POCKET MEDIA
MOBILE ADVERTISING

Version 1 - 19 Jun 2015

Native ads with Objective-C

Introduction

Native ads match both the style and function of the user experience in which they're placed. They match the visual design of the application they live within, and look and behave like natural content on the publisher property in which they're displayed. These ads increase the user experience by providing value through relevant content delivered in your apps.

Our native ads feature allows publishers to show ads that are seamless with the content. Ads are no longer confined to a box, like a traditional banner ad.

Getting started

Before adding native ads in your project, you should first check the Native Ad Guidelines. This guide helps to integrate the ads with your data, protecting the integrity of the app.

Integrating Pocket Media Native Ads in your app is very easy and customizable: you need to retrieve the ads from a JSON with some parameters (described below). These parameters are necessary for improving the results depending on the user.

You are free to use the libraries you want for getting the JSON, parsing it, etc. If you aren't sure where to start you may wish to check the example project here.

These are the main steps you need to follow:

1. Create a unique token for every user of the app when this is opened for the first time and save it for later use. This token has to be a string with a maximum of 64 characters.

2. Perform a request to

```
http://offerwall.12trackway.com/ow.php?
output=json&os=ios&limit=1&version=7.1&model=iphone&token=TOKEN&affiliate_id=XX
```

with the following parameters:

Name	Description	Type	Required	Possible values
os	The OS	String	Yes	ios or android
limit	Number of ads	Unsigned integer	No (default: 1)	Between 1 and X
version	The iOS version	String	Yes	6.1, 7.1, etc
model	The type of device	String	Yes	iPad, iPhone or iPod
token	Unique identifier	String	Yes	Maximum 64 characteres
affiliate_id	Your identifier	String	Yes	Request this string to your account manager

Name	Description	Type	Required	Possible values
size	Size of the creative	String	No (default: 300x300)	Check the section “Retrieving different size of images"

3. Parse the JSON results and mix it with your data. The results contain an array of ads with the following fields:

- `campaign_name`: the ad headline.
- `campaign_description`: the summary of the advertisement.
- `campaign_image`: a URL to the square image with max size 300x300.
- `click_url`: the url necessary for opening the app.

4. Finally, open the browser with `click_url` when the user touch on the native ad.

Retrieving ads with Objective-C

You can download the Objective-C project example here, where we propose a solution to parse native ads from a JSON and place them in a random position in a `UITableView`.

For opening this project, it is required to have Xcode 6 installed.

Creating and saving a token

For saving and loading the token you can use `NSUserDefaults` and methods like `objectForKey` and `setObject`.

```
-(NSString *)loadToken {
  NSString * tokenAdKey = @"nativeAdToken";
  long timestamp = [[NSDate date] timeIntervalSince1970]*10000;
  NSUserDefaults *preferences = [NSUserDefaults standardUserDefaults];
  NSString * token = [preferences objectForKey:tokenAdKey];
  if (!token){
    token = [self generateRandomString];
    [preferences setObject:token forKey:tokenAdKey];
    [preferences synchronize];
  }
  return token;
}
```

Retrieving the JSON

First, we need to create the url with the parameters specified earlier.

```
  NSString *baseURL = @"http://offerwall.12trackway.com/ow.php?
output=json&os=ios";
  NSString *currentVersion = [[UIDevice currentDevice] systemVersion];
  NSString *model = [[[UIDevice currentDevice] model]
componentsSeparatedByString:@" "][0];
  NSString *affiliateId = @"XXXXXXX";
  return [NSString stringWithFormat:@"%@&limit=%i&version=%@&model=%@&token=
%@&affiliate_id=%@", baseURL, limit, currentVersion, model,[self loadToken],
affiliateId];
```

Afterwards, we perform a request to the previous URL with the library AFNetworking.

```
NSString *url = [self getNativeAdsURL:limit];
NSLog(@"URL: %@",url);
AFHTTPRequestOperationManager *manager = [AFHTTPRequestOperationManager manager];
[manager POST:url parameters:nil success:^(AFHTTPRequestOperation *operation, id
responseObject) {
  if ([responseObject isKindOfClass:[NSArray class]]) {
    // Parse the results
  }
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
  [self.delegate didReceiveError:error];
}];
```

Parsing the results

After getting the results we need to parse the JSON into the NativeAd object. In this case we are using JSONModel for transforming the JSON in a `NSDictionary` or `NSArray`. For a better understanding of this process, check the JSONModel webpage.

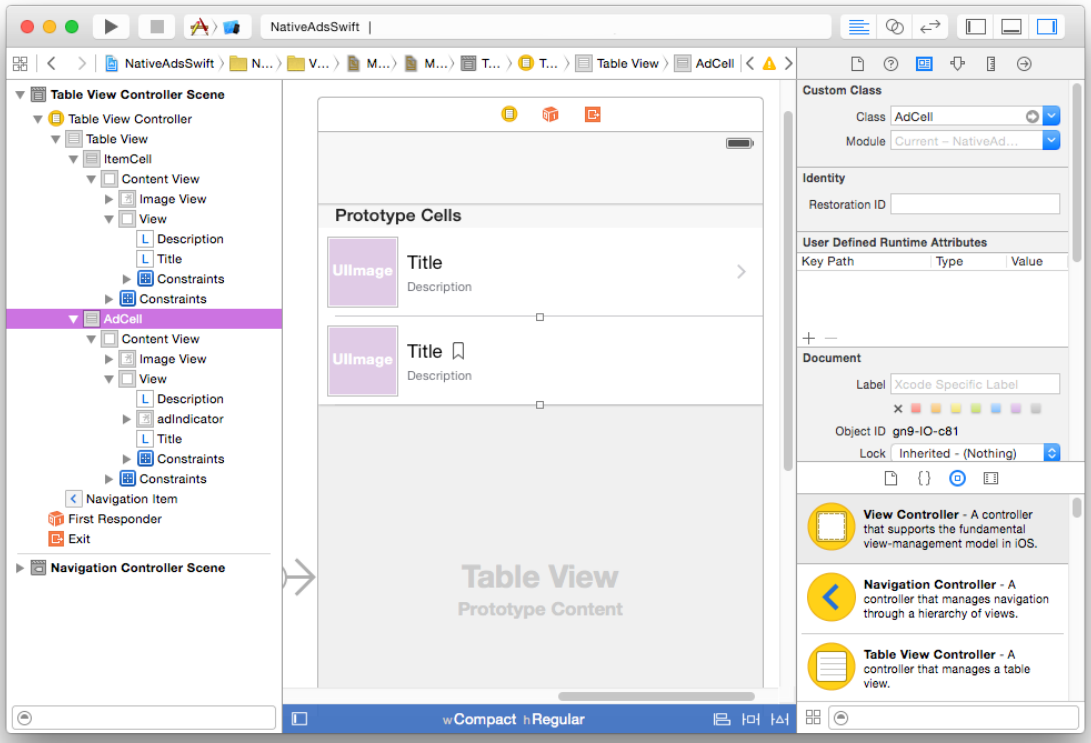
```
NSArray *responseArray = responseObject;
for(NSDictionary *nativeAdsDic in responseArray) {
  NSError *e = nil;
  NativeAd * nativeAd = [[NativeAd alloc] initWithDictionary:nativeAdsDic
error:nil];
  if (nativeAd) {
    [ads addObject:nativeAd];
  } else {
    NSLog(@"Error parsing JSON: %@", e);
  }
}

@interface NativeAd : JSONModel
 @property (strong, nonatomic) NSString* campaignName;
 @property (strong, nonatomic) NSString* campaignDescription;
 @property (strong, nonatomic) NSString* clickUrl;
 @property (strong, nonatomic) NSString* categoryName;
 @property (strong, nonatomic) NSString* campaignImage;
@end

@implementation NativeAd
 +(JSONKeyMapper*)keyMapper
 {
   return [JSONKeyMapper mapperFromUnderscoreCaseToCamelCase];
 }
@end
```

Showing the ads in a UITableView

Once you have an array of native ads you can show it as you wish. In this case we are taking into consideration a table with two types of cells, one for the ads and the other for the other items.



The class `AdCell` is as follows:

```
@interface AdCell : UITableViewCell
 @property (weak, nonatomic) IBOutlet UIImageView *campaignImageView;
 @property (weak, nonatomic) IBOutlet UILabel *campaignNameLabel;
 @property (weak, nonatomic) IBOutlet UILabel *campaignDescriptionLabel;
@end
```

In the `UITableViewController` we'll have an array of `AnyObject` with `NativeAds` and the other type mixed in the order we want. In this case the ad position is a random number between 1 and 5, this way we ensure that the ad is going to be more visible for the user in the first screen.

```
self.itemsTableView = [[NSMutableArray alloc] initWithCapacity:20];
[self loadLocalJSON];
NativeAdsRequest *adRequest = [[NativeAdsRequest alloc] init];

-(void)didReceiveResults: (NSArray *) nativeAds {
  if (self.itemsTableView.count > 4){
    for(NativeAd *nativeAd in nativeAds) {
      int randomPosition = arc4random_uniform(4)+1;
      [self.itemsTableView insertObject:nativeAd atIndex:randomPosition];
    }
    [self.tableView reloadData];
  }
}
```

Later, when the table is populated, we need to check the type of object in the array: if it is a native ad, we are going to reuse a cell of type `AdCell`.

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:
(NSIndexPath *)indexPath
{

  if ([self.itemsTableView[indexPath.row] isKindOfClass:[NativeAd class]]){
    static NSString *tableIdentifier = @"AdCell";
    AdCell *cell = [tableView dequeueReusableCellWithIdentifier:tableIdentifier];
    NativeAd *item = self.itemsTableView[indexPath.row];
    if (cell) {
      cell.campaignDescriptionLabel.text = item.campaignDescription;
      cell.campaignNameLabel.text = item.campaignName;
      [cell.campaignImageView setImageWithURL:[NSURL
URLWithString:item.campaignImage]];
    }
    return cell;
  } else {
   // Returning other type of cell
 }
}
```

Adding the click behaviour in the native ads

Finally, when the user interacts with the native ad, you must open the browser with the value `clickURL`.

```
- (void) tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath
*)indexPath {
  NativeAd *ad = self.itemsTableView[indexPath.row];
  if (ad) {
    NSURL *url = [NSURL URLWithString:ad.clickUrl];
    [[UIApplication sharedApplication] openURL:url];
  }
}
```

Retrieving different size of images

You can retrieve native ads with the following sizes:

43x43	80x80	200x200
72x72	100x100	300x300

For getting the images with one specific size you need to perform the request with the parameter `size=NumxNum` like the following example:

```
http://offerwall.12trackway.com/ow.php?
output=json&os=ios&limit=1&version=7.1&model=iphone&token=TOKEN&affiliate_id=AFFI
LIATE_ID&size=100x100
```