iAd Programming Guide

User Experience

2010-11-15

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Introduction **About iAd 7**

At a Glance 7
- Banner Views Use a Portion of your Interface 7
- Pause Nonessential Activities While Users Interact with Advertisements 7
- Canceling Advertising Negatively Impacts Your Application 8

Prerequisites 8
See Also 8

Chapter 1 **Banner View Concepts 9**

Banner Views Require a View Controller 10
Creating a Banner View 10
Banner View Sizes 10
Thread Safety 11

Chapter 2 **Working with Banner Views 13**

Responding to Banner Events 13
- Responding to a Touch in the Banner View 13
- Responding When an Advertisement Loads 14
- Error Handling 15

Canceling an Advertising Action 15
Changing the Banner Size Dynamically 16
Testing Banner Advertisements 17

Document Revision History 19

Figures and Listings

Chapter 1 **Banner View Concepts 9**

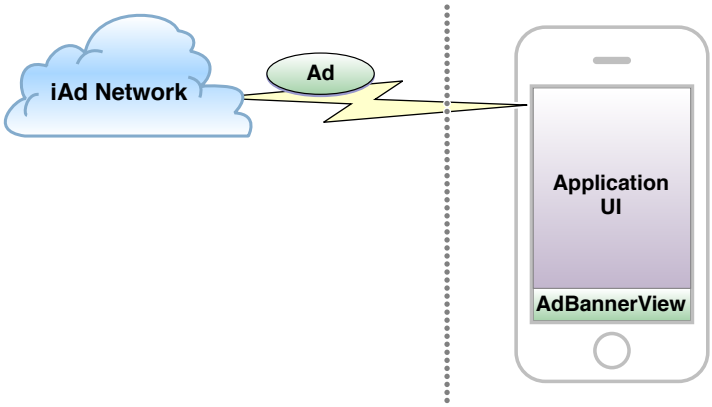
Figure 1-1 An advertisement in a banner view 9
Figure 1-2 A banner action 9
Listing 1-1 Programmatically creating a portrait banner view 10
Listing 1-2 Programmatically creating a landscape banner view 11

Chapter 2 **Working with Banner Views 13**

Listing 2-1 Allowing an action to be triggered 14
Listing 2-2 Animating in the banner view after a new advertisement is loaded 14
Listing 2-3 Removing a banner view when advertisements are not available 15
Listing 2-4 Configuring a banner view to handle landscape and portrait orientations 16
Listing 2-5 Responding to an orientation change 16

About iAd

iAd allows your applications to display advertising to the user. You are paid when users see and interact with the advertisements displayed by your application.



At a Glance

Advertising offers you an alternative to directly charging for your application. Apple sells the advertising and delivers it to your application.

Banner Views Use a Portion of your Interface

The `ADBannerView` class allows you to dedicate a portion of a user interface screen to display a banner ad. The banner view automatically downloads new advertisements to display to the user. The user may tap a visible advertisement to see additional content.

Pause Nonessential Activities While Users Interact with Advertisements

When the user taps on a banner advertisement, the most common behavior is for your application's user interface to be covered by the advertisement. Your application continues to run, but the user cannot see or interact with your application's user interface. While an advertisement is displayed, your application's activities should be scaled back, and features that require the user to see or hear the user interface should be suspended.

Canceling Advertising Negatively Impacts Your Application

While the user interacts with an advertisement, your application continues to receive events. Once the user finishes the advertising action, control returns to your application. However, if your application receives an event that requires the user's immediate attention, your application can programmatically cancel the advertisement to restore its user interface. Frequently canceling advertisements can reduce the income you receive from showing advertisements as well as the inventory of advertisements that are made available to your application.

Prerequisites

This guide assumes you know how to program in Objective-C. You should be familiar with view programming and the use of view controllers to manage your user interface. To learn more about view programming, see *iOS Application Programming Guide*. To learn more about view controllers, see *View Controller Programming Guide for iOS*.

See Also

Before you can add iAd to your application, you must join the iAd Network and sign the appropriate agreement. For more information on joining the iAd Network, see http://developer.apple.com/iad/.

For guidance and restrictions on how your application should display ad banners, see *iOS Human Interface Guidelines*.

For details about the classes and protocols in the iAd framework, see *iAd Framework Reference*.

An advertisement may move your application into the background. For example, an advertisement might launch Safari to display a web page. To learn more about what your application must do when moving to the background, see *iOS Application Programming Guide*.

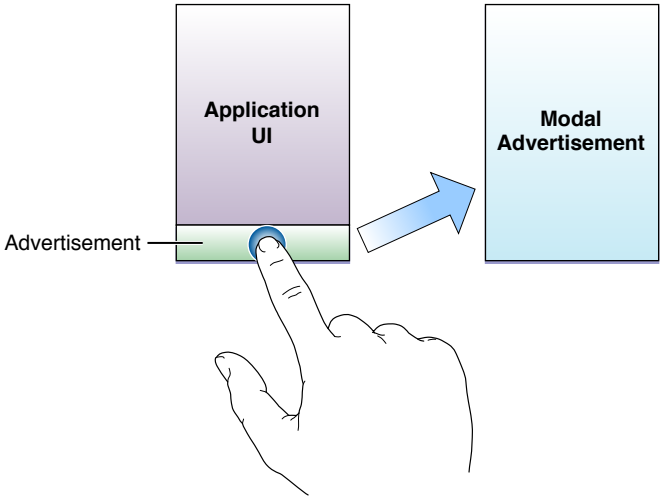
Banner View Concepts

A banner view periodically retrieves advertisements from the iAd service and displays them to the user. Your application dedicates a small portion of a user interface screen to a banner view, as shown in Figure 1-1.

Figure 1-1 An advertisement in a banner view

Advertisements define an action that takes place when the user taps on the banner. For example, an advertisement could launch another application or temporarily cover your application's user interface to play a movie or to present an interactive advertisement. Figure 1-2 shows the screen after the user taps the display.

Figure 1-2 A banner action



A key point is that the user decides when they want to see the content associated with the banner advertisement. The user remains in control.

Banner Views Require a View Controller

Any user interface screen that includes a banner view must be managed by a view controller (a class that subclasses `UIViewController`). This allows a triggered action to cover your user interface with an additional advertising screen. Whenever a banner view is visible, it must be part of a view hierarchy that is attached to the `view` property of a view controller. The simplest way to do this is to instantiate the view as part of the nib file used to instantiate the view controller's interface.

Creating a Banner View

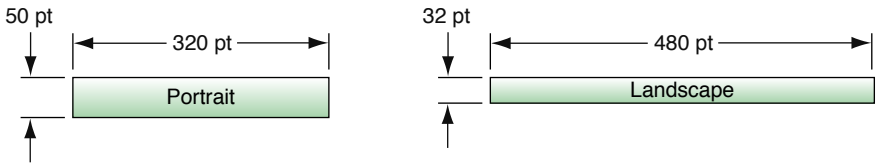
Listing 1-1 shows the simplest code that a view controller might use to programmatically create a banner view in portrait mode.

Listing 1-1 Programmatically creating a portrait banner view

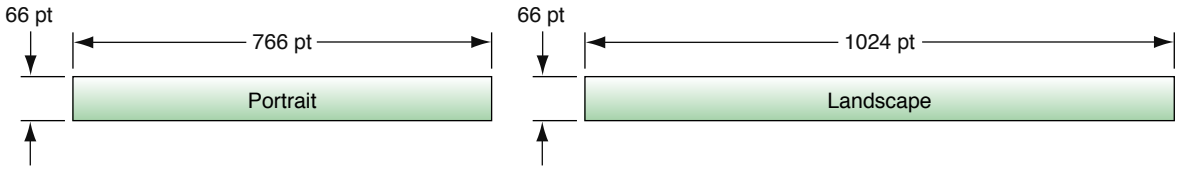
```
ADBannerView *adView = [[ADBannerView alloc] initWithFrame:CGRectZero];
adView.currentContentSizeIdentifier = ADBannerContentSizeIdentifierPortrait;
[self.view addSubview:adView];
```

Banner View Sizes

iAd supports different banner sizes for portrait and landscape applications. The exact size of advertisements depends on the device the banner is being shown on. On an iPhone, a portrait advertisement is 320 x 50 points and 480 x 32 points for a landscape advertisement. On an iPad, a portrait advertisement is 768 x 66 points and 1024 x 66 points for a landscape advertisement. In the future, additional sizes may be exposed by iAd.



iPhone iAd Sizes



iPad iAd Sizes

To ensure that advertisements are displayed properly, a banner view must always be sized to match one of the built-in advertising sizes. The proper way to do this is to set the `currentContentSizeIdentifier` property. Changing this property resizes the banner view's frame to the match this content size. The size identifier must also be included in the set attached to the `requiredContentSizeIdentifiers` property. For example, Listing 1-2 shows how your view controller could programmatically create a landscape banner view.

Listing 1-2 Programmatically creating a landscape banner view

```
ADBannerView *adView = [[ADBannerView alloc] initWithFrame:CGRectZero];
adView.requiredContentSizeIdentifiers = [NSSet
setWithObject:ADBannerContentSizeIdentifierLandscape];
adView.currentContentSizeIdentifier = ADBannerContentSizeIdentifierLandscape;
[self.view addSubview:adView];
```

If your application needs the exact size of the advertisement to use at runtime, it can call the `sizeFromBannerContentSizeIdentifier:` class method, passing in either `ADBannerContentSizeIdentifierLandscape` or `ADBannerContentSizeIdentifierPortrait`.

Thread Safety

A banner view is a user interface element. As with other user interface elements, you should only reference it from your application's main thread.

Working with Banner Views

Your application implements a banner view delegate to handle common events. Your application must:

- Respond when the user taps the banner.
- Respond when the banner view loads an advertisement.
- Respond when the banner view encounters an error.

If your application supports orientation changes, your view controller must also change the banner view's size when the orientation changes.

Responding to Banner Events

Banner views use a delegate to communicate events to your application. Although you are not required to implement a delegate, your application almost always provides one to respond to banner events. A typical pattern is to implement these methods in your custom view controller class.

Responding to a Touch in the Banner View

Before the banner view triggers an action, it calls the delegate's `bannerViewActionShouldBegin:willLeaveApplication:` method. Your delegate method performs two tasks:

- It decides whether to allow the action to be triggered.
- If the action will cover your application's user interface, this method pauses any activities that require user interaction.

Your delegate should return `YES` from this method if it wants to allow the action to be triggered. It can prevent the action from being triggered by returning `NO`. Your application should always allow actions to be triggered unless it cannot safely do so.

- If the `willLeave` parameter is `YES`, then your application is going to be moved to the background after it returns from this delegate method. This process is described in "Understanding an Application's States and Transitions" in *iOS Application Programming Guide* in the *iOS Application Programming Guide*.
- If the `willLeave` parameter is `NO`, iAd is going to cover the application's user interface after it returns from this delegate method. Your application should disable sounds, animations or other activities that require user interaction. For example, a real-time game should pause gameplay before allowing the action to be triggered.

"Creating a Banner View" shows the structure for how your application should implement this delegate method:

Listing 2-1 Allowing an action to be triggered

```
- (BOOL)bannerViewActionShouldBegin:(ADBannerView *)banner
willLeaveApplication:(BOOL)willLeave
{
    NSLog(@"Banner view is beginning an ad action");
    BOOL shouldExecuteAction = [self allowActionToRun]; // your application
implements this method
    if (!willLeave && shouldExecuteAction)
    {
        // insert code here to suspend any services that might conflict with
the advertisement
    }
    return shouldExecuteAction;
}
```

If the banner view covered the application's user interface, it calls the delegate's `bannerViewActionDidFinish:` method after the interface is restored. Your implementation of this method should restore any services paused by your application.

Important: If your application was moved into the background because the `willLeave` parameter was `YES`, then the application's user interface is never covered by the banner view and your application does not receive a call to `bannerViewActionDidFinish:`. However, if your interface was covered by the banner view, your application could still be moved into the background later, either because the advertisement launched another application or because the user chose to do so. In all cases, if your user interface was covered by the banner view, it is uncovered and your delegate's `bannerViewActionDidFinish:` is invoked before your application moves to the background. Because the application may be moving into the background, your delegate should return quickly from its `bannerViewActionDidFinish:` method.

While the user is interacting with the advertisement, your application should not delete the banner view object until after the banner view delegate's `bannerViewActionDidFinish:` method is called.

Responding When an Advertisement Loads

The iAd framework makes it easy to adopt an asynchronous model and only display an ad when one is available. Your application should never display an empty banner view. Instead, it should show the banner when an advertisement is available and hide it when the banner has nothing new to show.

When a banner view has a new advertisement to display, it calls the delegate's `bannerViewDidLoadAd:` method. This method is called even if the banner view is not currently part of the view hierarchy. Your application can use this method to attach the view to a view hierarchy or to move the banner view on screen. "Banner View Sizes" uses a property to track whether the banner view is visible. If the banner is not visible and a new advertisement is loaded, the method animates the view onto the screen.

Listing 2-2 Animating in the banner view after a new advertisement is loaded

```
- (void)bannerViewDidLoadAd:(ADBannerView *)banner
{
    if (!self.bannerIsVisible)
    {
        [UIView beginAnimations:@"animateAdBannerOn" context:NULL];
// Assumes the banner view is just off the bottom of the screen.
       banner.frame = CGRectOffset(banner.frame, 0, -banner.frame.size.height);
```

```
        [UIView commitAnimations];
        self.bannerIsVisible = YES;
    }
}
```

Error Handling

If an error occurs, the banner view calls the delegate's `bannerView:didFailToReceiveAdWithError:` method. When this happens, your application must hide the banner view. Listing 2-3 shows one way you might implement this. It uses the same property as Listing 2-2 (page 14) to keep track of whether the banner is visible. If the banner is visible and an error occurs, it moves the banner off the screen.

Even after an error is sent to your delegate, the banner view continues to try to download new advertisements. The combination of these two delegate methods allows you to display the banner only when advertisements are loaded.

Listing 2-3 Removing a banner view when advertisements are not available

```
- (void)bannerView:(ADBannerView *)banner didFailToReceiveAdWithError:(NSError
 *)error
{
   if (self.bannerIsVisible)
    {
        [UIView beginAnimations:@"animateAdBannerOff" context:NULL];
// Assumes the banner view is placed at the bottom of the screen.
        banner.frame = CGRectOffset(banner.frame, 0, banner.frame.size.height);
        [UIView commitAnimations];
        self.bannerIsVisible = NO;
    }
}
```

Canceling an Advertising Action

When the banner view covers your application's user interface to perform its action, your application continues to receive events. Your application can read the `bannerViewActionInProgress` property of the banner view to determine whether a banner action is executing. Your application should scale back its activities and avoid actions that require interaction with the user.

If an event occurs that requires the user's attention, you can invoke the banner view's `cancelBannerViewAction` method to cancel the advertising action. The action ends, and the banner view calls the delegate's `bannerViewActionDidFinish:` method.

Important: Canceling an advertising action, or preventing an advertising action from executing, can potentially impact the advertisements your application receives, and the revenue you receive through showing advertisements. Your application should cancel the action only when it urgently requires the user's attention. For example, an application that provides Voice over Internet Protocol (VoIP) might cancel an advertisement when the application receives a call from another user.

Changing the Banner Size Dynamically

Applications that intend to resize the banner view after creation should configure the `requiredContentSizeIdentifiers` property with the set of all possible sizes the view can take in your application. The most common reason to support multiple sizes in your application is to support orientation changes. If your application changes its interface in response to an orientation change, it should resize the banner view to fit the new orientation.

Listing 2-4 shows how a view controller could configure the banner view to download advertisements that have both portrait and landscape images:

Listing 2-4 Configuring a banner view to handle landscape and portrait orientations

```
self.bannerView.requiredContentSizeIdentifiers = [NSSet setWithObjects:
ADBannerContentSizeIdentifierPortrait, ADBannerContentSizeIdentifierLandscape,
 nil];
```

When an orientation change occurs, your view controller's `willRotateToInterfaceOrientation:duration:` method changes the banner size.

Listing 2-5 Responding to an orientation change

```
-
(void)willRotateToInterfaceOrientation:(UIInterfaceOrientation)toInterfaceOrientation
 duration:(NSTimeInterval)duration
{
    if (UIInterfaceOrientationIsLandscape(toInterfaceOrientation))
        self.bannerView.currentContentSizeIdentifier =
            ADBannerContentSizeIdentifierLandscape;
    else
        self.bannerView.currentContentSizeIdentifier =
            ADBannerContentSizeIdentifierPortrait;
}
```

When your application configures the `requiredContentSizeIdentifiers` property with more than one banner size, the iAd service only downloads advertisements that provide images for *all* of the specified sizes. This allows the banner view to seamlessly change the displayed advertisement when your application changes the size of the view. As this restricts the ads available to the banner view, you should only configure the `requiredContentSizeIdentifiers` property with sizes that are actually used by your application.

Note: The current version of iAd only supports portrait and landscape advertisements, so this restriction may not make much sense today. In fact, Apple expects most advertisers to provide both portrait and landscape views, so including both in the `requiredContentSizeIdentifiers` property should not significantly limit the inventory of advertisements. When additional sizes are added in the future, including unused content sizes may unnecessarily restrict your application to a smaller subset of advertisements.

Testing Banner Advertisements

While you are developing your application, iAd Network sends test advertisements to your application. To assist you in validating your implementation, the iAd Network occasionally returns errors to test your error handling code. You can also test your error handling support by turning your device's wireless capability off.

To receive advertisements from iAd Network, you need to enable the advertising service for your application. Information on how to enable advertising in your shipping application will be available before the release of iOS 4.0.

iAd Network automatically displays the correct ad depending on the application binary:

Application	Audience	Displayed Ads
Developer build	Developer	iAd Network serves test ads.
Ad-hoc distribution build	Beta Testers	iAd Network serves test ads.
Signed Distribution build	End Users	iAd Network serves live ads if you signed the iAd Network Agreement and enabled advertising for your application.

Document Revision History

This table describes the changes to *iAd Programming Guide*.

Date	Notes
2010-11-15	Updated for iOS 4.2. iAd is now supported on iPad.
2010-08-26	Clarified the expected behaviors for iAd applications.
2010-05-27	New document that describes how to display ads in your application.

Document Revision History