
Accessibility Overview

User Experience: Accessibility



2008-03-11



Apple Inc.
© 2004, 2008 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

.Mac is a registered service mark of Apple Inc.

Apple, the Apple logo, Carbon, Cocoa, Mac, Mac OS, and Macintosh are trademarks of Apple Inc., registered in the United States and other countries.

Finder is a trademark of Apple Inc.

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION,

EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Introduction Introduction to Accessibility Overview 9

Who Should Read This Document? 9
Organization of This Document 9
See Also 10

Chapter 1 Why Make Your Application Accessible? 11

Increase Your User Base 11
Enter New Markets 11
Take Advantage of Mac OS X Assistive Features 12
It's Not Difficult 12

Chapter 2 Developing an Accessible Application 13

Basic Design Requirements 13
Considering Specific Disabilities 14
 Visual Disabilities 14
 Hearing Disabilities 14
 Motor and Cognitive Disabilities 15

Chapter 3 The Mac OS X Accessibility Protocol 17

The Accessibility Model 17
The Accessibility Object 20
 Attributes 21
 Actions 23
Communication With Accessibility Objects 24
 Messages 25
 Notifications 25
 Hit-Testing and Keyboard Focus 26
An Example of Accessibility 26

Chapter 4 Testing for Accessibility 29

Using Accessibility Inspector 29
Using Accessibility Verifier 29

Appendix A Accessibility Keyboard Shortcuts 31

Appendix B Roles and Associated Attributes 35

Application Role	35
Browser Role	36
Busy Indicator Role	37
Button Role	38
Checkbox Role	39
Color Well Role	40
Combo Box Role	40
Disclosure Triangle Role	42
Dock Item Role	42
Drawer Role	43
Group Role	44
Grow Area Role	45
Image Role	45
Incrementor Role	46
List Role	47
Menu Role	47
Menu Bar Item Role	48
Menu Bar Role	49
Menu Button Role	50
Menu Item Role	51
Outline Role	52
Pop-Up Button Role	53
Progress Indicator Role	53
Radio Button Role	54
Radio Group Role	55
Relevance Indicator Role	56
Row Role	57
Ruler Role	58
Ruler Marker Role	58
Scroll Area Role	59
Scroll Bar Role	60
Sheet Role	61
Slider Role	62
Split Group Role	62
Splitter Role	63
Static Text Role	64
Tab Group Role	65
Table Role	65
Text Area Role	66
Text Field Role	68
Toolbar Role	69

Unknown Role 70
Value Indicator Role 70
Window Role 71

Document Revision History 73

Figures and Tables

Chapter 3 **The Mac OS X Accessibility Protocol 17**

Figure 3-1	Communication between an assistive application and other applications	18
Figure 3-2	The accessibility object hierarchy	19
Figure 3-3	Complete inheritance hierarchy of a button in a window	19
Figure 3-4	Appropriate accessibility hierarchy of a button in a window	20

Appendix A **Accessibility Keyboard Shortcuts 31**

Table A-1	Key combinations used with screen zooming	31
Table A-2	Key combinations for moving focus in full keyboard access mode	32

Appendix B **Roles and Associated Attributes 35**

Table B-1	Attributes associated with the application role	35
Table B-2	Attributes associated with the browser role	36
Table B-3	Attributes associated with the busy indicator role	37
Table B-4	Attributes associated with the button role	38
Table B-5	Attributes associated with the checkbox role	39
Table B-6	Attributes associated with the color well role	40
Table B-7	Attributes associated with the combo box role	40
Table B-8	Attributes associated with the disclosure triangle role	42
Table B-9	Attributes associated with the Dock item role	42
Table B-10	Attributes associated with the drawer role	44
Table B-11	Attributes associated with the group role	44
Table B-12	Attributes associated with the grow area role	45
Table B-13	Attributes associated with the image role	45
Table B-14	Attributes associated with the incrementor role	46
Table B-15	Attributes associated with the list role	47
Table B-16	Attributes associated with the menu role	48
Table B-17	Attributes associated with the menu bar item role	48
Table B-18	Attributes associated with the menu bar role	49
Table B-19	Attributes associated with the menu button role	50
Table B-20	Attributes associated with the menu item role	51
Table B-21	Attributes associated with the outline role	52
Table B-22	Attributes associated with the pop-up button role	53
Table B-23	Attributes associated with the progress indicator role	54
Table B-24	Attributes associated with the radio button role	54
Table B-25	Attributes associated with the radio group role	55
Table B-26	Attributes associated with the relevance indicator role	56
Table B-27	Attributes associated with the row role	57
Table B-28	Attributes associated with the ruler role	58

Table B-29	Attributes associated with the ruler marker role	59
Table B-30	Attributes associated with the scroll area role	59
Table B-31	Attributes associated with the scroll bar role	60
Table B-32	Attributes associated with the sheet role	61
Table B-33	Attributes associated with the slider role	62
Table B-34	Attributes associated with the split group role	63
Table B-35	Attributes associated with the splitter role	63
Table B-36	Attributes associated with the static text role	64
Table B-37	Attributes associated with the tab group role	65
Table B-38	Attributes associated with the table role	66
Table B-39	Attributes associated with the text area role	67
Table B-40	Attributes associated with the text field role	68
Table B-41	Attributes associated with the toolbar role	69
Table B-42	Attributes associated with the unknown role	70
Table B-43	Attributes associated with the value indicator role	71
Table B-44	Attributes associated with the window role	71

Introduction to Accessibility Overview

Accessibility is the successful access to information and information technologies by people with disabilities. Apple's commitment to accessibility is rooted in the Macintosh's legendary ease-of-use and is enhanced by the Universal Access features in Mac OS X. Beginning in Mac OS X version 10.2, Apple introduced the accessibility architecture, which defines how an assistive technology, such as a screen reader or head-tracking mouse, communicates with applications running in Mac OS X.

This document describes why you should make your application accessible, a process Apple calls access enabling. It then provides an overview of the design considerations you should make when developing an accessible application. Finally, it describes Mac OS X accessibility architecture that supports both the access enabling of applications and the development of assistive technologies.

Who Should Read This Document?

To reach the broadest range of users, all applications should be accessible. Therefore, all application developers should read this document to learn how accessibility affects their applications and how Mac OS X supports accessibility. This document is a prerequisite to the Cocoa and Carbon framework-specific documents listed in ["See Also"](#) (page 10) that describe how to access-enable these types of applications.

Note: Java developers should implement the `javax.accessibility` APIs to ensure their applications are accessible (both Swing and AWT interfaces are accessible).

If you're developing an assistive application, you should read this document for an introduction to the Mac OS X accessibility architecture. In particular, you'll learn about some of the information you can expect to receive from an accessible application.

Organization of This Document

This document contains the following chapters:

- ["Why Make Your Application Accessible?"](#) (page 11) will help you to make a business justification for spending the development time to make your applications accessible.
- ["Developing an Accessible Application"](#) (page 13) describes design considerations to keep in mind during the design process and outlines how to access-enable an application.
- ["The Mac OS X Accessibility Protocol"](#) (page 17) provides an overview of the Mac OS X accessibility architecture.
- ["Testing for Accessibility"](#) (page 29) describes how to use tools Apple provides to test the accessibility of your application.

INTRODUCTION

Introduction to Accessibility Overview

- [“Accessibility Keyboard Shortcuts”](#) (page 31) lists the keyboard shortcuts reserved by Mac OS X for accessibility purposes.
- [“Roles and Associated Attributes”](#) (page 35) lists the attributes associated with each role defined in the Mac OS X accessibility protocol.
- [“Document Revision History”](#) (page 73) lists the changes to this document.

See Also

In addition to *Accessibility Overview*, Apple developer documentation includes several documents that cover accessibility. Documents that describe specific areas of accessibility are listed below.

- *Getting Started with Accessibility* provides a brief introduction to accessibility and describes learning paths you might choose to follow.
- *Accessibility Programming Guidelines for Carbon* describes how to access-enable a Carbon application.
- *Accessibility Programming Guidelines for Cocoa* describes how to access-enable a Cocoa application.
- *Carbon Accessibility Reference* describes the functions, data types, and constants used in accessible Carbon applications.
- *NSAccessibility Reference* describes the NSAccessibility protocol and its methods and constants.

In addition to these documents, Apple maintains a website devoted to accessibility in Mac OS X, with links to more information about compatible assistive technologies:

- <http://www.apple.com/accessibility>

Why Make Your Application Accessible?

Accessibility encompasses more than just providing an alternative to a mouse-driven user interface. At its core, it's about enabling individuals and supporting their unique viewpoints and working styles.

This chapter presents several compelling reasons why you should access-enable every application you develop. If you're considering making your application accessible, you should read this chapter. Then, you should read the following chapters to learn how Mac OS X supports accessibility and how easy it is for most applications to incorporate it.

If you're an assistive application developer, you're more interested in how Mac OS X supports accessibility. You may choose to skip ahead to [“The Mac OS X Accessibility Protocol”](#) (page 17) to learn about the Mac OS X accessibility protocol.

Increase Your User Base

Millions of people have a disability or special need. These include visual and hearing impairments, physical disabilities, and cognitive and learning challenges. Access to computers is vitally important for this population, because computers can provide a level of independence that is difficult to attain any other way.

As populations around the world age, an increasing number of people will experience age-related disabilities, such as vision or hearing loss. Unlike earlier generations, members of the currently aging population are more accustomed to using computers in their daily lives. They are more likely to store a large portion of their meaningful data in digital form and to embrace digital communication. Current and future generations of the elderly will expect to be able to continue using their computers and accessing their data, regardless of the state of their vision and hearing. Applications that support customizable text displays, access by a screen reader, or the replacement of visual cues by audible ones can serve this population well.

Even people who don't necessarily identify themselves as disabled can benefit from alternate ways of interacting with applications. Think of a person suffering from carpal tunnel syndrome (a painful condition caused by the compression of a nerve in the wrist) who prefers an application that provides keyboard alternatives to its mouse-driven user interface. By providing alternate ways of using your application, you allow users to choose their own ways to work and express themselves, which ultimately broadens your user base.

Enter New Markets

Federal regulations like section 508 of the Rehabilitation Act of 1973 in the United States stipulate that computers used in government or educational settings must provide reasonable access for people with disabilities. As this regulation and others like it take effect, entrance into these markets is dependent upon your ability to supply accessible applications.

Like the localization issue a few years ago, accessibility has evolved from a good idea to an essential component of competitive applications. Apple is committed to providing the best platform from which to enter these markets. By access enabling your application and deploying it on Mac OS X, you make your application more attractive to these markets.

Take Advantage of Mac OS X Assistive Features

Mac OS X is designed to accommodate assistive technologies and has many built-in features to help people with disabilities. Users access most of this functionality through the Universal Access pane of System Preferences. Some of these built-in technologies take advantage of the same accessibility architecture that allows external assistive technologies to access your application.

For example, VoiceOver, the built-in spoken interface introduced in Mac OS X version 10.4, relies on the accessibility architecture to make the navigation and use of the system accessible to users with visual disabilities. If you access-enable your application, VoiceOver helps a visually impaired user use it.

It's Not Difficult

Mac OS X version 10.2 and later builds in support for accessibility. In particular, both Carbon and Cocoa integrate accessibility into their APIs. This means that most of accessibility comes for free when you develop applications with these frameworks. This lets you focus on providing your application-specific information to assistive technologies, which enhances the user's experience and highlights your application's unique features.

If you have an established application, you'll find that the Mac OS X accessibility architecture is designed to allow you to access-enable your application in a selective way. This allows you to enjoy the benefits of accessibility without having to redesign your application. When you're ready to access-enable your application, be sure to read the document that pertains to your chosen application framework:

- *Accessibility Programming Guidelines for Cocoa*
- *Accessibility Programming Guidelines for Carbon*

Developing an Accessible Application

Designing your application with accessibility in mind not only allows you to reach a larger group of users, it results in a better experience for all your users. You've already made the design decision to develop an application that runs in Mac OS X. Now, make sure you can deliver the Macintosh experience to all your users.

This chapter describes some of the things you should consider when developing your application. If your application includes support for the accessibility-related features this chapter describes, it will be even easier to access-enable.

Although this chapter is of most use to developers who are in the design phase of an application, it presents information about accessibility concerns all developers should know.

Basic Design Requirements

As a first step in the design process, you should familiarize yourself with the information in *Apple Human Interface Guidelines*. That document presents the best practices of application design that help you create a first-rate application for Mac OS X. In addition, *Apple Human Interface Guidelines* provides detailed specifications for designing and implementing an intuitive, consistent, and aesthetically pleasing user interface that delivers the superlative experience Macintosh users have come to expect.

During the design process, you also should be aware of the accessibility perspective on many basic design considerations. This section describes how you can support accessibility in your most fundamental design decisions. As with most accessibility design considerations, incorporating them with your design will result in a better user experience for all users.

The design principles described here are especially important to consider when developing an accessible application:

- **Support full keyboard navigation.** For many users, a mouse is difficult, if not impossible, to use. Consequently, a user should be able to perform all your application's functions using the keyboard alone. Taking this consideration into account will make access enabling your application an even easier task.
- **Don't override built-in keyboard shortcuts.** This applies both to the keyboard shortcuts Mac OS X reserves (listed in the *Apple Human Interface Guidelines* appendix "Keyboard Shortcuts Quick Reference") and to the accessibility-related keyboard shortcuts (listed in "[Accessibility Keyboard Shortcuts](#)" (page 31)). As a general rule, you should never override reserved keyboard shortcuts. In particular, you should take care not to override accessibility-related keyboard shortcuts or your application will not be accessible to users who enable full keyboard access.

A corollary to this principle is to avoid creating too many new keyboard shortcuts that are specific to your application. Users should not have to memorize a new set of keyboard commands for each application they use.

- **Provide alternatives for drag-and-drop operations.** If your application relies on drag-and-drop operations in its workflow, you should provide alternate ways to accomplish the same tasks. This may not be easy; in fact, it may require the design of an alternate interface for applications that are heavily dependent on drag and drop.

For example, the original Mac OS X Finder application was designed to provide a simple drag-and-drop interface to the file system. In keeping with its accessibility goals, however, the Finder in Mac OS X version 10.2 and later adds keyboard support that allows users to copy and move files using keyboard commands instead of the mouse.

- **Make sure there's always a way out of your application's workflow.** This is important for all users, of course, but it's essential for users of assistive technologies. A user relying on an assistive application to use your application may have a somewhat narrower view of your application's user interface. For this reason, it's especially important to make cancelling operations and retracing steps easy.

Considering Specific Disabilities

Following the guidelines in “[Basic Design Requirements](#)” (page 13) will help you design an easy-to-use application that will be easy to access-enable. There may be specific information about particular disabilities you don't know, however, and this information is useful to keep in mind during the design process.

The following sections describe some broad categories of disabilities and offer suggestions for specific design solutions and adaptations you can make. The main theme of these suggestions is to provide as many alternate modes of content display as possible. The more ways your application presents information, the more likely your users will be to find the way that suits them best.

Visual Disabilities

Visual disabilities include blindness, color blindness, and low vision. In addition to making your application accessible to assistive applications, such as screen readers, you should also consider the following:

- Although color can greatly enhance a user interface, make sure it is not the only source of information. A color blind user may not be able to distinguish between two objects that differ only in color.
- Provide an audio option to all visual cues and feedback. Your application should make it easy to replace visual communication with audio communication.
- Provide an option to present images and animated content in an alternate manner. If your application displays an image or animation, consider providing your own succinct descriptions of these elements so blind or low-vision users can benefit from the information they convey.

Hearing Disabilities

People with hearing disabilities may have difficulty distinguishing your application's sound effects from ambient noise or may not be able to hear them at all. Users without hearing disabilities may find themselves in circumstances in which audio output from an application is inappropriate (in a library, for example). Be sure to consider these points as you design the audio output of your application.

Your application should not override the audio-output settings the user selects in System Preferences. In addition, you should provide a visual option to all audio cues and feedback. Your application should make it easy to replace audio communication with visual communication. For example, a “beep” can be replaced or accompanied by a flash of the display screen.

Motor and Cognitive Disabilities

People with motor disabilities may need to use alternatives to the standard mouse and keyboard input devices. Other users may have difficulty with the fine motor control required to double-click a mouse or to press key combinations on the keyboard. Users with cognitive or learning disabilities may need extra time to complete tasks or respond to alerts.

For the most part, support for motor disabilities is provided at the hardware or operating system level. Mac OS X provides many such solutions in the Universal Access preferences. The Sticky Keys feature, for example, allows a user to type the keys in a key combination sequentially, instead of simultaneously. As an application developer, therefore, the most important thing you can do is to access-enable your application so your users can deploy the assistive technologies of their choice.

A feature such as Sticky Keys can also be helpful to a user with a cognitive or learning disability that makes it difficult to perform simultaneous tasks. An application that provides its output in both visual and auditory modes (especially simultaneously) can enhance comprehension. Users with such disabilities also benefit from the redundancy provided by an application that employs both audio and visual output.

In addition to making your application accessible, you should consider incorporating the following features:

- Provide options to adjust the length of expected response times. Users who have difficulty quickly responding to application events benefit from having extra time to respond. When a timed response is required—such as notification that a regularly scheduled action is about to take place—you should provide at least one response method that does not require users to respond within the timed interval. Alternatively, you should provide at least one method that allows users to adjust the response time to at least five times the default setting.
- Avoid using regularly blinking cursors or other objects on screen. The frequency of a blinking object must not be in the range of 2 hertz to 55 hertz, inclusive. Objects that blink in this frequency range can cause medical complications, such as seizures, in some people.

The Mac OS X Accessibility Protocol

In Mac OS X version 10.2, Apple introduced the accessibility framework. This framework includes:

- The accessibility protocol that both Carbon and Cocoa frameworks implement to allow applications to represent themselves to assistive applications and technologies
- APIs an assistive application uses to drive the user interface of another application running in Mac OS X

This chapter introduces the accessibility protocol. It describes:

- The model that represents accessible applications to assistive technologies
- The accessibility object that represents user interface objects
- Some of the ways an assistive application interacts with an accessible application

If you're an application developer, you should read this chapter to learn about the Mac OS X accessibility protocol. Then, if you're ready to access-enable your application, you should read *Accessibility Programming Guidelines for Cocoa* or *Accessibility Programming Guidelines for Carbon*.

Important: If your application uses only standard, noncustom Carbon or Cocoa objects, most of your application is already accessible. There remain a few things you must do, however. This chapter provides fundamental information about the accessibility protocol that helps you understand the reasons for these things.

Note: Java developers should implement the Java Accessibility API (the `javax.accessibility` package) to ensure their applications are accessible (both Swing and AWT interfaces are accessible). See for more information on this API, see the Java 1.4.2 API reference in the Java Reference Library.

If you are developing an assistive application, you should read this chapter to learn how accessible applications represent themselves in Mac OS X. You'll find out what information your assistive application can expect to get from an accessible application.

The Accessibility Model

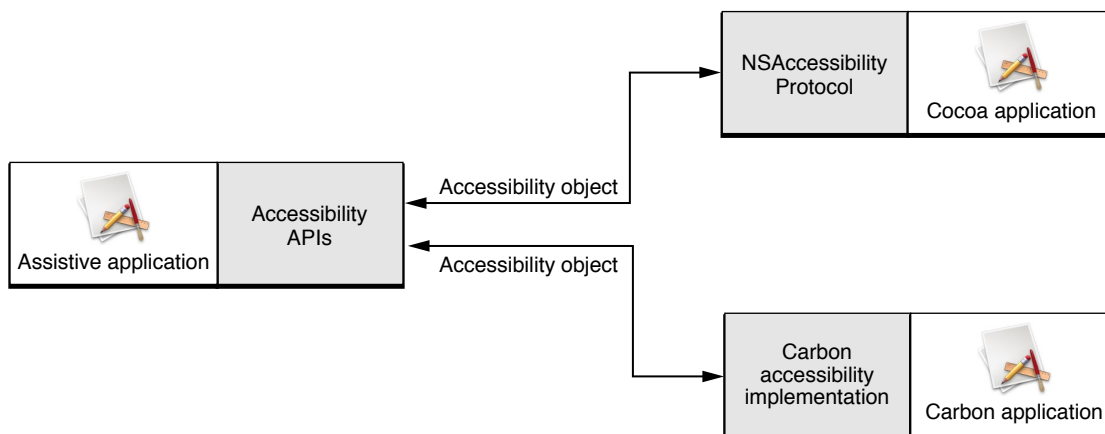
An assistive application helps a user interact with the applications on the user's computer. To do this, an assistive application must be able to access everything in an application's user interface and perform all the application's functions. To be accessible, therefore, an application must provide information about its user interface and capabilities in a standard manner that any assistive application or technology can understand.

This is a challenge because each application type has its own native way of representing its user interface. Cocoa applications, for example, use the `NSWindow` and `NSControl` classes to display windows and controls. A modern Carbon application, on the other hand, uses `HIObj` and `HIView` objects to implement its user interface. Other types of applications use other native constructs.

Apple solved this problem in Mac OS X version 10.2 with the introduction of a generic object, called an accessibility object. In an accessible application, a user interface element, such as a window, a control, and even the application itself is represented by an accessibility object. To learn more about the accessibility object and the information it provides, see [“The Accessibility Object”](#) (page 20).

Accessibility objects provide a uniform representation of an application’s user interface elements, regardless of the application framework on which the application depends. Figure 3-1 shows how an assistive application communicates with different types of applications using the accessibility objects the applications provide.

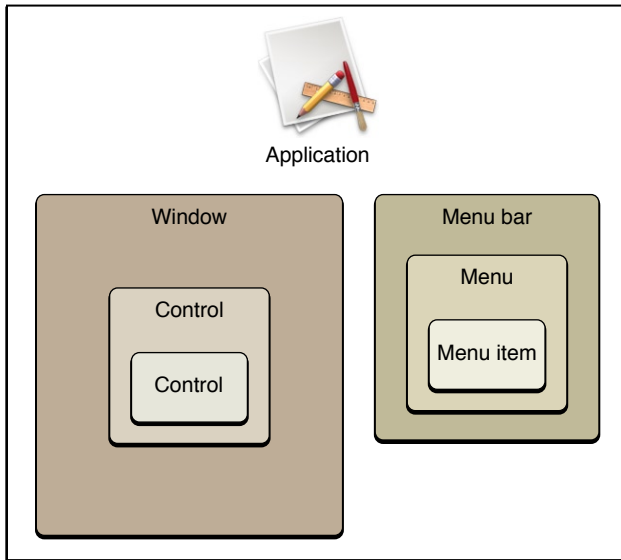
Figure 3-1 Communication between an assistive application and other applications



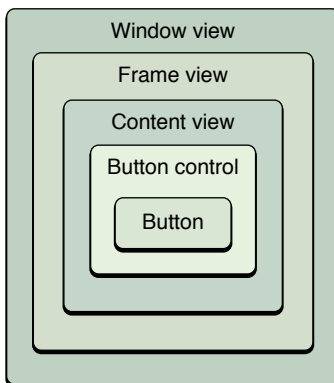
The Mac OS X accessibility model represents an application’s user interface as a hierarchy of accessibility objects. For the most part, the hierarchy is defined by the parent-child relationships among accessibility objects. For example, an application’s dialog window might contain several buttons. The accessibility object representing the dialog contains a list of child accessibility objects, each representing a button in the dialog. In turn, each accessibility object representing one of the buttons knows its parent is the accessibility object representing the dialog.

Of course, the accessibility objects representing the menu bar and windows in an application are children of the application-level accessibility object. Even the application-level accessibility object has a parent, which is the system-wide accessibility object. An application never needs to worry about its system-wide parent because it is out of the application’s scope. On the other hand, an assistive application might query the system-wide accessibility object to find out which running application has keyboard focus.

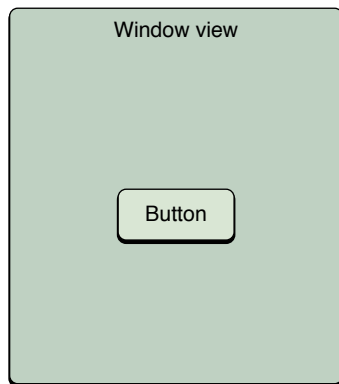
[Figure 3-2](#) (page 19) shows the hierarchy of accessibility objects in a simple application.

Figure 3-2 The accessibility object hierarchy

A strength of the accessibility object hierarchy is that it can leave out implementation-specific details that are irrelevant to an assistive application and, by extension, to the user. For example, in Cocoa, a button in a window is usually implemented as a button cell within a button control, which is in a content view within a window. A user has no interest in this detailed containment hierarchy; she only needs to know that there's a button in a window. If the application's accessibility hierarchy contains an accessibility object for each of these intermediate objects, however, an assistive application has no choice but to present them to the user. This results in a poor user experience because the user is forced to take several steps to get from the window to the button. Figure 3-3 shows how such an inheritance hierarchy might look.

Figure 3-3 Complete inheritance hierarchy of a button in a window

To exclude this unnecessary information, the accessibility protocol allows an application to specify some accessibility objects as ignored. Continuing the Cocoa button example, the application can designate as ignored the accessibility objects representing the button control and the content view. This allows an application to present to an assistive application only the significant objects in its accessibility hierarchy. [Figure 3-4](#) (page 20) shows how the same hierarchy shown in [Figure 3-3](#) (page 19) might be presented to an assistive application.

Figure 3-4 Appropriate accessibility hierarchy of a button in a window

An assistive application helps a user perform tasks by telling an application's accessibility objects to perform actions. For the most part, actions correspond to things a user can do with a single mouse click. Each accessibility object contains information about which actions it supports, if any. The accessibility object representing a button, for example, supports the press action. When a user wants to press a button, he communicates this to the assistive application. The assistive application determines that the button's accessibility object supports the press action and sends a request to the button to perform it.

The Mac OS X accessibility protocol defines only a handful of actions that accessibility objects can support. At first, this might seem restrictive, because applications can perform huge numbers of tasks. It's essential to remember, however, that an assistive application helps a user drive an application's user interface, it does not simulate the application's functions. Therefore, an assistive application has no interest in how your application responds to a button press, for example. Its job is to tell the user that the button exists and to tell the application when the user wants to press it. It's up to your application to respond appropriately to that button press, just as it would if a user used a mouse to click the button.

The Accessibility Object

An accessibility object provides to assistive applications information about the user-interface object it represents. This information includes the object's position in the accessibility hierarchy, its position on the display, details about what it is, and what actions it can perform. In addition, an accessibility object responds to messages sent by assistive applications and sends out notifications that describe changes in its state.

This section describes the accessibility object. It describes the information an accessibility object provides and the actions it can perform and outlines the communication between accessibility objects and assistive applications.

Note: Carbon and Cocoa implement the accessibility object in different native ways. This section describes the accessibility object in general terms that are not specific to either application framework. For framework-specific details about the implementation of the accessibility object and how to use it, see *Accessibility Programming Guidelines for Cocoa* and *Accessibility Programming Guidelines for Carbon*.

Attributes

An accessibility object has many attributes associated with it. The number and kind of attributes vary depending on the type of user interface object the accessibility object represents. A few attributes are required for every accessibility object, but most are optional.

Attributes have values that assistive applications use to find out about the user interface object. For example, an assistive application gets the value of an accessibility object's role attribute to find out what type of user interface object it represents.

Some attribute values are settable by assistive applications. An example is the value attribute in an accessibility object that represents an editable text field. When a user types in the text field, an assistive application sets the value of the value attribute to the text the user enters.

If you use standard, noncustom Cocoa or Carbon objects most of the attribute values are already in place. There are a few attribute values, however, that you must provide because they contain application-specific information, such as the description of a user interface object's function.

The `AXAttributeConstants.h` file in the `HServices` framework defines all the accessibility object attributes in the Mac OS X accessibility protocol. The following sections describe some of the most common required attributes, paying particular attention to the attributes for which you must provide values:

- [“The Role and Role Description Attributes”](#) (page 21)
- [“The Description Attribute”](#) (page 22)
- [“Title Attributes”](#) (page 22)
- [“Relationship Attributes”](#) (page 23)
- [“Value Attributes”](#) (page 23)

The Role and Role Description Attributes

An accessibility object's most important attribute is its **role** attribute. This is because the accessibility object's role determines which other attributes the object contains and tells an assistive application how to handle it. You can think of the role as the accessibility object's class—it defines a standard set of behaviors and capabilities to which the object conforms. For more information on the attributes associated with specific roles, see [“Roles and Associated Attributes”](#) (page 35).

The value of the role attribute is a nonlocalized string. An assistive application can programmatically test the value of the role attribute to find out what type of user interface object the accessibility object represents.

In `AXRoleConstants.h` (in the `HServices` framework), Mac OS X defines a set standard roles that describe the vast majority of user interface object types. Although it may be tempting to define new roles for custom objects in your application, it is not recommended. An assistive application may not know how to handle an

accessibility object with an arbitrary role and additional roles add unnecessary complexity to your code. Instead, you should examine the behavior of your objects and choose the standard role that best represents them.

The **role description** attribute contains a human-intelligible, localized string that names the accessibility object's role. An assistive application presents this string to the user (a screen reader application, for example, speaks the string). Mac OS X defines a role description string for each role in `AXRoleConstants.h`, so you do not have to provide strings such as "button" or "window". In the very unlikely event that your application needs to define a new role, however, you are responsible for providing the value of the role description attribute.

The Description Attribute

The **description** attribute is almost as important as the role attribute. The value of the description attribute is a human-intelligible, localized string that describes what the object does. Because it describes the application-specific function of a user interface object, the accessibility protocol cannot supply the description value. Therefore, it is essential to provide a description for all accessibility objects in your application that do not already include a title attribute (described in "[Title Attributes](#)" (page 22)).

To see why the description attribute is so important, suppose your application window contains a button to left-justify text and which displays a left-pointing arrow. An assistive application can accurately identify this control as a button because it is represented by an accessibility object with the role attribute "button". However, unless you provide an appropriate description, the assistive application has no way to know that this button left-justifies text, and therefore no way to communicate this to the user.

Title Attributes

The **title** attribute is required for user interface objects that include a text title in their display. For example, the title of a button is the text that appears on the button, such as the text "OK" on an OK button, and the title of a window is the text that appears in its title bar.

A user cannot change the title of such an object directly, but the title might change programmatically if the state of the object changes. For example, a Connect button's title might change to Disconnect after a connection is made, but not because a user chose to change the button's title. An accessibility object that represents such an object must include the title attribute and the attribute's value must be the title string.

Many applications display static text that serves as the title for a user interface object, but that is not contained in that object. An example is the word "Search" displayed below a search field or "Address:" displayed next to a set of editable text fields. To a sighted user, the proximity of the string to the object (or objects) it describes is usually enough to imply the relationship between them. To an assistive application, however, these strings are unrelated to the objects they describe (if they are visible to the assistive application at all).

Mac OS X version 10.4 introduced two related attributes that give assistive applications information about such titles. The **TitleUIElement** and **ServesAsTitleForUIElements** attributes allow you to define the relationship between a piece of static text and the object (or objects) it describes.

The **TitleUIElement** attribute belongs in the accessibility object representing the object being described. The value of this attribute is the accessibility object you create to represent the static text. The **ServesAsTitleForUIElements** attribute belongs in the static text accessibility object you've created and its value is an array containing an arbitrary number of accessibility objects. This allows you to link the static text title with any number of user interface objects. Although these attributes are not required, you should provide them if your user interface includes such static text titles.

Relationship Attributes

To participate in the accessibility hierarchy, an accessibility object must include links to its immediate ancestor and descendents (if any). This helps an assistive application traverse the hierarchy. In addition, an accessibility object can express other relationships, such as between views that affect each other, but that are not linked by containment.

All accessibility objects, with the exceptions of the application-level and system-wide accessibility objects, include a **parent** attribute. The value of this attribute is usually the accessibility object representing the closest accessible container of the user interface object.

If a user interface object contains other accessible user interface objects, the UIElement representing it must include the **children** attribute. The value of this attribute is an array containing the UIElements of the accessible descendents.

Some relationships are conceptual rather than containment-based. For example, it's not unusual for an application to display two separate views of the same or related content. One example is the Mac OS X Mail application. The Mail application's upper view can display the subject, sender, and receive date of each message in the selected mail box. In the lower view, Mail can display the body of a message selected in the upper view. To a sighted user, the relationship between the selected message in the upper view and the message content in the lower view is apparent. To an assistive application, on the other hand, this direct relationship doesn't exist. If an assistive application can't express such a relationship to a blind user, for example, the user can't jump back and forth between the related elements the way a sighted user can. Instead, such a user might have to step through all intervening controls and views to move between the message description and its content.

Mac OS X version 10.4 introduced the **LinkedUIElements** attribute to allow you to define such relationships. As you would expect, the UIElement of each related object should contain this attribute. The value of the attribute is an array of UIElements so you can specify one-to-one and one-to-many relationships.

Value Attributes

The optional **value** attribute describes the accessibility object's state. The value might be the state of a check box or the contents of an editable text field.

The value attribute is often settable. For example, an accessibility object that represents a user-modifiable object, such as an editable text field, has a settable value attribute. This allows an assistive application to set the value of the accessibility object's value attribute to contain the user's input. Optionally, accessibility objects may also include attributes that define a range or set of values the object can accept, such as minimum and maximum values for a slider.

Actions

Technically, an action is an attribute of an accessibility object. However, the accessibility protocol supports actions differently from the way it supports attributes, so this chapter describes them separately.

An accessibility object can support one or more actions. An action describes how a user interacts with the user interface object the accessibility object represents. It does not describe the application-defined function of the object. This is because an assistive application is concerned with driving the user interface and the results of an action are irrelevant to it. If your application displays a print button, for example, the button's accessibility object supports a press action, not a print action.

Because actions are generic and refer to the capabilities of user interface objects, there are only a few of them. This means that the set of actions an assistive application has to understand is small and well-defined.

In `AXActionConstants.h` (located in the `HServices` framework), Mac OS X defines seven actions an accessibility object can support:

- Press (a button)
- Increment (the value of a scroller or slider indicator)
- Decrement (the value of a scroller or slider indicator)
- Confirm (the selection of an object)
- Cancel (a selection or action)
- Raise (a window)
- ShowMenu (display a contextual menu associated with an object)

When a user performs an action, an assistive application sends a message to the accessibility object, requesting it to perform the action. Your application should invoke the same code to carry out this action as it does when the request comes directly from your user interface.

Each action attribute has a description property. An assistive application may speak this description to tell the user what action is available for a specific object. The value of this description property is similar to the value of the role description attribute in that it is a human-intelligible, localized word or short phrase. Unlike the role description, however, the action description is not automatically supplied by the accessibility protocol. If your application creates its own accessibility objects which support actions, you must supply the appropriate action descriptions.

Communication With Accessibility Objects

At the heart of accessibility is the communication between an assistive application and the accessibility objects that represent your application's user interface. This communication can be divided into two categories:

- Messages sent by an assistive application to get information about an accessibility object and to request the performance of actions. An accessibility object responds to these messages by returning attribute values, performing actions, or changing attribute values.
- Notifications triggered by accessibility objects that assistive applications can listen for. These notifications tell an interested assistive application about changes in the state of an accessibility object.

If you use only standard, noncustom Cocoa or Carbon objects in your application, most of this communication is transparent to you. In some cases, you might have to create a custom response to a message, but this is unlikely. It is even less likely that you will have to handle notifications if you use only standard objects.

Messages

An assistive application communicates with your application by sending messages to accessibility objects. In Carbon, these messages are transmitted as Carbon events. In Cocoa, these messages result in calls to methods of the NSAccessibility protocol. For more details about the framework-specific implementation of messages, see *Accessibility Programming Guidelines for Cocoa* and *Accessibility Programming Guidelines for Carbon*.

In `HIObject.h`, Mac OS X defines a handful of messages. The following lists the types of messages an assistive application can send to an accessibility object:

- Get a list of the accessibility object's attributes
- Get the value of a specific attribute
- Check to see if the value of a specific attribute can be set
- Set the value of a specific attribute
- Get a list of the accessibility object's actions
- Get the description of an action
- Tell the accessibility object to perform a specific action
- Determine which accessibility object is under the mouse pointer (hit testing)
- Register or unregister for notifications

Like the set of actions, the set of messages to which an accessibility object can respond is small. These messages give the assistive application a great deal of control, however. For example, by getting and setting attributes, an assistive application can do things like the following:

- Read the value of a slider control
- Traverse the object hierarchy to find all the accessibility objects (such as controls, embedded controls, and table cells) within a window
- Check to see if a control is enabled

Notifications

In addition to responding to messages from assistive applications, accessibility objects also broadcast any significant changes that occur in the user interface objects they represent. For example, if the keyboard focus changes to a new text field, a new window becomes active, or a control's title changes, the accessibility objects for these objects send out notifications.

An assistive application chooses to register for the notifications it is interested in.

An accessibility object can send notifications to indicate any of the following status changes:

- The object's value changed
- The object was destroyed
- The keyboard focus changed

Unless you are creating accessibility objects to represent custom user interface objects, it is unlikely you will have to write code to send notifications. Both Carbon and Cocoa automatically broadcast the appropriate notifications for standard objects.

Hit-Testing and Keyboard Focus

To a sighted user, the location of the cursor is easy to discern. Similarly, a sighted user can usually tell which object in the user interface has keyboard focus. An assistive application, on the other hand, must query an application to determine which object has keyboard focus or is under the mouse pointer. An accessibility object provides answers to these queries by returning the values of various attributes.

Although you implement hit-testing differently depending on whether you're using Carbon or Cocoa, the basic procedure is for an assistive application to ask the application to return the accessibility object under the cursor. The request is recursively passed down the application's accessibility hierarchy until it reaches the deepest, unignored accessibility object that contains the mouse pointer.

Accessibility objects also must support queries regarding keyboard focus. An accessibility object stores focus information in its focused attribute. The initial query from the assistive application is for the focused attribute of the application-level accessibility object. This query, too, is passed down the application's accessibility hierarchy until it reaches the deepest, unignored accessibility object whose focused attribute is `true`. As with hit-testing, how the keyboard focus is actually determined from lower-level objects varies depending on whether you use Carbon or Cocoa.

An Example of Accessibility

This example gives a detailed description of how a fictitious screen reader with speech recognition and speech synthesis capability might communicate with your application:

1. The user says, "Open Preferences window."
2. The screen reader sends a message to the application accessibility object, asking for its menu bar attribute, which is a reference to the menu bar accessibility object. It then queries the menu bar for a list of its children, and queries each child for its title attribute until it finds the one whose title is the application's name (that is, the application menu). A second iteration lets it find the Preferences menu item within the application menu. Finally, the screen reader tells the Preferences menu item to perform the press action.
3. The application opens the Preferences window and then the window sends a notification broadcasting that a new window is now visible and active.
4. The screen reader, assuming that it registered to be notified when a new window opens, queries the window for a list of its attributes. Assuming that the window accessibility object contains a children attribute, it then queries the accessibility object for the value of its children attribute.
5. To each child of the window accessibility object, the screen reader sends a query asking for a list of its attributes. It then queries the child for the values of its role, role description, and (if it exists) children attributes.
6. Among the responses, the screen reader learns that the pane contains several children (for example, three checkboxes).

7. The screen reader queries each checkbox, asking for the values of the following attributes:
 - role (checkbox in this case)
 - role description (“checkbox”)
 - value (checked or unchecked)
 - children (none in this case)
8. The screen reader, having learned what objects (controls in this case) are accessible in the window, reports this information to the user using speech synthesis.
9. The user might then ask for more information about one of the checkboxes.
10. The screen reader queries the specified checkbox, asking for the value of its help attribute (assuming it exists). It reports this string to the user using speech synthesis.
11. The user then tells the screen reader to check the checkbox.
12. The screen reader sends a message requesting that the checkbox’s value attribute be set to 1.
13. The checkbox accessibility object broadcasts that the value of its value attribute has changed.

Testing for Accessibility

Whether you're designing a new application or access enabling an existing one, you should plan to test the accessibility of your product. Testing for accessibility is a bit different than standard user-interface testing and Apple provides a couple of tools that make the process easier.

Application developers should read this chapter to find out how to exercise the accessibility of their applications.

Using Accessibility Inspector

In Mac OS X v10.4 and later, Apple provides the Accessibility Inspector testing tool in `/Developer/Applications/Utilities/Accessibility Tools`. You can download the Developer Tools from the Apple Developer Connection website (developer.apple.com). Accessibility Inspector presents a utility window that displays the attributes (and values), actions, and position in the accessibility hierarchy of the object currently under the mouse pointer. To use Accessibility Inspector, be sure to enable assistive applications in the Universal Access Preferences.

If you're beginning to access-enable your application, try using Accessibility Inspector to view the accessibility information other applications provide. Although Accessibility Inspector is not an assistive application, it uses the same APIs assistive applications use to get information from the accessibility objects it encounters.

You can focus Accessibility Inspector on a specific object to examine its attributes, perform its actions, and access its parent and children (if any), by pressing Command-F7. When you do this, the Accessibility Inspector display freezes, allowing you to move the mouse without changing the object on which the tool is focused. A second utility window then appears that displays information about the selected object. In this second window, you can go to the object's parent, children, or other related objects, such as the containing window or the top-level accessibility object. You can also perform the actions supported by this accessibility object, allowing you to see how this affects the values of various attributes and the application itself.

As you access-enable your application, use Accessibility Inspector to make sure your objects contain the appropriate information. If you find that a specific object is not accessible, you can focus Accessibility Inspector on that object and examine its attribute values and perform its actions to find the problem.

Using Accessibility Verifier

In Mac OS X v10.4 and later, Apple provides the Accessibility Verifier tool in `/Developer/Applications/Utilities/Accessibility Tools`. Accessibility Verifier displays the accessibility hierarchy comprising all currently instantiated objects in the selected application. To use Accessibility Verifier, be sure to enable assistive applications in the Universal Access Preferences.

Use Accessibility Verifier to perform any or all of the following tests (select the tests you want to run by clicking the Choose Tests... button):

- **Parent/Child.** This test checks the integrity of the accessibility hierarchy by making sure each parent-child pair forms a closed loop. For example, if a child listed in a parent object's children attribute does not refer to that object as its parent, this parent-child pair is invalid. Invalid parent-child pairs can prevent an assistive application from correctly traversing an application's accessibility hierarchy.
- **Window.** This test checks that all objects contained in a window include attributes that refer to the containing window. An object contained in a window is not necessarily the child of that window, but it should refer to its containing window as a convenience for assistive applications.
- **Missing AXDescription.** This test checks for the presence of the description attribute in accessibility objects that require one. Recall from [“The Description Attribute”](#) (page 22) that all accessibility objects must provide some context-specific descriptive information to assistive applications. If an object does not display a descriptive title, it should include the description attribute and an appropriate value.
- **Role Verification.** This test verifies that an accessibility object's attributes are appropriate for its role. See [“Roles and Associated Attributes”](#) (page 35) for more information on the attributes that are associated with each role.

When you click **Verify**, Accessibility Verifier runs the tests you selected and displays the results for each. The problems are reported as warnings or errors, depending on their severity (you can filter the results by selecting a severity level in the Report Level pop-up menu).

It's important to note that eliminating all errors and warnings Accessibility Verifier displays does not guarantee a perfectly accessible application. You should always test your application with various assistive applications to make sure there are no problems.

Accessibility Keyboard Shortcuts

This appendix lists keyboard shortcuts that are reserved for use with various Universal Access features in Mac OS X.

To turn VoiceOver on or off, use Command-F5.

Table A-1 lists the keyboard shortcuts that are reserved for use with screen zooming in Mac OS X. A user turns on the zoom feature in the Seeing pane of Universal Access preferences.

Table A-1 Key combinations used with screen zooming

Key combination	Action
Option-Command-8	Turns screen zooming on or off
Option-Command=	Zooms in
Option-Command-- (hyphen)	Zooms out
Command-Option-Control-8	Inverts the screen colors
Command-Option-Control-,	Reduces contrast
Command-Option-Control-.	Increases contrast

Mac OS X version 10.2 and later provides the option of **full keyboard access mode**, in which users can navigate through windows, dialogs, menus, toolbars, and the Dock using the keyboard alone, without a mouse or other pointing device.

Users can turn on full keyboard access in the Keyboard Shortcuts pane of Keyboard & Mouse preferences. Control-F1 is a reserved keyboard shortcut for turning full keyboard access on or off. Control-F7 toggles between keyboard access for all controls in windows and dialogs and the default state, in which only text fields and scrolling lists are accessed with the keyboard. Don't use these combinations for any other purpose.

With keyboard access turned on in windows and dialogs, the arrow keys move between values within a control. For example, if the user selects a slider with the Tab key, the arrow keys move the slider control along the slider track. For vertically oriented choices, such as menu items, the Up Arrow and Down Arrow keys move the selection. For horizontally oriented choices, such as a row of tabs, the Right Arrow and Left Arrow keys move the selection. In some cases, it makes sense to support both orientations. For example, a vertical slider could use both the Up Arrow and the Right Arrow to increase the value.

In some cases, such as radio buttons, moving the focus to an item selects it as well. In other cases, such as push buttons, the user activates a selected item by pressing the Space bar. In full keyboard access mode, pressing the Space bar is equivalent to clicking the mouse button.

The Esc (Escape) key is used to cancel a dialog and to cancel a selection in a pop-up menu or list. In a Dock menu, Esc dismisses the menu and moves focus to the foremost window.

The user can also quickly place focus in the menu bar, the Dock, toolbars, and utility windows using key combinations described in Table A-2.

This behavior is provided for all applications that use the standard controls. If you are implementing your own controls, you need to provide these behaviors for your users.

Table A-2 lists the key combinations used in full keyboard access mode.

Table A-2 Key combinations for moving focus in full keyboard access mode

Key combination	Action
Control-F1	Turns full keyboard access on or off
Control-F2	Moves focus to the menu bar
Control-F3	Moves focus to the Dock
Control-F4	Moves focus to the active (or next) window
Shift-Control-F4	Moves focus to the previous window
Control-F5	Moves focus to the toolbar
Control-F6	Moves focus to the first (or next) utility window
Shift-Control-F6	Moves focus to the previous utility window
Control-F7	Toggles the keyboard access mode in windows and dialogs between all controls and just text fields and scrolling lists
Control-Tab	Moves focus to the next grouping of controls in a dialog or the next table (when Tab moves to next cell)
Shift-Control-Tab	Moves focus to the previous grouping of controls
Command-Tab	Moves focus to the first (or next) open application's Dock icon
Command-Shift-Tab	Moves focus to the previous open application's Dock icon
Arrow key	Moves focus to the next or previous value in a text field or certain controls, such as menus; also opens Dock menus
Control-arrow key	Moves focus to another value or cell within a control such as a table
Command-` (grave accent)	Activates the next open window in the frontmost application
Command-Shift-` (grave accent)	Activates the previous open window in the frontmost application
Command-Option-` (grave accent)	Moves focus to the window drawer
Space bar	Selects the highlighted control (equivalent to clicking the mouse button)
Return (Enter)	Selects the default button

Key combination	Action
Esc	Cancels a dialog or a selection in a pop-up menu or list; in a Dock menu, Esc closes the menu and moves the focus to the frontmost window

Note: Users can change the default combinations listed above in the Keyboard Shortcuts pane of Keyboard & Mouse preferences or in the Keyboard pane of Universal Access preferences. When full keyboard access is on, user-defined combinations override any combinations used in applications.

Roles and Associated Attributes

This appendix lists many of the roles Mac OS X defines, along with the attributes an object of that role should support. If you must create accessibility objects to represent custom user interface objects in your application, you can use this information to ensure you support the appropriate set of attributes. If you are an assistive application developer, you can use this information to learn what values you can get from objects of different roles.

Because different application frameworks have different ways of implementing the accessibility object, the type of an attribute's value also differs according to the application framework. For example, a string value in the Carbon accessibility implementation is usually an object of type `CFStringRef`, whereas in Cocoa, a string is implemented as an `NSString`. The value types listed in the tables in this appendix are therefore generic, such as string or Boolean. The exception to this is when the value type is an accessibility object. In these cases, in the interests of space, the tables list `UIElement` instead of `Accessibility Object`.

The application frameworks also provide different constant names for attributes, roles, actions, and notifications. In Carbon, for example, the radio-button role is defined as the constant `kAXRadioButtonRole` and the `CFString` `AXRadioButton`. In Cocoa, the same role is defined as the constant `NSString` `NSAccessibilityRadioButtonRole`. This document, therefore, uses the base of the name, which is common to both frameworks, and displays it as a title. To refer to the radio-button role, for example, this appendix uses the phrase `Radio Button`. Attribute names are listed in a similar manner.

The application frameworks also differ somewhat in how they define which attributes are required for each role. In general, Cocoa requires all of a role's required attributes to be present whether or not the value of a particular attribute is `NULL` in a specific instance of that role. Carbon, on the other hand, sometimes allows a specific instance of a role to leave out a normally required attribute if the value of the attribute would be `NULL`. Be sure to check the application framework-specific documentation to learn whether you can leave out an attribute whose value is `NULL`.

Note: All accessibility objects include the `Role` and `Role Description` attributes. To conserve space, these attributes are not listed in the role-specific sections below. For more information on these attributes, see [“The Role and Role Description Attributes”](#) (page 21).

Application Role

An accessibility object of the application role includes the following attributes as shown in Table B-1:

Table B-1 Attributes associated with the application role

Attribute	Required	Type	Description
Frontmost	Yes	Boolean	<code>true</code> if the application is active; <code>false</code> otherwise
Hidden	Yes	Boolean	<code>true</code> if the application is hidden; <code>false</code> otherwise

Attribute	Required	Type	Description
Main Window	Yes	UIElement	Accessibility object representing the main window
Focused Window	Yes	UIElement	Accessibility object representing the key window
Title	Yes	String	Visible title of the application's main window
Menu Bar	Yes	UIElement	Accessibility object representing the application's menu bar
Windows	Yes	UIElement array	Accessibility objects representing the application's open windows
Focused UIElement	Yes	UIElement	Accessibility object representing the object that currently has keyboard focus
Children	Yes	UIElement array	Accessibility objects representing the application's children

Browser Role

An accessibility object of the browser role includes the following attributes as shown in Table B-2:

Table B-2 Attributes associated with the browser role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the browser's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the browser's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the browser's containing window, sheet, or drawer
Columns	Yes	UIElement array	Accessibility objects representing the browser's columns
Help	No	String	Help-tag text for the browser
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the browser; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the browser is focused; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the columns, titles, and scroll area

Attribute	Required	Type	Description
Header	Yes	UIElement	Accessibility object representing the browser's headers
Column Titles	Yes	UIElement array	Accessibility objects representing the titles of the columns
Visible Columns	Yes	UIElement array	Accessibility objects representing currently visible columns
Selected Columns	Yes	UIElement array	Accessibility objects representing currently selected columns
Horizontal Scroll Bar	No	UIElement	Accessibility object representing the horizontal scroll bar (if one exists)
Vertical Scroll Bar	No	UIElement	Accessibility object representing the vertical scroll bar (if one exists)

Busy Indicator Role

An accessibility object of the busy indicator role can represent the indeterminate progress bar or the asynchronous progress indicator (a determinate progress bar is represented by an object of the progress indicator role, discussed in “[Progress Indicator Role](#)” (page 53)). This object includes the following attributes as shown in Table B-3:

Table B-3 Attributes associated with the busy indicator role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the busy indicator's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the busy indicator's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the busy indicator's containing window, sheet, or drawer
Help	No	String	Help-tag text for the busy indicator
Focused	Yes	Boolean	<code>true</code> if the busy indicator is focused; <code>false</code> otherwise
Value	Yes	Boolean	<code>true</code> if the busy indicator is displaying busyness; <code>false</code> otherwise

Button Role

An accessibility object of the button role includes the following attributes as shown in Table B-4:

Table B-4 Attributes associated with the button role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the button's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the button's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the button's containing window, sheet, or drawer
Help	No	String	Help-tag text for the button
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the button; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the button is focused; <code>false</code> otherwise
Title	Yes	String	Visible title of the button. Not required if the button does not display a title in its visible interface.
Subrole	No	String	Type of the button. Required if the button is a close, zoom, minimize, toolbar, or sort button.
Edited	No	Boolean	<code>true</code> if the contents of the window have been edited; <code>false</code> otherwise. Required only if the button is a close button.
Description	No	String	Human-intelligible description of the button. Required if the button does not display a title.

An accessibility object of the button role supports the following action:

- **Press.** When the press action is performed, the button is selected.

An accessibility object of the button role can have one of the following subroles:

- **Close Button.** The button represents the close button in a window.
- **Minimize Button.** The button represents the minimize button in a window.
- **Zoom Button.** The button represents the zoom button in a window.
- **Toolbar Button.** The button represents the button to hide and reveal the toolbar in a window.
- **Increment Arrow.** The button represents the increment scroll arrow in a scroll bar object.

- **Decrement Arrow.** The button represents the decrement scroll arrow in a scroll bar object.
- **Increment Page.** The button represents the increment-page area of the scroll track in a scroll bar object.
- **Decrement Page.** The button represents the decrement-page area of the scroll track in a scroll bar object.
- **Sort Button.** The button represents a sort button.

Checkbox Role

An accessibility object of the checkbox role includes the following attributes as shown in Table B-5:

Table B-5 Attributes associated with the checkbox role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the checkbox's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the checkbox's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the checkbox's containing window, sheet, or drawer
Help	No	String	Help-tag text for the checkbox
Enabled	Yes	Boolean	true if the user can interact with the checkbox; false if interaction is disabled
Focused	Yes	Boolean	true if the checkbox is focused; false otherwise
Title	Yes	String	Visible title of the checkbox. Not required if the checkbox does not display a title in its visible interface.
Value	Yes	Number	Value of the checkbox (checked is 1, unchecked is 0, and mixed state is 2)
Description	No	String	Human-intelligible description of the checkbox. Required if the checkbox does not display a title.

An accessibility object of the checkbox role supports the following action:

- **Press.** When the press action is performed, the checkbox is clicked.

Color Well Role

An accessibility object of the color well role includes the following attributes as shown in Table B-6:

Table B-6 Attributes associated with the color well role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the color well's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Help	No	String	Help-tag text for the color well
Window	Yes	UIElement	Accessibility object representing the color well's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the color well's containing window, sheet, or drawer
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the color well; <code>false</code> otherwise
Value	Yes	String	RGB value of currently displayed color

An accessibility object of the color well role can support the following action:

- **Press.** When the press action is performed, the Color Picker window is displayed.

Combo Box Role

An accessibility object of the combo box role includes the following attributes as shown in Table B-7:

Table B-7 Attributes associated with the combo box role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the combo box's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the combo box's containing window

Attribute	Required	Type	Description
Top-Level UIElement	Yes	UIElement	Accessibility object representing the combo box's containing window, sheet, or drawer
Help	No	String	Help-tag text for the combo box
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the combo box; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the combo box is focused; <code>false</code> otherwise
Value	Yes	String	Text of the currently selected item
Children	Yes	UIElement array	Accessibility objects representing the disclosure button and (when expanded) the scroll area
Expanded	Yes	Boolean	<code>true</code> when the list is displayed; <code>false</code> otherwise
Selected Text	No	String	Selected portion of the text of the currently selected item
Selected Text Range	No	Range value	Position and length (in characters) of the selected portion of the text of the currently selected item
Number of Characters	No	Number	Number of characters in the selected portion of the text of the currently selected item
Visible Character Range	No	Range value	Position and length (in characters) of the text of the currently selected item
Description	Yes	String	Human-intelligible description of the combo box. Not required if the title UIElement attribute is included.
Title UIElement	No	UIElement	Accessibility object representing the static text that serves as the title for the combo box
Insertion Point Line Number	No	Number	Line number of current cursor position in combo box

Note: Combo boxes also support parameterized attributes. See the framework-specific documentation for more information on these attributes.

An accessibility object of the combo box role supports the following action:

- **Confirm.** When the confirm action is performed, the selected list item is displayed in the combo box's text field.

Disclosure Triangle Role

An accessibility object of the disclosure triangle role includes the following attributes as shown in Table B-8:

Table B-8 Attributes associated with the disclosure triangle role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the disclosure triangle's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the disclosure triangle's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the disclosure triangle's containing window, sheet, or drawer
Help	No	String	Help-tag text for the disclosure triangle
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the disclosure triangle; <code>false</code> otherwise
Value	Yes	Number	1 if disclosure triangle is open; 0 if it is closed

An accessibility object of the disclosure triangle role supports the following action:

- **Press.** When the press action is performed, the disclosure triangle changes state (from open to closed or from closed to open).

Dock Item Role

An accessibility object of the Dock item role includes the following attributes as shown in Table B-9:

Table B-9 Attributes associated with the Dock item role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the Dock item's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Help	No	String	Help-tag text for the Dock item

Attribute	Required	Type	Description
Subrole	Yes	String	Type of item represented by the Dock item
Title	Yes	String	Title displayed when the mouse cursor hovers over the Dock item
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the color well; <code>false</code> otherwise
Top-Level UIElement	Yes	UIElement	Accessibility object representing the Dock item's containing object
Selected	Yes	Boolean	<code>true</code> if the Dock item is visible; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility object representing the Dock item's Dock menu
Shown Menu UIElement	Yes	UIElement array	Accessibility object representing the Dock item's Dock menu (NULL if Dock menu is hidden)
URL	Yes	String	File path or URL of item represented by the Dock item
Is Application Running	Yes	Boolean	<code>true</code> if the application represented by the Dock item is currently running; <code>false</code> otherwise

An accessibility object of the Dock item role can support the following actions:

- **Press.** When the press action is performed, the item represented by the Dock item is activated.
- **Show Menu.** When the show menu action is performed, the Dock item's Dock menu is displayed.

An accessibility object of the Dock item role should have one of the following subroles:

- **Application Dock Item.** The Dock item represents an application.
- **URL Dock Item.** The Dock item represents a website.
- **Minimized Window Dock Item.** The Dock item represents a minimized window.
- **Trash Dock Item.** The Dock item represents the Trash.
- **Document Dock Item.** The Dock item represents a document.
- **Folder Dock Item.** The Dock item represents a folder.
- **Dock Extra Dock Item.** The Dock item represents a Dock Extra.

Drawer Role

An accessibility object of the drawer role includes the following attributes as shown in Table B-10:

Table B-10 Attributes associated with the drawer role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the drawer's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Focused	Yes	Boolean	<code>true</code> if the drawer has keyboard focus; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the drawer's children
Window	Yes	UIElement	Accessibility object representing the drawer's containing window
Description	Yes	String	Human-intelligible description of the drawer (required because a drawer doesn't usually display a title)

An accessibility object of the drawer role supports the following action:

- **Raise.** When the raise action is performed, this drawer becomes frontmost.

Group Role

An accessibility object of the group role includes the following attributes as shown in Table B-11:

Table B-11 Attributes associated with the group role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the group's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the group's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the group's containing window, sheet, or drawer
Title	No	String	Visible title of the group
Help	No	String	Help-tag text for the group
Focused	Yes	Boolean	<code>true</code> if the group is focused; <code>false</code> otherwise

Attribute	Required	Type	Description
Children	Yes	UIElement array	Accessibility objects representing the objects in the group
Contents	No	UIElement array	Accessibility objects representing the children of the group, excluding ancillary elements such as the title of a box or the scroll bars of a scroll view

Grow Area Role

An accessibility object of the grow area role includes the following attributes as shown in Table B-12:

Table B-12 Attributes associated with the grow area role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the grow area's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the grow area's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the grow area's containing window, sheet, or drawer

Image Role

An accessibility object of the image role includes the following attributes as shown in Table B-13:

Table B-13 Attributes associated with the image role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the image's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the image's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the image's containing window, sheet, or drawer

Attribute	Required	Type	Description
Help	No	String	Help-tag text for the image
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the image; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the image is focused; <code>false</code> otherwise
Description	Yes	String	Human-intelligible description of the image
Title	No	String	Required if the image displays a visible title

Incrementor Role

An accessibility object of the incrementor role represents the stepper control (the little arrows). This object includes the following attributes as shown in Table B-14:

Table B-14 Attributes associated with the incrementor role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the incrementor's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the incrementor's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the incrementor's containing window, sheet, or drawer
Help	No	String	Help-tag text for the incrementor
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the incrementor; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the incrementor is focused; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the up and down arrows in the stepper control
Increment Button	Yes	UIElement	Accessibility object representing the up arrow
Decrement Button	Yes	UIElement	Accessibility object representing the down arrow

An accessibility object of the incrementor role supports the following actions:

- **Increment.** When the increment action is performed, the value in the text field or other view associated with the stepper control increments to the next allowed value.
- **Decrement.** When the decrement action is performed, the value in the text field or other view associated with the stepper control decrements to the next allowed value.

List Role

An accessibility object of the list role includes the following attributes as shown in Table B-15:

Table B-15 Attributes associated with the list role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the list's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the list's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the list's containing window, sheet, or drawer
Help	No	String	Help-tag text for the list
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the list; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the list is focused; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the members of the list
Visible Children	Yes	UIElement array	Accessibility objects representing the currently visible members of the list
Selected Children	Yes	UIElement array	Accessibility objects representing the currently selected members of the list
Orientation	Yes	String	Horizontal or vertical orientation of the list; see framework-specific reference documentation for string constant definitions

Menu Role

An accessibility object of the menu role includes the following attributes as shown in Table B-16:

Table B-16 Attributes associated with the menu role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the menu's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Help	No	String	Help-tag text for the menu
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the menu; <code>false</code> otherwise
Title UIElement	Yes	UIElement	Accessibility object representing the object that serves as title of the menu (such as a menu bar item or menu item)
Children	Yes	UIElement array	Accessibility object representing the menu items the menu displays
Selected Children	Yes	UIElement array	Accessibility objects representing currently selected menu items in the menu
Visible Children	Yes	UIElement array	Accessibility objects representing currently visible menu items in the menu

An accessibility object of the menu role can support the following actions:

- **Cancel.** When the confirm action is performed, menu tracking is canceled.
- **Press.** When the press action is performed, the selected menu item in the menu is chosen.

Menu Bar Item Role

An accessibility object of the menu bar item role includes the following attributes as shown in Table B-17:

Table B-17 Attributes associated with the menu bar item role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the menu bar item's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Help	No	String	Help-tag text for the menu bar item

Attribute	Required	Type	Description
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the menu bar item; <code>false</code> otherwise
Title	Yes	String	Text title of the menu bar item
Children	Yes	UIElement array	Accessibility object representing the menu the menu bar item displays
Selected Children	Yes	UIElement array	Accessibility objects representing currently selected menu items in the menu the menu bar item displays
Visible Children	Yes	UIElement array	Accessibility objects representing currently visible menu items in the menu the menu bar item displays
Serves as Title for UIElements	No	UIElement array	The accessibility object representing the menu that is the child of this object

An accessibility object of the menu bar item role can support the following actions:

- **Cancel.** When the confirm action is performed, menu tracking is canceled.
- **Press.** When the press action is performed, the selected menu item in the menu bar item's menu is chosen.

Menu Bar Role

An accessibility object of the menu bar role includes the following attributes as shown in Table B-18:

Table B-18 Attributes associated with the menu bar role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the menu bar's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Help	No	String	Help-tag text for the menu bar
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the menu bar; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the menu bar items in the menu bar
Selected Children	Yes	UIElement array	Accessibility objects representing currently selected menu bar items

Attribute	Required	Type	Description
Visible Children	Yes	UIElement array	Accessibility objects representing currently visible menu bar items

An accessibility object of the menu bar role can support the following action:

- **Cancel.** When the confirm action is performed, menu tracking is canceled.

Menu Button Role

An accessibility object of the menu button role is a command pop-down menu or an icon or bevel button with a pull-down menu. It includes the following attributes as shown in Table B-19:

Table B-19 Attributes associated with the menu button role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the menu button's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the menu button's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the menu button's containing window, sheet, or drawer
Help	No	String	Help-tag text for the menu button
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the menu button; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the menu button is focused; <code>false</code> otherwise
Title	Yes	String	Text of the menu button's title. Not required if the button does not display a title.
Children	Yes	UIElement array	Accessibility object representing the menu
Description	No	String	Human-intelligible description of the button's function. Required if the button does not display a title.

An accessibility object of the menu button role supports the following actions:

- **Press.** When the press action is performed, the menu button's menu items are available for selection.
- **Show Menu.** When the show menu action is performed, the menu button's menu items are revealed.

Menu Item Role

An accessibility object of the menu item role includes the following attributes as shown in Table B-20:

Table B-20 Attributes associated with the menu item role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the menu item's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Help	No	String	Help-tag text for the menu item
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the menu item; <code>false</code> otherwise
Title	Yes	String	Accessibility object representing the title of the menu item
Selected	Yes	Boolean	<code>true</code> if the menu item is selected; <code>false</code> otherwise
Menu Item Command Character	No	String	Primary key in the keyboard shortcut for the command this menu item represents. Required if applicable.
Menu Item Command Virtual Key	No	String	Key code associated with the physical key in the keyboard shortcut for the command this menu item represents. Required if applicable.
Menu Item Command Glyph	No	String	Glyph displayed for a physical key in the keyboard shortcut for the command this menu item represents. Required if applicable.
Menu Item Command Modifiers	No	String	Integer mask representing the modifier keys held down in the keyboard shortcut for the command this menu item represents. Required if applicable.
Menu Item Mark Character	No	String	Symbol displayed to the left of this menu item. Required if applicable.
Menu Item Primary UIElement	No	UIElement	Accessibility object representing the object that displays the menu in which this menu item is contained
Serves as Title for UIElements	No	UIElement array	The accessibility object representing the menu that is the child of this object

An accessibility object of the menu item role can support the following actions:

- **Cancel.** When the confirm action is performed, menu tracking is canceled.
- **Press.** When the press action is performed, this menu item is chosen, if selected.

Outline Role

An accessibility object of the outline role includes the following attributes as shown in Table B-21:

Table B-21 Attributes associated with the outline role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the outline's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the outline's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the outline's containing window, sheet, or drawer
Columns	Yes	UIElement array	Accessibility objects representing the outline's columns
Rows	Yes	UIElement array	Accessibility objects representing the outline's rows
Help	No	String	Help-tag text for the outline
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the outline; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the outline is focused; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the columns, rows, and group
Header	Yes	UIElement	Accessibility object representing the outline headers
Visible Columns	Yes	UIElement array	Accessibility objects representing currently visible columns
Selected Columns	Yes	UIElement array	Accessibility objects representing currently selected columns
Visible Rows	Yes	UIElement array	Accessibility objects representing currently visible rows
Selected Rows	Yes	UIElement array	Accessibility objects representing currently selected rows

Pop-Up Button Role

An accessibility object of the pop-up button role is a pop-up menu. It includes the following attributes as shown in Table B-22:

Table B-22 Attributes associated with the pop-up button role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the pop-up menu's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the pop-up menu's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the pop-up menu's containing window, sheet, or drawer
Help	No	String	Help-tag text for the pop-up menu
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the pop-up menu; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the pop-up menu is focused; <code>false</code> otherwise
Value	Yes	String	Text of the pop-up menu item currently selected
Children	Yes	UIElement array	Accessibility object representing the menu

An accessibility object of the pop-up button role supports the following actions:

- **Press.** When the press action is performed, the pop-up menu's menu items are available for selection.
- **Show Menu.** When the show menu action is performed, the pop-up menu's menu items are revealed.

Progress Indicator Role

An accessibility object of the progress indicator role represents the determinate progress bar (an indeterminate progress bar or asynchronous progress indicator is represented by an object of the busy indicator role, discussed in “[Busy Indicator Role](#)” (page 37)). This object includes the following attributes as shown in Table B-23:

Table B-23 Attributes associated with the progress indicator role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the progress indicator's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the progress indicator's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the progress indicator's containing window, sheet, or drawer
Help	No	String	Help-tag text for the progress indicator
Focused	Yes	Boolean	<code>true</code> if the progress indicator is focused; <code>false</code> otherwise
Value	Yes	Number	The current numerical value associated with the fill of the progress bar
Min Value	Yes	Number	The smallest numerical value that can be associated with the fill of the progress bar (typically 0)
Max Value	Yes	Number	The largest numerical value that can be associated with the fill of the progress bar (typically 1)

Radio Button Role

An accessibility object of the radio button role includes the following attributes as shown in Table B-24:

Table B-24 Attributes associated with the radio button role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the radio button's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the radio button's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the radio button's containing window, sheet, or drawer
Help	No	String	Help-tag text for the radio button

Attribute	Required	Type	Description
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the radio button; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the radio button is focused; <code>false</code> otherwise
Title	Yes	String	Visible title of the radio button. Not required if the radio button does not display a title in its visible interface.
Value	Yes	Number	Value of the radio button (on is 1, off is 0)
Description	No	String	Human-intelligible description of the radio button. Required if the radio button does not display a title.

An accessibility object of the button role supports the following action:

- **Press.** When the press action is performed, the button is clicked.

Radio Group Role

An accessibility object of the radio group role includes the following attributes as shown in Table B-25:

Table B-25 Attributes associated with the radio group role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the radio group's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the radio group's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the radio group's containing window, sheet, or drawer
Help	No	String	Help-tag text for the radio group
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the radio group; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the radio group is focused; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the members of the radio group

Attribute	Required	Type	Description
Value	Yes	UIElement or UIElement array	Accessibility object representing the currently selected member of the radio group (see Note below for more information)
Visible Children	Yes	UIElement array	Accessibility objects representing the currently visible members of the radio group

Note: If more than one member of the radio group is selected, the value attribute contains an array of accessibility objects representing all selected members. If no member is selected, the value attribute contains NULL.

Relevance Indicator Role

An accessibility object of the relevance indicator role includes the following attributes as shown in Table B-26:

Table B-26 Attributes associated with the relevance indicator role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the relevance indicator's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the relevance indicator's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the relevance indicator's containing window, sheet, or drawer
Help	No	String	Help-tag text for the relevance indicator
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the relevance indicator; <code>false</code> otherwise
Value	Yes	Number	The current numerical value associated with the relevance displayed
Min Value	Yes	Number	The smallest numerical value that can be associated with the relevance displayed
Max Value	Yes	Number	The largest numerical value that can be associated with the relevance displayed

Row Role

An accessibility object of the row role includes the following attributes as shown in Table B-27:

Table B-27 Attributes associated with the row role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the row's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the row's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the row's containing window, sheet, or drawer
Children	Yes	UIElement array	Accessibility objects representing the row's contents
Visible Children	Yes	UIElement array	Accessibility objects representing the row's contents that are currently visible (for example, not scrolled out of view)
Help	No	String	Help-tag text for the row
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the row; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the row is focused; <code>false</code> otherwise
Subrole	Yes	String	Type of row
Index	Yes	Number	Numerical value indicating position of the row
Selected	Yes	Boolean	<code>true</code> if the row is currently selected; <code>false</code> otherwise
Disclosing	No	Boolean	<code>true</code> if the row is associated with a disclosure triangle that is currently disclosing contents; <code>false</code> otherwise
Disclosed Rows	No	UIElement array	Accessibility objects representing the items the row's disclosure triangle is disclosing
Disclosed By Row	No	UIElement	Accessibility object representing the row by which this row is disclosed
Disclosure Level	No	Number	Numerical value indicating how deep this row is in the disclosure hierarchy

An accessibility object of the row role can have the following subroles:

- **Table Row.** The row is a row in a table.

- **Outline Row.** The row is a row in an outline view.

Ruler Role

An accessibility object of the ruler role includes the following attributes as shown in Table B-28:

Table B-28 Attributes associated with the ruler role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the ruler's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the ruler's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the ruler's containing window, sheet, or drawer
Children	Yes	UIElement array	Accessibility objects representing the ruler's ruler markers and other children
Help	No	String	Help-tag text for the ruler
Focused	Yes	Boolean	<code>true</code> if the ruler is focused; <code>false</code> otherwise
Marker UIElements	Yes	UIElement array	Accessibility objects representing the ruler's ruler markers
Units	Yes	String	Ruler unit type (such as inches); see framework-specific reference documentation for string constant definitions
Unit description	Yes	String	Human-intelligible description of the ruler's unit type
Orientation	Yes	String	Horizontal or vertical orientation of the ruler; see framework-specific reference documentation for string constant definitions

An accessibility object of the ruler role supports the following optional action:

- **Press.** When the press action is performed, the ruler can be adjusted.

Ruler Marker Role

An accessibility object of the ruler marker role includes the following attributes as shown in Table B-29:

Table B-29 Attributes associated with the ruler marker role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the ruler marker's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the ruler marker's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the ruler marker's containing window, sheet, or drawer
Help	No	String	Help-tag text for the ruler marker
Focused	Yes	Boolean	<code>true</code> if the ruler marker is focused; <code>false</code> otherwise
Marker type	Yes	String	Ruler marker type (such as left tab stop); see framework-specific reference documentation for string constant definitions
Marker type description	Yes	String	Human-intelligible description of the ruler marker's unit type
Value	No	String	The positional value on the ruler of the ruler marker

An accessibility object of the ruler marker role supports the following optional action:

- **Delete.** When the delete action is performed, the ruler marker is removed from the ruler.

Scroll Area Role

An accessibility object of the scroll area role includes the following attributes as shown in Table B-30:

Table B-30 Attributes associated with the scroll area role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the scroll area's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the scroll area's containing window

Attribute	Required	Type	Description
Top-Level UIElement	Yes	UIElement	Accessibility object representing the scroll area's containing window, sheet, or drawer
Help	No	String	Help-tag text for the scroll area
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the scroll area; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the scroll area is focused; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the table or list and scroll bars
Contents	Yes	UIElement array	Accessibility objects representing the table or list displayed in the scroll area
Horizontal Scroll Bar	No	UIElement	Accessibility object representing the horizontal scroll bar (if one exists)
Vertical Scroll Bar	No	UIElement	Accessibility object representing the vertical scroll bar (if one exists)

Scroll Bar Role

An accessibility object of the scroll bar role includes the following attributes as shown in Table B-31:

Table B-31 Attributes associated with the scroll bar role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the scroll bar's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the scroll bar's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the scroll bar's containing window, sheet, or drawer
Help	No	String	Help-tag text for the scroll bar
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the scroll bar; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the scroll bar is focused; <code>false</code> otherwise

Attribute	Required	Type	Description
Children	Yes	UIElement array	Accessibility objects representing the scroller, scroll track, and scroll arrows (see Note below)
Value	Yes	Number	Numeric value representing the position of the scroller
Orientation	Yes	String	Horizontal or vertical orientation of scroll bar; see framework-specific reference documentation for string constant definitions

Note: A scroll bar's children include an object of the value indicator role (representing the scroller) and 4 objects of the button role that can have the following subroles: Increment Arrow, Increment Page, Decrement Arrow, or Decrement Page. See ["Button Role"](#) (page 38) for more information.

Sheet Role

An accessibility object of the sheet role includes the following attributes as shown in Table B-32:

Table B-32 Attributes associated with the sheet role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the sheet's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Focused	Yes	Boolean	<code>true</code> if the sheet has keyboard focus; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the sheet's children
Window	Yes	UIElement	Accessibility object representing the sheet's containing window
Grow Area	Yes	UIElement	Accessibility object representing the grow area
Default	No	UIElement	Accessibility object representing the default button
Cancel	No	UIElement	Accessibility object representing the Cancel button

An accessibility objects of the sheet role supports the following action:

- **Raise.** When the raise action is performed, this sheet becomes frontmost.

Slider Role

An accessibility object of the slider role includes the following attributes as shown in Table B-33:

Table B-33 Attributes associated with the slider role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the slider's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the slider's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the slider's containing window, sheet, or drawer
Help	No	String	Help-tag text for the slider
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the slider; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the slider is focused; <code>false</code> otherwise
Value	Yes	Number	The value associated with the position of the slider thumb (an object of the value indicator role)
Children	Yes	UIElement array	Accessibility object representing the slider's thumb (an object of the value indicator role)
Min Value	Yes	Number	The smallest numerical value the slider can have
Max Value	Yes	Number	The largest numerical value the slider can have
Allowed Values	No	Number array	Array of specific values the slider can have

An accessibility object of the slider role supports the following actions:

- **Increment.** When the increment action is performed, the slider increments to the next allowed value.
- **Decrement.** When the decrement action is performed, the slider decrements to the next allowed value.

Split Group Role

An accessibility object of the split group role includes the following attributes as shown in Table B-34:

Table B-34 Attributes associated with the split group role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the split group's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Help	No	String	Help-tag text for the split group
Window	Yes	UIElement	Accessibility object representing the split group's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the split group's containing window, sheet, or drawer
Focused	Yes	Boolean	<code>true</code> if the split group is focused; <code>false</code> otherwise
Selected	Yes	Boolean	<code>true</code> if the menu item is selected; <code>false</code> otherwise
Splitters	Yes	UIElement array	Accessibility objects representing the splitter bars

Splitter Role

An accessibility object of the splitter role is a splitter bar. This object includes the following attributes as shown in Table B-35:

Table B-35 Attributes associated with the splitter role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the splitter's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Help	No	String	Help-tag text for the splitter
Window	Yes	UIElement	Accessibility object representing the splitter's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the splitter's containing window, sheet, or drawer
Focused	Yes	Boolean	<code>true</code> if the splitter is focused; <code>false</code> otherwise
Value	Yes	Number	Numerical value representing the position of the splitter bar

Attribute	Required	Type	Description
Min Value	Yes	Number	Numerical value representing the position of the splitter bar associated with revealing the smallest portion of the view
Max Value	Yes	Number	Numerical value representing the position of the splitter bar associated with revealing the largest portion of the view
Previous Contents	Yes	UIElement array	Accessibility objects representing the objects on one side of the splitter bar (side is determined by orientation)
Next Contents	Yes	UIElement array	Accessibility objects representing the objects on one side of the splitter bar (side is determined by orientation)
Orientation	Yes	String	Horizontal or vertical orientation of splitter bars; see framework-specific reference documentation for string constant definitions

Static Text Role

An accessibility object of the static text role includes the following attributes as shown in Table B-36:

Table B-36 Attributes associated with the static text role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the static text's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the static text's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the static text's containing window, sheet, or drawer
Help	No	String	Help-tag text for the static text
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the static text; <code>false</code> otherwise
Focused	Yes	Boolean	<code>true</code> if the static text is focused; <code>false</code> otherwise
Value	Yes	String	Text in the static text
Selected Text	Yes	String	Selected portion of the text in the static text

Attribute	Required	Type	Description
Selected Text Range	Yes	Range value	Position and length (in characters) of the selected portion of the text in the static text
Number of Characters	Yes	Number	Number of characters in the text in the static text
Visible Character Range	Yes	Range value	Position and length (in characters) of the text in the static text
Insertion Point Line Number	No	Number	Line number of current cursor position in static text

Tab Group Role

An accessibility object of the tab group role includes the following attributes as shown in Table B-37:

Table B-37 Attributes associated with the tab group role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the tab group's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the tab group's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the tab group's containing window, sheet, or drawer
Tabs	Yes	UIElement array	Accessibility objects representing the tab controls
Help	No	String	Help-tag text for the tab group
Focused	Yes	Boolean	<code>true</code> if the tab group is focused; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the tab controls and the objects in the currently displayed tab pane
Value	Yes	UIElement	Accessibility object representing the currently selected tab control

Table Role

An accessibility object of the table role includes the following attributes as shown in Table B-38:

Table B-38 Attributes associated with the table role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the table's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the table's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the table's containing window, sheet, or drawer
Columns	Yes	UIElement array	Accessibility objects representing the table's columns
Rows	Yes	UIElement array	Accessibility objects representing the table's rows
Help	No	String	Help-tag text for the table
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the table; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the table is focused; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the columns, rows, and headers
Header	Yes	UIElement	Accessibility object representing the table headers
Visible Columns	Yes	UIElement array	Accessibility objects representing currently visible columns
Selected Columns	Yes	UIElement array	Accessibility objects representing currently selected columns
Visible Rows	Yes	UIElement array	Accessibility objects representing currently visible rows
Selected Rows	Yes	UIElement array	Accessibility objects representing currently selected rows

Text Area Role

An accessibility object of the text area role includes the following attributes as shown in Table B-39:

Table B-39 Attributes associated with the text area role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the text area's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the text area's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the text area's containing window, sheet, or drawer
Help	No	String	Help-tag text for the text area
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the text area; <code>false</code> otherwise
Focused	Yes	Boolean	<code>true</code> if the text area is focused; <code>false</code> otherwise
Value	Yes	String	Text in the text area
Selected Text	Yes	String	Selected portion of the text in the text area
Selected Text Range	Yes	Range value	Position and length (in characters) of the selected portion of the text in the text area
Number of Characters	Yes	Number	Number of characters in the text in the text area
Visible Character Range	Yes	Range value	Position and length (in characters) of the text in the text area
Description	Yes	String	Human-intelligible description of the text area. Not required if the title UIElement attribute is included.
Title UIElement	No	UIElement	Accessibility object representing the static text that serves as the title for the text area. Not required if the description attribute is included.
Insertion Point Line Number	No	Number	Line number of current cursor position in text area
Shared Text UIElements	No	UIElement array	Accessibility objects representing the objects with which the text of this text area is shared.
Shared Character Range	No	Range value	Range of shared text this text area displays

Note: A text area object also supports parameterized attributes. See the framework-specific documentation for more information on these attributes.

An accessibility object of the text area role can support the following action:

- **Confirm.** When the confirm action is performed, the selected portion of the text in the text area can display an editing menu.

Text Field Role

An accessibility object of the text field role includes the following attributes as shown in Table B-40:

Table B-40 Attributes associated with the text field role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the text field's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the text field's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the text field's containing window, sheet, or drawer
Help	No	String	Help-tag text for the text field
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the text field; <code>false</code> otherwise
Focused	Yes	Boolean	<code>true</code> if the text field is focused; <code>false</code> otherwise
Value	Yes	String	Text in the text field
Selected Text	Yes	String	Selected portion of the text in the text field
Selected Text Range	Yes	Range value	Position and length (in characters) of the selected portion of the text in the text field
Number of Characters	Yes	Number	Number of characters in the selected portion of the text in the text field
Visible Character Range	Yes	Range value	Position and length (in characters) of the text in the text field
Description	Yes	String	Human-intelligible description of the text field. Not required if the title UIElement attribute is included.

Attribute	Required	Type	Description
Title UIElement	No	UIElement	Accessibility object representing the static text that serves as the title for the text field
Insertion Point Line Number	No	Number	Line number of current cursor position in text field
Subrole	No	String	Type of text field
Search Button	No	UIElement	Accessibility object representing the search button. Required if the text field is a search field.
Clear Button	No	UIElement	Accessibility object representing the clear button. Required if the text field is a search field.

Note: A text field object also supports parameterized attributes. See the framework-specific documentation for more information on these attributes.

An accessibility object of the text field role can support the following action:

- **Confirm.** When the confirm action is performed, the selected portion of the text in the text field can display an editing menu.

An accessibility object of the text field role can have the following subroles:

- **Search Field.** The text field functions as a search field.
- **Secure Text Field.** The text field can accept and obscure sensitive input, such as a password.

Toolbar Role

An accessibility object of the toolbar role includes the following attributes as shown in Table B-41:

Table B-41 Attributes associated with the toolbar role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the toolbar's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the toolbar's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the toolbar's containing window, sheet, or drawer

Attribute	Required	Type	Description
Help	No	String	Help-tag text for the toolbar
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the toolbar; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the icons and controls in the toolbar
Overflow Button	No	UIElement	Accessibility object representing which of the toolbar's children (if any) is the overflow button

Unknown Role

An accessibility object of the unknown role includes the following attributes as shown in Table B-42:

Table B-42 Attributes associated with the unknown role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the object's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the object's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the object's containing window, sheet, or drawer
Help	No	String	Help-tag text for the image
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the object; <code>false</code> if interaction is disabled

Value Indicator Role

An accessibility object of the value indicator role generally represents the scroller in a scroll bar or the thumb of a slider. This object includes the following attributes as shown in Table B-43:

Table B-43 Attributes associated with the value indicator role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the value indicator's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Window	Yes	UIElement	Accessibility object representing the value indicator's containing window
Top-Level UIElement	Yes	UIElement	Accessibility object representing the value indicator's containing window, sheet, or drawer
Help	No	String	Help-tag text for the value indicator
Enabled	Yes	Boolean	<code>true</code> if the user can interact with the value indicator; <code>false</code> if interaction is disabled
Focused	Yes	Boolean	<code>true</code> if the value indicator is focused; <code>false</code> otherwise

Note: The value indicated by an object of the value indicator role is contained in the value attribute of its parent object (for example, the slider or the scroll bar)

Window Role

An accessibility object of the window role includes the following attributes as shown in Table B-44:

Table B-44 Attributes associated with the window role

Attribute	Required	Type	Description
Parent	Yes	UIElement	Accessibility object representing the window's parent
Position	Yes	Point value	Structure containing screen-position coordinates
Size	Yes	Size value	Structure containing width and height values
Subrole	Yes	String	Type of the window
Title	Yes	String	Visible title of the window
Focused	Yes	Boolean	<code>true</code> if the window is the first to receive keyboard input; <code>false</code> otherwise
Children	Yes	UIElement array	Accessibility objects representing the window's children
Main	Yes	Boolean	<code>true</code> if the window is the main window; <code>false</code> otherwise

Attribute	Required	Type	Description
Minimized	Yes	Boolean	true if the window is minimized; false otherwise
Close Button	Yes	UIElement	Accessibility object representing the close button
Minimize Button	Yes	UIElement	Accessibility object representing the minimize button
Zoom Button	Yes	UIElement	Accessibility object representing the zoom button
Grow Area	Yes	UIElement	Accessibility object representing the grow area
Proxy	No	UIElement	Accessibility object representing a document window's icon
Toolbar Button	No	UIElement	Accessibility object representing the toolbar button
Default	No	UIElement	Accessibility object representing the default button
Cancel	No	UIElement	Accessibility object representing the Cancel button
Document	No	String	URL or path for the document displayed in the window
Modal	No	Boolean	true if the window is modal; false otherwise

An accessibility object of the window role supports the following action:

- **Raise.** When the raise action is performed, this window becomes frontmost.

An accessibility object of the window role should have one of the following subroles:

- **Standard Window.** The window is a standard application or document window.
- **Dialog.** The window is a modeless dialog, such as a Find dialog.
- **System Dialog.** The window is a system-wide modeless dialog, such as the restart dialog.
- **Floating Window.** The window is a utility or information window, such as the Color Picker window.
- **System Floating Window.** The window is a system-wide utility or information window, such as the About This Mac window.

Document Revision History

This table describes the changes to *Accessibility Overview*.

Date	Notes
2008-03-11	Removed the Children attribute from the Busy Indicator role information.
2007-12-11	Changed the description of the window role's focused attribute.
2007-09-04	Made minor fixes to required attributes for accessibility objects of the image and combo box roles.
2007-02-08	Added information on the group role and its attributes.
2006-06-28	Made minor corrections.
2005-11-09	Made minor bug fixes.
2005-04-29	Updated attributes supported by Menu, Menu Item, and Menu Bar Item roles.
	New document that explains the accessibility features built into Mac OS X. Some content was previously in "Making Carbon Applications Accessible to Users With Disabilities."

REVISION HISTORY

Document Revision History