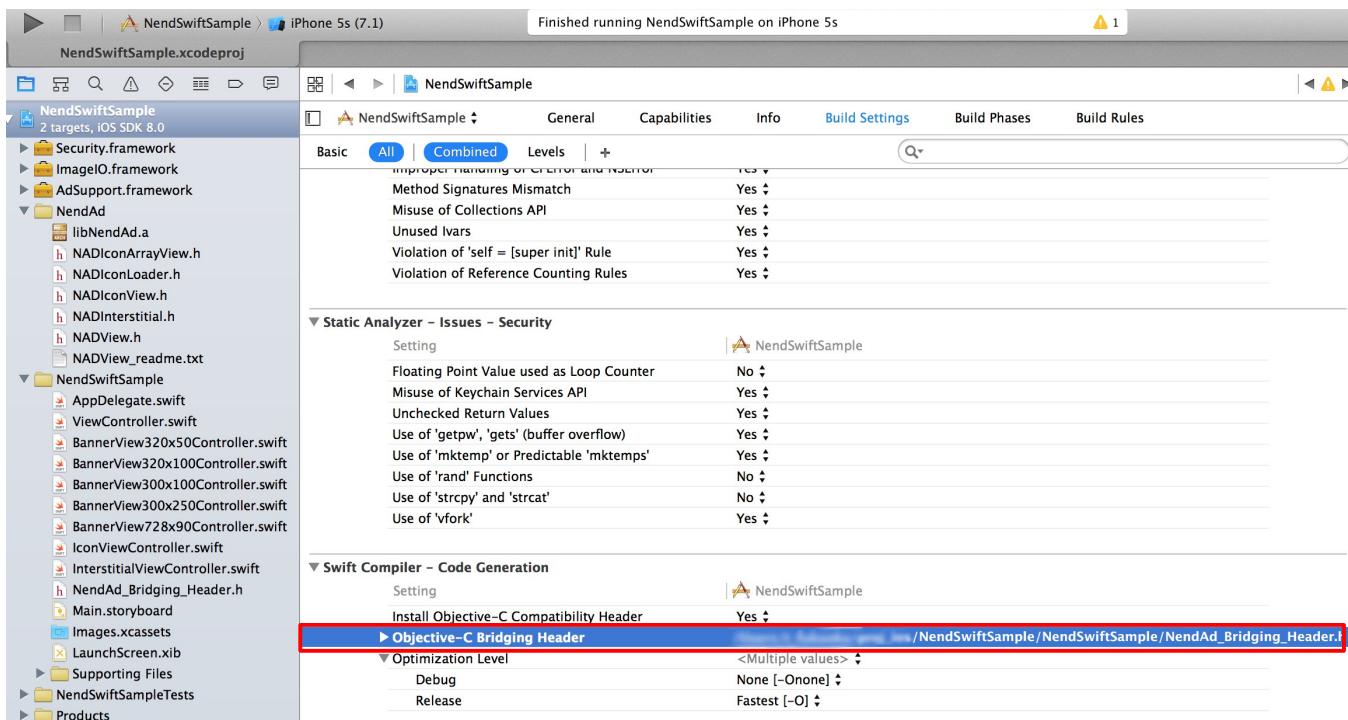
1. NendAd-Bridging-Header.h ファイルをプロジェクトに追加する



2. NendAd-Bridging-Header.h へのパスを設定する

Build Settings > Swift Compiler-Code Generation > Objective-C Bridging Header に、\$(SRCROOT)/\$(PRODUCT_NAME)/NendAd-Bridging-Header.h を追加します。

※パスは適宜変更してください



○すでに Objective-C Bridging Header を利用している場合

利用しているヘッダーファイル内に、利用したい広告タイプのヘッダーファイルをインポートします。

例) YourOriginal-Bridging-Header.h

```
#ifndef NendSwiftSample_Swift_Bridging_Header_h
#define NendSwiftSample_Swift_Bridging_Header_h

#import "NADView.h"

#endif
```

上記の例では、nendSDK のバナー広告を利用する例になります。

2. 必須フレームワークの追加

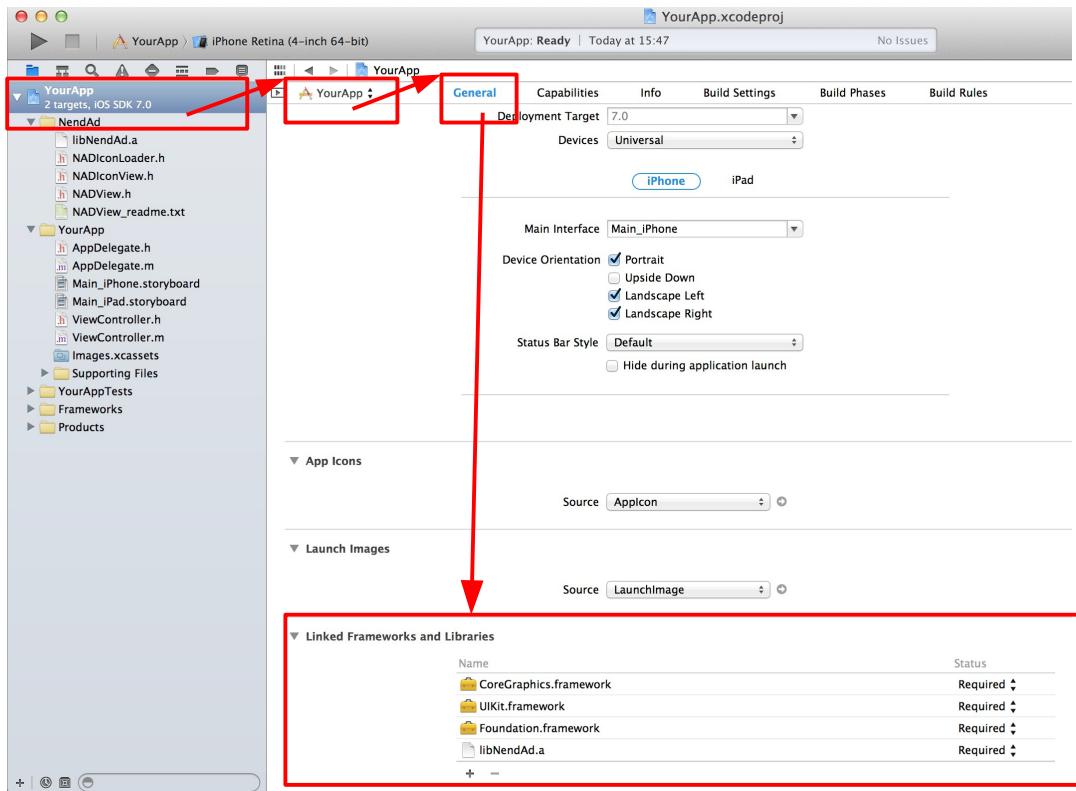
nendSDK の利用には、

- **AdSupport.framework**
- **Security.framework**
- **ImageIO.framework**

の追加が必要です。

左側のプロジェクトナビゲーターから、プロジェクトをクリックして

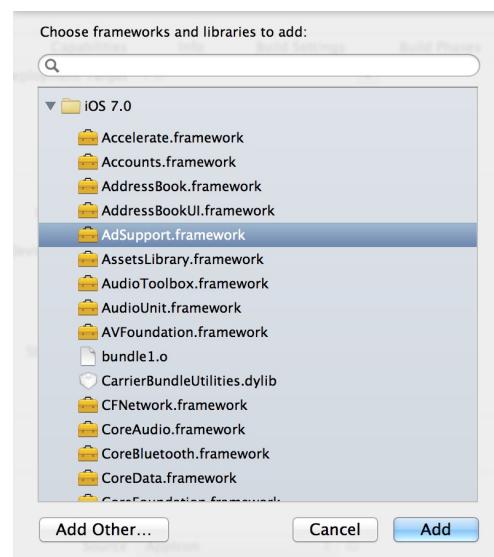
TARGETS> General> Link Frameworks With Libraries 項目を開きます。



「+」をクリックし

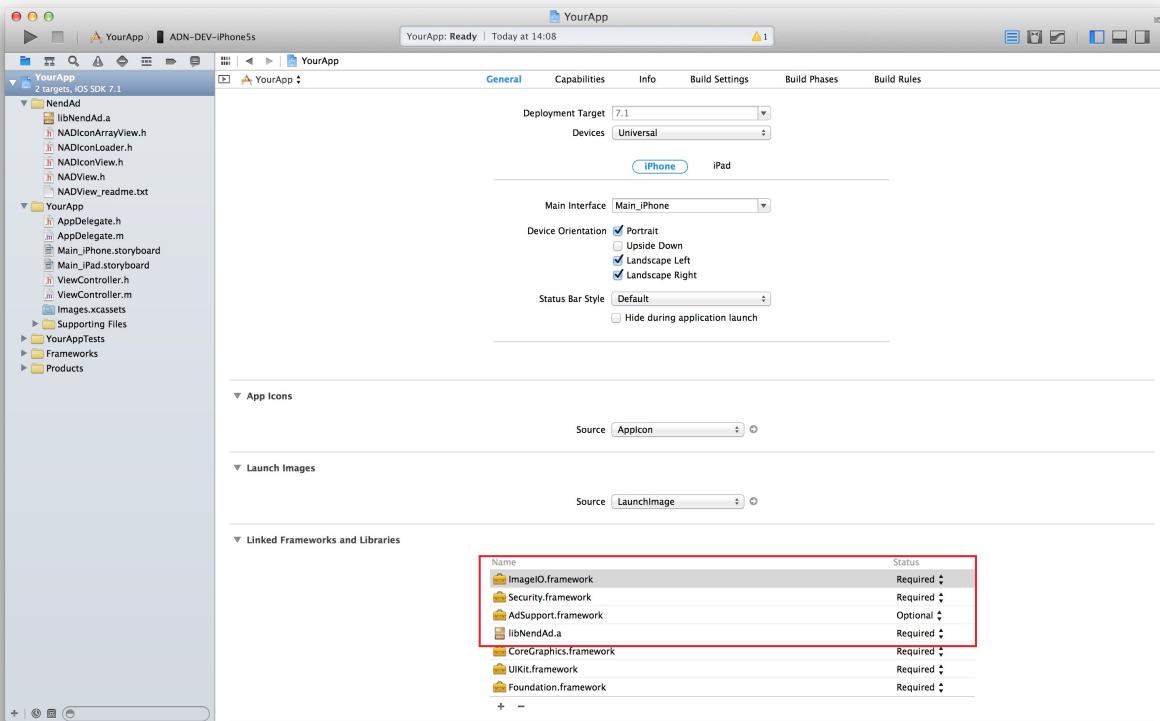
追加できるライブラリの一覧を開いて
下記の Framework を追加してください。

- **AdSupport.framework**
- **Security.framework**
- **ImageIO.framework**



Link Binary With Libraries に下記のリンクが正常に追加されていることを確認します。

- **AdSupport.framework**
- **Security.framework**
- **ImageIO.framework**
- **libNendAd.a**



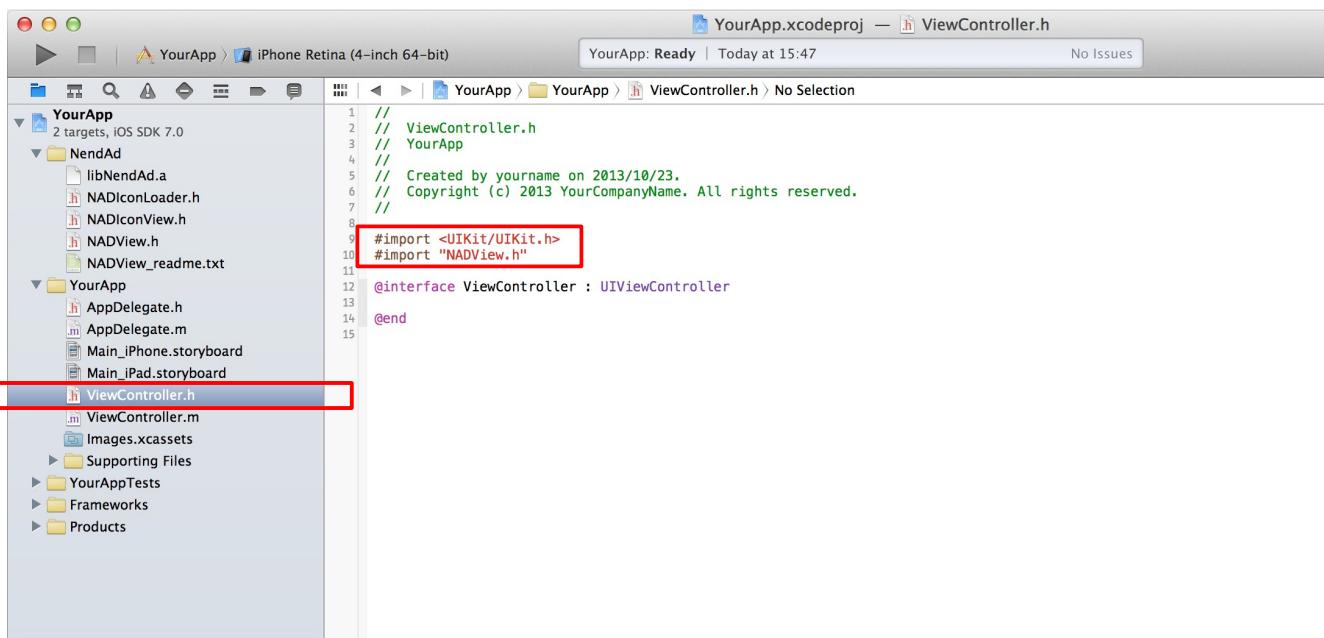
このとき、AdSupport.framework を Required → Optional に変更します。

**※iOS6 未満を動作対象に含めている (Deployment Target の指定が iOS 6 未満)
アプリケーションである場合は、必ず Optional に変更してください。**

3. ビルド

Objective-Cの場合、任意のヘッダファイルに「**#import “NADView.h”**」を記述します。Swiftの場合、利用している Objective-C Bridging Header に「**#import “NADView.h”**」を記述しますので、任意のファイルへの記述は不要です。

一旦ビルド (Clear&Build) を行い、エラーが発生せずにビルドが成功すれば、正常に SDK が追加できています。



※この時点でエラーが発生する場合は、Link Binary With Libraries の内容を再度確認してください。

4. 広告ビューの設置

4-1. バナー型広告

4-1-1. 広告サイズについて

SDK ver2.2.0 から以下の広告サイズが使用出来るようになりました。

| | | |
|-----------|--------------------------|---------------|
| 320 × 50 | iPhone, iPod Touch, iPad | 従来のバナーサイズ |
| 320 × 100 | iPhone, iPod Touch, iPad | |
| 300 × 100 | iPhone, iPod Touch, iPad | |
| 300 × 250 | iPhone, iPod Touch, iPad | |
| 728 × 90 | iPad のみ | タブレット専用バナーサイズ |

※広告サイズの指定は、nend 管理画面での広告枠作成時に行います。

1つの広告枠に対して、1つの広告サイズが指定出来ます。

(1) 広告ビューサイズの指定

広告ビューを作成する際に「該当広告枠で指定したサイズ」を指定するようにしてください。

※アプリ側からのサイズ指定と異なる場合は、管理画面でのサイズ指定が優先されます。

(2) 端末のディスプレイサイズよりも大きい広告サイズを指定した場合

端末ディスプレイサイズよりも大きい広告サイズが指定されている場合は、広告を表示しません。アプリ側へは受信エラーが通知されます。



上図のようにディスプレイに収まりきらない広告サイズの場合は表示されません。

4 - 1 - 2. バナー型広告の実装手順

ここでは最もシンプルな構造を例にして NADView の実装方法について説明します。
それ以外の詳しい実装についてはサンプルソースをご参照ください。

○ヘッダーファイル

(1) delegate 準拠

Objective-C

広告ビューを保持するクラスのヘッダーファイルで NADView.h をインポートして
NADViewDelegate に準拠します。

例) YourAppViewController.h

```
#import <UIKit/UIKit.h>
#import "NADView.h"

@interface YourAppViewController : UIViewController <NADViewDelegate> {}

@property (nonatomic, retain) NADView * nadView;

@end
```

Swift

事前に利用している **Objective-C Bridging Header** のヘッダーファイルに
NADView.h ファイルをインポートしてください。

例) YourAppViewController.swift

```
import UIKit

class : YourAppViewController: UIViewController, NADViewDelegate {

    private var nadView: NADView!

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    ~省略~
}
```

注意：複数画面で構成されるアプリケーションで、各 ViewController ごとにインスタンス生成するような実装方法の場合には、画面ごとに必ず後述の **(8) 定期ロードの管理**を行ってください。

○実装ファイル

Objective-C

例) YourAppViewController.m (一部) - NADView 生成からロードまで

```
- (void)viewDidLoad {
    [super viewDidLoad];

    // (2) NADView の作成
    self.nadView = [[NADView alloc] initWithFrame:CGRectMake(0, 0, 320, 50)];

    // (3) ログ出力の指定
    [self.nadView setIsOutputLog:NO];
    // (4) set apiKey, spotId.
    [self.nadView setNendID:@"[管理画面より発行された apiKey]"
                      spotID:@"[管理画面より発行された spotID]"];
    [self.nadView setDelegate:self]; // (5)
    [self.nadView load]; // (6)

    [self.view addSubview:self.nadView]; // 最初から表示する場合
}
```

Swift

例) YourAppViewController.swift (一部) - NADView 生成からロードまで

```
import UIKit

class : YourAppViewController: UIViewController, NADViewDelegate {

    private var nadView: NADView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // NADViewクラスを生成
        nadView = NADView(frame: CGRect(x: 0, y: 0, width: 320, height: 50))

        // 広告枠のapikey/spotidを設定(必須)
        nadView.setNendID("[管理画面より発行された apiKey]",
                          spotID: "[管理画面より発行された spotID]")

        // nendSDKログ出力の設定(任意)
        nadView.isOutputLog = false

        // delegateを受けるオブジェクトを指定(必須)
        nadView.delegate = self

        // 読み込み開始(必須)
        nadView.load()

        // 通知有無にかかわらずViewに乗せる場合
        self.view.addSubview(nadView)
    }
}
```

(2) NADView 生成

任意の位置、320x50 サイズの CGRect を引数にインスタンスを生成します。

Objective-C

```
self.nadView = [[NADView alloc] initWithFrame: CGRectMake(0,0, 320, 50)];
```

Swift

```
nadView = NADView(frame: CGRect(x: 0, y: 0, width: 320, height: 50))
```

※上記は、addSubview する先の View の 0,0 の位置を左上に指定して広告を配置する例です。

(3) isOutputLog の設定

エラーログや警告ログを NSLog で出力するかどうかを指定します。

Objective-C

```
[self.nadView setIsOutputLog:NO];
```

Swift

```
nadView.isOutputLog = false
```

初期値は YES です。出力したくない場合は NO をセットしてください。

(4) apiKey,spotID の設定

nend 管理画面で発行した、該当アプリの広告枠の apiKey、spotID をセットします。

Objective-C

```
[self.nadView setNendID:@"[管理画面より発行された apiKey]"  
spotID:@"[管理画面より発行された spotID]"];
```

Swift

```
nadView.setNendID("[管理画面より発行された apiKey]",  
spotID: "[管理画面より発行された spotID]")
```

※この時点で広告設定情報へのアクセスを開始します。

広告枠ステータスが「承認中」の場合、広告のロードをしても受信エラーになります。nend 管理画面で該当アプリの広告枠ステータスが「アクティブ」であることを確認してください。

広告枠の申請は、承認済みになるとステータスが「アクティブ」に変わり、登録メールアドレス宛に広告枠承認のお知らせが届きます。メール到着の数時間後には広告配信ができる状態になります。

単に実装方法の確認を行う場合は一時的に表示テスト用 ID ([→◆検証](#)) を利用するなどしてください。

(5) デリゲートオブジェクトの設定

NADView が広告を受信開始した場合に指定されたデリゲートの nadViewDidFinishLoad メソッドを呼んで通知を行います。指定するデリゲートは「nadViewDidFinishLoad」メソッドを実装する「NADViewDelegate」プロトコルに準拠させたクラスを指定します。

Objective-C

```
[self.nadView setDelegate:self];
```

Swift

```
nadView.delegate = self
```

(6) 広告のロード

NADView の load メソッドで広告のロードを開始します。

Objective-C

```
[self.nadView load];
```

Swift

```
nadView.load()
```

※リトライ間隔を標準値以外で指定したい場合には、
以下のメソッドを利用してロードを開始します。

Objective-C

```
// 例) 問い合わせエラー時には 60 分間隔で再問い合わせする  
[self.nadView load:[NSDictionary dictionaryWithObjectsAndKeys: @"3600", @"retry", nil]];
```

Swift

```
// 例) 問い合わせエラー時には 60 分間隔で再問い合わせする  
var dictionary: Dictionary<String, String> = ["3600" : "retry"];  
nadView.load(dictionary)
```

(7) Delegate 通知

※ver2.0.0より、delegate メソッドでの通知の際、一律メインスレッドで通知を行うよう変更されました。旧バージョンからの入れ替えの際、サブスレッドで通知を受ける前提での実装がある場合には十分にご注意ください。

【任意】ロード完了

広告のロードが完了すると(5)で指定したデリゲートの nadViewDidFinishLoad メソッドに通知されます。ロードが完了してから NADView を表示したい場合はここで行うことができます。

Objective-C

```
- (void)nadViewDidFinishLoad:(NADView *)adView {  
    NSLog(@"delegate nadViewDidFinishLoad:");  
}
```

Swift

```
func nadViewDidFinishLoad(adView: NADView!) {  
    println("delegate nadViewDidFinishLoad:")  
    self.view.addSubview(nadView) // ロードが完了してから NADView を表示する場合  
}
```

【任意】広告バナークリック通知

表示されている広告バナーをクリックした際に通知されます。
ただし、このイベントをカウントしても、ネットワーク状況や環境により「実際に任意の広告表示ができた数(サーバ側でのクリック数としてのカウント)」とは、異なる場合がありますので、注意してください。

Objective-C

```
- (void)nadViewDidClickAd:(NADView *)adView {  
    NSLog(@"delegate nadViewDidClickAd:");  
}
```

Swift

```
func nadViewDidClickAd(adView: NADView!) {  
    println("delegate nadViewDidClickAd")  
}
```

【任意】広告受信通知

広告の受信に成功した場合通知されます。
広告を受信するたびに任意の処理を行いたい場合に利用します。

Objective-C

```
- (void)nadViewDidReceiveAd:(NADView *)adView {  
    NSLog(@"delegate nadViewDidReceiveAd:");  
}
```

Swift

```
func nadViewDidReceiveAd(adView: NADView!) {  
    println("delegate nadViewDidReceiveAd")  
}
```

【任意】広告受信エラー通知

広告の受信に失敗した場合に通知されます。

通信エラー、広告在庫がなくなった場合など、何らかの理由で広告を表示できない場合に通知します。エラー時に広告を非表示にするなどの処理が必要な場合に利用します。

Objective-C

```
- (void)nadViewDidFailToReceiveAd:(NADView *)adView {  
    NSLog(@"delegate nadViewDidFailToReceiveAd:");  
}
```

Swift

```
func nadViewDidFailToReceiveAd(adView: NADView!) {  
    println("delegate nadViewDidFailToReceiveAd")  
}
```

エラー内容によって処理を分ける必要がある場合は、次ページのように実装してください。

【任意】広告受信エラー通知

エラー内容によって処理を分けるサンプル

Objective-C

```
- (void)nadViewDidFailToReceiveAd:(NADView *)adView
{
    NSLog(@"delegate nadViewDidFailToLoad:");

    // エラーごとに分岐する
    NSError* error = adView.error;
    NSString* domain = error.domain;
    int errorCode = error.code;

    // isOutputLog = NO でも、domain を利用してアプリ側で任意出力が可能
    NSLog(@"log %d", adView.isOutputLog);
    NSLog(@"%@",[NSString stringWithFormat:@"code=%d, message=%@",
                errorCode, domain]);

    switch (errorCode) {
        case NADVIEW_AD_SIZE_TOO_LARGE:
            // 広告サイズがディスプレイサイズよりも大きい
            break;
        case NADVIEW_INVALID_RESPONSE_TYPE:
            // 不明な広告ビュータイプ
            break;
        case NADVIEW_FAILED_AD_REQUEST:
            // 広告取得失敗
            break;
        case NADVIEW_FAILED_AD_DOWNLOAD:
            // 広告画像の取得失敗
            break;
        case NADVIEW_AD_SIZE_DIFFERENCES:
            // リクエストしたサイズと取得したサイズが異なる
            break;
        default:
            break;
    }
}
```

Swift

```
func nadViewDidFailToReceiveAd(adView: NADView!) {  
  
    // エラーごとに処理を分岐する  
    var error: NSError = adView.error  
  
    switch (error.code){  
    case NADViewErrorCode.ADVIEW_AD_SIZE_TOO_LARGE.hashValue:  
        // 広告サイズがディスプレイサイズよりも大きい  
        break  
    case NADViewErrorCode.ADVIEW_INVALID_RESPONSE_TYPE.hashValue:  
        // 不明な広告ビュータイプ  
        break  
    case NADViewErrorCode.ADVIEW_FAILED_AD_REQUEST.hashValue:  
        // 広告取得失敗  
        break  
    case NADViewErrorCode.ADVIEW_FAILED_AD_DOWNLOAD.hashValue:  
        // 広告画像の取得失敗  
        break  
    case NADViewErrorCode.ADVIEW_AD_SIZE_DIFFERENCES.hashValue:  
        // リクエストしたサイズと取得したサイズが異なる  
        break  
    default:  
        break  
    }  
}
```

NADViewErrorCode (NADView.NSError.code) の内容

| NADViewErrorCode | エラー内容 |
|-------------------------------|-------------------------------------|
| NADVIEW_FAILED_AD_REQUEST | 広告取得失敗 (ネットワークエラー、サーバエラー、在庫切れなど) |
| NADVIEW_AD_SIZE_TOO_LARGE | 広告サイズがディスプレイサイズよりも大きい |
| NADVIEW_AD_SIZE_DIFFERENCES | リクエストしたサイズと取得したサイズが異なる※ |
| NADVIEW_INVALID_RESPONSE_TYPE | 不明な広告ビュータイプ※ |
| NADVIEW_FAILED_AD_DOWNLOAD | 広告画像の取得失敗 |

※基本的に発生することは稀です

(8) 定期ロードの管理

広告のロードを開始した後に画面内に表示しないケースがある場合には、必ず以下の処理を行ってください。

これには、

- ・複数画面で構成された画面遷移が発生するアプリケーションにおいて各画面で個別に広告のインスタンスを生成している場合
- ・**複数広告の切り替え処理（広告スイッチング SDK の利用含む）を行う場合**などが該当します。

定期ロードの中断

広告を画面内に表示しない、もしくは画面遷移等で View 自体が表示されない場合には pause メッセージを送信、広告の定期ロードを中断します。

例) 画面が隠れたら定期ロードを中断

Objective-C

```
- (void)viewWillDisappear:(BOOL)animated {
    [self.nadView pause];
}
```

Swift

```
override func viewWillDisappear(animated: Bool) {
    super.viewWillDisappear(animated)
    nadView.pause()
}
```

定期ロードの再開

広告を再び画面内に表示、または画面遷移等で View 自体を表示する場合には resume メッセージを送信、広告の定期ロードを再開します。

例) 画面が表示されたら定期ロードを再開

Objective-C

```
- (void)viewWillAppear:(BOOL)animated {
    [self.nadView resume];
}
```

Swift

```
override func viewWillAppear(animated: Bool) {
    super.viewWillAppear(animated)
    nadView.resume()
}
```

(9) リリース

dealloc 時には、必ず delegate プロパティに nil をセットしてからリリースを行うようにしてください。

Objective-C

```
- (void) dealloc {  
    [self.nadView setDelegate:nil]; // delegate に nil をセット  
    self.nadView = nil;           // プロパティ経由で release、nil をセット  
  
    // [super dealloc]; // MRC(非 ARC 時には必要)  
}
```

Swift

```
deinit {  
    nadView.delegate = nil  
    nadView = nil  
}
```

【重要】

delegate プロパティに nil をセットしない場合、メモリの解放が適切に行われず、稀に予期しないクラッシュを引き起こす場合があります。必ず delegate に nil をセットしてから release を行うようにしてください。

また、delegate に nil がセットされた際、自動的に広告受信ローテーションを中断(pause)します。これにより release 前の pause メッセージ送信が不要になります。

○Interface Builder を使用した実装手順

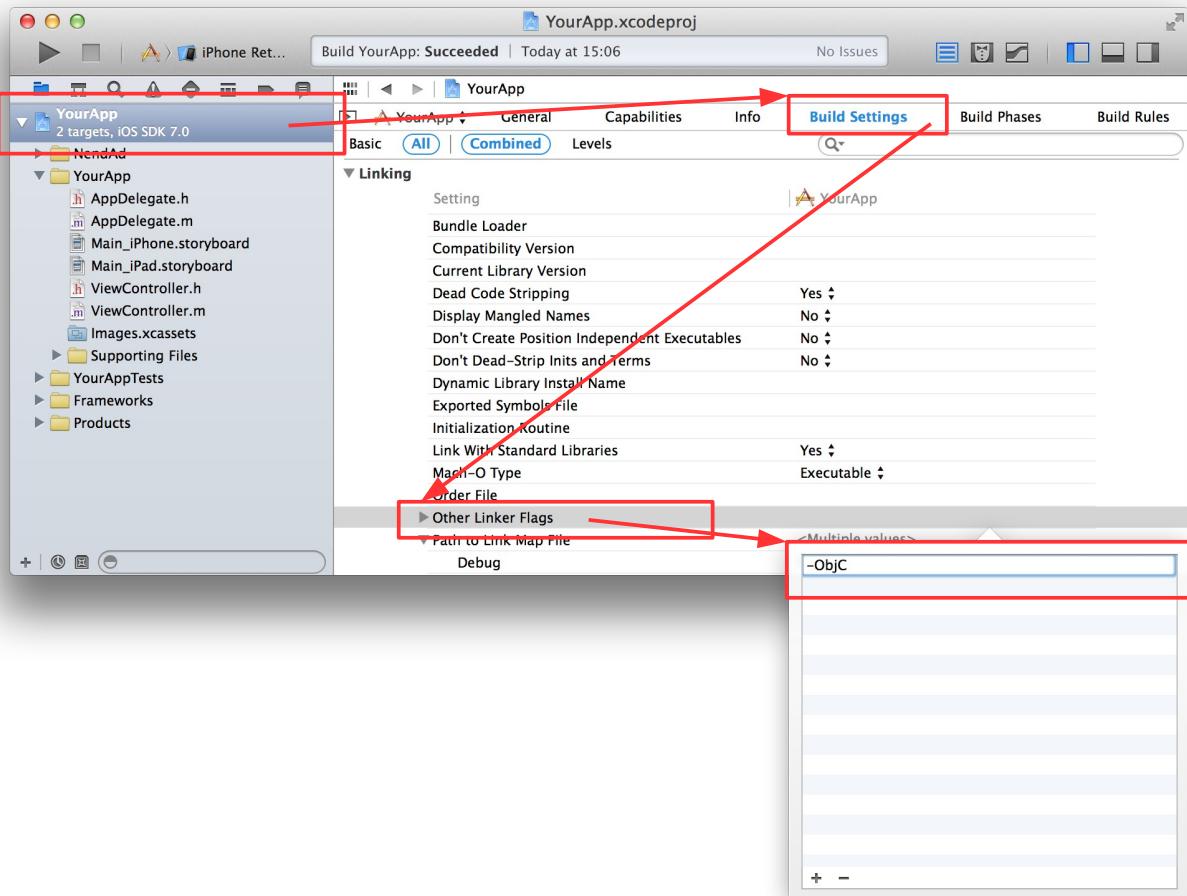
Interface Builder を使用してバナー広告の実装が可能です。

(※Deployment Target の設定が 5.1 未満ではご利用できません。)

(1)リンクフラグの追加

左側のプロジェクトナビゲーターから、プロジェクトをクリックして

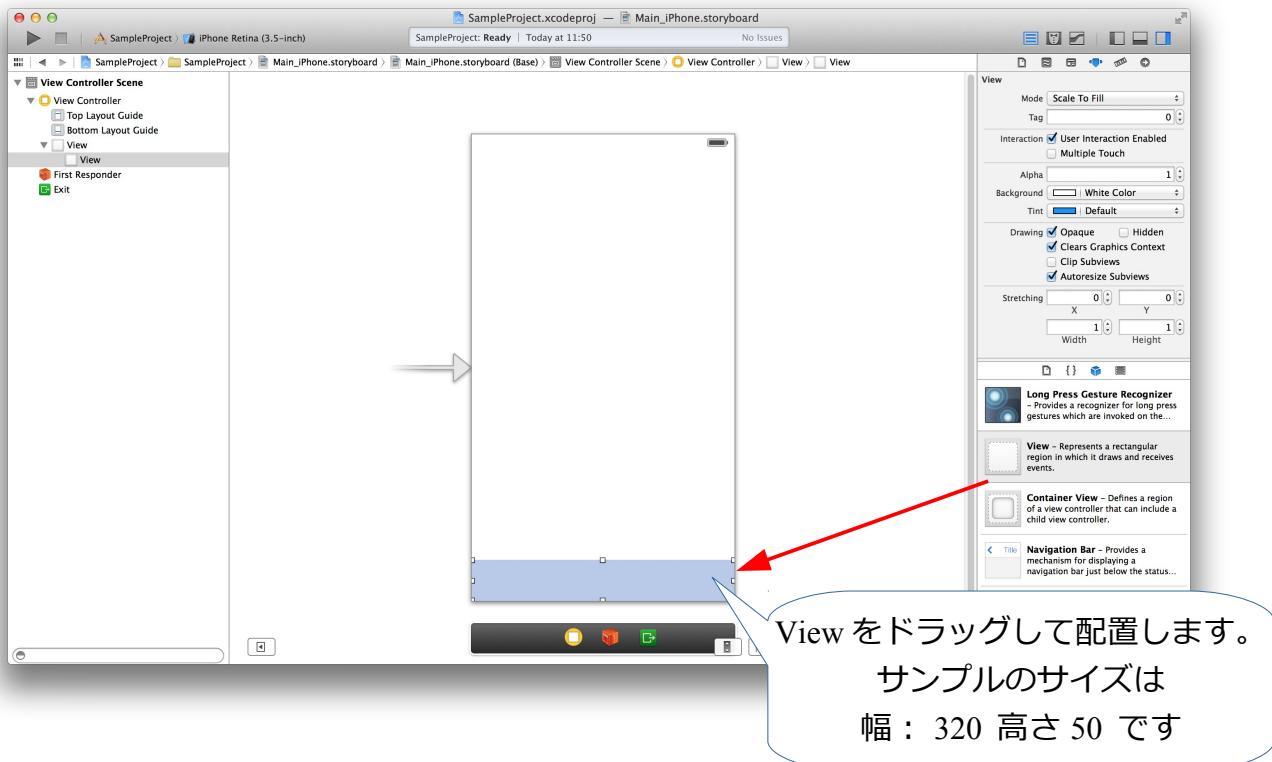
TARGETS > Build Setting > Other Linker Flags 項目を開き、“-ObjC”を追加します。



(2)View の追加

広告を表示する View 上に View を追加します。

また、配置する際に View のサイズを広告サイズに合わせます。



(3)クラスの変更、プロパティの設定

Utility area から Identity Inspector からクラスの変更とプロパティの変更を行います。

(1)CustomClass に NADView を指定

(2)User Defined Runtime Attributes に以下の値を追加する

- nendApiKey に管理画面より発行された apiKey を設定
- nendSpotID に管理画面より発行された spotID を設定

| Custom Class | | |
|--|---------|--|
| Class | NADView | |
| Identity | | |
| Restoration ID <input type="text"/> | | |
| User Defined Runtime Attributes | | |
| Key Path | Type | Value |
| nendApiKey | String | a6eca9dd074372c898dd1df549301f277c53f2b9 |
| nendSpotID | String | 3172 |

(4) User Defined Runtime Attributes で設定可能な項目

User Defined Runtime Attributes で設定可能な項目および、初期値は以下となります。

| Key Path | Type | Value | 必須 or 任意 | 初期値 | 説明 |
|-------------|---------|--------------|----------|-----|--------------------|
| nendApiKey | String | 発行された apiKey | 必須 | - | 管理画面より発行された apiKey |
| nendSpotID | String | 発行された SpotID | 必須 | - | 管理画面より発行された SpotID |
| isOutputLog | Boolean | YES or NO | 任意 | NO | ログ出力可否 |

以上で設定完了となります。

4 – 1 – 3 . NADView の内容

○メソッド

- **(void)setNendID:(NSString *)apiKey spotID:(NSString *)spotID;**

広告枠の apiKey と spotID をセットします。

- **(void)load;**

ロードを開始します。

- **(void)load:(NSDictionary *)parameter;**

ロードを開始します。

接続エラーや広告設定受信エラーなどの場合にリトライする間隔を、 NSDictionary で任意指定出来ます。ただし key は「retry」、value は 30 - 3600 の間で指定してください。範囲外指定された場合は標準で 60 秒が適用されます。標準で問題ない場合は parameter のない load; を利用してください。

- **(void) pause;**

広告の定期ロード中断を要求します

- **(void) resume;**

広告の定期ロード再開を要求します

○プロパティ

@property (nonatomic, assign) id < NADViewDelegate > delegate;

delegate オブジェクトの指定（任意）

@property (nonatomic) BOOL isOutputLog;

エラーログや警告ログを、 NSLog として出力するかどうかの指定（任意）

@property (nonatomic, assign) NSError error;

参照すると受信エラー時にその内容を動的に知ることが出来ます。

【廃止予定】

@property (nonatomic, assign) UIViewController *rootViewController;

モーダルビュー表示元のビューコントローラの指定(任意)

○Delegate

- (void) nadViewDidFinishLoad:(NADView *)adView;

広告コードが初めて成功した際に通知されます。 (任意)

- (void) nadViewDidReceiveAd:(NADView *)adView;

広告受信が成功した際に通知されます。 (任意)

- (void) nadViewDidFailToReceiveAd:(NADView *)adView;

広告受信に失敗した際に通知されます。 (任意)

4 - 2. アイコン型広告

4 - 2 - 1. アイコン型広告のサイズについて

(1) アイコン型広告ビューのサイズ指定

アイコン型広告ビューは、サイズ設定を行わない場合、幅 75px、高さ 75px となります。

(2) アイコン型広告ビュー内の配置

アイコン型広告ビューは、アイコン画像とテキストを表示します。

テキストは、表示、非表示の選択、色の変更が可能です。

サンプルプログラムと同様の手順で実装した際の配置は以下になります。

・デフォルト (75×75)



・縦 100、横 50 と指定した場合

縦横で異なるサイズが指定された場合、



小さい方のサイズで縦横を揃えます。

(3) アイコン型広告サイズと各種設定について

アイコン型広告は、余白の有無・タイトル文字列の有無・サイズ指定の有無によって下記の通りに画面上に表示されます。レイアウトを決める上での参考にしてください。

例) アイコンサイズを 50×50 と指定した場合

| | | タイトル文字列 | |
|----|---|---|---|
| | | 有 | 無 |
| 余白 | 有 |  |  |
| | 無 |  |  |
| | | 余白を含めた全体が 50×50 になります アイコン画像のサイズはSDK側で最適化 されます | 余白を含めた全体が 50×50 になります アイコン画像のサイズはSDK側で最適化 されます |
| | | アイコン画像のサイズが 50×50 になります タイトル文字列分の余白が下部に追加されます | アイコン画像のサイズが 50×50 になります |

4 – 2 – 2. アイコン型広告の実装手順

ここでは最もシンプルな構造を例にして NADIconView の実装方法について説明します。
それ以外の詳しい実装についてはサンプルソースをご参照ください。

○ヘッダーファイル

(1) delegate 準拠

Objective-C

広告ビューを保持するクラスのヘッダーファイルで NADIconLoader.h をインポート
DIconLoaderDelegate に準拠します。

例) YourAppViewController.h

```
#import <UIKit/UIKit.h>
#import "NADIconLoader.h"

@interface YourAppViewController : UIViewController <NADIconLoaderDelegate>
{
    NADIconLoader* iconLoader;
    NADIconView* iconView;
}

@end
```

Swift

事前に利用している **Objective-C Bridging Header** のヘッダーファイルに
NADIconLoader.h ファイルをインポートしてください。

例) YourAppViewController.swift

```
import UIKit

class YourAppViewController: UIViewController, NADIconLoaderDelegate {

    private var iconLoader: NADIconLoader!
    private var iconView: NADIconView!

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    ~省略~

}
```

注意：複数画面で構成されるアプリケーションで、各 ViewControllerごとにインスタンス生成するような実装方法の場合には、画面ごとに必ず後述の **(10) 定期ロードの管理**を行ってください。

○実装ファイル

Objective-C

例) YourAppViewController.m (一部) -NADIconLoader, NADIconView 生成からロードまで

```
- (void)viewDidLoad {
    [super viewDidLoad];

    // NADIconView クラスの生成
    iconView = [[NADIconView alloc] initWithFrame:CGRectMake(0, 0, 75, 75)];
    // NADIconView の配置
    [self.view addSubview:iconView];

    // NADIconLoader クラスの生成
    iconLoader = [[NADIconLoader alloc] init];

    // ログ出力の指定
    [iconLoader setIsOutputLog:YES];

    // NADIconLoaderへNADIconView を追加
    [iconLoader addIconView:iconView];

    // APIキーとSPOTID を設定
    [iconLoader setNendID:@"[管理画面より発行された apiKey]"
                  spotID:@"[管理画面より発行された spotID]"];

    // デリゲートオブジェクトの設定
    [iconLoader setDelegate:self];

    // 広告のロード
    [iconLoader load];
}
```

Swift

例) YourAppViewController.swift (一部) - NADIconLoader, NADIconView 生成からコードまで

```
import UIKit

class IconViewController: UIViewController, NADIconLoaderDelegate {

    private var iconLoader: NADIconLoader!
    private var iconView: NADIconView!

    override func viewDidLoad() {
        super.viewDidLoad()

        //NADIconViewクラスの生成
        iconView = NADIconView(frame: CGRect(x: 0, y: 100, width: 75, height: 75))

        //テキスト色の設定(任意)
        iconView.setTextColor(UIColor.whiteColor())

        //テキストの表示or非表示(任意)
        iconView.textHidden = true

        //画面上へ追加
        self.view.addSubview(iconView)

        //NADIconLoaderクラスの生成
        iconLoader = NADIconLoader()

        //loaderへ追加
        iconLoader.addIconView(iconView)

        //loaderへの設定
        iconLoader.setNendID("[管理画面より発行されたapiKey]",spotID: "[管理画面より発行されたspotID]")
        iconLoader.delegate = self
        iconLoader.isOutputLog = true

        //load開始
        iconLoader.load()
    }
}
```

(2) NADIconView 生成

任意の位置、75x75 サイズのCGRect を引数にインスタンスを生成します。

Objective-C

```
iconView = [[NADIconView alloc] initWithFrame:CGRectMake(0, 0, 75, 75)];
```

Swift

```
iconView: NADIconView = NADIconView(frame: CGRect(x: 0, y: 0, width: 75, height: 75))
```

※上記は、addSubview する先の View の 0,0 の位置を左上に指定して広告を配置する例です。

(3) NADIconLoader 生成

アイコン広告の情報を制御するクラスを生成します。

Objective-C

```
iconLoader = [[NADIconLoader alloc] init];
```

Swift

```
iconLoader = NADIconLoader()
```

(4) isOutputLog の設定

エラーログや警告ログを NSLog で出力するかどうかを指定します。

Objective-C

```
[iconLoader setIsOutputLog:YES];
```

Swift

```
iconLoader.isOutputLog = true
```

※初期値は YES です。出力したくない場合は NO をセットしてください。

(5) NADIconLoader に NADIconView の登録

NADIconLoader へ NADIconView を登録します。

Objective-C

```
[iconLoader addIconView:iconView];
```

Swift

```
iconLoader.addIconView(iconView)
```

※NADIconView は最大 6 つまで登録可能です。

(6) apiKey,spotID の設定

nend 管理画面で発行した、該当アプリの広告枠の apiKey、spotID をセットします。

Objective-C

```
[iconLoader setNendID:@"[管理画面より発行された apiKey]"  
    spotID:@"[管理画面より発行された spotID]"];
```

Swift

```
iconLoader.setNendID("[管理画面より発行された apiKey]",  
    spotID: "[管理画面より発行された spotID]")
```

※この時点で広告設定情報へのアクセスを開始します。

広告枠ステータスが「承認中」の場合、広告のロードをしても受信エラーになります。
nend 管理画面で該当アプリの広告枠ステータスが「アクティブ」であることを
確認してください。

広告枠の申請は、承認済みになるとステータスが「アクティブ」に変わり、登録メールアドレス宛に広告枠承認のお知らせが届きます。メール到着の数時間後には広告配信ができる状態になります。

単に実装方法の確認を行う場合は一時的に表示テスト用 ID ([→◆検証](#)) を利用するなどしてください。

(7) デリゲートオブジェクトの設定

NADIconLoader が広告を受信開始した場合に指定されたデリゲートの
nadIconLoaderDidFinishLoad メソッドを呼んで通知を行います。
指定するデリゲートは「nadIconLoaderDidFinishLoad」メソッドを実装する
NADIconViewDelegate」プロトコルに準拠させたクラスを指定します。

Objective-C

```
[iconLoader setDelegate:self];
```

Swift

```
iconLoader.delegate = self
```

(8) 広告のロード

NADView の load メソッドで広告のロードを開始します。

Objective-C

```
[iconLoader load];
```

Swift

```
iconLoader.load()
```

(9) Delegate 通知

【任意】ロード完了

広告のロードが完了すると(7)で指定したデリゲートの nadIconLoaderDidFinishLoad メソッドに通知されます。ロードが完了してから NADIconView を表示したい場合はここで行うことができます。

Objective-C

```
- (void)nadIconLoaderDidFinishLoad:(NADIconLoader *)iconLoader{
    NSLog(@"delegate nadIconLoaderDidFinishLoad:");
}
```

Swift

```
func nadIconLoaderDidFinishLoad(iconLoader: NADIconLoader!) {
    println("delegate nadIconLoaderDidFinishLoad")
}
```

【任意】アイコン広告クリック通知

表示されているアイコン広告をクリックした際に通知されます。

引数にクリックされた NADIconView が設定されます。

ただし、このイベントをカウントしても、ネットワーク状況や環境により「実際に任意の広告表示ができた数(サーバ側でのクリック数としてのカウント)」とは、異なる場合がありますので、注意してください。

Objectie-C

```
- (void)nadIconLoaderDidClickAd:(NADIconLoader *)iconLoader
    nadIconView:(NADIconView *)nadIconView{
    NSLog(@"delegate nadIconLoaderDidClickAd:");
}
```

Swift

```
func nadIconLoaderDidClickAd(iconLoader: NADIconLoader!, nadIconView:
    NADIconView!) {
    println("delegate nadIconLoaderDidClickAd")
}
```

【任意】広告受信通知

広告の受信に成功した場合通知されます。引数に広告を受信した NADIconView が設定されます。広告を受信するたびに任意の処理を行いたい場合に利用します。

Objective-C

```
- (void)nadIconLoaderDidReceiveAd:(NADIconLoader *)iconLoader  
    nadIconView:(NADIconView *)nadIconView{  
    NSLog(@"delegate nadIconLoaderDidReceiveAd:");  
}
```

Swift

```
func nadIconLoaderDidReceiveAd(iconLoader:NADIconLoader!, nadIconView:  
    NADIconView!) {  
    println("delegate nadIconLoaderDidReceiveAd")  
}
```

【任意】広告受信エラー通知

広告の受信に失敗した場合に通知されます。

通信エラー、広告在庫がなくなった場合など、何らかの理由で広告を表示できない場合に通知します。エラー時に広告を非表示にするなどの処理が必要な場合に利用します。

Objective-C

```
- (void)nadIconLoaderDidFailToReceiveAd:(NADIconLoader *)iconLoader  
    nadIconView:(NADIconView *)nadIconView{  
    NSLog(@"delegate nadIconLoaderDidFailToReceiveAd:");  
}
```

Swift

```
func nadIconLoaderDidFailToReceiveAd(iconLoader: NADIconLoader!, nadIconView:  
    NADIconView!) {  
    var error: NSError = iconLoader.error  
    println("delegate nadIconLoaderDidFailToReceiveAd")  
}
```

エラー内容によって処理を分ける必要がある場合は、次ページのように実装してください。

【任意】広告受信エラー通知

エラー内容によって処理を分けるサンプル

Objective-C

```
- (void)nadIconLoaderDidFailToReceiveAd:(NADIconLoader *)iconLoader
    nadIconView:(NADIconView *)nadIconView
{
    NSLog(@"delegate nadIconLoaderDidFailToReceiveAd:");

    // エラーごとに分岐する
    NSError* error = adView.error;
    NSString* domain = error.domain;
    int errorCode = error.code;

    // isOutputLog = NO でも、domain を利用してアプリ側で任意出力が可能
    NSLog(@"log %d", adView.isOutputLog);
    NSLog(@"%@",[NSString stringWithFormat: @"code=%d, message=%@",
                errorCode, domain]);

    switch (errorCode) {
        case NADVIEW_AD_SIZE_TOO_LARGE:
            // 広告サイズがディスプレイサイズよりも大きい
            break;
        case NADVIEW_INVALID_RESPONSE_TYPE:
            // 不明な広告ビュータイプ
            break;
        case NADVIEW_FAILED_AD_REQUEST:
            // 広告取得失敗
            break;
        case NADVIEW_FAILED_AD_DOWNLOAD:
            // 広告画像の取得失敗
            break;
        case NADVIEW_AD_SIZE_DIFFERENCES:
            // リクエストしたサイズと取得したサイズが異なる
            break;
        default:
            break;
    }
}
```

Swift

```
func nadIconLoaderDidFailToReceiveAd(iconLoader: NADIconLoader!, nadIconView: NADIconView!) {  
  
    // エラーごとに処理を分岐する  
    var error: NSError = iconLoader.error  
  
    switch (error.code){  
    case NADViewErrorCode.ADVIEW_AD_SIZE_TOO_LARGE.hashValue:  
        // 広告サイズがディスプレイサイズよりも大きい  
        break  
    case NADViewErrorCode.ADVIEW_INVALID_RESPONSE_TYPE.hashValue:  
        // 不明な広告ビュータイプ  
        break  
    case NADViewErrorCode.ADVIEW_FAILED_AD_REQUEST.hashValue:  
        // 広告取得失敗  
        break  
    case NADViewErrorCode.ADVIEW_FAILED_AD_DOWNLOAD.hashValue:  
        // 広告画像の取得失敗  
        break  
    case NADViewErrorCode.ADVIEW_AD_SIZE_DIFFERENCES.hashValue:  
        // リクエストしたサイズと取得したサイズが異なる  
        break  
    default:  
        break  
    }  
}
```

NADViewErrorCode (NADView.NSError.code) の内容

| NADViewErrorCode | エラー内容 |
|-------------------------------|-------------------------------------|
| NADVIEW_FAILED_AD_REQUEST | 広告取得失敗 (ネットワークエラー、サーバエラー、在庫切れなど) |
| NADVIEW_AD_SIZE_TOO_LARGE | 広告サイズがディスプレイサイズよりも大きい |
| NADVIEW_AD_SIZE_DIFFERENCES | リクエストしたサイズと取得したサイズが異なる※ |
| NADVIEW_INVALID_RESPONSE_TYPE | 不明な広告ビュータイプ※ |
| NADVIEW_FAILED_AD_DOWNLOAD | 広告画像の取得失敗 |

※基本的に発生することは稀です

(10) 定期ロードの管理

広告のロードを開始した後に画面内に表示しないケースがある場合には、必ず以下の処理を行ってください。

これには、

- 複数画面で構成された画面遷移が発生するアプリケーションにおいて各画面で個別に広告のインスタンスを生成している場合
- 複数広告の切り替え処理（広告スイッチング SDK の利用含む）を行う場合**
などが該当します。

定期ロードの中断

広告を画面内に表示しない、もしくは画面遷移等で View 自体が表示されない場合には pause メッセージを送信、広告の定期ロードを中断します。

例) 画面が隠れたら定期ロードを中断

Objective-C

```
- (void)viewWillDisappear:(BOOL)animated {
    [iconLoader pause];
}
```

Swift

```
override func viewWillDisappear(animated: Bool) {
    super.viewWillDisappear(animated)
    iconLoader.pause()
}
```

定期ロードの再開

広告を再び画面内に表示、または画面遷移等で View 自体を表示する場合には resume メッセージを送信、広告の定期ロードを再開します。

例) 画面が表示されたら定期ロードを再開

Objective-C

```
- (void)viewWillAppear:(BOOL)animated {
    [iconLoader resume];
}
```

Swift

```
override func viewWillAppear(animated: Bool) {
    super.viewWillAppear(animated)
    iconLoader.resume()
}
```

(11) リリース

dealloc 時には、必ず delegate プロパティに nil をセットしてからリリースを行うようにしてください。

Objective-C

```
- (void) dealloc {  
    [iconLoader setDelegate:nil]; // delegate に nil をセット  
    iconLoader = nil;           // プロパティ経由で release、nil をセット  
  
    // [super dealloc]; // MRC(非ARC 時には必要)  
}
```

Swift

```
deinit {  
    iconLoader.delegate = nil  
    iconLoader = nil;  
}
```

delegate プロパティに nil をセットしない場合、メモリの解放が適切に行われず、稀に予期しないクラッシュを引き起こす場合があります。必ず delegate に nil をセットしてから release を行うようにしてください。

また、delegate に nil がセットされた際、自動的に広告受信ローテーションを中断(pause)します。これにより release 前の pause メッセージ送信が不要になります。

○Interface Builder を使用した実装手順

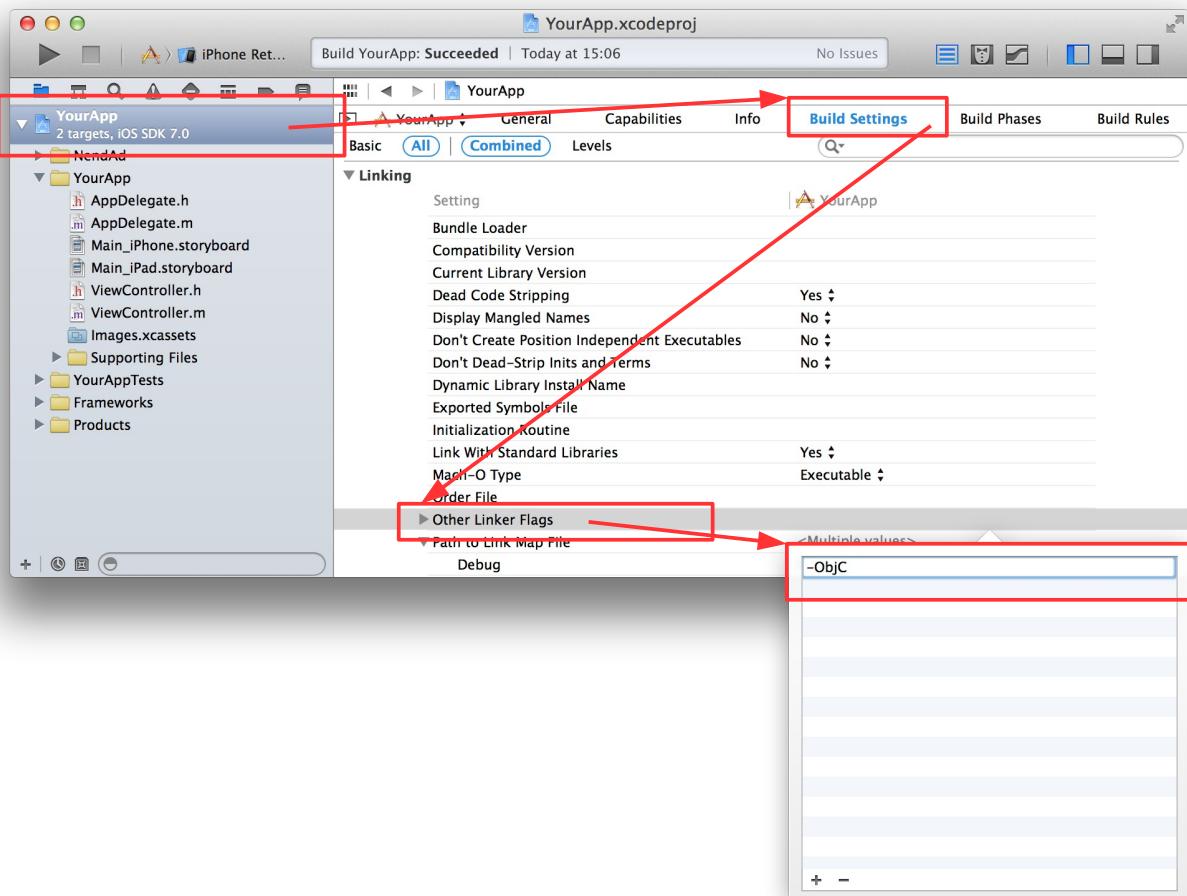
Interface Builder を使用してバナー広告の実装が可能です。

(※Deployment Target の設定が 5.1 未満ではご利用できません。)

(1)リンクフラグの追加

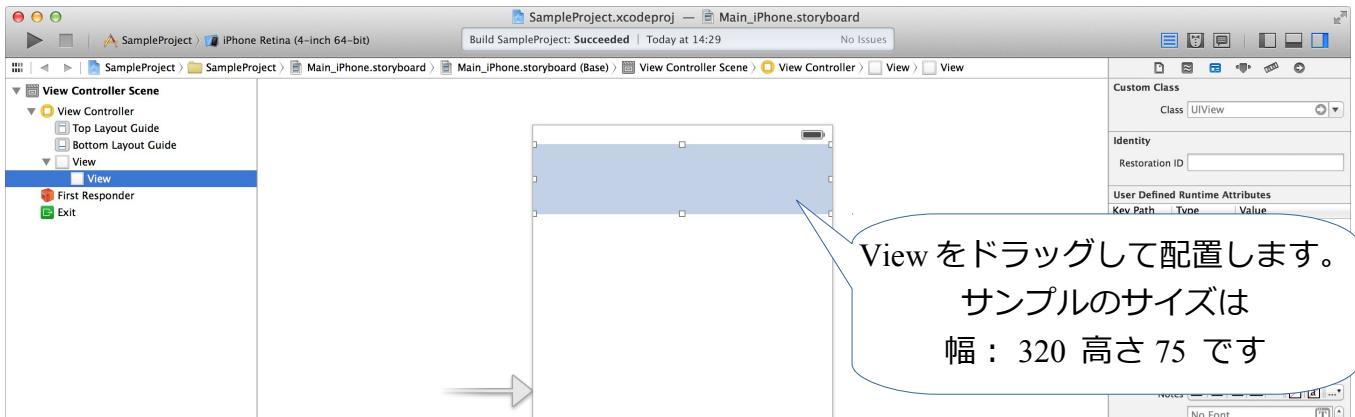
左側のプロジェクトナビゲーターから、プロジェクトをクリックして

TARGETS > Build Setting > Other Linker Flags 項目を開き、“-ObjC”を追加します。



(2)View の追加

広告を表示する View 上に View を追加します。



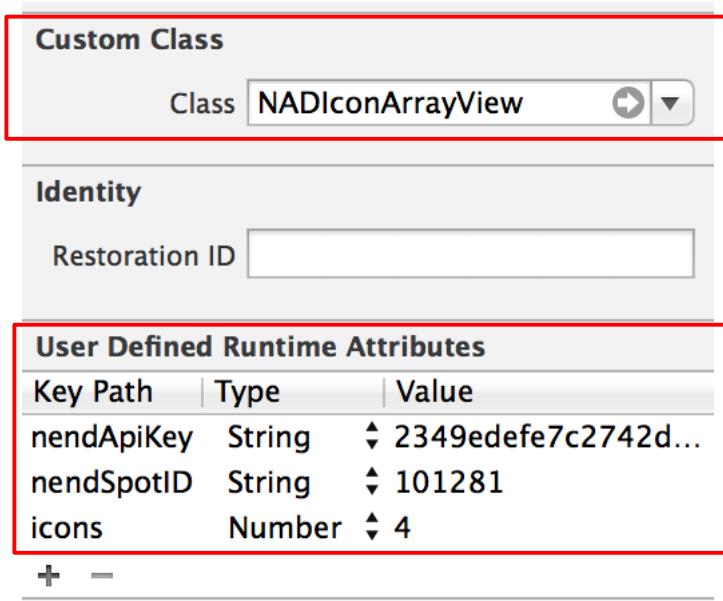
(3)クラスの変更、プロパティの設定

Utility area - Identity Inspector からクラスの変更とプロパティの変更を行います。

(1)CustomClass に **NADIconArrayView** を指定

(2)User Defined Runtime Attributes に以下の値を追加する

- nendApiKey に管理画面より発行された apiKey を設定
- nendSpotID に管理画面より発行された spotID を設定
- icons に表示するアイコン個数を設定

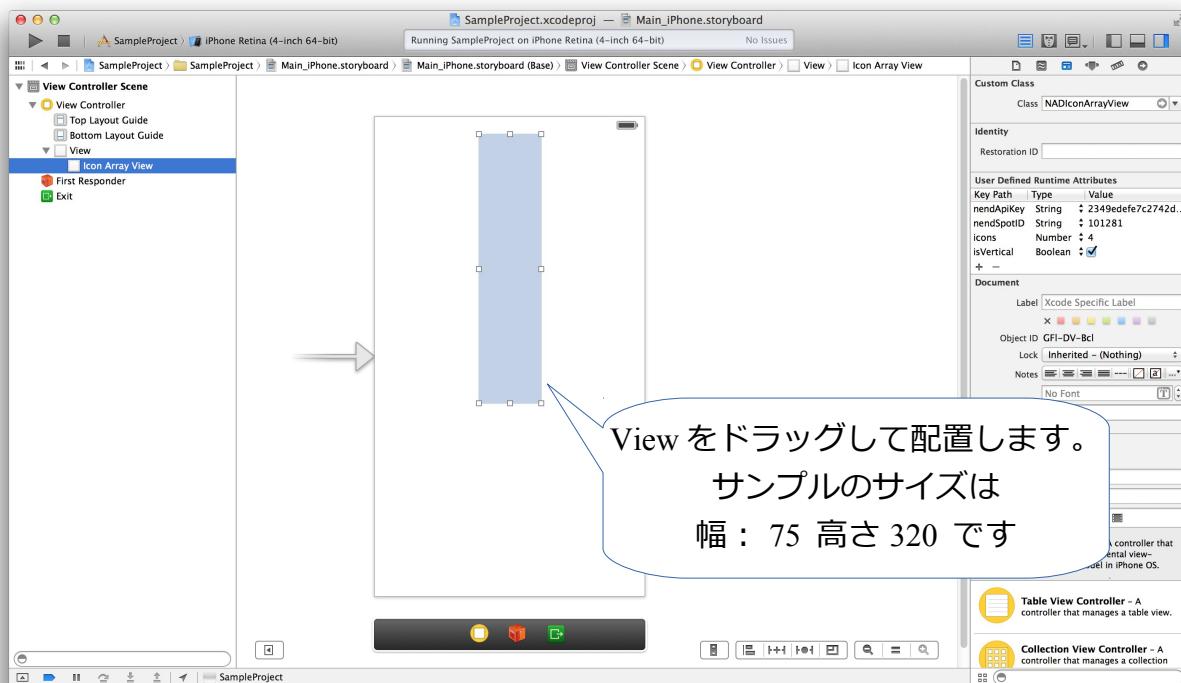


ビルドを行い、追加したViewにアイコンが表示できれば組込み完了です。

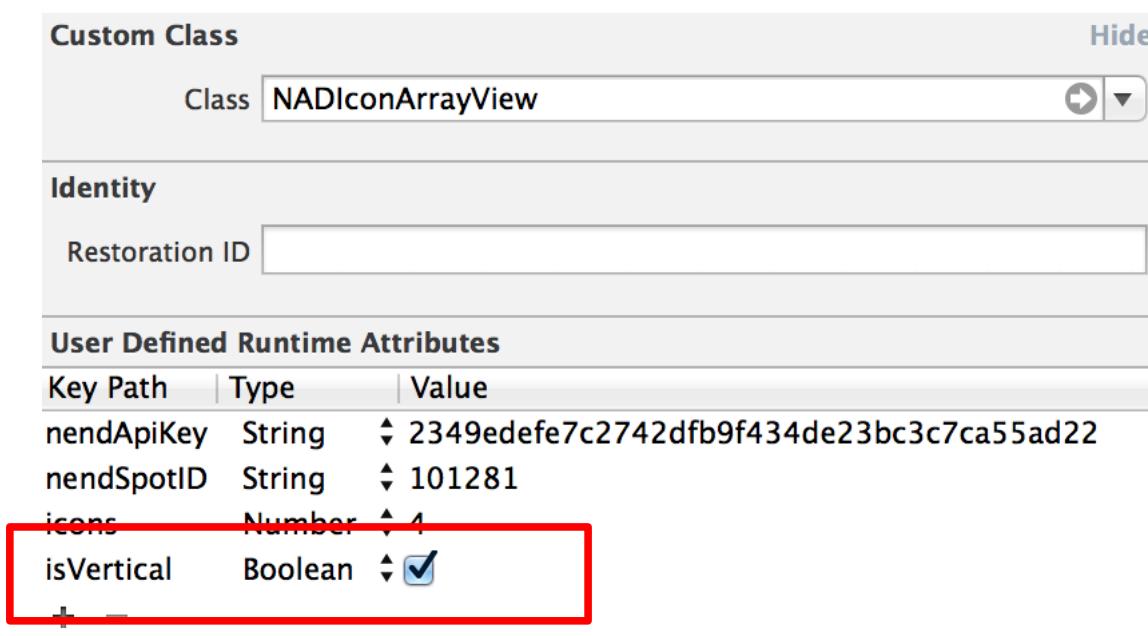


(4)アイコンを縦に並べる

縦に並べるための View を追加します。



User Defined Runtime Attributes に **isVertical** を追加し、チェックを入れます。



ビルドを行い、追加した View にアイコンが表示できれば組込み完了です。

(5) User Defined Runtime Attributes で設定可能な項目

User Defined Runtime Attributes で設定可能な項目および、初期値は以下となります。



| Key Path | Type | Value | 必須 or 任意 | 初期値 | 説明 |
|------------------|---------|--------------|----------|-----|--------------------|
| nendApiKey | String | 発行された apiKey | 必須 | - | 管理画面より発行された apiKey |
| nendSpotID | String | 発行された SpotID | 必須 | - | 管理画面より発行された SpotID |
| isOutputLog | Boolean | YES or NO | 任意 | NO | ログ出力可否 |
| textHidden | Boolean | YES or NO | 任意 | NO | タイトル表示可否 |
| iconSpaceEnabled | Boolean | YES or NO | 任意 | YES | 余白有無 |
| isVertical | Boolean | YES or NO | 任意 | NO | 表示方向(YES=縦, NO=横) |
| icons | Number | 1 ~ 6 | 任意 | 1 | アイコン表示数 |

以上で設定完了となります。

4 – 2 – 3 . NADIconLoader の内容

○メソッド

- **(void)addIconView:(NADIconView *)iconView;**

ローダーにアイコン広告のビューを登録します。

- **(void)removeIconView:(NADIconView *)iconView;**

ローダーからアイコン広告のビューを登録解除します。

- **(void)setNendID:(NSString *)apiKey spotID:(NSString *)spotID;**

広告枠の apiKey と spotID をセットします。

- **(void)load;**

ロードを開始します。

- **(void) pause;**

広告の定期ロード中断を要求します

- **(void) resume;**

広告の定期ロード再開を要求します

○プロパティ

@property (nonatomic, assign) id <NADIconLoaderDelegate> delegate;

delegate オブジェクトの指定 (任意)

@property (nonatomic) BOOL isOutputLog;

エラーログや警告ログを、 NSLog として出力するかどうかの指定 (任意)

@property (nonatomic, assign) NSError error;

参照すると受信エラー時にその内容を動的に知ることが出来ます。

○Delegate

- **(void) nadIconLoaderDidFinishLoad:(NADIconLoader *)iconLoader;**

広告ロードが初めて成功した際に通知されます。 (任意)

-**(void)nadIconLoaderDidReceiveAd:(NADIconLoader *)iconLoader**

nadIconView:(NADIconView*)nadIconView;

広告受信が成功した際に通知されます。 (任意)

```
- (void)nadIconLoaderDidFailToReceiveAd:(NADIconLoader *)iconLoader  
    nadIconView:(NADIconView*)nadIconView;
```

広告受信に失敗した際に通知されます。 (任意)

```
- (void)nadIconLoaderDidClickAd:(NADIconLoader *)iconLoader  
    nadIconView:(NADIconView*)nadIconView;
```

広告バナークリック時に通知されます。 (任意)

4 – 2 – 4 . NADIconView の内容

○メソッド

```
- (void)setTextColor:(UIColor *)setColor;
```

テキストの色を変更します。

○プロパティ

```
@property (nonatomic) BOOL textHidden;
```

テキストを表示するかどうかの指定

```
@property (nonatomic) BOOL iconSpaceEnabled;
```

余白部分を表示するかどうかの指定

4 – 2 – 5 . NADIconArrayView の内容

○メソッド

```
- (void) pause;
```

広告の定期ロード中断を要求します

```
- (void) resume;
```

広告の定期ロード再開を要求します

```
- (void)setTextColor:(UIColor *)setColor;
```

テキストの色を変更します。

○プロパティ

@property (nonatomic) BOOL isOutputLog;

エラーログや警告ログを、 NSLog として出力するかどうかの指定（任意）

@property (nonatomic) BOOL textHidden;

テキストを表示するかどうかの指定

@property (nonatomic) BOOL iconSpaceEnabled;

余白部分を表示するかどうかの指定

@property (nonatomic, assign) NSString* nendApiKey;

広告枠の apiKey の指定

@property (nonatomic, assign) NSString* nendSpotID;

広告枠の spotID の指定

@property (nonatomic) BOOL isVertical;

アイコンを縦に並べて表示するかどうかの指定

@property (nonatomic, copy) NSNumber* icons;

表示するアイコンの個数の指定

@property (nonatomic, readonly) NADIconLoader* iconLoader;

NADIconLoader の参照

4 – 3. インタースティシャル広告

4 – 3 – 1. インタースティシャル広告の実装手順

(1) 広告のロード

まず、NADInterstitial クラスに

```
[-(void) loadAdWithApiKey:(NSString *)apiKey spotId:(NSString *)spotId]  
メッセージを送信し、広告のロードを行います。
```

複数の広告枠を実装する場合は、広告枠ごとに別々の apiKey と SpotID を設定して広告のロードを実装してください。

※アプリ起動直後にロード処理を実装されることを推奨します。

Objective-C

例) UIApplicationDelegate プロトコルを実装したクラスの

```
[-(BOOL) application:(UIApplication *)application didFinishLaunchingWithOptions:  
(NSDictionary *)launchOptions] など
```

```
#import "NADInterstitial.h" // ヘッダファイルをインポートします  
  
@implementation YourAppDelegate  
  
- (BOOL) application:(UIApplication *)application  
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions  
{  
    ...  
    // NADInterstitial はシングルトンパターンで実装されています。  
    // 以下のように、sharedInstance メッセージでインスタンスを取得できます。  
    [[NADInterstitial sharedInstance] loadAdWithApiKey:@"管理画面より発行された ApiKey"  
                                                spotId:@"管理画面より発行された SpotId"];  
  
    return YES;  
}  
  
@end
```

Swift

事前に利用している **Objective-C Bridging Header** のヘッダーファイルに **NADInterstitial.h** ファイルをインポートしてください。

例) UIApplicationDelegate プロトコルを実装したクラスの

```
func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool など
```

```
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        // Override point for customization after application launch.

        NADInterstitial.sharedInstance().loadAdWithApiKey("[管理画面より発行されたapiKey]",
spotId: "[管理画面より発行されたspotID]")

        return true
    }

    ~省略~

}
```

また、NADInterstitial に delegate を設定することで、広告ロードの結果通知を受けることができます。詳細は後述の○プロパティおよび○NADInterstitialDelegate をご参照ください。

(2) 広告の表示

NADInterstitial クラスに[-(NADInterstitialShowResult) showAd]メッセージを送信することで、広告表示が行えます。広告の表示を行いたい場面で本メッセージを送信してください。

[- (NADInterstitialShowResult) showAdWithSpotId:(NSString *)spotId]で SpotID を指定してメッセージを送信することで、複数の広告枠を切り替えて表示することができます。

なお、広告の読み込みが完了していない場合、インアースティシャル広告は表示されません。

右上×ボタンまたはインアースティシャル広告の範囲外をタップすると、広告が非表示になります。
下部のインストールボタン（またはバナー広告）をタップすると、AppStoreへ遷移します。



戻り値の NADInterstitialShowResult の内容につきましては、以下のサンプルコードをご参照ください。

Objective-C

```
#import "NADInterstitial.h" // ヘッダファイルをインポートします

@implementation YourOriginalClass

- (void) showInterstitialAdSample
{
    NADInterstitialShowResult result = [[NADInterstitial sharedInstance] showAd];
    switch ( result )
    {
        case AD_SHOW_SUCCESS:
            NSLog(@"広告の表示に成功しました。");
            break;
        case AD_SHOW_ALREADY:
            NSLog(@"既に広告が表示されています。");
            break;
        case AD_FREQUENCY_NOT_REACHABLE:
            NSLog(@"広告のフリークエンシーカウントに達していません。");
            Break;
        case AD_LOAD_INCOMPLETE:
            NSLog(@"抽選リクエストが実行されていない、もしくは実行中です。");
            break;
        case AD_REQUEST_INCOMPLETE:
            NSLog(@"抽選リクエストに失敗しています。");
            break;
        case AD_DOWNLOAD_INCOMPLETE:
            NSLog(@"広告のダウンロードが完了していません。");
            break;
    }
}

- (void) showInterstitialAdSampleWithSpotID
{
    // SpotID を指定する場合
    NADInterstitialShowResult result = [[NADInterstitial sharedInstance]
showAdWithSpotId:@"管理画面より発行された SpotId"];
    ~省略~
}

@end
```

Swift

```
import UIKit

class YourOriginalClass: UIViewController {

    ~省略~

    func showButtonClicked(sender: UIButton) {

        var showResult: NADInterstitialShowResult
        showResult = NADInterstitial.sharedInstance().showAd()

        switch(showResult.value){
        case AD_SHOW_SUCCESS.value:
            println("広告の表示に成功しました。")
            break
        case AD_SHOW_ALREADY.value:
            println("既に広告が表示されています。")
            break
        case AD_FREQUENCY_NOT_REACHABLE.value:
            println("広告のフリークエンシーカウントに達していません。")
            break
        case AD_LOAD_INCOMPLETE.value:
            println("抽選リクエストが実行されていない、もしくは実行中です。")
            break
        case AD_REQUEST_INCOMPLETE.value:
            println("抽選リクエストに失敗しています。")
            break
        case AD_DOWNLOAD_INCOMPLETE.value:
            println("広告のダウンロードが完了していません。")
            break
        default:
            break
        }
    }

    func showWithSpotIdButtonClicked(sender: UIButton) {
        // SpotIDを指定する場合
        var showResult: NADInterstitialShowResult
        showResult = NADInterstitial.sharedInstance().showAdWithSpotId("管理画面より発行されたSpotId")

        ~省略~
    }
}
```

(3) 広告の非表示

NADInterstitial クラスに[-(BOOL) dismissAd]メッセージを送信することで、広告を非表示にすることができます。

Objective-C

```
#import "NADInterstitial.h" // ヘッダファイルをインポートします

@implementation YourOriginalClass

- (void) dismissInterstitialAdSample
{
    [[NADInterstitial sharedInstance] dismissAd];
}

@end
```

Swift

```
import UIKit

class YourOriginalClass: UIViewController {

    ~省略~

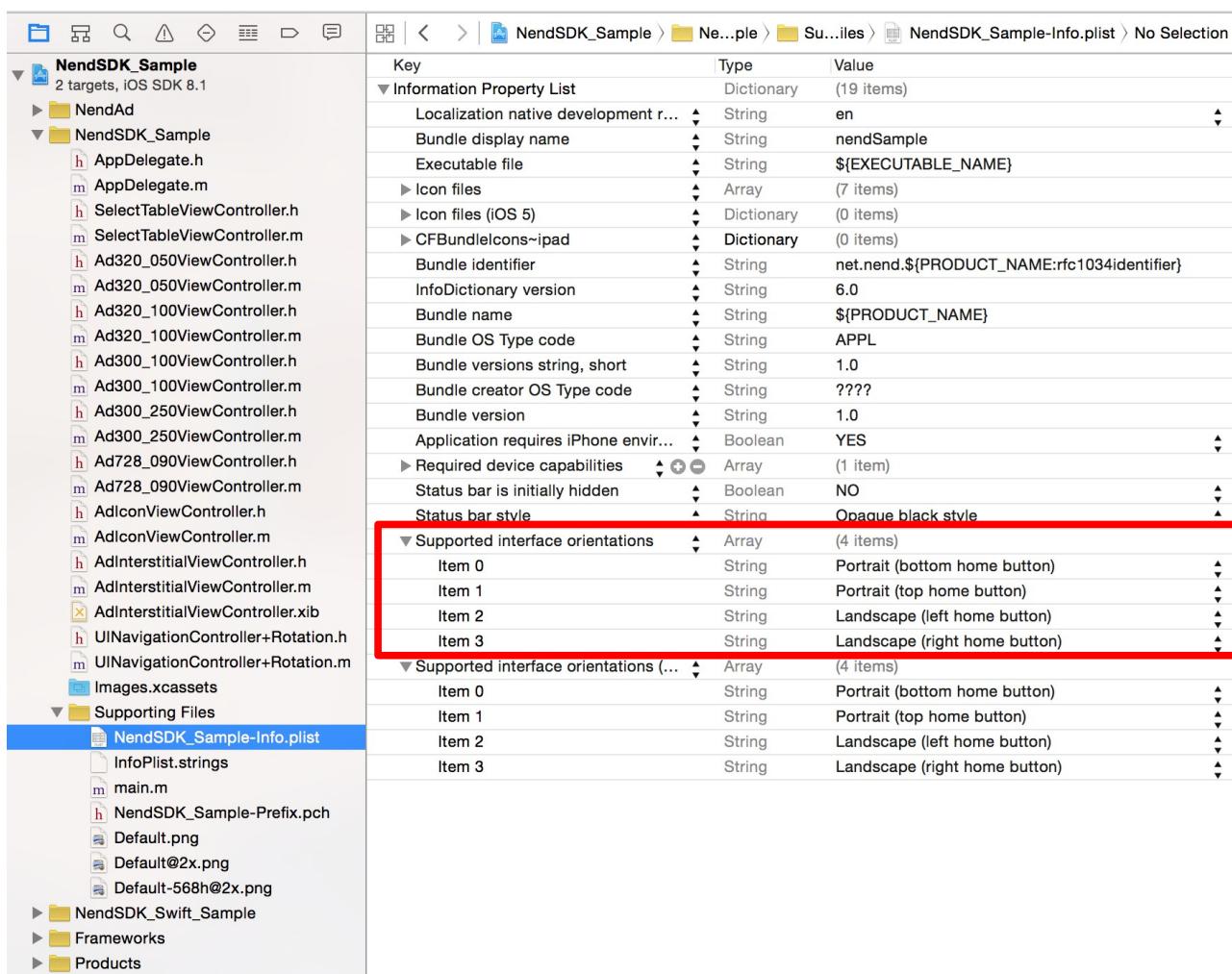
    func dismissInterstitialAdSample(){
        NADInterstitial.sharedInstance().dismissAd()
    }
}
```

4 – 3 – 2. NADInterstitial の内容

○インスタディシャル広告の端末の向き設定条件について

nendSDK for iOS ver 2.5.6 から、インスタディシャル広告のサポートする端末の向きを設定するには、**Info.plist** の **Supported interface orientations** で設定を行ってください。

例) 端末の全方向にインスタディシャル広告を対応させる場合



○プロパティ

@property (nonatomic, assign) id<NADInterstitialDelegate> delegate;
delegate オブジェクトの指定（任意）

※delegate に設定したオブジェクトが解放されるタイミングで、本プロパティに nil を設定してください。

@property (nonatomic) BOOL isOutputLog;
エラーログや警告ログを、 NSLog として出力するかどうかの指定（任意）

@property (nonatomic, retain) NSArray* supportedOrientations;
アプリがサポートする端末の向きを指定

以下の条件に当てはまるアプリの場合は、設定が必須となります。

- **Info.plist の Supported interface orientations の設定と異なる向きを特定の画面で設定する場合**

※以下のサンプルのように、 UIInterfaceOrientation 列挙型の値を NSNumber 型にし、 NSArray に格納してください。

Objective-C

```
#import "NADInterstitial.h" // ヘッダファイルをインポートします

@implementation YourOriginalClass

- (void) initNADInterstitialSample
{
    ...
    // 横向きのみのアプリの場合
    NSArray* array = @[[NSNumber
        numberWithInt:UIInterfaceOrientationLandscapeLeft],
        [NSNumber
        numberWithInt:UIInterfaceOrientationLandscapeRight]];
    [NADInterstitial sharedInstance].supportedOrientations = array;
}
@end
```

Swift

```
import UIKit

class YourOriginalClass: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // 横向きのみのアプリの場合
        var orientationArray =
            [UIInterfaceOrientationIsLandscape(UIInterfaceOrientation.LandscapeLeft),
             UIInterfaceOrientationIsLandscape(UIInterfaceOrientation.LandscapeRight)]
        NADInterstitial.sharedInstance().supportedOrientations = orientationArray

    }
    ~省略~
}
```

○メソッド

- **(instancetype) sharedInstance;**

インスタンスの生成および取得を行います

- **(void) loadAdWithApiKey:(NSString *)apiKey
spotId:(NSString *)spotId;**

広告のロードを開始します

- **(NADInterstitialShowResult) showAd;**

広告を表示します

- **(NADInterstitialShowResult) showAdWithSpotId:(NSString *)spotId;**

管理画面より発行された SpotID を指定して広告を表示します

- **(BOOL) dismissAd;**

広告を非表示にします

(広告が表示から非表示に変化する場合に YES を返します)

○NADInterstitialDelegate

- **(void) didFinishLoadInterstitialAdWithStatus**

: (NADInterstitialStatusCode)status;

広告のロード結果を通知します。 (任意)

- **(void) didFinishLoadInterstitialAdWithStatus**

: (NADInterstitialStatusCode)status

spotId:(NSString *)spotId;

ロード結果と対象の広告の SpotID を通知します。 (任意)

- **(void) didClickWithType:(NADInterstitialClickType)type;**

クリックイベントを通知します。 (任意)

- **(void) didClickWithType:(NADInterstitialClickType)type**

spotId:(NSString *)spotId;

クリックイベントと対象の広告の SpotID を通知します。 (任意)

引数の NADInterstitialStatusCode および NADInterstitialClickType の内容につきましては、次のサンプルコードをご参照ください。

Objective-C

```
#import "NADInterstitial.h" // ヘッダファイルをインポートします

@implementation YourOriginalClass <NADInterstitialDelegate>

- (void) initNADInterstitialSample
{
    // NADInterstitialDelegate プロトコルを実装したクラスを delegate プロパティにセットします。
    [NADInterstitial sharedInstance].delegate = self;
}

- (void) didFinishLoadInterstitialAdWithStatus:(NADInterstitialStatusCode)status
{
    switch ( status )
    {
        case SUCCESS:
            NSLog(@"広告のロードに成功しました。");
            break;
        case INVALID_RESPONSE_TYPE:
            NSLog(@"不正な広告タイプです。");
            break;
        case FAILED_AD_REQUEST:
            NSLog(@"抽選リクエストに失敗しました。");
            break;
        case FAILED_AD_DOWNLOAD:
            NSLog(@"広告のロードに失敗しました。");
            break;
    }
}

// ロード結果と対象の広告の SpotID の通知を受け取る場合
- (void) didFinishLoadInterstitialAdWithStatus:(NADInterstitialStatusCode)status spotId:
(NSString *)spotId
{
    ~省略~
}
```

```
- (void) didClickWithType:(NADInterstitialClickType)type
{
    switch ( type )
    {
        case DOWNLOAD:
            NSLog(@"ダウンロードボタンがクリックされました。");
            Break;
        case CLOSE:
            NSLog(@"閉じるボタンあるいは広告範囲外の領域がクリックされました。");
            break;
    }
}

// クリックイベントと対象の広告のSpotIDの通知を受け取る場合
- (void) didClickWithType:(NADInterstitialClickType)type spotId:(NSString *)spotId
{
    ~省略~
}

@end
```

Swift

```
import UIKit

class YourOriginalClass: UIViewController, NADInterstitialDelegate {

    override func viewDidLoad() {
        super.viewDidLoad()
        NADInterstitial.sharedInstance().delegate = self
    }
    ~省略~
    // MARK: NADInterstitialDelegate
    func didFinishLoadInterstitialAdWithStatus(status: NADInterstitialStatusCode) {
        switch(status.value){
        case SUCCESS.value:
            println("広告のロードに成功しました。")
            break
        case INVALID_RESPONSE_TYPE.value:
            println("不正な広告タイプです。")
            break
        case FAILED_AD_REQUEST.value:
            println("抽選リクエストに失敗しました。")
            break
        case FAILED_AD_DOWNLOAD.value:
            println("広告のロードに失敗しました。")
            break
        default:
            break
        }
    }

    // ロード結果と対象の広告のSpotIDの通知を受け取る場合
    func didFinishLoadInterstitialAdWithStatus(status: NADInterstitialStatusCode,
spotId: String!) {

    ~省略~

    }
}
```

```
func didClickWithType(type: NADInterstitialClickType) {  
    switch(type.value){  
        case DOWNLOAD.value:  
            println("ダウンロードボタンがクリックされました。")  
            break  
        case CLOSE.value:  
            println("閉じるボタンあるいは広告範囲外の領域がクリックされました。")  
            break  
        default:  
            break  
    }  
}  
  
// クリックイベントと対象の広告のSpotIDの通知を受け取る場合  
func didClickWithType(type: NADInterstitialClickType, spotId: String!) {  
  
    ~省略~  
}  
}
```

◆検証

iOS アプリ向け表示テスト用の apiKey と spotID を設定していただくことで
対象アプリケーション用の広告枠のステータスが「承認中」(非アクティブ)である場合でも
広告表示の確認ができます。

iOS アプリ向け表示テスト用 ID

| サイズ | apiKey | spotID |
|-----------------|--|--------|
| 320 x 50 | a6eca9dd074372c898dd1df549301f277c53f2b9 | 3172 |
| 320 x100 | eb5ca11fa8e46315c2df1b8e283149049e8d235e | 70996 |
| 300 x100 | 25eb32adddc4f7311c3ec7b28eac3b72bbca5656 | 70998 |
| 300 x250 | 88d88a288fdea5c01d17ea8e494168e834860fd6 | 70356 |
| 728 x 90 | 2e0b9e0b3f40d952e6000f1a8c4d455fffc4ca3a | 70999 |
| アイコン | 2349edefe7c2742dfb9f434de23bc3c7ca55ad22 | 101281 |
| インターナル ディシタル | 308c2499c75c4a192f03c02b2fcebd16dcb45cc9 | 213208 |

※確認後は必ず各アプリケーション用広告枠の apiKey と spotID に書き換えてください。

※テスト用 ID のままアプリケーションをストアへ申請、配布された場合、広告効果は
正しく集計されませんのでご注意ください。また、テスト用 ID のまま申請してしまった
場合、nend 側での保障等はできかねますのでご了承ください。

広告受信エラーの最も簡単な再現方法はオフラインにすることです。現状、nendSDK 内でオ
ンライン状況のチェックは行っておりませんので、オフライン時に何らかの処理を行う場合
は、アプリケーション側で実装する必要があります。

◆よくある質問

詳しくはメディアパートナー様向けヘルプをご覧ください

<https://www.nend.net/m/help/index/20>

お問合せ先

nend - お問合せフォーム

<https://www.nend.net/inquiries/form/>

※お問い合わせ時には、メディア登録名、apikey/spotID、SDKバージョン番号、必要に応じてご利用中の開発環境や端末機種の詳細などを情報としてお寄せ頂けますとより回答がスムースになります。