

# nendSDK for Android ver 2.5.2

## 設定ガイド

更新履歴	更新内容
バージョン 1.0 2011/11/11	新規作成
バージョン 1.1 2011/11/18	-
バージョン 1.2 2012/08/03	ANDROID_ID の利用を廃止、UUID へ変更 受信成功通知、受信エラー通知 追加 定期ロードの手動中断、手動再開メソッド 追加 広告サイズを 320*50 へ変更
バージョン 1.2.1 2012/08/16	メモリ関連の不具合を修正
バージョン 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 その他不具合修正
バージョン 2.0.1 2013/05/20	不具合修正
バージョン 2.1.0 2013/05/29	AdMob メディエーション対応
バージョン 2.2.0 2013/07/22	広告サイズ追加 NendError プロパティの追加
バージョン 2.2.1 2013/10/07	不具合修正
バージョン 2.2.2 2013/10/08	不具合修正
バージョン 2.3.0 2013/11/05	アイコン型広告追加
バージョン 2.3.1 2013/12/2	アイコン余白削除機能の実装 不具合修正
バージョン 2.3.2 2013/12/16	不具合修正
バージョン 2.3.3 2014/01/20	不具合修正
バージョン 2.4.0 2014/06/18	アニメーション GIF 対応 広告識別子 Google Advertising ID 利用開始
バージョン 2.4.1 2014/07/15	不具合修正
バージョン 2.5.0 2014/08/05	インタースティシャル広告追加
バージョン 2.5.1 2014/09/16	不具合修正 AdMob メディエーション用アダプタの更新
バージョン 2.5.2 2014/10/22	不具合修正

## 目次

◆nendSDK android について.....	5
1. nendSDK 導入のおおまかな流れ.....	5
2. ファイル構成.....	5
3. 対応環境.....	5
4. nendSDK 取得項目について.....	6
5. インフォメーションボタンについて.....	7
◆SDK の組み込み.....	8
1. jar ファイルの追加.....	8
2. Google Play services の追加.....	9
2-1. Google Play services のインストール.....	9
2-2. Google Play services プロジェクトをインポート.....	10
2-3. Google Play services への参照を追加.....	11
2-4. 注意事項.....	12
3. マニフェスト設定.....	13
4. 広告ビューの配置.....	14
4-1. バナー型広告.....	14
4-1-1. レイアウトファイルで固定配置する場合.....	14
4-1-2. Java プログラムから動的に呼び出す場合 .....	15
4-1-3. 広告サイズについて.....	17
4-1-4. ログ出力.....	19
4-1-5. 受信エラー通知について.....	19
4-1-6. 定期ロードについて.....	20
4-2. アイコン型広告.....	21
4-2-1. アイコン型広告のサイズについて.....	21
1) アイコン型広告ビューのサイズ指定.....	21
2) アイコン型広告ビュー内の配置.....	21
3) アイコン型広告サイズと各種設定について.....	22
4-2-2. アイコンを横（もしくは縦）に並べて表示する場合.....	22
1) XML レイアウトファイルで配置する場合.....	23
2) Java プログラムから動的に呼び出す場合.....	24
3) 方向指定.....	24
4) イベントリスナ.....	25
5) タイトル文字列の設定.....	26
6) 余白の設定.....	27
4-2-3. アイコンを 1 個ずつ個別に扱う場合.....	28
1) XML レイアウトファイルで配置する場合.....	28
2) Java プログラムから動的に呼び出す場合.....	30

3) アイコンの個数について.....	30
4) イベントリスナ.....	31
5) タイトル文字列の設定.....	32
6) 余白の設定.....	33
4-2-4. 受信エラー通知について.....	34
4-2-5. ログ出力について.....	35
4-2-6. 定期ロードについて.....	35
4-3. インタースティシャル広告.....	36
4-3-1. インタースティシャル広告.....	36
1) 広告のロード.....	36
2) 広告の表示.....	36
4-3-2. 終了時広告.....	37
1) 広告のロード.....	37
2) 広告の表示.....	37
4-3-3. 画面回転について.....	38
4-3-4. イベントリスナ.....	38
1) ロードイベントリスナ.....	38
2) クリックイベントリスナ.....	39
4-3-5. 表示結果取得.....	40
◆検証.....	41
◆サンプルについて.....	42
◆よくある質問.....	43

## ◆nendSDK android について

### 1. nendSDK 導入のおおまかな流れ

① nend の管理画面でアプリ登録、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。

- ※ 本マニュアルは、android アプリ用 nendSDK 導入に関してのマニュアルとなります。

### 2. ファイル構成

NendSDK_Android-2.5.2.zip		
AdMobMediationAdapter/		
nendAdapter_readme.txt		AdMob メディエーション用アダプタについて
nendAdapter-1.2.0.jar		AdMob メディエーション用アダプタ
Nend/		
nendSDK_Android_Guide.pdf		本マニュアル
NendSample/		サンプルソースフォルダ
SDK/		SDK フォルダ
nendSDK-2.5.2.jar		ライブラリ

### 3. 対応環境

OSバージョンは、AndroidOS 2.3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。

## 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](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](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いいたします。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 I D                              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 I D）
- ・Google Advertising ID                      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 Google 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

- ・パッケージ名                      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ナーに「インフォメーションボタン」が表示されます。

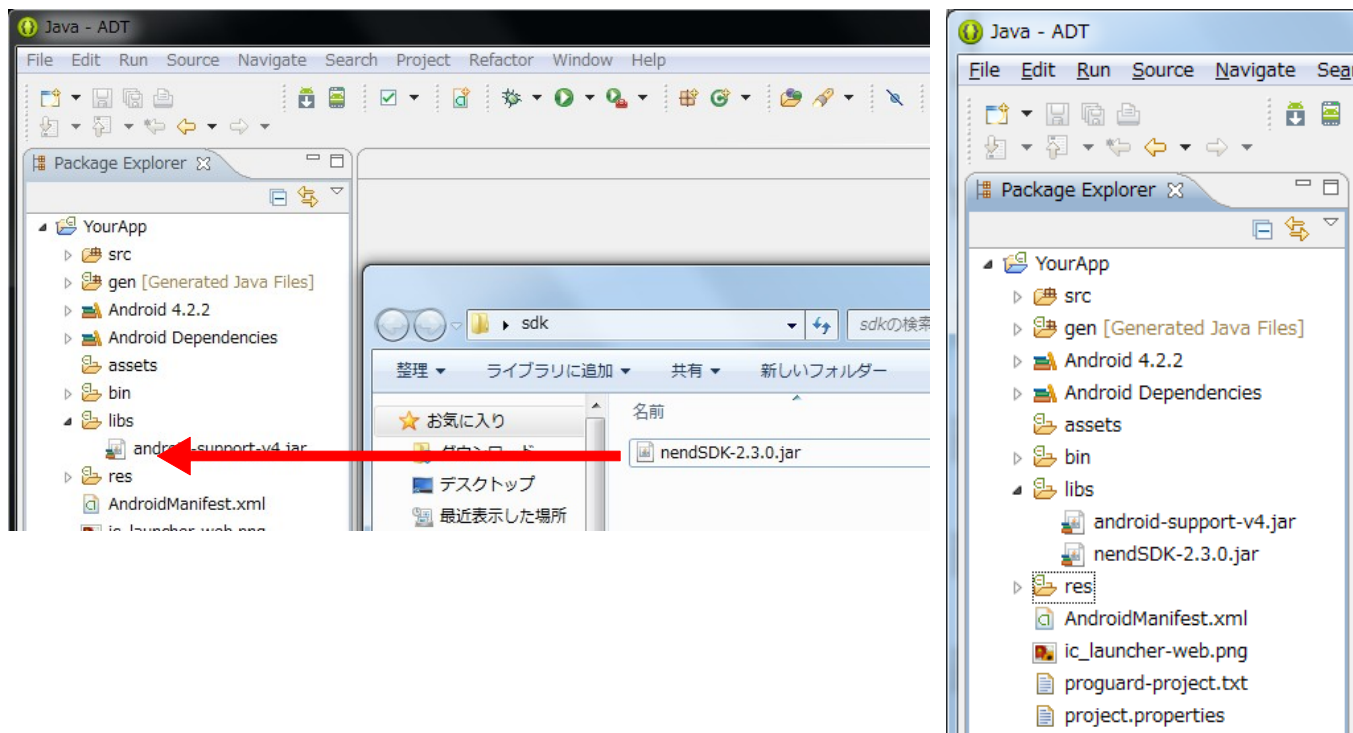
この機能が追加された nendSDK を利用することにより、nend が提供する一部ターゲティング広告に対して、よりユーザ（アプリ利用者）自身が容易にオプトアウトの設定をすることが可能になります。

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. jar ファイルの追加

Eclipse プロジェクト内の libs フォルダに『nendSDK-X.X.X.jar』を配置します。



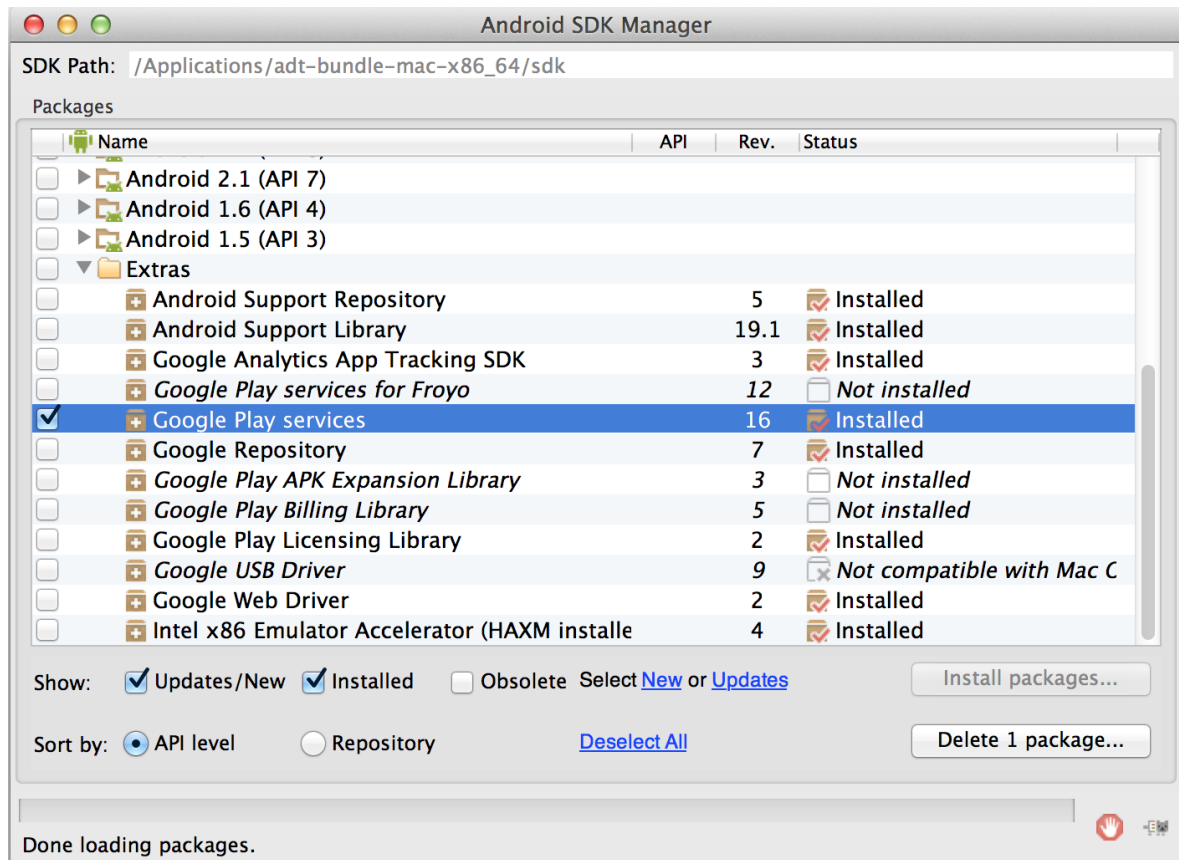


## 2. Google Play services の追加

### 2-1. Google Play services のインストール

Android SDK Manager を開き、Google Play Services をインストールします。

(※既にインストール済みの場合でも、更新可能な場合は最新版にアップデートしてください)

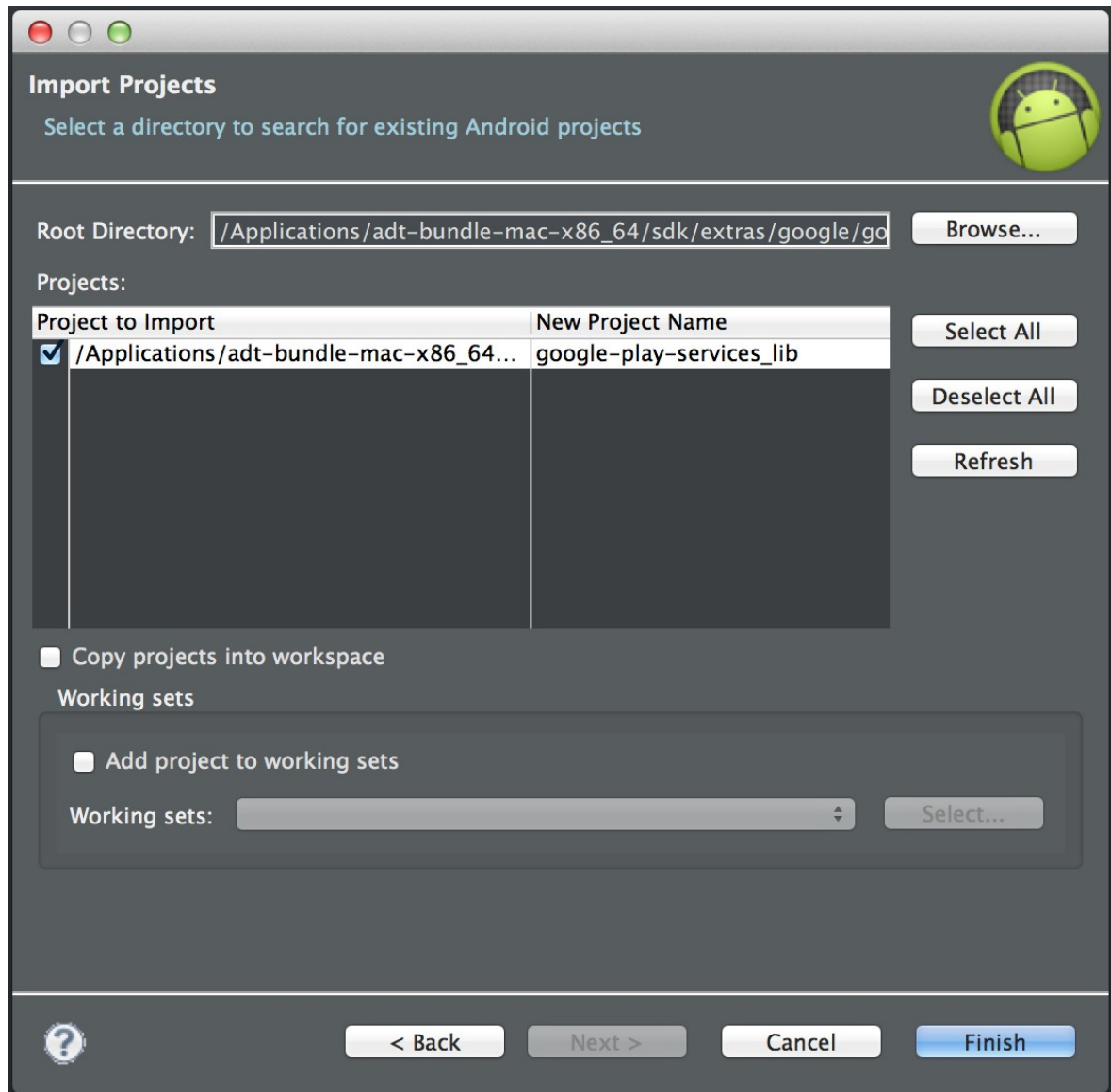


## 2-2. Google Play servicesプロジェクトをインポート

前項でインストールした google-play-services\_lib プロジェクトをインポートします。

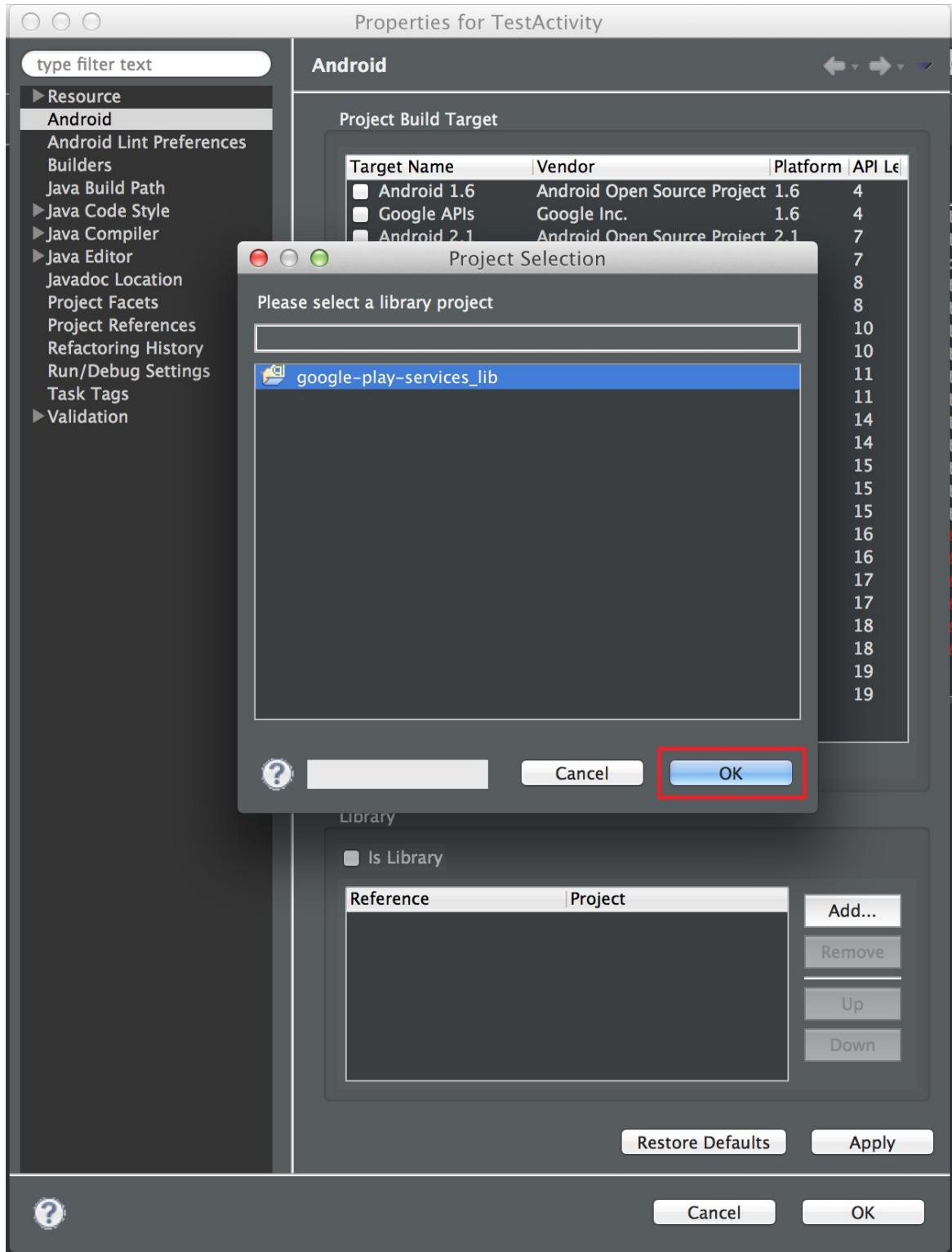
※google-play-services\_lib は以下のフォルダにインストールされています。

**AndroidSDK フォルダ/sdk/extras/google/google\_play\_services/libproject**



## 2-3. Google Play services への参照を追加

開発中のプロジェクトにて、google-play-services\_lib プロジェクトへの参照を追加します。



## 2-4. 注意事項

※Google Play services には Google Mobile Ads SDK が含まれています。  
そのため、メディエーションで Google Mobile Ads SDK をプロジェクトに追加している場合は、クラスの多重定義でビルドエラーが発生してしまいます。  
このエラーは、Google Mobile Ads SDK をプロジェクトから削除していただくことで回避することができます。

### 3. マニフェスト設定

#### AndroidManifest.xml の編集

1. <manifest>～</manifest>セクション内に以下の記述を追加します

```
<uses-permission android:name="android.permission.INTERNET" />
```

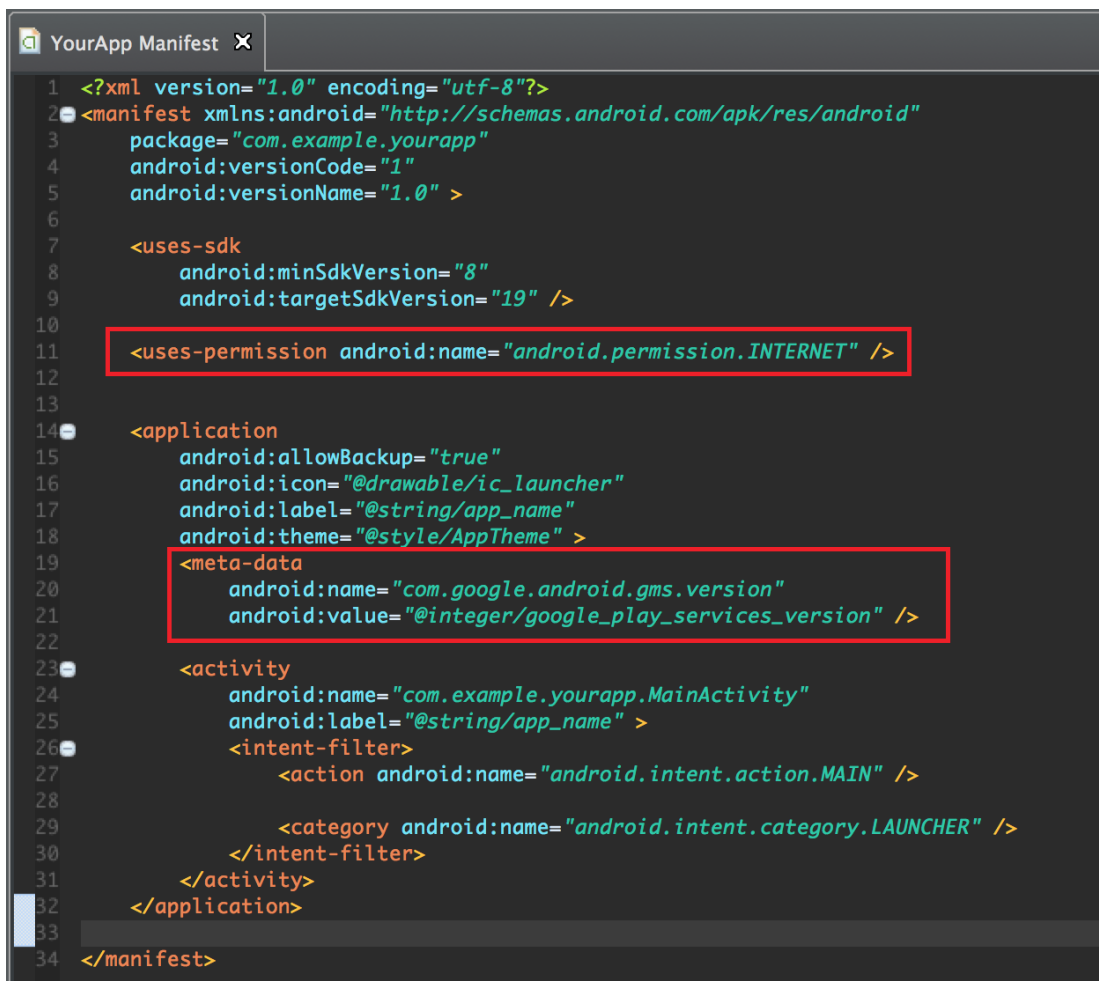
nendSDK を利用する際に設定が必要なパーミッション

#### INTERNET

ネットワーク経由にて広告を取得するために設定するパーミッションです。

2. <application>～</application>セクション内に以下の記述を追加します

```
<meta-data  
    android:name="com.google.android.gms.version"  
    android:value="@integer/google_play_services_version" />
```



```
1 <?xml version="1.0" encoding="utf-8"?>  
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"  
3     package="com.example.yourapp"  
4     android:versionCode="1"  
5     android:versionName="1.0" >  
6  
7     <uses-sdk  
8         android:minSdkVersion="8"  
9         android:targetSdkVersion="19" />  
10  
11     <uses-permission android:name="android.permission.INTERNET" />  
12  
13  
14     <application  
15         android:allowBackup="true"  
16         android:icon="@drawable/ic_launcher"  
17         android:label="@string/app_name"  
18         android:theme="@style/AppTheme" >  
19         <meta-data  
20             android:name="com.google.android.gms.version"  
21             android:value="@integer/google_play_services_version" />  
22  
23         <activity  
24             android:name="com.example.yourapp.MainActivity"  
25             android:label="@string/app_name" >  
26             <intent-filter>  
27                 <action android:name="android.intent.action.MAIN" />  
28  
29                 <category android:name="android.intent.category.LAUNCHER" />  
30             </intent-filter>  
31         </activity>  
32     </application>  
33  
34 </manifest>
```

## 4. 広告ビューの配置

### 4-1. バナー型広告

#### 4-1-1. レイアウトファイルで固定配置する場合

ここではレイアウトファイルのみで設定する場合の方法を説明します。  
付属の NendSample プロジェクト内の『XmlLayoutActivity.java』と同様です。

該当する画面のレイアウトファイル内に  
『net.nend.android.NendAdView』のタグを追加し  
『apiKey』と『spotID』の値を挿入してください。

#### サンプル 1

```
<net.nend.android.NendAdView
    android:id="@+id/nend"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    NendApiKey="[管理画面より発行された apiKey]"
    NendSpotId="[管理画面より発行された spotID]" />
```

#### サンプル 2

画面中央最下部への設置例

詳しいレイアウト方法は android ドキュメントをご確認ください。

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center|bottom">
    <net.nend.android.NendAdView
        android:id="@+id/nend"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        NendApiKey="[管理画面より発行された apiKey]"
        NendSpotId="[管理画面より発行された spotID]" />
</LinearLayout>
```

## 4-1-2. Java プログラムから動的に呼び出す場合

ここでは Java プログラムのみで設定する場合の方法を説明します。

付属の NendSample プロジェクト内の

『JavaCallLinearActivity.java』 『JavaCallWithListenerActivity.java』 と同様です。

※Java プログラムから動的に呼び出す場合、広告取得のために NendAdView オブジェクトの loadAd() メソッドの呼び出しが必要になりました。  
旧バージョン SDK から利用しているアプリなどでメソッドの呼び出しがない場合、  
下記サンプルを参照して、loadAd() メソッドの呼び出しを追加してください。

### (1) 配置するだけのサンプル

```
public class JavaCallLinearActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.javacall_linear);

        LinearLayout rootLayout = (LinearLayout) findViewById(R.id.root);

        // ①NendAdView をインスタンス化
        NendAdView nendAdView = new NendAdView(
            getApplicationContext(), [発行された spotID], "[発行された apiKey]");

        // 中央下部表示の場合
        rootLayout.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.BOTTOM);
        // ②NendAdView をレイアウトに追加
        rootLayout.addView(nendAdView,
            new LinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        // ③ 広告の取得を開始
        nendAdView.loadAd();
    }
}
```

## (2) イベントリスナを利用する場合のサンプル

```
public class JavaCallWithListenerActivity extends Activity implements NendAdListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.javacall_relative);

        RelativeLayout rootLayout = (RelativeLayout) findViewById(R.id.root);

        NendAdView nendAdView = new NendAdView(
            getApplicationContext(), [発行された spotID], "[発行された apiKey]");
        // ② リスナーを登録
        nendAdView.setListener(this);

        RelativeLayout.LayoutParams params = new RelativeLayout.LayoutParams(
            LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
        params.addRule(RelativeLayout.CENTER_HORIZONTAL);
        params.addRule(RelativeLayout.ALIGN_PARENT_BOTTOM);

        rootLayout.addView(nendAdView, params);
        nendAdView.loadAd();
    }

    // ③ 通知を受けるメソッドを用意
    // 受信エラー通知
    public void onFailedToReceiveAd(NendAdView nendAdView) {
        Toast.makeText(getApplicationContext(), "onFailedToReceiveAd", Toast.LENGTH_LONG).show();
    }
    // 受信成功通知
    public void onReceiveAd(NendAdView nendAdView) {
        Toast.makeText(getApplicationContext(), "onReceiveAd", Toast.LENGTH_LONG).show();
    }
    // クリック通知
    public void onClick(NendAdView nendAdView) {
        Toast.makeText(getApplicationContext(), "onClick", Toast.LENGTH_LONG).show();
    }
    // 復帰通知
    public void onDismissScreen(NendAdView nendAdView) {
        Toast.makeText(getApplicationContext(), "onDismissScreen", Toast.LENGTH_LONG).show();
    }
}
```

このサンプルの場合は、受信が成功/失敗するたびにトースト表示されます。

SDKver2.1.0 からクリック通知、復帰通知用のメソッドが追加されました。旧 SDK の NendAdListener を使用している場合、SDK のバージョンアップの際に適宜メソッドの実装の追加をお願い致します。

その他のサンプルについては「[サンプルについて](#)」を参照してください。



### 4-1-3. 広告サイズについて

以下の広告サイズが使用出来ます。

- 320 × 50
- 320 × 100
- 300 × 100
- 300 × 250
- 728 × 90

※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。

### 広告ビューサイズの指定

広告ビューを作成する際に、レイアウトの幅、高さは「wrap\_content」もしくは「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。

異なったサイズを指定した場合、広告が拡大/縮小表示されてしまいます。

その場合、正常な広告表示としてカウントされない場合があります。

#### 【例】

下記広告枠がサイズ 300×250 で作成されていた場合

wrap\_content を指定する例

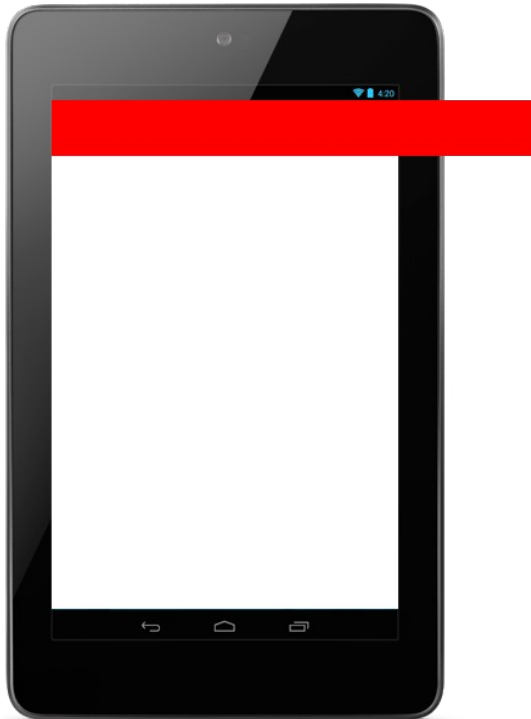
```
<net.nend.android.NendAdView
    android:id="@+id/nend"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    NendApiKey="499f011dbec5d37cfa388b749aed2bfff440a794"
    NendSpotId="70357 " />
```

数値を指定する例

```
<net.nend.android.NendAdView
    android:id="@+id/nend"
    android:layout_width="300dp"
    android:layout_height="250dp"
    NendApiKey="499f011dbec5d37cfa388b749aed2bfff440a794"
    NendSpotId="70357 " />
```

## 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、広告の表示を行いません。  
アプリ側へは受信エラーが通知され、定期ロードも自動的に停止します。



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。

【例：Nexus7 (1280×800px) に 728×90px の広告を表示する場合】

Nexus7 は解像度グループが tvdpi で、解像度比率が 1.3312501 であるため、728×90px の広告を表示するためには、969×120px 以上のディスプレイサイズが必要です。  
Nexus7 では、728×90px の広告は縦では表示されず、横では表示されます。

#### 4-1-4. ログ出力

AndroidManifest.xml の<application>セクション内に、以下のメタタグを追加することにより、LogCat にログ出力を行うことが出来ます。

value="false"もしくは設定自体を削除した場合はエラーログを出力しません。

```
<meta-data
    android:name="NendDebuggable"
    android:value="true" />
```

#### 4-1-5. 受信エラー通知について

受信エラー通知でエラー内容を取得出来ます。

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

NendAdView から getNendError メソッドで NendError オブジェクトを取得できます。

NendError からはエラーコードとエラーメッセージを取得出来ます。

```
@Override
public void onFailedToReceiveAd(NendAdView adView) {

    NendError nendError = adView.getNendError();
    switch (nendError) {
        case INVALID_RESPONSE_TYPE:
            // 不明な広告ビュータイプ
            break;
        case FAILED_AD_DOWNLOAD:
            // 広告画像の取得失敗
            break;
        case FAILED_AD_REQUEST:
            // 広告取得失敗
            break;
        case 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 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;
    }
    // エラーメッセージをログに出力
    Log.e(TAG, nendError.getMessage());
}
```

#### Enum NendError の内容

定数	エラー内容
INVALID_RESPONSE_TYPE	不明な広告ビュータイプ
FAILED_AD_DOWNLOAD	広告画像の取得失敗
FAILED_AD_REQUEST	広告取得失敗
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
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

#### 4-1-6. 定期ロードについて

android 版 nendSDK では、ネットワーク利用ができない場合、また、フォーカスを失った場合や広告が乗った画面が背面にある等広告が表示されていない間には、自動的に広告受信ローテーションを中断します。

それ以外で任意にローテーションを中断させたい場合には、以下のメソッドを利用してください。

#### NendAdView

中断

```
void pause();
```

再開

```
void resume();
```

**注意：**手動での中断が優先になります。手動で中断させている間は、広告が乗った画面が最前面に表示されても、ローテーションは再開されません。手動でローテーションを中断した場合は、必ず手動でローテーションを再開するようにしてください。

## 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 75dip、高さ 75dip 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。

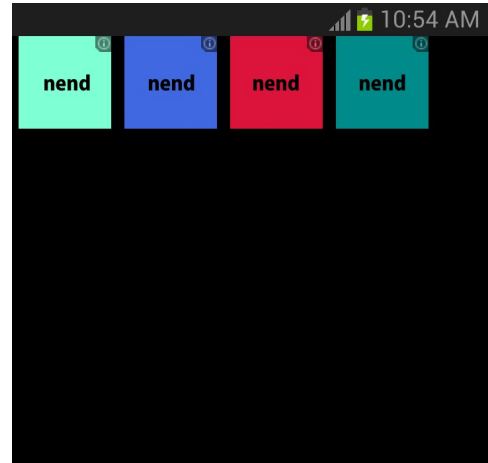
レイアウトを決める上での参考にしてください。

#### 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	 <p>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</p>	 <p>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</p>
	無	 <p>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</p>	 <p>アイコン画像のサイズが50×50になります</p>

#### 4-2-2. アイコンを横（もしくは縦）に並べて表示する場合

ここでは最も簡単な実装方法として、右の画像のように横（もしくは縦）にアイコンを並べて表示する方法を説明します。



### 1) XML レイアウトファイルで配置する場合

ここではレイアウトファイルを用いて、アイコン広告の表示を行う場合の方法を説明します。付属の NendSample プロジェクト内の『IconLayoutActivity.java』と同様です。

該当する画面の XML レイアウトファイル内に『net.nend.android.NendAdIconLayout』のタグを追加します。

NendIconCount に表示するアイコンの個数（最大 6）、NendApiKey に『apiKey』、NendSpotId に『spotID』の値をそれぞれ挿入してください。

※NendIconCount に 7 以上の数値を設定しても、6 として処理が行われます。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <net.nend.android.NendAdIconLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        NendIconCount="4"
        NendApiKey="e6f2c6be591b3b2583844e8ea4f73ea16dfc8512"
        NendSpotId="22706" />
</RelativeLayout>
```

これでアイコン広告の表示が可能です。

上記の例では、画面上部にアイコン 4 個を横並びに表示しています。

## 2) Java プログラムから動的に呼び出す場合

ここでは Java プログラムのみで、アイコン広告の表示を行う場合の方法を説明します。

広告を表示するだけの基本的な実装例になります。

- ① spotID と apiKey、アイコン数を指定して、NendAdIconLayout を生成
- ② loadAd メソッドを実行し、広告取得を開始する。
- ③ レイアウトに追加する。

```
public class IconLayoutActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.icon_layout);  
  
        RelativeLayout rootLayout = (RelativeLayout) findViewById(R.id.root);  
        // ①spotID と apiKey、アイコン数を指定して、NendAdIconLayout を生成  
        NendAdIconLayout iconLayout = new NendAdIconLayout(getApplicationContext(),  
            [発行された spotID], [発行された apiKey], [表示するアイコンの個数]);  
        // ②loadAd メソッドを実行し、広告取得を開始する  
        iconLayout.loadAd();  
        // ③ レイアウトに追加する。  
        rootLayout.addView(iconLayout);  
    }  
}
```

## 3) 方向指定

アイコンの並ぶ方向を指定する場合は、horizontal（デフォルト）もしくは vertical を指定します。

Java プログラムで設定

```
NendAdIconLayout iconLayout = new NendAdIconLayout(getApplicationContext(),  
    [発行された spotID], [発行された apiKey], [表示するアイコンの個数]);  
iconLayout.setOrientation(NendAdIconLayout.VERTICAL);
```

XML レイアウトファイルで設定

```
<net.nend.android.NendAdIconLayout  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    NendIconCount="[表示するアイコンの個数]"  
    NendApiKey="[発行された apiKey]"  
    NendSpotId="[発行された spotID]"  
    NendOrientation="vertical" />
```



#### 4) イベントリスナ

下記のように、必要に応じて NendAdIconLayout にリスナを登録することで、通知を受けることができます。

```
NendAdIconLayout iconLayout = new NendAdIconLayout(getApplicationContext(),
    [発行された spotID], "[発行された apiKey]", [表示するアイコンの個数]);

// 受信成功通知
iconLayout.setOnReceiveLisner(new OnReceiveListner() {
    @Override
    public void onReceiveAd(NendAdIconView iconView) {
        // 通知元のアイコンによって処理を変える場合
        switch(iconView.getId()){
            case R.id.icon1:
                // TODO
                break;
            case R.id.icon2:
                // TODO
                break;
            case R.id.icon3:
                // TODO
                break;
            case R.id.icon4:
                // TODO
                break;
        }
        Toast.makeText(getApplicationContext(), "Recieved", Toast.LENGTH_SHORT).show();
    }
});

// クリック通知
iconLayout.setOnClickListner(new OnClickListner() {
    @Override
    public void onClick(NendAdIconView iconView) {
        Toast.makeText(getApplicationContext(), "Clicked", Toast.LENGTH_SHORT).show();
    }
});

// エラー通知
iconLayout.setOnFailedListner(new OnFailedListner() {
    @Override
    public void onFailedToReceiveAd(NendIconError iconError) {
        Toast.makeText(getApplicationContext(), "Failed", Toast.LENGTH_SHORT).show();
    }
});
```

## 5) タイトル文字列の設定

アイコン下部に表示されるタイトル文字列に対して、次のように設定を行うことができます。

※設定した内容が全てのアイコンに適用されます。

### ・表示/非表示設定

表示する場合は true（デフォルト）、非表示にする場合は false を設定します。

Java プログラムで設定

```
NendAdIconLayout iconLayout = new NendAdIconLayout(getApplicationContext(),  
    [発行された spotID], "[発行された apiKey]", [表示するアイコンの個数]);  
iconLayout.setTitleVisible(false);
```

XML レイアウトファイルで設定

```
<net.nend.android.NendAdIconLayout  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    NendIconCount="[表示するアイコンの個数]"  
    NendApiKey="[発行された apiKey]"  
    NendSpotId="[発行された spotID]"  
    NendTitleVisible="false" />
```

### ・文字色設定

デフォルトの文字色は黒になります。

Java プログラムで設定

```
NendAdIconLayout iconLayout = new NendAdIconLayout(getApplicationContext(),  
    [発行された spotID], "[発行された apiKey]", [表示するアイコンの個数]);  
iconView.setTitleColor(Color.WHITE);  
// iconView.setTitleColor(Color.parseColor("#FFFFFF"));
```

XML レイアウトファイルで設定

```
<net.nend.android.NendAdIconLayout  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    NendIconCount="[表示するアイコンの個数]"  
    NendApiKey="[発行された apiKey]"  
    NendSpotId="[発行された spotID]"  
    NendTitleColor="#FFFFFF" /> <!-- #rrggbb 形式で指定 -->
```

## 6) 余白の設定

アイコンの余白に対して、次のように有効/無効の設定を行うことができます。

余白を有効にする場合は true（デフォルト）、無効にする場合は false を設定します。

※タイトル文字列を表示する場合は、横の余白のみ上記の設定が適用されます。

Java プログラムで設定

```
NendAdIconLayout iconLayout = new NendAdIconLayout(getApplicationContext(),  
    [発行された spotID], "[発行された apiKey]", [表示するアイコンの個数]);  
iconLayout.setIconSpaceEnabled(false);
```

XML レイアウトファイルで設定

```
<net.nend.android.NendAdIconLayout  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    NendIconCount="[表示するアイコンの個数]"  
    NendApiKey="[発行された apiKey]"  
    NendSpotId="[発行された spotID]"  
    NendIconSpaceEnabled="false" />
```

### 4-2-3. アイコンを1個ずつ個別に扱う場合

#### 1) XML レイアウトファイルで配置する場合

ここではレイアウトファイルを用いて、アイコン広告の表示を行う場合の方法を説明します。  
付属の NendSample プロジェクト内の『IconActivity.java』と同様です。

以下のような手順で実装を行います。

1. 表示する個数分 NendAdIconView を生成し、任意の位置に配置する。
2. NendAdIconLoader を生成し、1 で作成した NendAdIconView を登録する。
3. NendAdIconLoader の loadAd メソッドを実行し、広告取得を開始する。

#### NendAdIconView の配置

該当する画面の XML レイアウトファイル内に必要な個数分の  
『net.nend.android.NendAdIconView』のタグを追加します。  
以下の例では、画面の4隅にアイコンを配置しています。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <net.nend.android.NendAdIconView
        android:id="@+id/icon1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <net.nend.android.NendAdIconView
        android:id="@+id/icon2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true" />
    <net.nend.android.NendAdIconView
        android:id="@+id/icon3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true" />
    <net.nend.android.NendAdIconView
        android:id="@+id/icon4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true" />
</RelativeLayout>
```

サイズは「wrap\_content」を指定した場合は、デフォルトサイズ（75dp）に設定されます。  
（サイズには余白とタイトル文字列の描画領域が含まれます。）

任意の大きさを指定することが可能ですが、極端に大きいサイズや認識出来ないほど小さい  
サイズで表示している場合、正常な広告表示としてカウントされない場合があります。

## NendAdIconLoader の生成と NendAdIconView の登録

広告を表示するだけの基本的な実装例になります。

- ① XML レイアウトファイルで配置した NendAdIconView を取得
- ② spotID と apiKey を指定して、NendAdIconLoader を生成
- ③ NendAdIconLoader に NendAdIconView を全て登録
- ④ loadAd メソッドを実行し、広告取得を開始する。

```
public class IconActivity extends Activity {

    private NendAdIconLoader mIconLoader;
    private NendAdIconView mIconView1;
    private NendAdIconView mIconView2;
    private NendAdIconView mIconView3;
    private NendAdIconView mIconView4;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.icon);

        // ①XML レイアウトファイルで配置した NendAdIconView を取得
        mIconView1 = (NendAdIconView) findViewById(R.id.icon1);
        mIconView2 = (NendAdIconView) findViewById(R.id.icon2);
        mIconView3 = (NendAdIconView) findViewById(R.id.icon3);
        mIconView4 = (NendAdIconView) findViewById(R.id.icon4);

        // ②spotID と apiKey を指定して、NendAdIconLoader を生成
        mIconLoader = new NendAdIconLoader(getApplicationContext(), [発行された spotID], "[発行された apiKey]");
        // ③NendAdIconLoader に NendAdIconView を全て登録
        mIconLoader.addIconView(mIconView1);
        mIconLoader.addIconView(mIconView2);
        mIconLoader.addIconView(mIconView3);
        mIconLoader.addIconView(mIconView4);

        // ④ 広告取得を開始
        mIconLoader.loadAd();
    }

    @Override
    protected void onResume() {
        super.onResume();
        mIconLoader.resume();
    }

    @Override
    protected void onPause() {
        super.onPause();
        mIconLoader.pause();
    }
}
```

## 2) Java プログラムから動的に呼び出す場合

ここでは Java プログラムのみでアイコン広告の表示を行う場合の方法を説明します。  
付属の NendSample プロジェクト内の『IconJavaCallActivity.java』と同様です。

XML レイアウトファイルで配置していた NendAdIconView を、Java プログラムで  
インスタンス化するように変更するだけです。

```
// Java プログラムで NendAdIconView を生成
mIconView1 = new NendAdIconView(getApplicationContext());
mIconView2 = new NendAdIconView(getApplicationContext());
mIconView3 = new NendAdIconView(getApplicationContext());
mIconView4 = new NendAdIconView(getApplicationContext());
```

## 3) アイコンの個数について

1 つの NendAdIconLoader に登録出来るアイコンの数は 6 個までになります。  
7 個目以上は登録を行っても無視されます。

```
mIconLoader = new NendAdIconLoader(getApplicationContext(), [発行された spotID], "[発行された apiKey]");
mIconLoader.addIconView(mIconView1);
mIconLoader.addIconView(mIconView2);
mIconLoader.addIconView(mIconView3);
mIconLoader.addIconView(mIconView4);
mIconLoader.addIconView(mIconView5);
mIconLoader.addIconView(mIconView6);
mIconLoader.addIconView(mIconView7); // 登録自体は可能ですが、広告表示は行われません
```

#### 4) イベントリスナ

下記のように、必要に応じて NendAdLoader にリスナを登録することで、通知を受けることができます。

どのアイコンからの通知であるかは、NendAdIconView の ID で判別が可能です。

```
mIconLoader = new NendAdIconLoader(getApplicationContext(), [発行された spotID], "[発行された apiKey]");
mIconLoader.addIconView(mIconView1);
mIconLoader.addIconView(mIconView2);
mIconLoader.addIconView(mIconView3);
mIconLoader.addIconView(mIconView4);

// 受信成功通知
mIconLoader.setOnReceiveLisner(new OnReceiveListner() {
    @Override
    public void onReceiveAd(NendAdIconView iconView) {
        // 通知元のアイコンによって処理を変える場合
        switch(iconView.getId()){
            case R.id.icon1:
                // TODO
                break;
            case R.id.icon2:
                // TODO
                break;
            case R.id.icon3:
                // TODO
                break;
            case R.id.icon4:
                // TODO
                break;
        }
        Toast.makeText(getApplicationContext(), "Recieved", Toast.LENGTH_SHORT).show();
    }
});
// クリック通知
mIconLoader.setOnClickListner(new OnClickListner() {
    @Override
    public void onClick(NendAdIconView iconView) {
        Toast.makeText(getApplicationContext(), "Clicked", Toast.LENGTH_SHORT).show();
    }
});
// エラー通知
mIconLoader.setOnFailedListner(new OnFailedListner() {
    @Override
    public void onFailedToReceiveAd(NendIconError iconError) {
        Toast.makeText(getApplicationContext(), "Failed", Toast.LENGTH_SHORT).show();
    }
});
```

## 5) タイトル文字列の設定

アイコン下部に表示されるタイトル文字列に対して、次のように設定を行うことができます。

### ・表示/非表示設定

表示する場合は true（デフォルト）、非表示にする場合は false を設定します。

Java プログラムで設定

```
NendAdIconView iconView = new NendAdIconView(getApplicationContext());  
iconView.setTitleVisible(false);
```

XML レイアウトファイルで設定

```
<net.nend.android.NendAdIconView  
    android:id="@+id/icon1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    NendTitleVisible="false" />
```

### ・文字色設定

デフォルトの文字色は黒になります。

Java プログラムで設定

```
NendAdIconView iconView = new NendAdIconView(getApplicationContext());  
iconView.setTitleColor(Color.WHITE);  
// iconView.setTitleColor(Color.parseColor("#FFFFFF"));
```

XML レイアウトファイルで設定

```
<net.nend.android.NendAdIconView  
    android:id="@+id/icon1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    NendTitleColor="#FFFFFF" /> <!-- #rrggbb 形式で指定 -->
```



## 6) 余白の設定

アイコンの余白に対して、次のように有効/無効の設定を行うことができます。

余白を有効にする場合は true（デフォルト）、無効にする場合は false を設定します。

※タイトル文字列を表示する場合は、横の余白のみ上記の設定が適用されます。

Java プログラムで設定

```
NendAdIconView iconView = new NendAdIconView(getApplicationContext());  
iconView.setIconSpaceEnabled(false);
```

XML レイアウトファイルで設定

```
<net.nend.android.NendAdIconView  
    android:id="@+id/icon1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    NendIconSpaceEnabled="false" />
```

#### 4-2-4. 受信エラー通知について

受信エラー通知でエラー内容を取得出来ます。

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

NendIconError から getNendError メソッドで NendError オブジェクトを取得できます。

NendError からはエラーコードとエラーメッセージを取得出来ます。

```
@Override
public void onFailedToReceiveAd(NendIconError iconError) {

    NendError nendError = iconError.getNendError();
    switch (nendError) {
        case INVALID_RESPONSE_TYPE:
            // 不明な広告ビュータイプ
            break;
        case FAILED_AD_DOWNLOAD:
            // 広告画像の取得失敗
            break;
        case FAILED_AD_REQUEST:
            // 広告取得失敗
            break;
        case 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 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;
    }
    // エラーメッセージをログに出力
    Log.e(TAG, nendError.getMessage());
}
```

#### 4-2-5. ログ出力について

AndroidManifest.xml の<application>セクション内に、以下のメタタグを追加することにより、LogCat にログ出力を行うことが出来ます。

value="false"もしくは設定自体を削除した場合はエラーログを出力しません。

```
<meta-data
    android:name="NendDebuggable"
    android:value="true" />
```

#### 4-2-6. 定期ロードについて

アイコン広告がバックグラウンドに回るなど、広告の表示が必要ないと判断した場合には、広告受信ローテーションを自動的に停止させます。

それ以外で任意にローテーションを中断させたい場合には、以下のメソッドを利用してください。

NendAdIconLoader 使用時

```
// ローテーション再開
mIconLoader.resume();

// ローテーション停止
mIconLoader.pause();
```

NendAdIconLayout 使用時

```
// ローテーション再開
mIconLayout.resume();

// ローテーション停止
mIconLayout.pause();
```

**注意：**手動での中断が優先になります。手動で中断させている間は、広告が乗った画面が最前面に表示されても、ローテーションは再開されません。手動でローテーションを中断した場合は、必ず手動でローテーションを再開するようにしてください。

## 4-3. インターstitial広告

### 4-3-1. インターstitial広告

[※GooglePlayService、AndroidManifest.xml の設定を行ってください。](#)

#### 1) 広告のロード

apiKey、spotID を指定して NendAdInterstitial の loadAd メソッドを実行します。

広告のロードは、Activity の onCreate 直後に実行してください。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    NendAdInterstitial.loadAd(getApplicationContext(), "[発行された apiKey]", [発行された spotID]);
}
```

#### 2) 広告の表示

広告を表示したい場面で NendAdInterstitial の showAd メソッドを実行します。

なお、広告の読み込みが完了していない場合、インターstitial広告は表示されません。

```
NendAdInterstitial.showAd(this);
```

右上×ボタンまたはインターstitial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、GooglePlayへ遷移します。



NendAdInterstitial の dismissAd メソッドを実行することで、広告を閉じることが出来ます。

```
NendAdInterstitial.dismissAd();
```

## 4-3-2. 終了時広告

[※GooglePlayService、AndroidManifest.xml の設定を行ってください。](#)

### 1) 広告のロード

apiKey、spotID を指定して NendAdInterstitial の loadAd メソッドを実行します。

広告のロードは、Activity の onCreate 直後に実行してください。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    NendAdInterstitial.loadAd(getApplicationContext(), "[発行された apiKey]", [発行された spotID]);
}
```

### 2) 広告の表示

アプリが終了されるタイミングで NendAdInterstitial の showFinishAd メソッドを実行します。

なお、広告の読み込みが完了していない場合、インタースティシャル広告はされず、

アプリケーション終了を確認するダイアログが表示されます。

```
@Override
public void onBackPressed() {
    // super.onBackPressed();
    NendAdInterstitial.showFinishAd(this);
}
```

右上×ボタンをタップすると対象 Activity を終了します。

インタ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、GooglePlay へ遷移します。



NendAdInterstitial の dismissAd メソッドを実行することで、広告を閉じることが出来ます。

```
NendAdInterstitial.dismissAd ();
```

### 4-3-3. 画面回転について

画面の向きに応じて自動的にレイアウトを変更します。

ただし、インタースティシャル広告を表示した状態で画面回転が行われる可能性がある場合、

必ず AndroidManifest.xml で Activity に以下の設定を行ってください。

```
android:configChanges="orientation|screenSize"
```

### 4-3-4. イベントリスナ

イベントリスナはインタースティシャル広告、終了時広告で共通です。

#### 1) ロードイベントリスナ

下記のように、必要に応じて NendAdInterstitial にリスナを登録することで、  
広告のロードに関する通知を受けることが出来ます。

```
NendAdInterstitial.addListener(new OnCompletionListener(){
    @Override
    public void onCompletion(NendAdInterstitialStatusCode status){
        switch(status){
            case SUCCESS:
                // 成功
                break;
            case INVALID_RESPONSE_TYPE:
                // 不明な広告タイプ
                break;
            case FAILED_AD_REQUEST:
                // 広告取得失敗
                break;
            case FAILED_AD_INCOMPLETE:
                // 広告取得未完了
                break;
            case FAILED_AD_DOWNLOAD:
                // 広告画像取得失敗
                break;
            default:
                break;
        }
    }
});
```

## 2) クリックイベントリスナ

下記のように、必要に応じて showAd、showFinishAd にリスナを登録することで、広告のクリック通知を受けることができます。

```
NendAdInterstitial.showAd(this, new NendAdInterstitial.OnClickListener(){
    @Override
    public void onClick(NendAdInterstitialClickType clickType){
        switch (clickType) {
            case CLOSE:
                // ダウンロードボタン
                break;
            case DOWNLOAD:
                // ×ボタンまたは範囲外タップ
                break;
            default:
                break;
        }
    }
});

NendAdInterstitial.showFinishAd(this, new NendAdInterstitial.OnClickListener(){
    @Override
    public void onClick(NendAdInterstitialClickType clickType) {
        switch (clickType) {
            case CLOSE:
                // 範囲外タップ
                break;
            case EXIT:
                // ×ボタン
                break;
            case DOWNLOAD:
                // ダウンロードボタン
                break;
            default:
                break;
        }
    }
});
```

#### 4-3-5. 表示結果取得

下記のように、showAd および showFinishAd メソッドの戻り値を取得することで、  
必要に応じて表示結果を判定することが出来ます。

```
NendAdInterstitialShowResult result = NendAdInterstitial.showAd(this);
switch (result) {
case AD_SHOW_SUCCESS:
    // 表示成功
    break;
case AD_LOAD_INCOMPLETE:
    // ロードが完了していない
    break;
case AD_REQUEST_INCOMPLETE:
    // 抽選が正常完了していない
    break;
case AD_DOWNLOAD_INCOMPLETE:
    // WebViewのロードが完了していない
    break;
case AD_FREQUENCY_NOT_RECHABLE:
    // フリークエンシーカウントに到達していない
    break;
case AD_SHOW_ALREADY:
    // 既に表示されている
    break;
}
```



## ◆検証

Android 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。

### Androidアプリ向け表示テスト用ID

サイズ	apiKey	spotID
320 × 50	c5cb8bc474345961c6e7a9778c947957ed8e1e4f	3174
320 × 100	8932b68d22d1d32f5d7251f9897a6aa64117995e	71000
300 × 100	1e36d1183d1ab66539998df4170a591c13028416	71001
300 × 250	499f011dbec5d37cfa388b749aed2bfff440a794	70357
728 × 90	02e6e186bf0183105fba7ce310dafe68ac83fb1c	71002
アイコン	0c734134519f25412ae9a9bff94783b81048ffbfe	101282
インター ショナル	8c278673ac6f676dae60a1f56d16dad122e23516	213206

※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。

## ◆サンプルについて

サンプルソースは以下のようになっています。

実装例になりますので、アプリケーションの実装方法やビルドターゲット等に合わせて適宜変更をしてください。

Android アプリケーションの実装、レイアウトに関しては、Android 公式ドキュメントを参照してください。

xml layout	xml レイアウトファイルでの表示
Java call/	Java からの動的表示サンプル
Java call LinearLayout	Java から LinearLayout に表示
Java call RelativeLayout	Java から RelativeLayout に表示
Java call with listner	Java から表示し、イベントリスナを使用
layout sample/	レイアウト
top	上部中央
bottom	下部中央
attach and dettach	広告ビューの削除と再利用
ListView	ListView での表示
ViewPager	ViewPager での表示
ViewPager and ListView	ViewPager 内の ListView での表示
Fragment replace	Fragment での表示
dialog sample/	ダイアログ上での表示サンプル
dialog	任意時表示
exit dialog	終了時表示
size sample/	サイズ表示サンプル
320 × 50	320×50 サイズ表示
320 × 100	320×100 サイズ表示
300 × 100	300×100 サイズ表示
300 × 250	300×250 サイズ表示
728 × 90	728×90 サイズ表示
icon sample/	アイコン型広告表示サンプル
xml layout	xml レイアウトファイルでの表示
xml layout separate	アイコンを個別に扱うサンプル
Java call	Java からの動的表示サンプル
dialog sample	ダイアログ上での表示サンプル
dialog	任意時表示
exit dialog	終了時表示
attach and dettach	アイコンの追加と削除
space example	アイコンの余白なしサンプル
interstitial sample/	インタースティシャル広告表示サンプル
interstitial	インタースティシャル
finish ad	終了時広告

## ◆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 と spotID、SDK バージョン番号を情報として頂ければ幸いです。