iAd Programming Guide

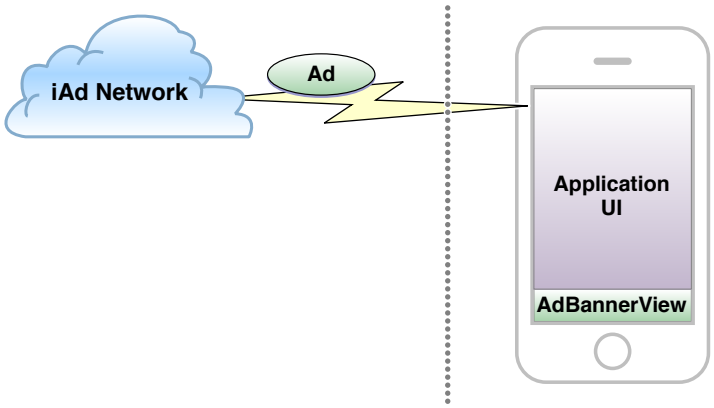
Developer

Contents

Figures, Tables, and Listings

About iAd

The iAd advertising platform provides developers new opportunities to generate revenue and promote their apps. You add banner or full-screen advertisements to your application's user interface; Apple sells advertising space and delivers ads to fill these spaces. You earn revenue when users view or interact with ads displayed by your application.



At a Glance

The iAd framework was introduced in iOS 4.0, and does the work necessary to download advertisements from the iAd Network. Your primary responsibility is to design your user interface to accomodate space for advertisements.

To Use iAd In Your Application, You Must Join the iAd Network

Before you start adding advertising support to your application, you must first agree to the iAd Network agreement. Further, you must explicitly enable iAd for each application in iTunes Connect. As a part of the iAd Network, you control the kinds of ads that are delivered to your application.

For more information on the iAd Network, see http://developer.apple.com/iad/.

For more information in enabling iAd in your application, see iTunes Connect Developer Guide.

Banner Views Use a Portion of the Screen to Display a Banner Ad

The `ADBannerView` class allows you to dedicate a portion of a user interface screen to display a banner ad. Once created, a banner view automatically downloads new advertisements to display to the user. A user taps a banner to interact with the ad's content.

Relevant Chapters "Banner View Concepts" (page 8)

Full-Screen Ads Provide Larger Advertisements to iPad Applications

The `ADInterstitialAd` class allows you to display full-screen advertisements in your application. You can add a full-screen advertisement as a page of content alongside other pages of content provided by your application, or you can display a full-screen advertisement modally when your application transitions from one screen to another. As with a banner ad, a user taps a full-screen advertisement to launch the ad's rich content.

Relevant Chapters "Full-Screen Advertisements" (page 17)

Pause Nonessential Activities While Users Interact with Advertisements

When the user taps on an advertisement, iAd typically covers the screen and displays an interactive advertisement. Your application continues to run, but the user cannot see or interact with your application's user interface. Instead, the user interacts with the rich media experience provided by the displayed advertisement. While an advertisement is displayed, your application's activities should be scaled back, and features that require the user to see or hear your application's user interface should be paused. Your application must respond to low-memory warnings and release objects that can be easily recreated after the user finishes interacting with the advertisement.

Relevant Chapters "Working with Banner Views" (page 11) and "Full-Screen Advertisements" (page 17)

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. Your application should only cancel advertising when it urgently needs the user's attention. Frequently canceling advertisements can reduce the revenue generated by your application. It may also affect the inventory of advertisements that are offered to your application.

Relevant Chapters "Working with Banner Views" (page 11) and "Full-Screen Advertisements" (page 17)

Validate Your iAd Support Before Releasing Your Application

When you build and run your application in Xcode, iAd automatically serves test advertisements to your application. You should confirm that your application properly supports the guidelines and recommendations found in this programming guide as well as those found in the *iOS Human Interface Guidelines*.

Relevant Chapters "Testing iAd Applications" (page 28)

Prerequisites

This guide assumes you know how to program in Objective-C. iAd uses views and view controllers to display advertisements, so 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 *View Programming Guide for iOS*. To learn more about view controllers, see *View Controller Programming Guide for iOS*.

As a result of a user tapping an ad, iAd may move your application into the background to launch another application. You should be familiar with the multitasking behavior introduced in iOS 4.0. See *iOS App Programming Guide*.

See Also

For sample code that demonstrates how to use banner ads, see *iAdSuite*.

For sample code that demonstrates how to use full-screen ads, see *iAdInterstitialSuite*

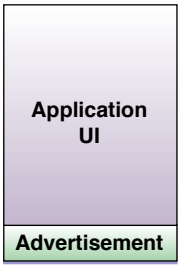
For guidance and restrictions on how your application should display advertisements, see *iOS Human Interface Guidelines*.

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

Banner View Concepts

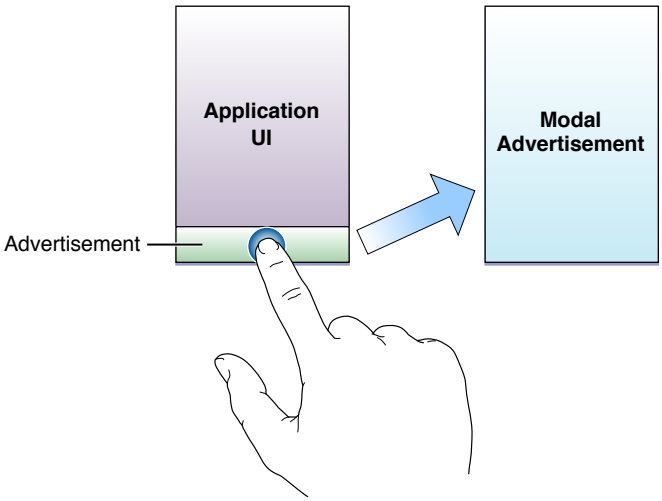
A banner view periodically retrieves advertisements from the iAd Network and displays them to the user. Your application dedicates a small portion of a user interface screen to a banner view, as shown in Figure 1-1.

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 present an interactive advertisement. Figure 1-2 shows the transition caused by a user tapping the display.

Figure 1-2 A banner action



The user always remains in control of the process; only the user decides when they want to see the content associated with the banner advertisement.

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 accomplish this is to instantiate the view as part of the nib file used to instantiate the view controller's interface.

Creating a Banner View

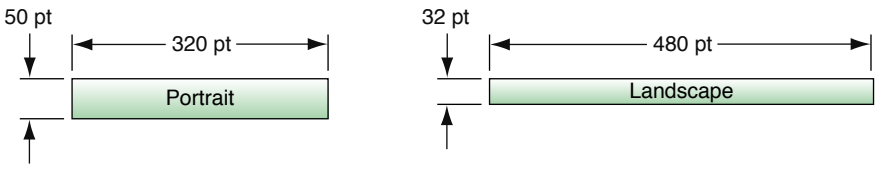
Listing 1-1 shows the simplest code that a view controller might use to programmatically create a banner view in portrait mode.

Listing 1-1 Programmatically creating a portrait banner view

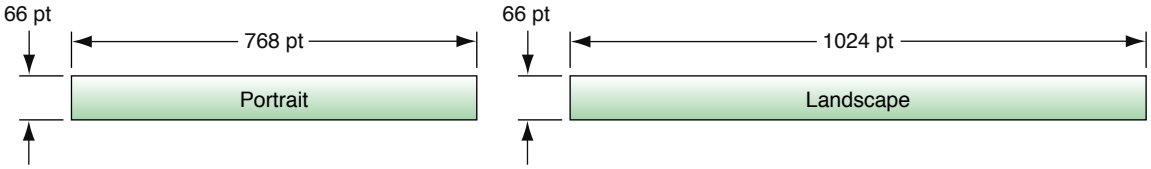
```
ADBannerView *adView = [[ADBannerView alloc] initWithFrame:CGRectZero];
adView.currentContentSizeIdentifier = ADBannerContentSizeIdentifierPortrait;
[self.view addSubview:adView];
```

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 `ADBannerView` class enforces this by preventing you from changing the frame directly. Instead, you change a banner view's frame by setting the `currentContentSizeIdentifier` property. Changing the value stored in this property resizes the banner view's frame to the match the size for the provided identifier. Before you can set a particular size identifier, the size identifier must also be included in the set of size identifiers included in 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 calls 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

Banner views use a delegate to communicate with your application. Your application implements a banner view delegate to handle common events in a banner view's lifecycle. 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.

A typical pattern is to implement these methods in your custom view controller, but you may implement them on another object if you prefer.

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

Responding to Banner Events

Most banner view delegates implement all of the following behaviors:

Responding to a Touch in the Banner View

Before the banner view triggers an advertising 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 App Programming Guide*.

- If the `willLeave` parameter is `NO`, iAd covers 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 before returning. For example, a real-time game should pause gameplay, then return `YES` to allow the action to be triggered.

"Creating a Banner View" shows the typical pattern 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 your 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.

On iOS 4.2 and earlier, your application should not delete the banner view object while the user is interacting with the advertisement. Only delete the banner view 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 the banner when it has nothing 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 add 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 13) 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.

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;
    }
}
```

Even after an error is sent to your delegate, the banner view continues to try to download new advertisements. Thus, implementing both of these delegate methods allows your application to display the banner only when advertisements are loaded.

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 never 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.

Banner View Best Practices

When designing your application, keep the following principles in mind:

- Only create a banner view when you intend to display it to the user. Otherwise, it may cycle through ads and deplete the list of available advertising for your application.
- If the user navigates from a screen of content with a banner view to a screen that does not have a banner view, and you expect them to be on that screen for a long period of time, remove the banner view from the view hierarchy, set its delegate to `nil` and release it before transitioning to the new screen of content. More generally, avoid keeping a banner view around when it is invisible to the user.
- When your application creates a banner view, there is a delay before the view can actually display an advertisement. If you intend to use that banner view on a screen of content that is only visible to the user for a short period of time, the banner may not have enough time to download an advertisement before a user finishes interacting with that screen of content. Instead, your application should create a single banner view and use it throughout your user interface. As the user navigates around your application, your application moves the banner view onto any screen that is expected to display a banner. The *iAdSuite* sample demonstrates how to implement this technique.
- When an ad transitions to a rich media experience, iAd consumes additional memory so that it can display an interactive ad to the user. This memory comes from your application's available memory. Your application must scale back its activities to allow the ad to run smoothly and respond quickly to low-memory conditions by releasing large objects that can be easily recreated after the user finishes interacting with the ad.

Full-Screen Advertisements

Full-Screen advertisements are a new kind of advertisement introduced in iOS 4.3. Full-screen advertisements are implemented by the `ADInterstitialAd` class.

Full-Screen Advertisements are Only Available on iPad

To test whether full-screen advertisements are available, retrieve the value of the `UI_USER_INTERFACE_IDIOM` macro; if the value returned from this macro is `UIUserInterfaceIdiomPad`, you may use full-screen advertisements.

Full-Screen Advertising Concepts

The full-screen advertisements provided by iAd have two common use cases:

- The advertisement is displayed as a page of content that is a peer to the content provided by your application. Typically, this means adding the full-screen ad to the contents of a scroll view.
- As a transitional screen between two portions of your application.

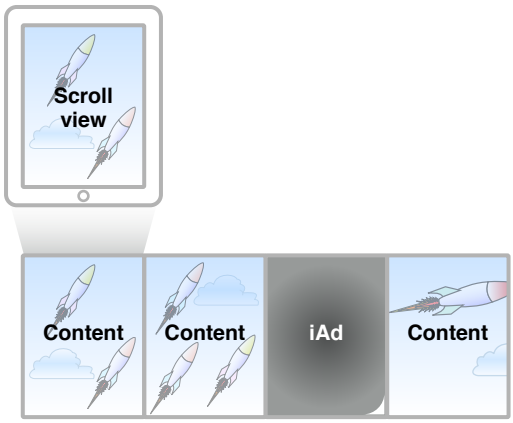
Before you embark on adding full-screen advertisements to your application, be sure you understand the expected behavior from your application for each use case you intend to introduce in your application. The *iAdInterstitialSuite* sample includes examples of both use cases.

Presenting a Full-Screen Advertisement as Content

In this use case, the advertisement is displayed alongside other content provided by your application. Your content is organized into screen-sized pages and placed into scroll view configured to scroll the visible content area on page boundaries. Figure 3-1 (page 18) illustrates this concept. The scroll view contains four views. Three views hold application content, while the fourth view contains a full-screen advertisement. The user can swipe left or right to navigate through the content. If the user taps on the iAd, the advertisement launches its rich media experience.

When your application displays full-screen advertisements alongside its own content, iAd blocks other behaviors in your application only when the user chooses to interact with an ad.

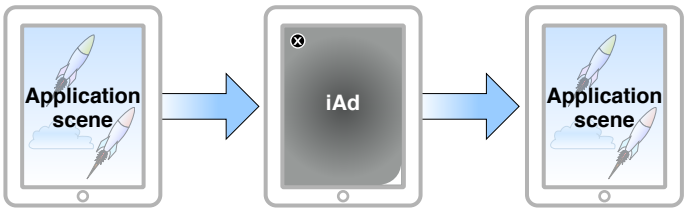
Figure 3-1 A full-screen advertisement as a modeless page



Presenting a Full-Screen Advertisement as a Transitional Screen

In this use case, the advertisement appears as a transitional screen between two other screens displayed by your application. Figure 3-2 (page 18) shows an example of this behavior. For example, the game sample in *iAdInterstitialSuite* displays an advertisement after the players finish playing a match and before allowing them to play another match. When an advertisement is displayed as a transitional page, it is displayed *modally*, which allows the user to interact with the ad. To continue to the next screen in your application, the user explicitly taps the ad's close button.

Figure 3-2 A full-screen advertisement as a modal page



The Programming Model For Full-Screen Ads Differs From Banner Views

The programming model for an `ADInterstitialAd` object is similar to that of an `ADBannerView` object, but there are a couple of important differences.

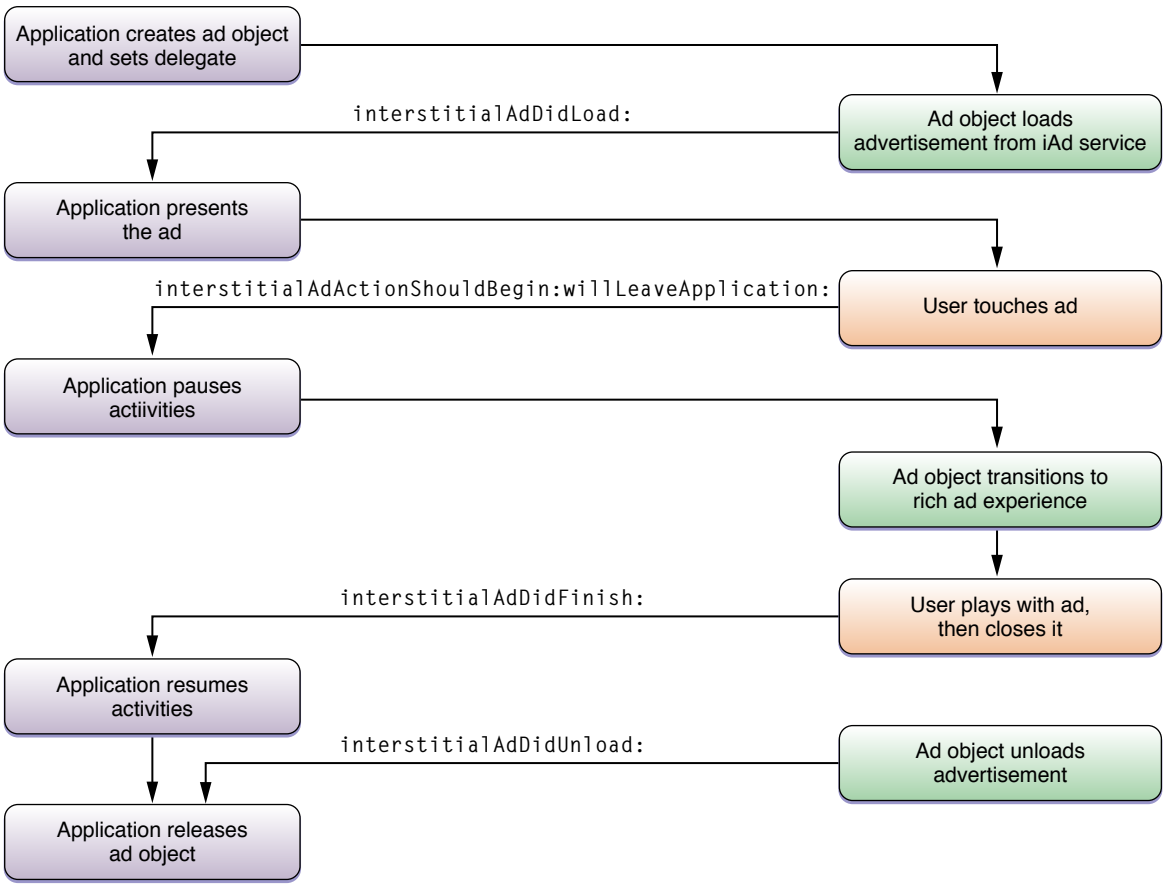
An `ADInterstitialAd` object is not actually a view. The ad object manages downloading its content, and it may create views to display the ad. However, to display the full screen ad, your application explicitly *presents* the advertisement. When your application displays a modal advertisement, it presents the advertisement using a view controller. In contrast, to add the advertisement as a modeless page, your application creates a custom view with the appropriate dimensions and adds it to the view hierarchy associated with a view controller. Then, your application presents the ad using this view. iAd uses your view to host the ad's content.

The second major difference between a banner view and an full-screen advertisement is that the full-screen advertisement does not cycle through new content. An full-screen ad object loads a single advertisement; once that content expires, your application must release the ad object. Each time your application needs to show a new advertisement, it must explicitly create a new ad object.

The Lifecycle of an Full-Screen Ad Object

Figure 3-3 shows the process your application uses to implement full-screen advertising in your application. The lifecycle begins when your application creates an `ADInterstitialAd` object and sets its delegate. The delegate is necessary because the ad object sends messages to the delegate at critical points in its lifecycle. Figure 3-3 shows most of the essential critical delegate methods.

Figure 3-3 How your application interacts with a full-screen advertisement.



After the ad object is created, it automatically downloads an advertisement from the iAd Network. When the ad object finishes downloading the advertisement, it signals your delegate. Your application takes the loaded ad and presents it to the user. The actual process your application uses to present the advertisement varies depending on whether you want to present it modally or as a modeless page.

If the touches the ad, your application is informed so that it can pause any activities that might interfere with the advertisement — this procedure is similar to the procedure a banner view uses to inform your application of a user's interests. After your application suspends its activities, the ad object loads the interactive advertisement from the iAd Network and displays it to the user. After the user finishes interacting with the ad, the ad object notifies your delegate so that your application can resume the activities it paused.

At some point after creating the ad object, the ad's content expires. The content may expire at different times, depending on how the ad is presented and how the user interacts with your application. When an ad object's content is unloaded, your delegate is called. Your application is required to release the ad object, but it often performs other tasks at this time. For example, the ADGame example found in *iAdInterstitialSuite* creates a new ad object to start the cycle over again, while the ADMagazine example removes shifts other content pages to fill the space vacated by the ad's view.

Next, you'll learn how to implement each step of the diagram above.

Creating an Ad Object

Listing 3-1 (page 21) shows a common implementation for creating a full-screen ad object. There are no other properties on the ad object to configure.

Listing 3-1 Creating a full-screen ad object

```
interstitial = [[ADInterstitialAd alloc] init];
interstitial.delegate = self;
```

After the ad object is created, it automatically starts downloading an advertisement from the iAd Network. Your application determines whether the ad object has successfully loaded its content in one of two ways:

- Your application can read the ad object's `loaded` property, which states whether the object has a loaded advertisement. Do not regularly poll this value to determine if the ad is ready to be presented. However, it may be useful for your application to check the value of this property at the moment it wants to transition to a new screen of content, particularly when displaying the advertisement is optional. For example, the ADGame example in *iAdInterstitialSuite* tests the `loaded` property on the ad object after a match finishes. If the value of this property is `YES`, then it displays the ad. If the value is `NO`, then the game skips advertising and begins a new match.
- Your delegate can implement the `interstitialAdDidLoad:` method to be informed as soon as the ad object loads the ad content. Typically, your delegate method presents the ad. Implementing a delegate method is best used when you want to insert the ad into your application's content as soon as the ad loads. You should not use this method to immediately present a modal ad to the user, as that would interrupt the task the user was performing at the moment the ad content loaded.

Important Don't create multiple ad objects at the same time; the time at which an advertisement's content expires is based in part on when the ad object finishes loading its content. If you create multiple ad objects simultaneously (to create a ready pool of ads), some ads may expire before you can display them.

Presenting an Ad

After the ad loads its content, you can present the advertisement to the user. The process you use depends on whether you plan to present it modally or as a modeless page of content.

Presenting an Ad Modally

An ad is presented modally by a view controller object. Your application creates a view controller and calls the ad object's `presentFromViewController:` method, passing in the view controller as the only parameter. Listing 3-2 (page 22) shows a minimal implementation of this behavior. In practice, because the ad is displayed modally on top of your application, your application may want to pause other activities now, before presenting the ad.

Listing 3-2 Presenting the Ad Modally

```
if (interstitial.loaded)
{
    [interstitial presentFromViewController:self];
}
```

After presenting the ad, your view controller is no longer the topmost view controller; the ad object creates a view controller and uses it to present the ad. The ad's view controller is removed after the user dismisses the ad.

Presenting an Ad as a Page of Content

Your application presents an ad modelessly by creating a view to host the ad. This view object must follow the following rules:

- The view must be contained in the view hierarchy of a view controller object; this restriction is identical to the behavior required of banner views.
- The width of the view must equal the width of the screen.

- The height of the view must be no smaller than 113 points smaller than the height of the screen and no larger than the height of the screen. For example, when an iPad is in landscape orientation, the height of the screen is 768 points; the hosting view must therefore be between 655 and 768 points in height, inclusive.
- If your view controller supports orientation changes, the view must be resized to match the new orientation.
- The view must be capable of hosting subviews.

Listing 3-3 shows a typical implementation. It uses an index for the new page to calculate the page's frame, allocates a view with that frame, and adds it to the scroll view. Then, it calls the ad's `presentInView:` method to associate the ad with the new view.

Listing 3-3 Adding an ad page to a scroll view

```
CGRect interstitialFrame = scrollView.bounds;
 interstitialFrame.origin = CGPointMake(interstitialFrame.size.width * index, 0);

 UIView *view = [[UIView alloc] initWithFrame:interstitialFrame];
 [scrollView addSubview:view];
[interstitial presentInView:view];
```

Important When your view hosts the advertisement, the ad object adds a subview to your view. Do not attempt to modify this view or add other views to obscure it. While the ad object's content is presented by your view, you should treat that view's contents as if it were owned by the ad object. You may change the view's position within its parent view; this allows you to move the view to a different page in your scroll view.

Handling User Interactions with the Ad

When a user taps on the ad, it triggers an action. The behavior is identical to that of a banner view.

Beginning an Advertising Action

Before the ad triggers an action, it calls the delegate's `interstitialAdActionShouldBegin: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 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 App 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.

Listing 3-4 shows the structure for how your application should implement this delegate method:

Listing 3-4 Allowing an action to be triggered

```
- (BOOL)interstitialAdActionShouldBegin:(ADInterstitialAd *)banner
willLeaveApplication:(BOOL)willLeave
{
    NSLog(@"Interstitial ad 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;
}
```

Completing an Advertising Action

If the full-screen ad displayed the rich media ad inside your application, it calls your delegate's `interstitialAdActionDidFinish:` method after the ad finishes. Your implementation of this method should restore any services paused by your application when the action started.

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 `interstitialAdActionDidFinish:`. 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 `interstitialAdActionDidFinish:` 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 `interstitialAdActionDidFinish:` method.

Canceling an Advertising Action

When the full-screen ad executes its action, your application continues to receive events. Your application can read the `actionInProgress` property of the ad object to determine whether an action is executing. While the ad is running, your application should scale back its activities and avoid actions that require interaction with the user.

If an event occurs that demands the user's attention, you can invoke the ad object's `cancelAction` method to cancel the advertising action. The action ends immediately. Your delegate's `interstitialAdActionDidFinish:` is not called, but your application is still expected to restart any services it paused before the ad action started.

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.

An Ad Object's Contents Only Persist For a Limited Period of Time

The content provided by an ad object eventually expires. When it does, ad object unloads the contents and calls your delegate. The exact circumstances where an ad's contents expire are not under the control of your application. Here are some common cases where the contents may expire:

- If the ad object encounters an error, it reports the error to your application and then unloads its contents.
- If an ad object is not presented by a view or a view controller, its contents expire after a period of 5 to 15 minutes.

- If an ad object is presented modelessly, its contents also expire after a period of 5 to 15 minutes. If the view associated with the ad object is offscreen when the content expires, the ad object unloads its contents immediately. If the advertisement is visible to the user when the contents expire, the contents are unloaded as soon as the view moves offscreen.
- If an ad object is presented modally, its contents expire immediately after the user dismisses the ad.

Listing 3-5 (page 26) shows the minimum behavior your application must implement after the ad object unloads its content. Your application may need to perform additional tasks. For example, a modal application might create a new ad object so that it has an ad to display next time it needs an advertisement. A modeless application might remove the view that hosted the ad, and shift other views to cover the gap in the displayed content. Both of these behaviors are demonstrated by the examples in the *iAdInterstitialSuite* sample.

Listing 3-5 Releasing an ad object after it unloads its content

```
- (void)interstitialAdDidUnload:(ADInterstitialAd *)interstitialAd
{
   [interstitialAd release];
}
```

Handling Full-Screen Ad Errors

When a full-screen ad object encounters an error, the ad object calls the delegate's `interstitialAd:didFailWithError:` method to allow your application to process the error. Full-screen ad objects share error codes with the `ADBannerView` class. Your application should not display the error to the user. The minimum response when an error occurs is to release the ad object. As with the case when an ad object unloads its content, you may want to perform other actions.

Listing 3-6 Logging full-screen ad errors

```
- (void)interstitialAd:(ADInterstitialAd *)interstitialAd didFailWithError:(NSError
 *)error
{
    NSLog(@"interstitialAd <%@> recieved error <%@>", interstitialAd, error);
    [interstitialAd release];
}
```

Most errors returned from a full-screen ad object can be handled equally. The exception is when a full-screen ad object returns `ADErrorConfigurationError` to your application. A configuration error indicates that you have not enabled iAd for your application in iTunes Connect; ads are served to your application only after you enable iAd.

Testing iAd Applications

Testing is critical for iAd applications; by working well with iAd, your application makes it easier for the user to explore the rich media ads provided by iAd. This, in turn, results in larger revenues generated by your application.

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 manually by turning your device's wireless capability off.

iAd Network automatically displays the correct ad depending on the how your application binary was downloaded onto your test device, as shown in Table 4-1 (page 28).

Table 4-1 Advertisements displayed by your application

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.

Testing Checklist

Use this checklist to assist you in testing your iAd applications.

Banner Views

- Do your iAd banners change orientation with the rest of your user interface? See "Changing the Banner Size Dynamically" (page 15).

- Does your application only show banners that have a loaded advertisement? See "Responding When an Advertisement Loads" (page 13) and "Error Handling" (page 14).
- Does your application pause other activities when the user taps a banner? See "Responding to a Touch in the Banner View" (page 11).
- Does your application observe low memory warnings? See "Use Memory Efficiently" in *iOS App Programming Guide*.
- When an iAd action finishes, are all services started again? See "Responding to a Touch in the Banner View" (page 11).
- If your application cancels an iAd action, are all services started again? See "Canceling an Advertising Action" (page 14).
- Do banner views appear quickly when a user moves to a new screen of content? See "Banner View Best Practices" (page 16).

Full-Screen Advertisements

- Do your full-screen banners also change orientation? See "Presenting an Ad" (page 22).
- Does your application pause other activities when the user taps a banner? See "Beginning an Advertising Action" (page 23)
- Does your application observe low memory warnings? See "Use Memory Efficiently" in *iOS App Programming Guide*.
- When an iAd action finishes, do all services you paused start up again? See "Completing an Advertising Action" (page 24).
- If your application cancels an iAd action, do all services you paused start up again? See "Canceling an Advertising Action" (page 25).
- Do your full-screen advertisements expire before being displayed to the user? See "Creating an Ad Object" (page 21).
- When a full-screen advertisement placed in a scroll view expires, does your application clean up the content view properly? See "An Ad Object's Contents Only Persist For a Limited Period of Time" (page 25).

Document Revision History

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

Date	Notes
2011-09-14	Corrected the size of an iAd banner in portrait mode.
2011-06-06	Added new best practices for banner ads.
2011-02-24	Added a chapter about full-screen advertisements on iPad. Added best practices for banner advertisements.
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.

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