Innity Mobile Ads

iPhone SDK v1.10 Implementation Guide

Getting Started

Thank you for including Innity Mobile Ads as a part of your mobile app monetisation strategy.

This document serves as a guide for app developers to implement Innity Mobile ads in iPhone apps. You'll see in this document that implementing the SDK is fairly easy, showing ads requires nothing more but a few lines of code. Happy coding!

Requirements

This Innity Mobile Ads iOS SDK requires:

- iOS 10 or newer

Ad Format Overview

This SDK comes with support for several ad formats:

Engage Ads

- **Spin Lite**
  This banner is presented as a circular disc overlay at the bottom of the screen taking up a full width 9:4 space (e.g. 414 x 184) The disc can be rotated by the user.

- **Spin**
  This banner is presented as a circular disc overlay at the bottom of the screen taking up a full width 9:4 space (e.g. 414 x 184) The disc can be rotated and when tapped, a series of swipeable cards are presented.

- **Spin+**
  This banner is presented as a circular disc overlay at the bottom of the screen taking up a full width 9:4 space (e.g. 414 x 184) The disc can be rotated to reveal a full screen ad behind the disc.

- **Revolver**
  This banner is presented as a series of cards arranged in a circle in 3D space. The cards can be rotated by swiping. The cards would expand to a series of larger cards, upon the second swipe by the user, also arranged in a circle in 3D space covering the full screen.

- **Pull**
  This banner is presented as a 320x50 unit that needs to be anchored to the bottom of the screen. The 320x50 unit can be "pulled" or swiped up to reveal a fullscreen banner. The fullscreen banner includes a close button. (Only use ONE pull banner per screen)

- **Interstitial**
  An interstitial banner is presented as a fullscreen unit with a close button.

- **Mobile Leaderboard (320x50)**
  This banner is a standard 320x50 unit without expansion.

- **Medium Rectangle (300x250)**
  This banner is a standard 320x50 unit without expansion.

- **Medium Rectangle Cube (300x250)**
  This banner is presented as a medium rectangle size banner that is periodically rotated like a cube.

- **Medium Rectangle Slider (300x250)**
  This banner is presented as a medium rectangle size slideshow that will periodically animated to the next slide. User can swipe the banner to navigate through the slideshow.

- **Mobile Cards**
  This banner is presented as a stack of cards. Only one card is visible to the user. The card at the front can be sent to the back by swiping the card to the left or right.

- **Splash Cards**
  This banner is presented with a brief splash screen. After the splash screen animation ended, the banner shrinks into a mobile cards banner.

- **Splash Spin Lite**
  This banner is presented with a brief splash screen. After the splash screen animation ended, the banner shrinks into a spin lite banner.

- **Reco**
  This banner is presented as a full-width carousel of cards and rounded thumbnails. The cards will automatically advance to the next card without intervention. User can swipe, or tap on the card of interest to expand the card.

IMPORTANT:

- All Engage ads served are for portrait orientation only.

- External links in banner ads will open in the embedded browser provided by the SDK for a less disruptive experience. The embedded browser includes the user to open the link in Safari.

Video Interstitial

The video interstitial unit is a fullscreen unit that presents an auto-played video typically lasting 15 to 30 seconds long. Once the video is played to completion, a button will appear allowing the user to learn more about the advertiser. Video units may be skippable or non-skippable or non-skippable only for a few seconds.

Native Ads

Unlike typical banner ad formats, native ads do not have have a fixed layout, components of the ad are returned for you to present the ad in a way that blends into the the look and feel of your app.

The components returned to you are the ad's title, description, advertiser's brand name, a thumbnail image and a larger portrait or landscape image.

Native Video Ads

Native Video ads are almost identical to native ads except you'll get a single video instead of images. As with native ads, you'll be able to present the ad with your app's look and feel as well.

If you need any assistance with choosing the right formats to implement in your app, feel free to contact your account manager at any time.

Adding the SDK to your App

1. Open your project in Xcode
2. Drag and drop the `IMAd.framework` into your project. Make sure the frameworks are located under your project group (not Unit Tests or Tests group).
3. Make sure "**Copy items if needed**" is checked when you paste the `IMAd.framework`
4. Select the project file from the project navigator on the left side of the project window.
5. Select the target for where you want to add frameworks in the project settings editor.
6. Select the "**General**" tab and click "**Embedded Binaries**" to see a list of all the frameworks in your application.
7. If you cannot locate the frameworks you added above, click the "**+**" below the list of frameworks. Then add `IMAd.framework`.
8. Check that there isn't any duplicate `IMAd.framework` files in "**Linked Frameworks and Libraries**" tab (just below "**Embedded Binaries**" tab)
9. Add the following permissions into your "**info.plist**"

   - ```
     <key>NSAppTransportSecurity</key>
     <dict>
           <key>NSAllowsArbitraryLoads</key>
     </dict>
     ```

   - ```
     <key>NSPhotoLibraryUsageDescription</key>
     <string>$(PRODUCT_NAME) requires access to the photo library</string>
     ```

     Some ads may include a call to action to save an image of a coupon.

   - ```
     <key>NSCalendarsUsageDescription</key>
     <string>$(PRODUCT_NAME) requires access to the calendar</string>
     ```

     Some ads may include a call to action to set a reminder for a new movie release date or concert date.

Implementing Engage Ads

Here are the steps to add a banner to your app:

1. Create an instance of the IMAd SDK Core.

2. Create an instance of the IMAd Engage ad format and set the **publisher ID** assigned to you by Innity as well as any targeting properties your app is able to provide, for example, the user's gender and age.

3. Load a banner from a designated zone (Zone IDs are issued by Innity, the IDs are created according to how you intent to target ads. You may want to have a zone for each category of content your app provides, for example, any content related to Fashion may be considered to be in the same zone and hence share the same zone ID.)

4. Once you receive the notification that an ad is available and has been loaded, add the banner ad's view to your layout.

   There are various events that fire such as when the banner expands or closes that you may want to handle. If your app is playing a video, you may want to pause the video when the ad is expanded to cover the full screen.

5. Finally, you'll want to clean up/destroy the ad when you're done using it.

IMPORTANT:

All Engage format ads served are for <u>portrait orientation</u> only. It is also your responsibility to ensure that the banner's view is the top-most view in your hierarchy.

Here is some sample code to do the above:

1. Create an IMAd SDK Core Object

```
import IMAd

imadSDKCore = IMAd().initIMAdSDK()
```

2. Create an IMAdEngage ad and set targeting

Once you've created an IMAdEngage object, we recommend that you set targeting properties if you know your user. There are a list of standard properties provided, please see the **Audience Targeting Properties** table below, however, if there are certain properties you can provide which are useful for targeting, you may also set your own custom key, value pairs.

```
let imadEngage = IMAdEngage().initIMAdEngage(imadSDK:AppDelegate.getImadSDKCore(),
publisherId: YOUR_PUBLISHER_ID)

imadEngage.targetingProperties.setTargetingProperties(
      key:IMAdTargetingProperties.IMATargetPropKey_Age, value:"28")

imadEngage.targetingProperties.setTargetingProperties(
      key:IMAdTargetingProperties.IMATargetPropKey_Gender, value:"F")

imadEngage.targetingProperties.setTargetingProperties(key:"customKey",
value:"customValue")
```

3. Load a banner

Every banner needs to be loaded from the server with a Zone ID. The Zone IDs are allocated for your app by Innity. There are 6 zone IDs provided for the purpose of testing, each zone is for one of the 6 Engage Ad formats. *See table above for Test Zone IDs.*

You'll also need to handle banner load events and other events related to the banner ad's creative. Specify the object to handle the load events via **adLoadCallback: self** and implement the mandatory **adReady** & **adFailed** methods.

Also specify the object to handle the banner ad's creative events via **adEventCallback: self** and implement the various methods as required. Take a look at the "**Handling Engage Ad Banner Events**" section below for an explanation of what these events are and what they do.

NOTE: The test zone IDs are **alphabetical** while the live zone IDs are **numerical**. You would still pass the zone ID **as a string**, e.g. zoneId: "8234"

```
imadEngage.load(
      zoneId: "interstitial", adLoadCallback: self, adEventCallback: self,
viewController: self)
```

Once a banner has successfully loaded, the **adReady** delegate method will be called. This delegate method will only be called if there is an ad available for display in the requested zone.

The **adFailed** delegate method will be called whenever there is no ad available or a timeout occurred during loading. The timeout is set to 30 seconds.

4. Handling Banner Load Events

There are various banner events that the ad unit may trigger, at minimum, you'll need to handle **adReady** - when the ad has loaded and is ready for you to ad to your view and **adFailed** when the ad fails to load.

```
func adReady(adUnit: AdUnit)
{
    self.adUnit = adUnit as! EngageAd
    self.view.addSubview(self.adUnit.getAdView()!)
}

func adFailed(zoneId: String, error: Error)
{
    //failed
}
```

5. Destroying the IMAdEngage and AdUnit

Remember to cleanup once you're done with IMAdEngage and AdUnit.

```
self.imadEngage.destroy()
self.imadEngage = nil

self.adUnit.destroy()
self.adUnit = nil
```

Zones

The following is a table of the test zone IDs. Remember that in a live environment, the zone IDs are numerical. Your Innity account manager will provide you with the live zone IDs.

Test Zone ID	Format
spin_lite	**Spin Lite** 9:4 (no expansion)
spin	**Spin** 9:4 expands to fullscreen
spin_plus	**Spin+** 9:4 expands to fullscreen
revolver	**Revolver** 9:4 expands to fullscreen
pull	**Pull** 320px x 50px expands to fullscreen
interstitial	**Interstitial** Fullscreen (no pre-expansion banner)
mobile_leaderboard	**Mobile Leaderboard** 320px x 50px (no expansion)
medium_rectangle	**Medium Rectangle** 300px x 250px (no expansion)
medium_rectangle_cube	**Medium Rectangle Cube** 300px x 250px (no expansion)
medium_rectangle_slider	**Medium Rectangle Slider** 300px x 250px (no expansion)
mobile_cards	**Mobile Cards** 9:4 expands to fullscreen
splash_cards	**Splash Cards** 9:4 expands to fullscreen
splash_spin_lite	**Splash Spin** 9:4 expands to fullscreen
reco	**Reco** Full-width x 405px (no expansion)

IMPORTANT:

Some ad formats such as the Spin family of formats may conflict with the gestures you wish to recognise in your app. The gestured required to spin the creative can conflict with the horizontal swipe gesture used to switch between tabs in a tab view.

We recommend you take some time to understand how the gestures required for each format before implementing them in your app.

Premium Ad Zones

Premium ads are reserved for ads that have a higher priority over standard ad zones. If you have been given premium ad zone IDs, please add the premium ad zone when requesting for ads. The SDK will prioritise premium ad zones when there are premium campaigns available.

If you have not received any premium ad zone IDs, please check with your account manager if you are eligible for premium ad zones.

The zone can be added with the following code:

```
let imadEngage = IMAdEngage().initIMAdEngage(imadSDK:AppDelegate.getImadSDKCore(),
publisherId: YOUR_PUBLISHER_ID)

imadEngage.setPremiumZone(premiumZoneId: YOUR_PREMIUM_ZONE_ID);

imadEngage.load(
        zoneId: YOUR_NORMAL_ZONE_ID, adLoadCallback: self, adEventCallback: self,
viewController: self);
```

Handling Engage Ad Banner Events

In addition to the mandatory `adDidLoad` and `adDidFailLoad`, optional delegates are available to better manage your app. They are:

- `func adDidBeginPreview(EngageAd)`

  When a **Pull** banner is tapped on, the ad is expanded upwards slightly to provide the user with a preview of the ad. The adDidBeginPreview delegate is called just before the banner expands slightly. If your app includes some animation or music, you may utilise this delegate to pause the animation or music when the preview begins. Also see `adDidEndPreview`.

- `func adDidEndPreview(EngageAd)`

  This delegate is called when the preview of a **Pull** banner ends, that is, when the banner returns to it's original position after expanding slightly. Also see `adDidBeginPreview`

- `func adDidExpand(EngageAd)`

  The delegate is called just after the the banner ad has expanded fully, for example when a **Pull, Spin** or **Revolver** banner is expanded. This delegate is also called when an **Interstitial** banner format appears.

- `func adDidCancelExpand(EngageAd)`

  The delegate is called after an expandable ad returns to its initial position but **did not** fully expand. This is only applicable to the **Pull** banner.

- `func adDidCloseExpanded(EngageAd)`

  The delegate is called after an expandable ad returns to its initial position from its **fully expanded** state. This delegate is also called when the user closes a **Pull**, **Interstitial, Spin** or **Revolver** banner.

- `func adDidUnload(EngageAd)`

  The delegate is called after **Spin Lite**, **Spin**, **Spin+** or **Revolver** is closed. The developer must remove the ad from the view hierarchy manually.

- `func adActionWillBegin(EngageAd)`

  The delegate is called just before a the calendar event UI is invoked (some ads may have a button for the user to add a reminder to an event such as a concert date and time) and just before an external link is opened with the SDK's embedded web browser.

- `func adActionDidEnd(EngageAd)`

  The delegate is called just after the calendar event UI has been dismissed.

- `func adActionWillLeaveApplication(EngageAd)`

  The delegate is called just before the "do you want make a call" confirmation dialog appears or before a messaging app is invoked to compose an SMS.

- `func adWasClicked(EngageAd)`

  The delegate is called when the user "clicks" on the ad and triggers a web browser to open the landing page.

Implementing Video Interstitial Ads

Here are the steps to add a banner to your app:

1. Create an instance of the IMAd SDK Core.

2. Create an instance of the IMAd Video Interstitial ad format and set any targeting properties your app is able to provide, for example, the user's gender and age.

3. Load a banner from a designated zone (Zone IDs are issued by Innity, the IDs are created according to how you intent to target ads. You may want to have a zone for each category of content your app provides, for example, any content related to Fashion may be considered to be in the same zone and hence share the same zone ID.)

4. Once you receive the notification that an ad is available and has been loaded, show the interstitial video when you're ready.

5. Finally, you'll want to clean up/destroy the ad when you're done using it.

IMPORTANT:

Please ensure that "Landscape Left" and "Landscape Right" are enabled in your project's **Device Orientation** settings.

1. Create an IMAd SDK Core Object

```
import IMAd
imadSDKCore = IMAd().initIMAdSDK()
```

2. Create an IMAdVideoInterstitial ad and set targeting

Once you've created an IMAdVideoInterstitial object, we recommend that you set targeting properties if you know your user. There are a list of standard properties provided, please see the **Audience Targeting Properties** table below, however, if there are certain properties you can provide which are useful for targeting, you may also set your own custom key, value pairs.

```
let videoInterstitial = IMAdVideoInterstitial().initIMAdVideoInterstitial(imadSDK:
AppDelegate.getImadSDKCore())

videoInterstitial.targetingProperties.setTargetingProperties(
     key:IMAdTargetingProperties.IMATargetPropKey_Age, value:"28"

videoInterstitial.targetingProperties.setTargetingProperties(
     key:IMAdTargetingProperties.IMATargetPropKey_Gender, value:"F")
```

```
videoInterstitial.targetingProperties.setTargetingProperties(key:"customKey",
value:"customValue")
```

3. Load a banner

Every banner needs to be loaded from the server with a Zone ID. The Zone IDs are allocated for your app by Innity.
NOTE: The test zone IDs are **alphabetical** while the live zone IDs are **numerical**. You would still pass the zone ID **as a string**, e.g. zoneId: "8234"

```
videoInterstitial.load(zoneId: "video_interstitial", adLoadCallback: self)
```

Once a banner has successfully loaded, the **adReady** delegate method will be called. This delegate method will only be called if there is an ad available for display in the requested zone.

The **adFailed** delegate method will be called whenever there is no ad available or a timeout occurred during loading. The timeout is set to 30 seconds.

4. Handling Banner Load Events

There are various banner events that the ad unit may trigger, at minimum, you'll need to handle **adReady** - when the ad has loaded and is ready for you to ad to your view and **adFailed** when the ad fails to load.

```
func adReady(adUnit: AdUnit)
{
      DispatchQueue.main.async
      {
            self.adUnit = adUnit as! VideoInterstitialAd
            // adUnit.show() has to be called from the main thread
            self.adUnit.show(viewController: self)
      }
}

func adFailed(zoneId: String, error: Error)
{
      //failed
}
```

5. Destroying the IMAdVideoInterstitial and AdUnit

Remember to cleanup once you're done with IMAdVideoInterstitial and AdUnit.

```
self.videoInterstitial = nil

self.adUnit.destroy()
self.adUnit = nil
```

Implementing Native Ads

Native Ads allow you to present ads according to the look and feel of your app. The SDK will return the following data to you for you to present in your app as you wish:

1. Title
2. Description
3. Advertiser's Brand Name
4. A Square Thumbnail Image
5. A Portrait Image
6. A Landscape Image

Once you present the ad, simply call the **displayed** method when the ad becomes visible the user and call the **clicked** method when the ad is tapped on.

The process is still very similar to Engage ads and Video Interstitial:

1. Create an instance of the IMAd SDK Core.

2. Create an instance of the IMAd Native Ad format and set any targeting properties your app is able to provide, for example, the user's gender and age.

3. Load an ad from a designated zone (Zone IDs are issued by Innity, the IDs are created according to how you intent to target ads. You may want to have a zone for each category of content your app provides, for example, any content related to Fashion may be considered to be in the same zone and hence share the same zone ID.)

4. Once you receive the notification that an ad is available and has been loaded, present the ad in your own style when you're ready and call the **displayed** method when the ad becomes visible to the user.

5. Respond to a click/tap and call the **clicked** method.

6. Finally, you'll want to clean up/destroy the ad when you're done using it.

1. Create an IMAd SDK Core Object

```
import IMAd

imadSDKCore = IMAd().initIMAdSDK()
```

2. Create an IMAdNative ad and set targeting

Once you've created an IMAdNative object, we recommend that you set targeting properties if you know your user. There are a list of standard properties provided, please see the **Audience Targeting Properties** table below, however, if there are certain properties you can provide which are useful for targeting, you may also set your own custom key, value pairs.

```
let nativeAd = IMAdNative().initIMAdNative(imadSDK: AppDelegate.getImadSDKCore()))

nativeAd.targetingProperties.setTargetingProperties(
       key:IMAdTargetingProperties.IMATargetPropKey_Age, value:"28"

nativeAd.targetingProperties.setTargetingProperties(
       key:IMAdTargetingProperties.IMATargetPropKey_Gender, value:"F")

nativeAd.targetingProperties.setTargetingProperties(key:"customKey", value:"customValue")
```

3. Load a banner

Every banner needs to be loaded from the server with a Zone ID. The Zone IDs are allocated for your app by Innity.

NOTE: The test zone IDs are **alphabetical** while the live zone IDs are **numerical**. You would still pass the zone ID **as a string**, e.g. zoneId: "8234"

```
nativeAd.load(zoneId: "native", adLoadCallback: self)
```

Once a banner has successfully loaded, the **adReady** delegate method will be called. This delegate method will only be called if there is an ad available for display in the requested zone.

The **adFailed** delegate method will be called whenever there is no ad available or a timeout occurred during loading. The timeout is set to 30 seconds.

4. Handling Banner Load Events

There are various banner events that the ad unit may trigger, at minimum, you'll need to handle **adReady** - when the ad has loaded and is ready for you to ad to your view and **adFailed** when the ad fails to load.

```
func adReady(adUnit: AdUnit)
{
       self.adUnit = adUnit as! NativeAd

       DispatchQueue.main.async
       {
              // Present the ad in the main thread
              adsTitle.text = self.adUnit.getTitle()
              adsDesc.text = self.adUnit.getDesc()
              thumbnailImage.image = UIImage(data: self.adUnit.getThumbnail())

              // Notify the SDK that the ad unit was displayed
              self.adUnit.displayed
       }
}

func adFailed(zoneId: String, error: Error)
{
       //failed
}
```

5. Respond to a Click/Tap

Once you have presented the ad, it is your responsibility to respond to the users' tap on the ad by calling the **clicked** method. This method will invoke an embedded web browser to display the advertiser's target URL.

```
self.adUnit.clicked(viewController: self)
```

6. Destroying the IMAdNative and AdUnit

Remember to cleanup once you're done with IMAdNative and AdUnit.

```
self.native = nil

self.adUnit.destroy()
self.adUnit = nil
```

Implementing Native Video Ads

Native Video Ads are almost identical to Native Ads. You can present ads according to the look and feel of your app however there are no static images for Native Video Ads.

The SDK will return the follow data to you for you to present in your app as you wish:

1. Title
2. Description
3. Advertiser's Brand Name

You'll also have to call the **displayed** method once the ad is visible within your app and call the **clicked** method when the ad is tapped on.

Once again, the process is very similar to Native Ads:

1. Create an instance of the IMAd SDK Core.

2. Create an instance of the IMAd Native Video Ad format and set any targeting properties your app is able to provide, for example, the user's gender and age.

3. Load an ad from a designated zone (Zone IDs are issued by Innity, the IDs are created according to how you intent to target ads. You may want to have a zone for each category of content your app provides, for example, any content related to Fashion may be considered to be in the same zone and hence share the same zone ID.)

4. Once you receive the notification that an ad is available and has been loaded, present the ad in your own style when you're ready. Also invoke the **displayed** method when the ad comes into view and play the video.

5. Respond to a click/tap and call the **clicked** method.

6. Finally, you'll want to clean up/destroy the ad when you're done using it.

1. Create an IMAd SDK Core Object

```
import IMAd

imadSDKCore = IMAd().initIMAdSDK()
```


2. Create an IMAdNativeVideo ad and set targeting

Once you've created an IMAdNativeVideo object, we recommend that you set targeting properties if you know your user. There are a list of standard properties provided, please see the **Audience Targeting Properties** table below, however, if there are certain properties you can provide which are useful for targeting, you may also set your own custom key, value pairs.

```
let nativeVideo
=IMAdNativeVideo().initIMAdNativeVideo(imadSDK:AppDelegate.getImadSDKCore())
```

```
nativeVideo.targetingProperties.setTargetingProperties(
      key:IMAdTargetingProperties.IMATargetPropKey_Age, value:"28"

nativeVideo.targetingProperties.setTargetingProperties(
      key:IMAdTargetingProperties.IMATargetPropKey_Gender, value:"F")

nativeVideo.targetingProperties.setTargetingProperties(key:"customKey",
value:"customValue")
```

3. Load a banner

Every banner needs to be loaded from the server with a Zone ID. The Zone IDs are allocated for your app by Innity.

NOTE: The test zone IDs are **alphabetical** while the live zone IDs are **numerical**. You would still pass the zone ID **as a string**, e.g. zoneId: "8234"

```
nativeVideo.load(zoneId: "native_video", adLoadCallback: self)
```

Once a banner has successfully loaded, the **adReady** delegate method will be called. This delegate method will only be called if there is an ad available for display in the requested zone.

The **adFailed** delegate method will be called whenever there is no ad available or a timeout occurred during loading. The timeout is set to 30 seconds.

4. Handling Banner Load Events

There are various banner events that the ad unit may trigger, at minimum, you'll need to handle **adReady** - when the ad has loaded and is ready for you to ad to your view and **adFailed** when the ad fails to load.

```
func adReady(adUnit: AdUnit)
{
      self.adUnit = adUnit as! NativeVideoAd

      DispatchQueue.main.async
      {
      // Present the ad in the main thread
            adsTitle.text = self.adUnit.getTitle()
            adsDesc.text = self.adUnit.getDesc()

            videoView = UIView(frame:CGRect(
                  x: 20, y: 100, width:UIScreen.main.bounds.width-40, height: height))
            self.adUnit.playVideo(videoView: videoView)

            // Notify the SDK that the ad unit was displayed
            self.adUnit.displayed()
      }
}

func adFailed(zoneId: String, error: Error)
{
    //failed
}
```

5. Respond to a Click/Tap

Once you have presented the ad, it is your responsibility to respond to the users' tap on the ad by calling the **clicked** method. This method will invoke an embedded web browser to display the advertiser's target URL.

```
self.adUnit.clicked(viewController: self)
```

6. Destroying the IMAdNativeVideo and AdUnit

Remember to cleanup once you're done with IMAdNativeVideo and AdUnit.

```
self.nativeVideo = nil

self.adUnit.destroy()
self.adUnit = nil
```

Implementing Alternate Ads

It is recommended that you attempt to load an ad through the Innity Mobile Ads SDK first for the best premium rates before invoking other mobile ad SDKs.

If the Innity Mobile Ads SDK does not return an ad, you may proceed to display potential ads from other mobile ad SDKs.

The general process to implement alternate ads is as follows:

1. Load Innity Ads as usual
2. Create a function to load alternate ads (e.g. from Google Mobile Ads, AdMob, MoPub…etc)
3. Listen to the **adFailed** callback and call the function above if the callback is triggered
4. You may also create a timeout after **loadBanner** to call the above alternate ad loading function if both **adFailed** and **adReady** are not triggered within your preferred timeframe. Remember to cancel the timeout inside the adReady function.

Sample Code

```
override func viewDidLoad()
{
        super.viewDidLoad()
        engageAd = IMAdEngage().initIMAdEngage(imadSDK: AppDelegate.getImadSDKCore())
        engageAd.targetingProperties.setTargetingProperties(
                key: IMAdTargetingProperties.IMATargetPropKey_Age, value: "28")
        engageAd.targetingProperties.setTargetingProperties(
                key: IMAdTargetingProperties.IMATargetPropKey_Gender, value: "M")
        engageAd.targetingProperties.setTargetingProperties(
                key: "customKey", value: "customValue")

        // Load the banner
        engageAd.load(zoneId: "spin", adLoadCallback: self,
                    adEventCallback: self, viewController: self)

        // Set a timeout
        timer = Timer.scheduledTimer(withTimeInterval: 3.0, repeats: false,
                block: { timer in self.loadAlternateAd() });
}

……

extension AdsController: AdLoadCallback
{
    func adReady(adUnit: AdUnit)
    {
        // Cancel the timeout
        timer.invalidate()

        // ad is loaded, no need to load alternate ad
        self.adUnit = adUnit as? EngageAd
        self.view.addSubview(self.adUnit.getAdView()!)

    }

    func adFailed(zoneId: String, error: Error)
    {
        // Cancel the timeout
        timer.invalidate()


        // Call to load alternate ad
        loadAlternateAd()
```

```
    }
}

func loadAlternateAd()
{
    // Your code to load an alternate ad here
    // load AdMob, MoPub, Google Ads here
}
```

Rate Limit for Loading Ads

There is a rate limit applied on ad loading. An app cannot request more than one ad per zone within 30 seconds.

, If an ad is requested from the same zone within 30 seconds, the **adFailed** callback will be fired with a **RateLimitException** object.

Example App

An example app has been included in this SDK. Explore the **IMAdSDKDemo** to see how the an app may create and display the available banner formats.

Right Audience, Right Message

The Innity Advenue server is capable of targeting audiences and serving banner ads with *dynamic creatives* to provide the most relevant ads and messaging.

With audience targeting, the Innity Advenue ad server can determine which ads to serve depending on your user's profile.

Audience Targeting Properties

The following table shows a list of standard properties provided for your use.

System Property Key	Key	Values
IMATargetPropKey_Gender	gender	Single character, for example: "**m**" or "**f**"
IMATargetPropKey_Age	age	Numeric string, for example: "**26**"
IMATargetPropKey_Latitude	lat	Numeric string, for example: "**37.0625**"
IMATargetPropKey_Longitude	lon	Numeric string, for example: "**-95.677068**"
IMATargetPropKey_Birthday	birthday	YYYY-MM-DD format, for example: "**1988-12-31**"
IMATargetPropKey_Language	lang	ISO-639-1 language code (see: http://www.loc.gov/standards/iso639-2/php/code_list.php), for example: "**en**"

Support & Feedback

If you encounter any technical issues implementing the SDK, please contact your account manager, we would be more than happy to assist you.

We would also love to hear your feedback on your experience or on the features of the SDK, please do let us know so we can improve.