UIAccessibility Protocol Reference

User Experience

2010-05-18

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

Simultaneously published in the United States and Canada.

Contents

UIAccessibility Protocol Reference

(informal protocol)

Adopted by	NSObject
Framework	/System/Library/Frameworks/UIKit.framework
Companion guide	Accessibility Programming Guide for iOS
Declared in	UIAccessibility.h UIAccessibilityConstants.h

Overview

The `UIAccessibility` informal protocol provides accessibility information about an application's user interface elements. Assistive applications, such as VoiceOver, convey this information to users with disabilities to help them use the application.

Standard UIKit controls and views implement the `UIAccessibility` methods and are therefore accessible to assistive applications by default. This means that if your application uses only standard controls and views, such as `UIButton`, `UISegmentedControl`, and `UITableView`, you need only supply application-specific details when the default values are incomplete. You can do this by setting these values in Interface Builder or by setting the properties in this informal protocol.

The `UIAccessibility` informal protocol is also implemented by the `UIAccessibilityElement` class, which represents custom user interface objects. If you create a completely custom `UIView` subclass, you might need to create an instance of `UIAccessibilityElement` to represent it. In this case, you would support all the `UIAccessibility` properties to correctly set and return the accessibility element's properties.

Tasks

Determining Accessibility

`isAccessibilityElement` (page 8) *property*

A Boolean value indicating whether the receiver is an accessibility element that an assistive application can access.

Configuring an Accessibility Element

`accessibilityLabel` (page 7) *property*
A succinct label that identifies the accessibility element, in a localized string.

`accessibilityHint` (page 6) *property*
A brief description of the result of performing an action on the accessibility element, in a localized string.

`accessibilityValue` (page 8) *property*
The value of the accessibility element, in a localized string.

`accessibilityTraits` (page 8) *property*
The combination of accessibility traits that best characterize the accessibility element.

`accessibilityFrame` (page 6) *property*
The frame of the accessibility element, in screen coordinates.

`accessibilityLanguage` (page 8) *property*
The language in which to speak the accessibility element's label, value, and hint.

Properties

For more about Objective-C properties, see “Properties” in *The Objective-C Programming Language*.

accessibilityFrame

The frame of the accessibility element, in screen coordinates.

```
@property(nonatomic) CGRect accessibilityFrame
```

Discussion
The default value for this property is `CGRectZero` unless the receiver is a `UIView` or subclass of `UIView`, in which case the value is the frame of the view.

You must set this property for an accessibility element that represents an object that is not a subclass of `UIView`, because the screen coordinates of such an object are not already known. (You do not have to set this property for an accessibility element that represents a subclass of `UIView` because such an object’s screen coordinates are already known.)

Declared In
`UIAccessibility.h`

accessibilityHint

A brief description of the result of performing an action on the accessibility element, in a localized string.

```
@property(nonatomic, copy) NSString *accessibilityHint
```

Discussion
The default value for this property is `nil` unless the receiver is a UIKit control, in which case the value is a system-provided hint based on the type of control.

An accessibility hint helps users understand what will happen when they perform an action on the accessibility element, when that result is not obvious from the accessibility label. For example, if you provide an Add button in your application, the button's accessibility label helps users understand that tapping the button adds values in the application. If, on the other hand, your application allows users to play a song by tapping its title in a list of song titles, the accessibility label for the list row does not tell users this. To help an assistive application provide this information to users with disabilities, an appropriate hint for the list row would be "Plays the song."

Follow these guidelines to create a hint for an accessibility element:

- The hint should be a very brief phrase that begins with a plural verb that names the results of the action, such as "Plays the song" or "Purchases the item."

  Avoid beginning the phrase with a singular verb, because this can make the hint sound like a command. For example, do not create a hint such as "Play the song" or "Purchase the item."
- Don't repeat the action type in the hint. For example, do not create hints such as "Tap to play the song" or "Tapping plays the song."
- Don't repeat the control or view type in the hint. For example, do not create hints such as "Plays the song in the row" or "Button that adds a contact name."

Declared In
`UIAccessibility.h`

accessibilityLabel

A succinct label that identifies the accessibility element, in a localized string.

```
@property(nonatomic, copy) NSString *accessibilityLabel
```

Discussion
The default value for this property is `nil` unless the receiver is a UIKit control, in which case the value is a label derived from the control's title.

Note: If you supply `UIImage` objects to display in a `UISegmentedControl`, you can set this property on each image to ensure that the segments are properly accessible.

If you implement a custom control or view, or if you display a custom icon on a UIKit control, you should set this property to make sure your accessibility elements have appropriate labels. If an accessibility element does not display a descriptive label, set this property to supply a short, localized label that succinctly identifies the element. For example, a "Play music" button might display an icon that shows sighted users what it does. To be accessible, however, the button should have the accessibility label "Play" or "Play music" so that an assistive application can provide this information to users with disabilities. Note, however, that the label should never include the control type (such as "button") because this information is contained in the traits associated with the accessibility element.

Declared In
`UIAccessibility.h`

accessibilityLanguage

The language in which to speak the accessibility element's label, value, and hint.

```
@property(nonatomic, retain) NSString *accessibilityLanguage
```

Discussion
The default value for this property is `nil`. If no language is set, the user's current language setting is used.

If you need to set this property, be sure to use a language ID tag that follows the format defined in the BCP 47 specification (a draft of this specification is available at http://www.rfc-editor.org/).

Declared In
`UIAccessibility.h`

accessibilityTraits

The combination of accessibility traits that best characterize the accessibility element.

```
@property(nonatomic) UIAccessibilityTraits accessibilityTraits
```

Discussion
The default value for this property is `UIAccessibilityTraitNone` (page 10) unless the receiver is a UIKit control, in which case the value is the standard set of traits associated with the control.

If you implement a custom control or view, you need to select all the accessibility traits that best characterize the object and OR them together with its superclass's traits (in other words, with `super.accessibilityTraits`). See "Accessibility Traits" (page 9) for a complete list of traits.

Declared In
`UIAccessibility.h`

accessibilityValue

The value of the accessibility element, in a localized string.

```
@property(nonatomic, copy) NSString *accessibilityValue
```

Discussion
The default value for this property is `nil` unless the receiver is a UIKit control, in which case value of the property represents the value of the control, when it differs from the label.

When an accessibility element has a static label, but a dynamic value, you should set this property to return the value. For example, although an accessibility element that represents a text field might have the label "Message," its value is the text currently inside the text field.

Declared In
`UIAccessibility.h`

isAccessibilityElement

A Boolean value indicating whether the receiver is an accessibility element that an assistive application can access.

```
@property(nonatomic) BOOL isAccessibilityElement
```

Discussion
The default value for this property is `NO` unless the receiver is a standard UIKit control, in which case the value is `YES`.

Assistive applications can only get information about objects that are represented by accessibility elements. Therefore, if you implement a custom control or view that should be accessible to users with disabilities, you should set this property to `YES`. The only exception to this is a view that merely serves as a container for other items that should be accessible. Such a view should implement the `UIAccessibilityContainer` protocol and set this property to `NO`.

Declared In
`UIAccessibility.h`

Constants

UIAccessibilityTraits

A mask that contains the OR combination of the accessibility traits that best characterize an accessibility element.

```
typedef uint64_t UIAccessibilityTraits;
```

Availability
Available in iOS 3.0 and later.

Declared In
`UIAccessibilityConstants.h`

Accessibility Traits

Accessibility traits that tell an assistive application how an accessibility element behaves or should be treated.

```
UIAccessibilityTraits UIAccessibilityTraitNone;
UIAccessibilityTraits UIAccessibilityTraitButton;
UIAccessibilityTraits UIAccessibilityTraitLink;
UIAccessibilityTraits UIAccessibilityTraitSearchField;
UIAccessibilityTraits UIAccessibilityTraitImage;
UIAccessibilityTraits UIAccessibilityTraitSelected;
UIAccessibilityTraits UIAccessibilityTraitPlaysSound;
UIAccessibilityTraits UIAccessibilityTraitKeyboardKey;
UIAccessibilityTraits UIAccessibilityTraitStaticText;
UIAccessibilityTraits UIAccessibilityTraitSummaryElement;
UIAccessibilityTraits UIAccessibilityTraitNotEnabled;
UIAccessibilityTraits UIAccessibilityTraitUpdatesFrequently;
UIAccessibilityTraits UIAccessibilityTraitStartsMediaSession;
UIAccessibilityTraits UIAccessibilityTraitAdjustable;
```

Constants

`UIAccessibilityTraitNone`

The accessibility element has no traits.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitButton`

The accessibility element should be treated as a button.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitLink`

The accessibility element should be treated as a link.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitSearchField`

The accessibility element should be treated as a search field.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitImage`

The accessibility element should be treated as an image.

This trait can be combined with the button or link traits.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitSelected`

The accessibility element is currently selected.

You can use this trait to characterize an accessibility element that represents, for example, a selected table row or a selected segment in a segmented control.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitPlaysSound`

The accessibility element plays its own sound when activated.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitKeyboardKey`
The accessibility element behaves as a keyboard key.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitStaticText`
The accessibility element should be treated as static text that cannot change.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitSummaryElement`
The accessibility element provides summary information when the application starts.

You can use this trait to characterize an accessibility element that provides a summary of current conditions, settings, or state, such as the current temperature in the Weather application.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitNotEnabled`
The accessibility element is not enabled and does not respond to user interaction.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitUpdatesFrequently`
The accessibility element frequently updates its label or value.

You can use this trait to characterize an accessibility element that updates its label or value too often to send update notifications. Including this trait allows an assistive application to avoid handling continual notifications and, instead, poll for changes when it needs updated information. For example, you might use this trait to characterize the readout of a stopwatch.

Available in iOS 3.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitStartsMediaSession`
The accessibility element starts a media session when it is activated.

You can use this trait to silence the audio output of an assistive technology, such as VoiceOver, during a media session that should not be interrupted. For example, you might use this trait to silence VoiceOver speech while the user is recording audio.

Available in iOS 4.0 and later.

Declared in `UIAccessibilityConstants.h`.

`UIAccessibilityTraitAdjustable`
The accessibility element allows continuous adjustment through a range of values.

You can use this trait to characterize an accessibility element that users can adjust in a continuous manner, such as a slider or a picker view. If you specify this trait on an accessibility element, you must also implement the `accessibilityIncrement` and `accessibilityDecrement` methods in the `UIAccessibilityAction` protocol.

Available in iOS 4.0 and later.

Declared in `UIAccessibilityConstants.h`.

UIAccessibilityNotifications

A notification that an accessible application can send.

```
typedef uint32_t UIAccessibilityNotifications;
```

Availability
Available in iOS 3.0 and later.

Declared In
`UIAccessibilityConstants.h`

Notifications

UIAccessibilityAnnouncementNotification

Posted by an application when an announcement needs to be conveyed to the assistive technology. This notification includes a parameter, which is an `NSString` object that contains the announcement. An assistive technology outputs the announcement string contained in the parameter.

Use this notification to provide accessibility information about events that do not update the application user interface (UI), or that update the UI only briefly.

Availability
Available in iOS 4.0 and later.

Declared In
`UIAccessibilityConstants.h`

UIAccessibilityLayoutChangedNotification

Posted by an application when the layout of a screen changes, such as when an element appears or disappears. This notification does not include a parameter.

Availability
Available in iOS 3.0 and later.

Declared In
`UIAccessibilityConstants.h`

UIAccessibilityScreenChangedNotification

Posted by an application when a new view appears that comprises a major portion of the screen. This notification does not include a parameter.

Availability
Available in iOS 3.0 and later.

Declared In
`UIAccessibilityConstants.h`

UIAccessibilityVoiceOverStatusChanged

Posted by UIKit when VoiceOver starts or stops. This notification does not include a parameter.

You can use this notification to customize your application's user interface (UI) for VoiceOver users. For example, if you display a UI element that briefly overlays other parts of your UI, you can make the display persistent for VoiceOver users, but allow it to disappear as designed for users who are not using VoiceOver. Note that you can also use `UIAccessibilityIsVoiceOverRunning` to determine whether VoiceOver is currently running.

Availability
Available in iOS 4.0 and later.

Declared In
`UIAccessibility.h`

Document Revision History

This table describes the changes to *UIAccessibility Protocol Reference*.

Date	Notes
2010-05-18	Updated setter and getter methods to properties, and added descriptions of a property, a notification, and two traits introduced in iOS 4.0.
2009-07-14	Made minor corrections and reformatted to conform to the informal protocol appearance.
2009-05-26	New document that describes the interface for providing accessibility information about accessible user interface elements in an iPhone application.

Document Revision History