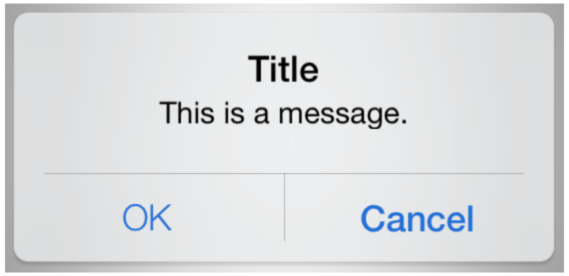
Alert Views

Alert views display a concise and informative alert message to the user. Alert views convey important information about an app or the device, interrupting the user and requiring them to stop what they're doing to choose an action or dismiss the alert. For example, iOS uses alerts to warn the user that battery power is running low, so they can connect a power adapter before their work is interrupted. An alert view appears on top of app content, and must be manually dismissed by the user before he can resume interaction with the app.



Purpose. Alert views allow users to:

- Be immediately informed of critical information
- Make a decision about a course of action

Implementation. Alert views are implemented in the `UIAlertView` class and discussed in the *UIAlertView Class Reference*.

Configuration. Alert views are created, initialized, and configured in code, typically residing in a view controller implementation file.

Content of Alert Views (Programmatic)

You cannot create or manipulate alert views in Interface Builder. Rather, an alert view floats over an existing view to interrupt its presentation, and it requires the user to dismiss it. If an alert view contains a custom button enabling the users to choose an alternative action, rather than simply dismissing the alert, that action is handled by the alert view’s delegate.

When setting alert view content. vou can control the number of buttons. their titles. displaved text. and inclusion of one

On This Page

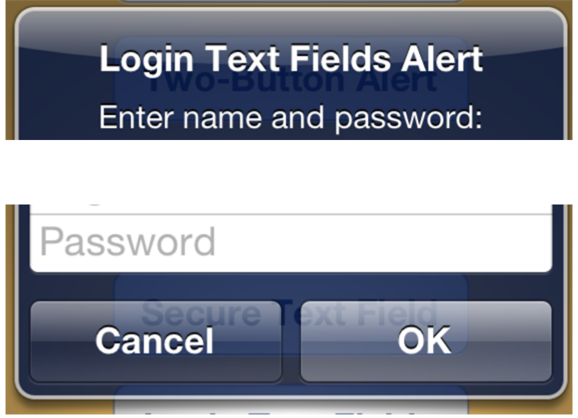
When you create an alert view object from the UIAlertView class, you can initialize its most important properties with one method, initWithTitle:message:delegate:cancelButtonTitle:otherButtonTitles:. Depending on your app’s needs, that single message may be enough to configure a fully functional alert object, as shown in the following code. After you have created the alert object, send it a show message to display the alert.

```
UIAlertView *theAlert = [[UIAlertView alloc] initWithTitle:@"Title"
                                                   message:@"This is the message."
                                                  delegate:self
                                         cancelButtonTitle:@"OK"
                                         otherButtonTitles:nil];
[theAlert show];
```

Every alert has a Cancel button so that the user can dismiss the alert. You can add additional, custom buttons to enable the user to perform some other action related to the alert, such as rectifying the problem the alert warned about. Although you can add multiple custom buttons to an alert, *iOS Human Interface Guidelines* recommend that you limit alerts to two buttons, and consider using an action sheet instead if you need more.

To add a titled custom button to an alert, send it an addButtonWithTitle: message. Alternatively, you can pass the custom button title, followed by a comma and nil terminator, with the otherButtonTitles: parameter when you initialize the alert view object.

Optionally, an alert can contain one or two text fields, one of which can be a secure text-input field. You add text fields to an alert after it is created by setting its alertViewStyle property to one of the styles specified by the UIAlertViewStyle constants. The alert view styles can specify no text field (the default style), one plain text field, one secure text field (which displays a bullet character as each character is typed), or two text fields (one plain and one secure) to accommodate a login identifier and password.



Behavior of Alert Views (Programmatic)

If your alert has a custom button, you must designate a delegate to handle the button's action, and the delegate must conform to the `UIAlertViewDelegate` protocol. You designate the delegate with the `delegate` parameter when you initialize the alert view object. The delegate must implement the `alertView:clickedButtonAtIndex:` message to respond when the custom button is tapped; otherwise, your custom buttons do nothing. For example, the following code shows an implementation that simply logs the title of the button that was tapped:

```
- (void)alertView:(UIAlertView *)theAlert clickedButtonAtIndex:
  (NSInteger)buttonIndex
{
    NSLog(@"The %@ button was tapped.", [theAlert buttonTitleAtIndex:buttonIndex]);
}
```

In the delegate method `alertView:clickedButtonAtIndex:`, depending on which button the user tapped, you can retrieve the values of text fields in your alert view with the `textFieldAtIndex:` method.

```
if (theAlert.alertViewStyle == UIAlertViewStyleLoginAndPasswordInput) {
    NSLog(@"The login field says %@, and the password is %@.",
          [theAlert textFieldAtIndex:0].text, [theAlert textFieldAtIndex:1].text);
}
```

The alert view is automatically dismissed after the `alertView:clickedButtonAtIndex:` delegate method is invoked. Optionally, you can implement the `alertViewCancel:` method to take the appropriate action when the system cancels your alert view. An alert view can be canceled at any time by the system—for example, when the user taps the

behavior is to si

On This Page

Appearance of Alert Views

You cannot customize the appearance of alert views.

Using Auto Layout with Alert Views

The layout of alert views is handled for you. You cannot create Auto Layout constraints between an alert view and another user interface element.

For general information about using Auto Layout with iOS views, see Using Auto Layout with Views.

Making Alert Views Accessible

Alert views are accessible by default.

Accessibility for alert views pertains to the alert title, alert message, and button titles. If VoiceOver is activated, it speaks the word “alert” when an alert is shown, then speaks its title followed by its message if set. As the user taps a button, VoiceOver speaks its title and the word “button.” As the user taps a text field, VoiceOver speaks its value and “text field” or “secure text field.”

For general information about making iOS views accessible, see Making Views Accessible.

Internationalizing Alert Views

To internationalize an alert view, you must provide localized translations of the alert title, message, and button titles. Screen metrics and layout may change depending on the language and locale.

For more information, see *Internationalization and Localization Guide*.

Debuggin

On This Page

When debugging issues with alert views, watch for this common pitfall:

Not testing localizations. Be sure to test the alert views in your app with the localizations you intend to ship. In particular, button titles can truncate if they are longer in localizations other than the one in which you designed your user interface. Following the HI guideline to give buttons short, logical titles helps to ameliorate this potential problem, but localization testing is also required.

Elements Similar to an Alert View

The following elements provide similar functionality to an alert view:

- **Action Sheet.** Present an action sheet when users tap a button in a toolbar, giving them choices related to the action they've initiated. For more information, see Action Sheets.
- **Modal View.** Present a modal view (that is, the view controller uses a modal presentation style) when users initiate a subtask in the context of their workflow or another task. For more information, see *UIViewController Class Reference*.