POCKET MEDIA
MOBILE ADVERTISING

Version 2 - 20 Oct 2015

Native ads with Swift

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 parameter 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
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
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 Swift 2

You can download the Swift project example here. It is required to have Xcode 7 installed.

Creating and saving a token

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

```
var token = NSUserDefaults.standardUserDefaults().objectForKey(
   Constants.NativeAds.tokenAdKey) as? String
if token == nil {
  token = self.generatingUniqueToken()
   NSUserDefaults.standardUserDefaults().setObject(token, forKey:
   Constants.NativeAds.tokenAdKey)
   NSUserDefaults.standardUserDefaults().synchronize()
}
```

Retrieving the JSON

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

```
struct Device {
 static let iosVersion = NSString(string:
       UIDevice.currentDevice().systemVersion).doubleValue
 static let model = UIDevice.currentDevice().model.characters.split{$0 == "
"}.map{ String($0) }[0]
}
let baseURL = “http://offerwall.12trackway.com/ow.php?output=json"
let affiliateId = “XXXXXXXXXXXXXXXX"
let url = baseURL + "&os=ios&limit=\(limit)&version=\
(Device.iosVersion)&model=\(Device.model)&token=\(token)&affiliate_id=\
(affiliateId)"
```

You can use a library for performing the request like Alamofire. Otherwise you can use `NSURLConnection.sendAsynchronousRequest(request, queue: NSOperationQueue.mainQueue())`.

```
if let url = NSURL(string: url) {
  let request = NSURLRequest(URL: url)
  NSURLConnection.sendAsynchronousRequest(request, queue:
   NSOperationQueue.mainQueue()) {(response, data, error) in
    if error != nil {
      //Error
    } else {
      if let json: NSArray = NSJSONSerialization.JSONObjectWithData(data,
       options: NSJSONReadingOptions.MutableContainers, error: nil) as? NSArray {
        //Parse the result
      }
    }
}
```

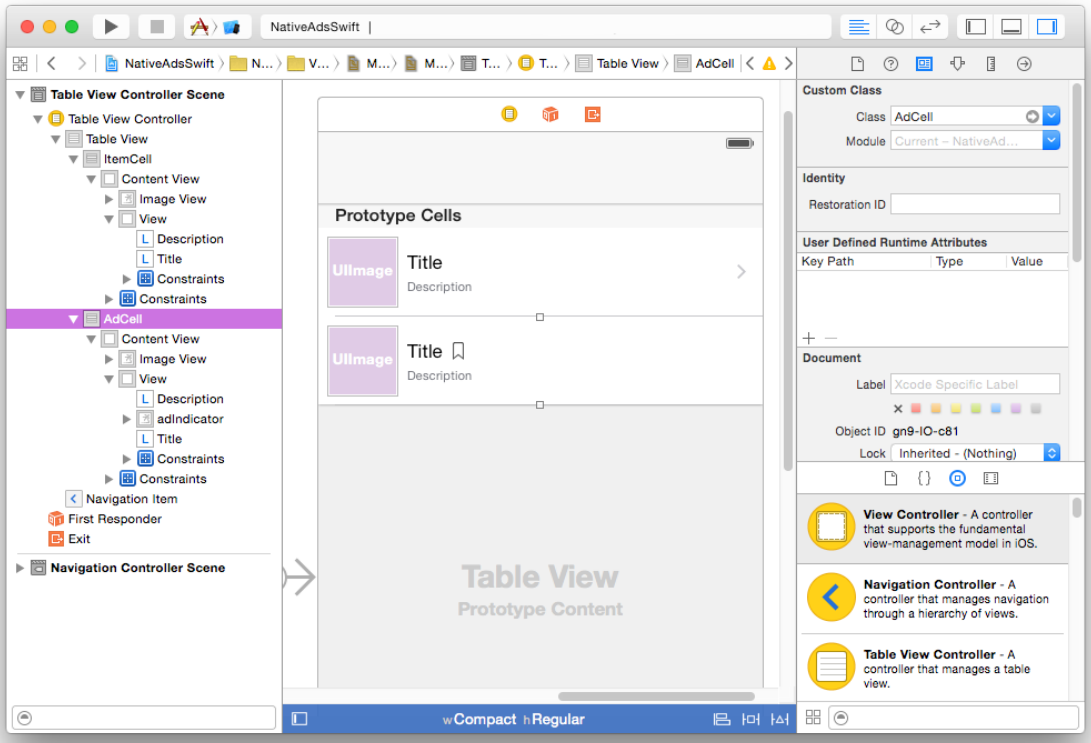
Parsing the results

You can use a library for parsing the results like SwiftyJSON or Argo. On the other hand, if you don't want to use any of this libraries, you can use `NSJSONSerialization` for transforming the JSON in a `NSDictionary` or `NSArray`.

```
var nativeAds: [NativeAd] = []
if let json: NSArray = NSJSONSerialization.JSONObjectWithData(data, options:
NSJSONReadingOptions.MutableContainers, error: nil) as? NSArray {
  for itemJson in json {
    if let itemAdDictionary = itemJson as? NSDictionary,
        ad = NativeAd(adDictionary: itemAdDictionary) {
      self.nativeAds.append(ad)
    }
}

struct NativeAd {

  var campaignName        : String
  var campaignDescription : String
  var clickURL            : NSURL
  var campaignImage       : NSURL

  init?(adDictionary: NSDictionary){

    guard let name = adDictionary["campaign_name"] as? String,
          urlClickString = adDictionary["click_url"] as? String,
          urlClick = NSURL(string: urlClickString),
          urlImageString = adDictionary["campaign_image"] as? String,
          urlImage = NSURL(string: urlImageString),
          description = adDictionary["campaign_description"] as? String else {
      return nil
    }

    self.clickURL = urlClick
    self.campaignName = name
    self.campaignImage = urlImage
    self.campaignDescription = description
  }
}
```

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:

```
class AdCell : UITableViewCell {
  @IBOutlet weak var campaignNameLabel: UILabel!
  @IBOutlet weak var campaignImageView: UIImageView!
  @IBOutlet var campaignDescriptionLabel: UILabel!
}
```

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 4, this way we ensure that the ad is going to be more visible for the user in the first screen.

```
var itemsTable: [AnyObject] = []
loadOtherItems() //Loading and appending data to itemsTable

func addNativeAdsInTableView(){
  for ad in nativeAds {
    itemsTable.insert(ad, atIndex: random(1...4))
    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`.

```
override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath:
NSIndexPath) -> UITableViewCell {
  switch itemsTable[indexPath.row] {
  case let item as ItemTableView :
    // returning other type of cell
  case let ad as NativeAd :
    let cell = tableView.dequeueReusableCellWithIdentifier("AdCell",
       forIndexPath:indexPath) as! AdCell
    cell.campaignNameLabel.text = ad.campaignName
    cell.campaignDescriptionLabel.text = ad.campaignDescription
    cell.campaignImageView.af_setImageWithURL(ad.campaignImage)
    return 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`.

```
override func tableView(tableView: UITableView, didSelectRowAtIndexPath
indexPath: NSIndexPath) {
  if let ad = itemsTable[indexPath.row] as? NativeAd {
    UIApplication.sharedApplication().openURL(ad.clickURL)
    tableView.deselectRowAtIndexPath(indexPath, animated: true)
  }
}
```

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
```