Help ToolKit for Motif
Programming Manual

Version 1.1.2

Software Components, Inc.
PMB 663
8775-M Centre Park Drive
Columbia, Maryland, USA 21045
www.softwarecomp.com
info@softwarecomp.com

January 2000

Help ToolKit for Motif Programming Manual

Table of Contents

Table of Code Listings

List of Figures

List of Tables

Preface

The Help ToolKit for Motif allows developers to easily add and modify various styles of on-line context-sensitive help to their Motif applications. Using a small set of functions to install the library, every widget and gadget in the application seemingly inherits new resources allowing help to be configured and changed via X resource files.

In addition, XscHelp provides an API that allows a developer to plug-in virtually any on-line help system, such as the Help system provided through CDE. In fact, a sample CDE Help system plug-in is provided with XscHelp.

The Help ToolKit also provides a rich API allowing help to be controlled and manipulated within your program.

Audience

The Help ToolKit is intended to be used by Motif application programmers. It is easy to use and configure, so even Motif beginners can achieve great results quickly. Configuring XscHelp is, in many ways, similar to configuring Motif widgets. As a result, it is useful to have a basic understanding of the X Window System Intrinsics and OSF/Motif before reading this manual or using XscHelp. O'Reilly & Associates, Inc. has a complete set of well-written books covering all aspects of the X Window System and Motif. If you have trouble understanding the Help ToolKit, you are encouraged to examine Volumes 4, 5, and 6a from the O'Reilly series.

Overview

The manual begins with an overview of the types of help supported by the Help ToolKit, followed by a discussion of how to install the Help ToolKit distribution on your system

The initial section is followed by a multi-chapter tutorial of the Help ToolKit. The tutorial includes terminology and initialization, as well as configuration of the various styles of supported help.

The manual concludes with a reference section broken into chapters based on specific topic areas.

The manual contains the following chapters:

Chapter 1, “Introduction,” on page 1, provides an overview of the capabilities of the Help ToolKit. This includes a brief examination of the supported styles of help.

Chapter 2, “Using the Help ToolKit,” on page 5, covers initialization of the Help ToolKit.

Chapter 3, “Tips,” on page 11, explains how to use, configure, and control Tip pop-up windows.

Chapter 4, “Cues,” on page 25, shows how to use, configure, and control Cue up windows.

Chapter 5, “Hints,” on page 33, covers how to install, configure, and control Hint help.

Chapter 6, “Context-Help,” on page 41, shows how to integrate any Intrinsics-based context-sensitive help system to a Motif application by using the Help ToolKit as the “glue.”

Chapter 7, “CDE Context-Help,” on page 49, examines how to use and configure the optional CDE Help System support.

Chapter 8, “Dynamic Updates,” on page 59, discusses how to modify Tips, Cues, and Hints at run-time.

Chapter 9, “Authoring Aids,” on page 71 describes some of the capabilities that are designed into the Help ToolKit specifically to assist in the creation and maintenance of the Help text and appearance.

Chapter 10, “Functions,” on page 77, begins the reference section of the manual with a discussion of all the public functions supported by the Help ToolKit.

Chapter 11, “Macros and Constants,” on page 107, covers the C preprocessor macros and constants defined by the ToolKit.

Chapter 12, “Data Types,” on page 109, contains a breakdown of public data structures defined by the ToolKit.

Chapter 13, “External Variables,” on page 111, provides a discussion of any external variables defined by the Help ToolKit that can be exploited by the application.

Chapter 14, “Resources,” on page 113, contains a break-down of all the resources that are supported by the Help ToolKit.

Software Versions

This manual was written for the Help ToolKit for Motif, Version 1.1.2. XscHelp is designed to work with Motif 1.2 and, optionally, with CDE 1.0.

XscHelp was developed and tested with RedHat Linux 4.1 (kernel 2.0.27).

Distribution

The distribution for the evaluation version has the name XscHelpUR-v1-1-2 instead of XscHelp-v1-1-2

The Help ToolKit for Motif is distributed as ANSI C source code. It is distributed as a single, GNU zip compressed tar file with a name similar to `XscHelp-v1-1-2.tar.gz`. The tar file can be uncompressed with the GNU `gunzip` command. The tar file can than be used to create a temporary distribution directory to install the product. The following commands show how to uncompress and untar the distribution file in a typical UNIX environment.

```
> gunzip XscHelp-v1-1-2.tar.gz
> tar xf XscHelp-v1-1-2.tar
```

This will create a single directory with a name similar to `XscHelp-v1-1-2`. The `INSTALLATION` file which explains how to install the toolkit will be located inside this directory.

Support

The Help ToolKit is developed by Software Components, Inc. If you purchased XscHelp directly from Software Components, then all technical questions and bug reports should be sent to Software Components via one of the following mechanisms:

- E-mail: `support@softwarecomp.com`
- Fax: 410-480-1422 (USA)
- Mail: **Software Components, Inc.**
  Technical Support
  PMB 663
  8775 Centre Park Drive
  Columbia, Maryland, USA 21045

Please include you product serial number with all requests.

If you purchased the product from a licensed reseller, then you should contact that reseller for technical support. Bug reports may still be forwarded directly to Software Components, Inc. via e-mail at `bugs@softwarecomp.com`; fixes for bugs reported to this e-mail address will be considered for inclusion in future updates.

All parties are encouraged to send comments, ideas, and suggestions directly to Software Components, Inc. via e-mail at `comments@softwarecomp.com`. If e-mail is not convenient, feel free to send comments and bug reports via the fax number or mail address shown above.

Check the Software Components, Inc. web page for additional information (`www.softwarecomp.com`).

CHAPTER 1

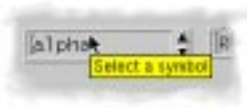
Introduction

This manual describes how to use the Help ToolKit in your Motif-based applications. XscHelp allows application developers to add various styles of on-line help to virtually any Motif-based application. One of the key features of XscHelp is that it does not levy any constraints on the design or implementation of the application; the toolkit operates almost completely behind the scenes with almost no required application interaction.

Supported Styles of Help

Four styles of on-line help are supported by XscHelp: Tips, Hints, Cues, and Context-Help[1]. Each of these help styles are briefly introduced in this section.

Tips are brief pop-up reminders that appear near a widget or gadget when the mouse cursor briefly comes to rest.

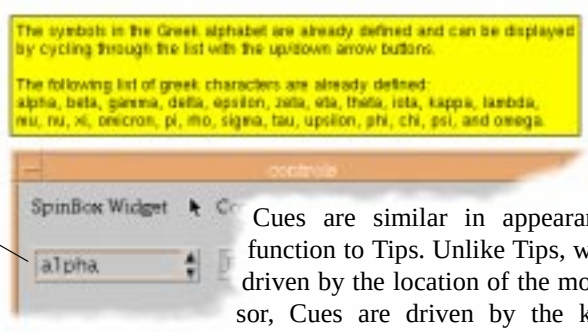


Tips are brief textual descriptions that appear in small pop-up windows over or near a screen object when the mouse cursor briefly rests over the object. Tips are commonly seen with icon buttons and where originally popularized in Windows 95 applications. The Help ToolKit allows tips to be assigned to any widget or gadget. In the Windows 95/NT implementation, Tips have a black foreground and yellow background and are displayed slightly offset from the pointer position. Tips in the Help ToolKit have a wide range of configuration options. XscHelp Tips are detailed in Chapter 3 starting on page 11.

1. In this manual, the term *Context-Help* refers to an infrastructure that allows integration with a context-sensitive help system.

Cues are driven by the keyboard focus instead of the mouse cursor. A Cue normally provides instructions about how to format or enter data.

This widget has the keyboard focus.

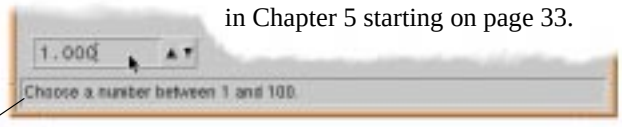


Cues are similar in appearance and function to Tips. Unlike Tips, which are driven by the location of the mouse cursor, Cues are driven by the keyboard focus. Cues are displayed in small pop-up windows that float above the shell widget that contains the keyboard focus. A Cue is displayed if the widget or gadget containing the keyboard focus defines a Cue or if an ancestor of the widget (up to the closest WM Shell) defines a Cue. Cues are discussed in detail in Chapter 4, “Cues,” on page 25.

XscHelp Hints are textual messages that appear in a fixed location on a window when the mouse cursor moves over a given widget or gadget contained in that window. The Hints are displayed in a widget created and positioned by the application. The hint widget is frequently positioned in the lower-left corner of a window. This type of help is commonly used with menus and text fields. XscHelp Hints also support a wide range of configuration options which are detailed in Chapter 5 starting on page 33.

Hints are located in a fixed area on a window. Information about the widget or gadget under the mouse cursor is displayed in the hint.

The hint display area is created and positioned by the programmer.



When using the Help ToolKit, both Tips and Hints are automatically invoked as the mouse pointer moves over a screen object. Cues are automatically displayed as the keyboard focus changes.

Context-Help provides developers with an infrastructure to easily integrate Motif applications with any Motif-compliant on-line context-sensitive help facility. The Help ToolKit, however, does not actually provide a context-sensitive help facility; instead, it provides the mechanics that allow the application to communicate with a third-party help facility, such as the help facility found under CDE. Context-Help is described in greater detail in Chapter 6 starting on page 41. In addition, the

Help ToolKit Context-Help capabilities for the CDE Help System are detailed in Chapter 7 starting on page 49.

Any Intrinsic-based object (except shell widgets), including gadgets, manager widgets, and custom or third-party widgets, can be given Tip, Hint, Cue, and Context-Help resources. These Help ToolKit resources can be configured via the resource database and, as a result, appear to be transparently inherited by the associated screen objects.

All of these topics are described in detail throughout the remainder of this manual.

CHAPTER 2

Using the Help ToolKit

This chapter examines how the Help ToolKit is installed in an application. All of the examples in this manual where created by installing the Help ToolKit into a sample program called “controls” which is distributed with CDE. This chapter shows all of the code added to the “controls” program to create the examples shown in Chapter 1[1].

Initializing the Library

FUNCTION
XscHelpInstall

The Help ToolKit must be initialized before it will function properly in an application. It is initialized by calling the `XscHelpInstall()` function. This function should be called immediately after the first shell widget in the application is created, but prior to the creation of any other widgets.

SOURCE CODE 2-1: Initializing the Help ToolKit

Almost all of the Help ToolKit functions are declared in the header file <Xsc/Help.h>

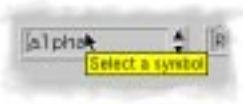
This installs the Help ToolKit providing complete support for Tips and Cues on the specified display

```
...
#include <Xsc/Help.h>
...
main( int argc, char** argv )
{
...
   toplevel = XtAppInitialize( &appContext,
      ApplicationClass, NULL, 0, &argc, argv,
      NULL, NULL, 0 );

   XscHelpInstall( toplevel );
...
}
```

1. In some cases, the source code has been slightly reformatted to satisfy the layout requirements for the manual.

The parameter to the `XscHelpInstall()` function is the first shell widget created by the application. Once this function is called, all widget created on the display of the specified shell completely support XscHelp Tips and Cues.



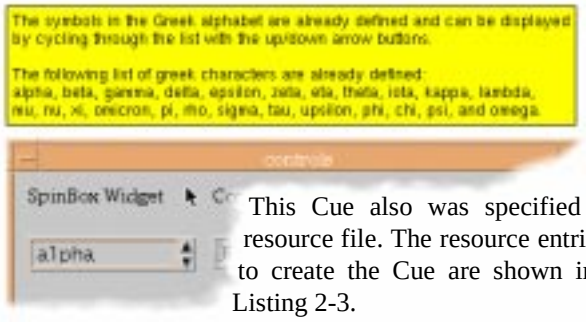
The actual Tips and Cues can be added using the application's resource file. Once XscHelp is installed, every widget (and gadget) appears to have new resources. The Tip shown here was created with the resource entries shown in Code Listing 2-2.

A compound widget looks and acts like a single widget but is actually implemented with one or more child widgets or gadgets. Normally, each child looks like a separate widget to the Help ToolKit; as a result, each child would have its own Tips. The xscTipCompound resource is used to specify that all widget children should be considered as part of a single widget.

SOURCE CODE 2-2: Simple resource file example for a Tip

```
*spinBox1.xscTipTopic:    Select a symbol
*spinBox1.xscTipCompound: True
```

The `xscTipTopic` resource name is used to specify the text to display in the tip. The `xscTipCompound` resource specifies that the widget should be treated as a primitive. Since a SpinBox widget is actually composed or many children, the `xscTipCompound` resource is needed to prevent each child from having its own Tip.



This Cue also was specified in the resource file. The resource entries used to create the Cue are shown in Code Listing 2-3.

The Cue's text is specified with the `xscCueTopic` resource. Lines that end with a '\' character are continued on the next line. In addition, a "`\n`" represents a newline character. The default background color of the Cue is modified with the resource `xscCueBackground`. The margins are specified with `xscCueMarginWidth` and `xscCueMarginHeight`.

Tips and Cues offer a large variety of configuration options which are explored in more detail in the appropriate chapters in this manual.

SOURCE CODE 2-3: Simple resource file example for a Cue

These lines compose a single entry in the resource database. Lines ending with a '\' are continued with the next line.

```
*spinBox1.xscCueTopic:     \
The symbols in the Greek alphabet are already \
defined and can be displayed\n\
by cycling through the list with the \
up/down arrow buttons.\n\
\n\
The following list of greek characters \
are already defiend:\n\
alpha, beta, gamma, delta, epsilon, zeta, \
eta, theta, iota, kappa, lambda,\n\
mu, nu, xi, omicron, pi, rho, sigma, tau, \
upsilon, phi, chi, psi, and omega.

*spinBox1.xscCueBackground:   Yellow
*spinBox1.xscCueMarginWidth:  4
*spinBox1.xscCueMarginHeight: 4
```

Adding Hints

Hints are not functional until a *hint display widget* is specified. Each toplevel window needs to have a separate hint display widget; those toplevel windows that do not have a hint display widget cannot display hints.

Any widget can be used as a hint display widget; gadgets cannot be used. The programmer may position the widget as desired. The hint display widget is specified by calling `XscHelpHintInstall()`, passing the widget as the only parameter.

The Hint shown above was created with the resources shown in Code Listing 2-4. The resource `xscHintTopic` specifies the actual Hint text for the widget. The `xscHintCompound` resource is used to make the compound widget act like a single primitive widget.

SOURCE CODE 2-4: Simple resource file example for a Hint

```
*spinBox5.xscHintTopic: \
Choose a number between 1 and 100.

*spinBox5.xscHintCompound:  True
```

Code Listing 2-5 shows how the hint display widget was created and added to the CDE "controls" demo program.

SOURCE CODE 2-5: Adding Hint support

```
...
Widget hintFrame, hintLabel;
XmString cs;
...
hintFrame = XtVaCreateManagedWidget(
   "hintFrame", xmFrameWidgetClass, workArea,
   XmNbottomAttachment, XmATTACH_FORM,
   XmNbottomOffset,     0,
   XmNleftAttachment,   XmATTACH_FORM,
   XmNleftOffset,       0,
   XmNrightAttachment,  XmATTACH_FORM,
   XmNrightOffset,      0,
   XmNshadowThickness,  1,
   NULL );

cs = XmStringCreateLocalized( " " );
hintLabel = XtVaCreateManagedWidget(
   "hintLabel", xmLabelWidgetClass, hintFrame,
   XmNlabelString,      cs,
   NULL );
XmStringFree( cs );
...
XscHelpHintInstall( hintLabel );
...
```

The Help ToolKit renders directly onto the surface of the hint display widget; shadow borders (or other decorations) can be overwritten during the rendering. A frame widget can be used to prevent this problem.

Labels work well for the hint display area. A compound string with a single space is used so that the widget will have a height while not rendering any text.

This actually installs Hint support for the toplevel window.

Adding Context-Help

CROSS-REFERENCE Refer to "Installing Context-Help Support" on page 41.

The function `XscHelpContextInstall()` is called to activate Context-Help support; this function is only called once and is needed to define the help resources that must be delivered to the application when context-sensitive help is requested.

Dealing with Multiple Displays

By definition, a display consists of a keyboard, a pointer device (e.g., mouse) and one or more screens.

The Help ToolKit will work with Motif applications that have multiple display connections. Each display connection is treated like the original; call `XscHelpInstall()` with the first widget created on the display, but before any additional widgets are created.

SOURCE CODE 2-6: Recognizing an additional display connection

```
...
#include <Xsc/Help.h>
...
   Display new_display;
   Widget shell;
   int argc = 0;

   new_display = XtOpenDisplay(
      app_context, “ravens:0”, NULL, “Demo”, NULL,
      0, &argc, NULL );
   shell = XtAppCreateShell(
      NULL, “Demo”, applicationShellWidgetClass,
      new_display, NULL, 0 );

   XscHelpInstall( shell );
...
```

Additional display connections can be specified; closed display connections are automatically “cleaned up.”

A workstation can have more that one screen which is usable from a single keyboard and pointer device. Each screen is part of a single logical display.

Nothing special needs to be done to work with multiple screens.

There are no functions to uninstall displays because the Help ToolKit automatically cleans-up when a display connection is closed.

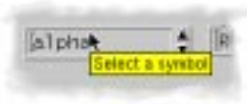
CHAPTER 3

Tips

Tips are small pop-up windows that appear over or near a widget or gadget; the text in the Tip window usually provides a brief description of the associated widget. Typically, the Tip appears after the mouse cursor has been at rest in the widget for a short period of time; the Tip disappears when the mouse cursor moves out of the widget.

Specifying the Tip Topic

RESOURCE
xscTipTopic

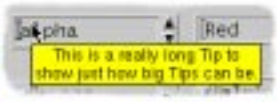


Regardless of what XscHelp resources are set, a given widget has an associated Tip if and only if the widget's `xscTipTopic` resource has a value. The value of the `xscTipTopic` resource is the string to display in the Tip.

For instance, the Tip shown above could have been created with the resource entry shown in Code Listing 3-1.

SOURCE CODE 3-1: Specifying a Tip topic via resource file

```
*spinBox1*xscTipTopic: Select a symbol
```



Although normally brief and limited to one line, a Tip can be as long as you desire. For instance, this Tip was specified with the resources shown in Code Listing 3-2. If the widget's `xscTipTopic` did not have a value, then a Tip would not be displayed under any circumstance.

The other Tip resources define how each Tip is rendered and how it behaves.

SOURCE CODE 3-2: Specifying a multi-lined Tip

```
*spinBox1*xscTipTopic: This is a really long tip \
to\nshow just how big Tips can be.
```

Rendering Options for Tip Text

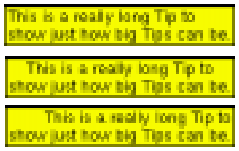
This section examines the different text rendering options available for Tips.

Text Alignment

RESOURCE
xscTipAlignment

The `xscTipAlignment` resource specifies the alignment of the Tip text. The allowed values for this resource are the same as the standard Motif alignment resource:

- `XmALIGNMENT_BEGINNING`
- `XmALIGNMENT_CENTER`
- `XmALIGNMENT_END`

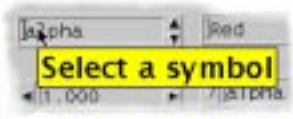


The Tip window shrink-wraps around the Tip text; as a result, the alignment resource is only useful if the Tip topic spans multiple lines. The default value is `XmALIGNMENT_CENTER`.

Changing the Font

RESOURCE
xscTipFontList

The font list for Tip text is set with the `xscTipFontList` resource. This font list can be set to any valid Motif font list using the standard Motif font list resource file syntax. As will be seen later, more than one font can be rendered in the text if multiple font are specified in the font list. If multiple fonts are specified, the first is used as the default.



This example shows a 24-point font used for the Tip. The font list was set using the resource name `xscTipFontList` as shown in Code Listing 3-3.

SOURCE CODE 3-3: Specifying a font for the Tip text

This line is used to specify one or more fonts.

```
*spinBox1*xscTipTopic:    Select a symbol
*spinBox1*xscTipFontList: lucidasans-24
```

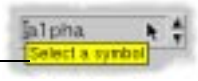
Default String Direction

RESOURCE
xscTipStringDirection

The default string direction, which is left-to-right, can also be specified with the `xscTipStringDirection` resource. The valid values are:

- `XmSTRING_DIRECTION_L_TO_R` — left-to-right string rendering
- `XmSTRING_DIRECTION_R_TO_L` — right-to-left string rendering.

Notice that the Tip appears to the left of the pointer



This example was created with the resources shown in Code Listing 3-4. Even though the string direction is set to `XmSTRING_DIRECTION_R_TO_L`, the text does not appear to be rendered right-to-left. Motif 1.2 expects right-to-left rendered strings to be generated backwards with the first character at the end of the string. Right to left rendering works properly when the direction is embedded in the compound string. Motif 2.x handles the default correctly.

SOURCE CODE 3-4: Changing the string direction for Tip text

This line is used to set the default string direction for the Tip text.

```
*spinBox1.xscTipTopic: Select a symbol
*spinBox1.xscTipStringDirection: \
                          XmSTRING_DIRECTION_R_TO_L
```

The string direction resource impacts the layout of the Tip. Notice that the Tip appears to the left of the arrow instead of the right; that is because the `xscTipStringDirection` resource is set to `XmSTRING_DIRECTION_R_TO_L`.

The text alignment is also impacted by the string direction, as shown in Table 3-1.

TABLE 3-1: Tip String Direction versus Alignment

Alignment	Left to Right	Right to Left
Beginning		
Center		
End		

String Converters

String converters are used to translate the ASCII string found in the resource database to a compound string that can be rendered by the Motif toolkit. By default, a simple sting converter is used that renders the text with the first font in the font list. However, other converters that support more complex rendering are available.

RESOURCE
xscTipStringConverter

A string converter for a Tip is specified with the `xscTipStringConverter` resource. The allowed values are:

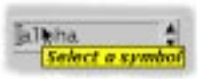
- `XmXSC_STRING_CONVERTER_STANDARD` (default)
- `XmXSC_STRING_CONVERTER_FONT_TAG`
- `XmXSC_STRING_CONVERTER_SEGMENTED`

Specifying Any Font from the FontList

A Motif font list is, as the name implies, a list of fonts. Normally, the first font in the list is used with strings specified in resource files. The Help ToolKit allows any font in the font list to be used when rendering the text.

RESOURCE
xscTipFontListTag

Each font entry in the font list is identified by a font tag. In order to explicitly specify an arbitrary font in the font list, the `xscTipStringConverter` resource is set to `XmXSC_STRING_CONVERTER_FONT_TAG` and the value of the `xscTipFontListTag` resource is set to the tag of the desired font. If the specified tag is not a member of the font list, then the first entry in the font list is used by default.



Code Listing 3-5 specifies a font list composed of two fonts; the first font in the list has a tag of "`norm`" and the second font has a tag of "`bold-italic`". By specifying the resource `xscTipFontListTag` as "`bold-italic`" and setting the resource `xscTipStringConverter` to the value `XmXSC_STRING_CONVERTER_FONT_TAG`, the Tip text is rendered with the bold italic font as shown. If either the `xscTipStringConverter` or the `xscTipFontListTag` resources had not been specified, then the font tagged as "norm" would have been used by default.

SOURCE CODE 3-5: Specifying one of many fonts from a font list

This font list has two font entries: one is tagged "norm" and the other is tagged "bold-italic".

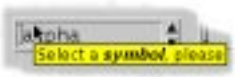
```
*spinBox1.xscTipFontList: \
   lucidasans-12=norm,\
   lucidasans-bolditalic-12=bold-italic
*spinBox1.xscTipTopic: Select a symbol
*spinBox1.xscTipStringConverter: \
    XmXSC_STRING_CONVERTER_FONT_TAG
*spinBox1.xscTipFontListTag: bold-italic
```

Using Multiple Fonts and Directions

Multiple fonts can be used and the string direction can be changed by setting the `xscTipStringConverter` resource to `XmXSC_STRING_CONVERTER_SEGMENTED`. This compound string converter scans the text for special *escape sequences* that are used to change the font or string direction. All of the defined escape sequences start with the character '@'.

The escape sequence "`@f`" indicates that a new font is going to be specified; the tag of the new font is enclosed in square brackets and should immediately follow the "`@f`". If no font tag is specified in the square brackets, then the default font is used. The default font is the first font in the font list.

Based on the resource values shown in Code Listing 3-6, the word "symbol" in the specified Tip topic is rendered in bold and italics while the remainder of the string is rendered with the default font.



SOURCE CODE 3-6: Specifying multiple fonts for a Tip

The "@f[bold-italic]" changes the font to the "bold-italic" font and "@f[]" changes it back to the default

In order to use multiple fonts, the string converter must be set as shown.

```
*spinBox1*xscTipFontList:  \
   lucidasans-12=norm, \
   lucidasans-bolditalic-12=bold-italic
*spinBox1*xscTipTopic: \
   Select a @f[bold-italic]symbol@f[], please
*spinBox1*xscTipStringConverter: \
   XmXSC_STRING_CONVERTER_SEGMENTED
```

The string direction can be altered by using the escape sequence "@d." followed immediately by a direction indicator — left-to-right is specified with the '>' character while right-to-left is specified with the '<' character.

In addition to having the word “symbol” rendered in bold-italic, this example has the first word — “Select” — rendered from right to left. The resource values used to create this example are shown in Code Listing 3-7.

CROSS-REFERENCE
Refer to “Default String Direction” on page 13 for more information about problems with Motif 1.2 and the default string direction.

Notice that the letters in the word “Select” are actually rendered from right to left, as you might expect. This is in contrast to using the `xscTipStringDirection` resource to set the default string direction for the Tip text to `XmSTRING_DIRECTION_R_TO_L`; recall that Motif 1.2 (and CDE 1.x) does not actually render the characters from right to left.

SOURCE CODE 3-7: Changing the direction multiple times in a Tip

The “@d<” changes the direction to right-to-left and “@d>” changes it back to left-to-right

```
*spinBox1*xscTipFontList:  \
   lucidasans-12=norm, \
   lucidasans-bolditalic-12=bold-italic
*spinBox1*xscTipTopic: \
   @d<Select@d> a @f[bold-italic]symbol@f[], please
*spinBox1*xscTipStringConverter: \
   XmXSC_STRING_CONVERTER_SEGMENTED
```

A ‘@’ character can be included in the output text by escaping it with another ‘@’ character. In other words, the converter replaces the string “@@” with a single ‘@’ character.

Tip Colors and Borders

By default, Tips have black text, a one pixel wide black border, and a yellow background. All of these attributes can be configured. In addition, a Tip can also be displayed with a traditional Motif shadow border.

Background and Border

RESOURCE
xscTipBackground
xscTipBorderColor
xscTipBorderWidth

The background color of a Tip can be specified by setting the `xscTipBackground` resource. Likewise, the border color and the border width are modified via the `xscTipBorderColor` and `xscTipBorderWidth` resources, respectively.

Code Listing 3-8 shows a resource file that specifies a background color of "Blue", a border color of "White", and a border width of 3 pixels. The resulting Tip is shown in this example.

SOURCE CODE 3-8: Setting the background color, border color, and border thickness of a Tip

The background color, border color, and border thickness are controlled with these resources

```
*spinBox1.xscTipTopic:  Select a symbol, please
*spinBox1.xscTipBackground:  Blue
*spinBox1.xscTipBorderColor: White
*spinBox1.xscTipBorderWidth: 3
```

Shadow Borders

RESOURCE
xscTipShadowThickness
xscTipShadowType

By default, the Tip shadow thickness is 0 pixels wide which results in no shadow border being rendered. If a shadow border is desired, the desired width is specified with the resource `xscTipShadowThickness` resource and the shadow style is specified with the `xscTipShadowType` resource.

All of the standard Motif shadow styles are supported; the allowed values are:

- `XmSHADOW_IN`
- `XmSHADOW_OUT` (default)
- `XmSHADOW_ETCHED_IN`
- `XmSHADOW_ETCHED_OUT`

The resource file entries shown in Code Listing 3-9 create a Tip with a blue background and a 4 pixel wide, etched out shadow border.

SOURCE CODE 3-9: Setting the shadow border for a tip

The shadow border size and shape is controlled with these resources

```
*spinBox1.xscTipTopic:    Select a symbol, please
*spinBox1.xscTipBackground:      Blue
*spinBox1.xscTipShadowThickness: 4
*spinBox1.xscTipShadowType:      XmSHADOW_ETCHED_OUT
*spinBox1.xscTipBorderWidth:     0
```

Overriding Automatic Color Selections

RESOURCE
xscTipMotifColorModel

The colors of the Tip text and shadow border are normally derived automatically by the Help ToolKit. The colors are derived from the specified Tip background color by using the Motif function `XmGetColors()`. This behavior can be disabled by setting the value of the resource `xscTipMotifColorModel` to False.

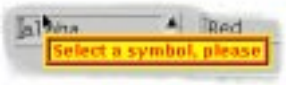
RESOURCE
xscTipForeground

When the `xscTipMotifColorModel` is set to `False`, the foreground color for the Tip is explicitly set using the resource `xscTipForeground`.

RESOURCE
xscTipColorBase

The top and bottom shadow colors cannot be set independently. Instead, the `xscTipColorBase` resource is used to specify a base color from which the top and bottom shadow colors are derived.

This example shows a Tip with a yellow border and background; the Tip also has red text and a reddish shadow border. The resource file code used to create this example is shown in Code Listing 3-10.



SOURCE CODE 3-10: Tip foreground and shadow border colors

The foreground text color and the shadow border color can be adjusted only if the `xscTipMotifColorModel` is set to False.

```
*spinBox1.xscTipTopic:    Select a symbol, please
*spinBox1.xscTipBackground:      Yellow
*spinBox1.xscTipBorderColor:     Yellow
*spinBox1.xscTipBorderWidth:     2
*spinBox1.xscTipShadowThickness: 4
*spinBox1.xscTipShadowType:     XmSHADOW_ETCHED_OUT
*spinBox1.xscTipBaseColor:       Red
*spinBox1.xscTipForeground:      Red
*spinBox1.xscTipMotifColorModel: False
```

Tip Margins

RESOURCE
xscTipMarginBottom
xscTipMarginHeight
xscTipMarginLeft
xscTipMarginRight
xscTipMarginTop
xscTipMarginWidth

Tips in the Help Toolkit provide all of the margin resources supported by the Motif label widget. The margin resource supported are as follows:

- `xscTipMarginBottom`
- `xscTipMarginHeight`
- `xscTipMarginLeft`

- `xscTipMarginRight`
- `xscTipMarginTop`
- `xscTipMarginWidth`

All of the margin resources default to 0 except `xscTipMarginHeight` and `xscTipMarginWidth`. The `xscTipMarginHeight` resource defaults to 1 and the `xscTipMarginWidth` resource defaults to 2.

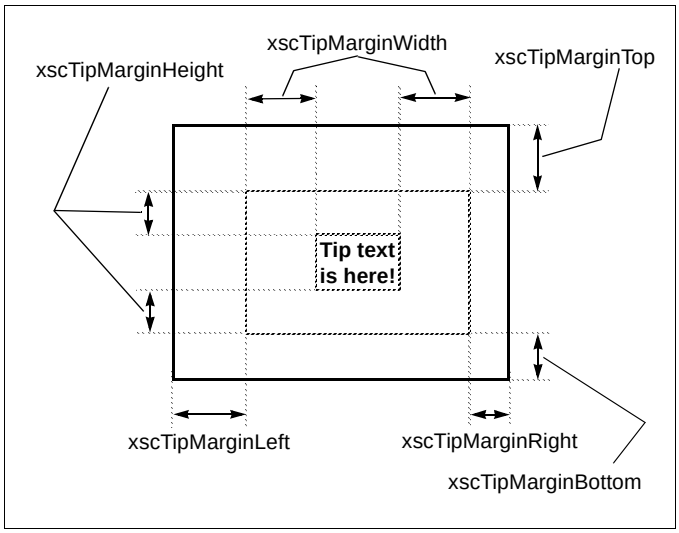
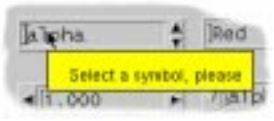


FIGURE 3-1: Tip margin resources

Figure 3-1 shows how each of the six Tip margin resources combine to create a desired margin layout. Notice that the margin width is used on both the left and the right sides of the Tip. Likewise, the margin height is used on both the top and bottom sides of the Tip.



This example uses the margin resources shown in Code Listing 3-11. The example was designed to mimic the image shown in Figure 3-1 to help reinforce how the relationship among each margin resource. The right and bottom margins were both set to 0; therefore, any margin seen on the right and bottom edge is due to the `xscTipMarginWidth` and `xscTipMarginHeight` resources, respectively.

SOURCE CODE 3-11: Tip margin resources

```
*spinBox1.xscTipTopic:  Select a symbol, please
*spinBox1.xscTipMarginHeight:  4
*spinBox1.xscTipMarginWidth:   8
*spinBox1.xscTipMarginLeft:    8
*spinBox1.xscTipMarginRight:   0
*spinBox1.xscTipMarginTop:     8
*spinBox1.xscTipMarginBottom:  0
```

Tip Control Resources

Rather than specifying how a Tip is rendered, some of the Tip resources control the Tip's behavior. For instance, some resources specify where the Tip is displayed while others indicate when it is popped-up.

When does a Tip pop-up?

RESOURCE
xscTipEnabled

The `xscTipEnabled` resource is used to enable and disable the Tip for a specific widget (or gadget). If `xscTipEnabled` is set to `True`, then the Help ToolKit will pop-up the Tip when the mouse pointer briefly comes to rest in the widget; however, if `xscTipEnabled` is set to `False`, then the Tip is not displayed under any circumstance.

FUNCTION
XscHelpSetTips...
...EnabledGlobally()
...EnabledOnShell()

The `xscTipEnabled` resource can be overridden either globally or on a shell-by-shell basis. Using the function `XscHelpSetTipsEnabledOnShell()`, the programmer can prevent all Tips associated with widgets in a given shell from being displayed. Likewise, the function `XscHelpSetTipsEnabledGlobally()` can be used to prevent all Tips in the application from being displayed. For a given Tip to display, the global, shell, and widget "tip enable" resources must all be set to `True`.

RESOURCE
xscTipPopupInterval

Assuming a given Tip is enabled, it will not actually pop-up until the mouse cursor briefly stops on the widget or gadget. The `xscTipPopupInterval` specifies how long the mouse cursor must be at rest over the widget. The interval value is specified in milliseconds. The default value is 1000 milliseconds which is equivalent to 1 second. If this interval is set to 0, then the Tip is popped-up immediately when the mouse cursor enters the widget.

RESOURCE
xscTipGroupId

Tips can be assigned to a Tip group via the `xscTipGroupId` resource. Once a Tip is displayed, it is redisplayed immediately as the mouse cursor moves from widget to widget if each widget (and the least common ancestors) are all in the same group. This effectively disables the `xscTipPopupInterval` resource once a Tip is displayed in a group.

The `xscTipGroupId` resource can be assigned an integer value between 1 and 10000, inclusive. There are two special values that can also be used:

- `XmXSC_TIP_GROUP_PARENT`
- `XmXSC_TIP_GROUP_SELF`

Setting the `xscTipGroupId` resource to the value `XmXSC_TIP_GROUP_PARENT` forces the widget to have the same group as its parent. Since this is the default, all widgets are members of a single group unless specified otherwise.

Using `XmXSC_TIP_GROUP_SELF` assigns a unique value to the widget's `xscTipGroupId` resource.

Beginning with version 1.1.2, there are two methods for deriving the default for this resource: the old *static* method and the new *dynamic* method.

There are two methods used to derive the default value. The *static* method was the only method prior to version 1.1.2. In the static method, the default value for this resource is always `XmXSC_TIP_GROUP_PARENT`. The new *dynamic* method makes the default value equal to `XmXSC_TIP_GROUP_SELF` as long as all of the widget's ancestors also use the default; if an ancestor has an explicit value for this resource (even if that value is the symbolic value `XmXSC_TIP_GROUP_SELF`), then the default value is `XmXSC_TIP_GROUP_PARENT`.

The dynamic algorithm is used by default.

FUNCTION
XscHelpIsDynamicTip-GroupIdDefaultActive()

You determine which method is being used with the function `XscHelpIsDynamicTipGroupIdDefaultActive()`. A result of True indicates that the new dynamic method is in use, while false indicates that the old static method is in use.

FUNCTION
XscHelpSetDynamicTip-GroupDefault()

The default method can be specified by calling the function `XscHelpSetDynamicTipGroupDefault()`. The function accepts a single `Boolean` argument: `True` for the new dynamic method and `False` for the old static method.

RESOURCE
xscTipPopdownInterval

In some cases, it is desirable to display a Tip only for a specified period of time. The `xscTipPopdownInterval`

resource can be used to specify how long the Tip should be displayed. The interval is specified in milliseconds and the default value is 0, indicating that the Tip should never be automatically popped-down.

For example, if the `xscTipPopdownInterval` for a widget is set to 3000, then the Tip is automatically popped-down 3 seconds after it is displayed.

Where does a Tip pop-up?

RESOURCE
xscTipPosition

A Tips's location is controlled via by `xscTipPosition` resource. The allowed values for this resource are:

- `XmXSC_TIP_POSITION_BOTTOM_BEGINNING`
- `XmXSC_TIP_POSITION_BOTTOM_END`
- `XmXSC_TIP_POSITION_BOTTOM_LEFT`
- `XmXSC_TIP_POSITION_BOTTOM_RIGHT`
- `XmXSC_TIP_POSITION_POINTER` (default)
- `XmXSC_TIP_POSITION_TOP_BEGINNING`
- `XmXSC_TIP_POSITION_TOP_END`
- `XmXSC_TIP_POSITION_TOP_LEFT`
- `XmXSC_TIP_POSITION_TOP_RIGHT`

The default value `XmXSC_TIP_POSITION_POINTER` is used to position the Tip relative to the position of the mouse cursor. The other value are used to position the Tip immediately above or below the widget with one edge of the Tip flush with the edge of the widget.

When the value is `XmXSC_TIP_POSITION_POINTER`, the upper-left corner of the Tip appears (by default) slightly below the hot-spot of the mouse cursor.

The Tip position values containing the token "`BOTTOM`" cause the Tip to appear immediately below the widget; values containing the token "`TOP`" cause the Tip to appear immediately above the widget.

The values containing the token "`LEFT`" cause the left edge of the Tip to be flush with the left edge of the widget. Likewise,

values containing the token "RIGHT" cause the right edge of the Tip to be flush with the right edge of the widget.

The values containing the token "BEGINNING" cause the beginning edge of the Tip to be flush with the equivalent widget edge. Similarly, the values containing the token "END" cause the ending edge of the Tip to be flush with the equivalent widget edge. The beginning and ending edge are based on the default rendering direction of the string in the Tip. Text rendered from left-to-right begins on the left side while text rendered right-to-left begins on the right side.

TABLE 3-2: Tip Positions versus String Direction

Tip Position	Left to Right	Right to Left
Bottom Beginning		
Bottom End		
Bottom Left		
Bottom Right		
Top Beginning		
Top End		
Top Left		
Top Right		

RESOURCE
xscTipXOffset
xscTipYOffset

A Tip's position can be adjusted with the `xscTipXOffset` and `xscTipYOffset` resources. The `xscTipXOffset` resource specifies the horizontal offset; likewise, the resource

xscTipYOffset specifies the vertical offset. These resource values are measured in pixels and can be positive, negative, or zero (unless otherwise indicated).

If the xscTipPosition resource is set to the value XmXSC_TIP_POSITION_POINTER, then the default value of the xscTipYOffset resource is 15. Furthermore, while in this mode the xscTipYOffset resource *cannot* be set to the value 0; if it is set to 0, the Help ToolKit will *quietly* set it to the default. If the xscTipPosition resource has any other value, then the default value of the xscTipYOffset is 0 and can freely be set to 0.

The default for the xscTipXOffset resource is always 0.

What is displayed?

RESOURCE
xscTipAutoDbReload

If the xscTipAutoDbReload resource is set to True, then the other Tip resources associated with the given widget are reloaded directly from the resource database each time the Tip is popped-up. If xscTipAutoDbReload is set to False, then the initial values are never automatically reread from the database. This can be used as a poor man's mechanism to dynamically change Tip resources.

RESOURCE
xscTipCompound

In some cases it may be desirable to have a collection of widgets behave like a single widget. For example, some widgets, such as the XmScale widget, are actually implemented as compound widgets containing one or more widget or gadget children. Normally, the Help ToolKit treats each child as a different widget instead of treating the XmScale as a singular widget.

The xscTipCompound resource is used to fix this problem. The default value is False. However, if it is set to True, then the associated widget is treated as if it contains no children.

CHAPTER 4

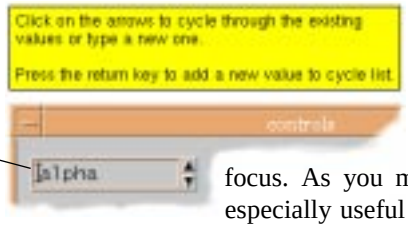
Cues

Cues are descriptive pop-up windows that appear above a window describing how to use the widget containing the keyboard focus.

Specifying the Cue Topic

RESOURCE
xscCueTopic

Regardless of what XscHelp resources are set, a given widget has an associated Cue if and only if the widget's `xscCueTopic` resource has a value. The value of the `xscCueTopic` resource is the string to display in the Cue.



This widget has the keyboard focus

Cues act like a mini-help system, providing helpful feedback to the user about the widget containing the keyboard focus. As you might imagine, Cues are especially useful for explaining syntax or validation requirements for text widgets (or other keyboard-centric widgets).

The Cue shown above could have been created with the resource entry shown in Code Listing 4-1.

SOURCE CODE 4-1: Specifying a Cue topic via resource file

```
*spinBox1*xscCueTopic: Click on the arrows to \
cycle through the existing\nvalues or type a new \
one.\n\nPress the return key to add a new value \
to cycle list.

*spinBox1*xscCueMarginWidth:  4
*spinBox1*xscCueMarginHeight: 4
```

Margins are discussed in more detail later in this chapter

A Cue is displayed for a widget if it (or any of its ancestors up to an including the nearest `WMShell` widget) has a value for the `xscCueTopic` resource.

The other resources define how each Cue is rendered and how it behaves.

Rendering Options for Cue Text

CROSS-REFERENCE Refer to "Rendering Options for Tip Text" on page 12 for more information about text rendering options.

This section examines the different text rendering options available for Cues. The rendering options for Cues are nearly identical to the rendering options for Tips. As a result, this section is an abridged version of the discussion about Tips. You should be familiar with the Tip discussion to completely understand text rendering in Cues.

Text Alignment

RESOURCE xscCueAlignment

The `xscCueAlignment` resource specifies the alignment of the Cue text. The allowed values for this resource are the same as the standard Motif alignment resource:

- `XmALIGNMENT_BEGINNING` (default)
- `XmALIGNMENT_CENTER`
- `XmALIGNMENT_END`

The default value is `XmALIGNMENT_BEGINNING`; this is in contrast to the equivalent default value for Tips.

Changing the Font

RESOURCE xscCueFontList

The font list for Cue text is set with the `xscCueFontList` resource.

Default String Direction

RESOURCE xscCueStringDirection

The default string direction, which is left-to-right, can also be specified with the `xscCueStringDirection` resource. The valid values are:

- `XmSTRING_DIRECTION_L_TO_R` (default)
- `XmSTRING_DIRECTION_R_TO_L`.

The string direction resource impacts the position of the Cue. Notice that the Cue appears flush with the right edge of the window; that is because the `xscCueStringDirection` resource is set to `XmSTRING_DIRECTION_R_TO_L`, as shown in Code Listing 4-2.

SOURCE CODE 4-2: Changing the string direction for Cue text

```
*spinBox1*xscCueTopic: Click on the arrows to \
cycle through the existing\nvalues or type a new \
one.\n\nPress the return key to add a new value \
to cycle list.
*spinBox1*xscCueMarginWidth:  4
*spinBox1*xscCueMarginHeight: 4

*spinBox1.xscCueStringDirection: \
                          XmSTRING_DIRECTION_R_TO_L
```

This line is used to set the default string direction for the Cue text

CROSS-REFERENCE Refer to Table 3-1 on page 13 for examples using Tips.

As with Tips, the text alignment is also impacted by the string direction; strings rendered (by default) from right to left begin on the right side and end on the left side.

String Converters

RESOURCE xscCueStringConverter

A string converter for a Cue is specified with the `xscCue-StringConverter` resource. The allowed values are:

- `XmXSC_STRING_CONVERTER_STANDARD` (default)
- `XmXSC_STRING_CONVERTER_FONT_TAG`
- `XmXSC_STRING_CONVERTER_SEGMENTED`

Specifying Any Font from the FontList

By default, the first font in the list is used with strings specified in resource files.

RESOURCE xscTipFontListTag

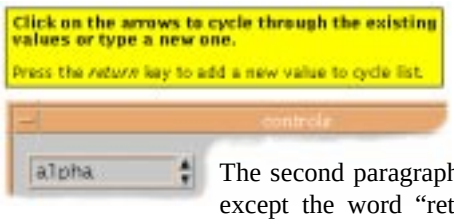
In order to explicitly specify an arbitrary font in the font list, the `xscCueStringConverter` resource is set to `XmXSC_STRING_CONVERTER_FONT_TAG` and the value of the `xscCueFontListTag` resource is set to the tag of the desired font.

Using Multiple Fonts and Directions

Multiple fonts can be used and the string direction can be changed by setting the `xscCueStringConverter` resource to `XmXSC_STRING_CONVERTER_SEGMENTED`.

The escape sequence "`@f`" indicates that a new font is going to be specified; the tag of the new font is enclosed in square brackets and should immediately follow the "`@f`". If no font tag is specified in the square brackets, then the default font is used. The default font is the first font in the font list.

The string direction can be altered by using the escape sequence "`@d`." followed immediately by a direction indicator — left-to-right is specified with the '`>`' character while right-to-left is specified with the '`<`' character.



Notice that this example uses three different fonts. The first paragraph is rendered in bold. The second paragraph uses a normal font except the word "return" is rendered in italics. The resource values used to create this example are shown in Code Listing 4-3.

SOURCE CODE 4-3: Using multiple fonts in a Cue

```
*spinBox1*xscCueTopic: @f[bold]Click on the \
arrows to cycle through the existing\n\
values or type a new one.@f[]\n\n\
Press the @f[italic]return@f[] key to add a new \
value to cycle list.

*spinBox1*xscCueFontList: \
   lucidasans-12=norm,\
   lucidasans-bold-12=bold, \
   lucidasans-italic-12=italic

*spinBox1*xscCueStringConverter: \
   XmXSC_STRING_CONVERTER_SEGMENTED

*spinBox1*xscCueMarginWidth:  4
*spinBox1*xscCueMarginHeight: 4
```

Tip Colors and Borders

By default, Cues have black text, a one pixel wide black border, and a yellow background. Although not visible in the default configuration, Cues can also be displayed with a traditional Motif shadow border.

Background and Border

RESOURCE
xscCueBackground
xscCueBorderColor
xscCueBorderWidth

The background color of a Cue can be specified by setting the `xscCueBackground` resource. Likewise, the border color and the border width are modified via the `xscCueBorderColor` and `xscCueBorderWidth` resources, respectively.

Shadow Borders

RESOURCE
xscCue-
ShadowThickness
xscCueShadowType

By default, the Cue shadow thickness is 0 pixels wide which results in no shadow border being rendered. If a shadow border is desired, the desired width is specified with the resource `xscCueShadowThickness` resource and the shadow style is specified with the `xscCueShadowType` resource.

All of the standard Motif shadow styles are supported; the allowed values are:

- `XmSHADOW_IN`
- `XmSHADOW_OUT` (default)
- `XmSHADOW_ETCHED_IN`
- `XmSHADOW_ETCHED_OUT`

Overriding Automatic Color Selections

RESOURCE
xscCueMotifColorModel

The colors of the Cue text and shadow border are normally derived automatically by the Help ToolKit. This behavior can be disabled by setting the value of the resource `xscCueMotifColorModel` to False.

RESOURCE
xscCueForeground

When the `xscCueMotifColorModel` is set to `False`, the foreground color for the Tip is explicitly set using the resource `xscCueForeground`.

RESOURCE
xscCueColorBase

The top and bottom shadow colors cannot be set independently. Instead, the `xscCueColorBase` resource is used to

specify a base color from which the top and bottom shadow colors are derived.

Cue Margins

RESOURCE
xscCueMarginBottom
xscCueMarginHeight
xscCueMarginLeft
xscCueMarginRight
xscCueMarginTop
xscCueMarginWidth

Cues in the Help Toolkit provide all of the margin resources supported by the Motif label widget. The margin resource supported are as follows:

- `xscCueMarginBottom`
- `xscCueMarginHeight`
- `xscCueMarginLeft`
- `xscCueMarginRight`
- `xscCueMarginTop`
- `xscCueMarginWidth`

CROSS-REFERENCE
Margins for Cues and Tips work the same. Refer to Figure 3-1 on page 19 to see how they work together.

All of the margin resources default to 0 except `xscCueMarginHeight` and `xscCueMarginWidth`. The `xscCueMarginHeight` resource defaults to 1 and the `xscCueMarginWidth` resource defaults to 2.

Cue Control Resources

Rather than specifying how a Cue is rendered, some of the Cue resources control the Cue's behavior. For instance, some resources specify where the Cue is actually displayed on the screen.

When does a Cue pop-up?

RESOURCE
xscCueEnabled

The `xscCueEnabled` resource is used to enable and disable the Cue for a specific widget (or gadget). If `xscCueEnabled` is set to `True`, then the Help ToolKit will display the Cue the moment the widget gains the keyboard focus. If `xscCueEnabled` is set to `False`, then the Cue is not displayed under any circumstance.

FUNCTION
XscHelpSetCues...
...EnabledGlobally()
...EnabledOnShell()

The `xscCueEnabled` resource can be overridden either globally or on a shell-by-shell basis. Using the function `XscHelpSetCuesEnabledOnShell()`, the programmer can prevent all Cues associated with widgets in a given shell

from being displayed. Likewise, the function XscHelpSetCuesEnabledGlobally() can be used to prevent all Cues in the application from being displayed. For a given Cue to display, the global, shell, and widget "cue enable" resources must all be set to True.

Where does a Cue pop-up?

RESOURCE
xscCuePosition

A Cue's location is controlled via by xscCuePosition resource. The allowed values for this resource are:

- XmXSC_CUE_POSITION_BOTTOM_BEGINNING
- XmXSC_CUE_POSITION_BOTTOM_END
- XmXSC_CUE_POSITION_BOTTOM_LEFT
- XmXSC_CUE_POSITION_BOTTOM_RIGHT
- XmXSC_CUE_POSITION_SHELL (default)
- XmXSC_CUE_POSITION_TOP_BEGINNING
- XmXSC_CUE_POSITION_TOP_END
- XmXSC_CUE_POSITION_TOP_LEFT
- XmXSC_CUE_POSITION_TOP_RIGHT

The default value XmXSC_CUE_POSITION_SHELL is used to position the Tip above the shell widget containing the Cue's widget. The other value are used to position the Cue immediately above or below the widget with one edge of the Cue flush with the edge of the widget.

When the value is XmXSC_CUE_POSITION_SHELL, the beginning edge of the Cue is even with the edge of the shell widget. The Cue is flush with the left or right edge of the shell based on the Cue's default string direction.

The Cue position values containing the token "BOTTOM" cause the Cue to appear immediately below the widget; values containing the token "TOP" cause the Cue to appear immediately above the widget.

The values containing the token "LEFT" cause the left edge of the Cue to be flush with the left edge of the widget. Likewise, values containing the token "RIGHT" cause the right edge of the Cue to be flush with the right edge of the widget.

The values containing the token "BEGINNING" cause the beginning edge of the Cue to be flush with the equivalent widget edge. Similarly, the values containing the token "END" cause the ending edge of the Cue to be flush with the equivalent widget edge. The beginning and ending edge are based on the default rendering direction of the string in the Cue. Text rendered from left-to-right begins on the left side while text rendered right-to-left begins on the right side.

RESOURCE
xscCueXOffset
xscCueYOffset

A Cue's position can be adjusted with the `xscCueXOffset` and `xscCueYOffset` resources. These resource values are measured in pixels and can be positive, negative, or zero.

If the `xscCuePosition` resource is set to the value `XmXSC_CUE_POSITION_SHELL`, then the default value of the `xscCueYOffset` resource is -10, which provides a 10 pixel gap between the bottom of the Cue and the top of the shell. If the `xscCuePosition` resource has any other value, then the default value of the `xscCueYOffset` is 0. The default for the `xscCueXOffset` resource is always 0.

What is displayed?

RESOURCE
xscCueAutoDbReload

If the `xscCueAutoDbReload` resource is set to True, then the other Cue resources associated with the given widget are reloaded directly from the resource database each time the Cue is displayed.

CHAPTER 5

Hints

Hints are brief textual messages that appear in a fixed area of a window describing the function of a widget. The Hint text is rendered based on resource values assigned to the widget (or gadget) located under the mouse cursor. Frequently, the Hint display area is located in the lower left region of the window in a status area; however, the Hints can be displayed in any portion of the window since it is positioned by the application developer.

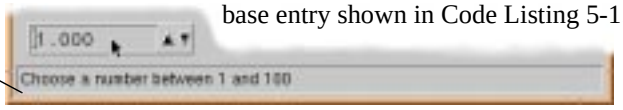
Specifying the Hint Topic

RESOURCE
xscHintTopic

Regardless of what Hint resources are set, a given widget has an associated Hint if and only if the widget's `xscHintTopic` resource is set. The value of this resource is the string to display in the Hint display area.

For instance, this Hint was generated with the resource database entry shown in Code Listing 5-1.

By default, a Hint always has the same background color as its parent



SOURCE CODE 5-1: Specifying a Hint topic in a resource file

```
*spinBox5.xscHintTopic: \
   Choose a number between 1 and 100
```

The other Hint resources define how each Hint is rendered and how it behaves.

Installing Hint Support

FUNCTION
XscHelpHintInstall

Unlike Tips and Cues which are automatically functional when the Help ToolKit is installed, Hints cannot be displayed until a display area is explicitly installed on a shell. Hint support is added to a window[1] by calling `XscHelpHintInstall()`. The only parameter to the function specifies the widget that should be used to rendering the Hint text within that shell; the specified widget is created and positioned by the programmer.

Caution
Care must be taken to configure the Hint display area widget so that it does not render its own text or graphics.

Any *widget* can be used for the Hint display; gadgets, however, cannot be used. The Help ToolKit does not override the normal rendering for the specified widget, so care must be taken to use a widget that does not render text or graphics. For instance, if a label widget is used, the string specified for the `XmNlabelString` resource is rendered along with the Hint text; this usually leads to a big mess. Therefore, if a Motif `XmLabel` widget is used, set the `XmNlabelString` resource to a null string, as shown in Code Listing 5-2.

SOURCE CODE 5-2: Using an XmLabel as the Hint display widget

The Hint display area widget is an XmLabel that displays an empty string.

This line installs the specified widget as the Hint for its own shell.

```
...
compound_string = XmStringCreateLocalized( "" );
hint = XtVaCreateManagedWidget(
   "hint", xmLabelWidgetClass, hint_frame,
   XmNlabelString, compound_string,
   NULL );
XmStringFree( compound_string );

XscHelpHintInstall( hint );
...
```

`XmLabel` widgets (with an `XmNlabelString` set to a null string) work well as the Hint display widget because the height easily can be set by specifying the largest Hint font as the default font in the label widget's `XmNfontList` resource. However, if the height can be set or controlled programmatically, then the `Core` widget provided by the Intrinsics can be used as a nice, lightweight Hint widget.

1. In this context, the term "window" refers to the generic WMShell and all of its non-WMShell descendants. It does not refer to an actual X window.

If a shadow border is desired, use an XmFrame widget as the parent of the Hint display widget even if the Hint display widget supports shadow borders; unlike the shadow border drawn by the Hint display widget, the XmFrame's shadow border cannot be overwritten by the Help ToolKit' rendering algorithm.

Rendering Options for Hint Text

CROSS-REFERENCE
Refer to "Rendering Options for Tip Text" on page 12 for more information about text rendering options.

Hints have the same text rendering options as Tips; therefore, Hint text rendering options are discussed only briefly in this chapter. You should be familiar with the Tip discussion to completely understand text rendering in Hints.

Text Alignment

RESOURCE
xscHintAlignment

The xscHintAlignment resource specifies the alignment of Hint text. The allowed values for this resource are the same as the standard Motif alignment resource:

- XmALIGNMENT_BEGINNING (default)
- XmALIGNMENT_CENTER
- XmALIGNMENT_END

The default value is XmALIGNMENT_BEGINNING; this is in contrast to the equivalent default value for Tips.

Changing the Font

RESOURCE
xscHintFontList

The font list used when rendering Hint text is specified with the xscHintFontList resource.

Default String Direction

RESOURCE
xscHintStringDirection

The default string direction can also be modified. By default, the string direction is assumed to be left-to-right. However, the string direction can be specified via the xscHintStringDirection resource. The valid values are:

- XmSTRING_DIRECTION_L_TO_R (default)
- XmSTRING_DIRECTION_R_TO_L

String Converters

RESOURCE
xscHintStringConverter

String converters are specified with the `xscHintStringConverter` resource. The allowed values are:

- `XmXSC_STRING_CONVERTER_STANDARD` (default)
- `XmXSC_STRING_CONVERTER_FONT_TAG`
- `XmXSC_STRING_CONVERTER_SEGMENTED`

Specifying Any Font from the FontList

By default, the first font in the list is used with strings specified in resource files.

RESOURCE
xscHintFontListTag

In order to explicitly specify an arbitrary font in the font list, the `xscHintStringConverter` resource is set to `XmXSC_STRING_CONVERTER_FONT_TAG` and the value of the `xscHintFontListTag` resource is set to the tag of the desired font.

Using Multiple Fonts and Directions

Multiple fonts can be used and the string direction can be changed by setting the `xscHintStringConverter` resource to `XmXSC_STRING_CONVERTER_SEGMENTED`.

The escape sequence "`@f`" indicates that a new font is going to be specified; the tag of the new font is enclosed in square brackets and should immediately follow the "`@f`". If no font tag is specified in the square brackets, then the default font is used. The default font is the first font in the font list.

The string direction can be altered by using the escape sequence "`@d`." followed immediately by a direction indicator — left-to-right is specified with the '`>`' character while right-to-left is specified with the '`<`' character.

The resource values used to create this example are shown in Code Listing 5-3. Notice that the word "between" is rendered in italics and the numbers "1" and "100" are rendered in bold.

SOURCE CODE 5-3: Using multiple fonts in a Cue

```
*spinBox5.xscHintTopic: Choose a number \
@f[italic]between@f[] @f[bold]1@f[] and \
@f[bold]100@f[]

*spinBox5*xscHintFontList: \
   lucidasans-12=norm, lucidasans-bold-12=bold, \
   lucidasans-italic-12=italic
*spinBox5*xscHintStringConverter: \
   XmXSC_STRING_CONVERTER_SEGMENTED
```

Colors

RESOURCE xscHint-InheritBackground

By default, the `xscHintInheritBackground` resource is set to True; this causes the background color of the Hint display area to be the same as the Hint display widget's parent. The background color is derived each time the Hint is rendered, so if the parent's background color changes, the Hint's background color changes as well.

RESOURCE xscHintBackground

If the `xscHintInheritBackground` resource is set to False, then the background color for the Hint display widget is derived from the value of the `xscHintBackground` resource. Since each widget (and gadget) in the application can have its own Hint value, the background color of the Hint display widget can change as different Hints are displayed.

RESOURCE xscHintMotifColorModel

RESOURCE xscHintForeground

By default, the foreground color of the Hint display widget is derived automatically to obtain reasonable contrast. To use a specific foreground color, the `xscHintMotifColorModel` resource must be set to False and the desired text color must be specified with the `xscHintForeground` resource.

The Hint in this example is rendered with a foreground of Red3 and a background of Yellow as shown in Code Listing 5-4.

SOURCE CODE 5-4: Foreground and background colors for a Hint

```
*spinBox5.xscHintTopic: \
   Choose a number between 1 and 100
*spinBox5.xscHintInheritBackground: False
*spinBox5.xscHintBackground:        Red3
*spinBox5.xscHintForeground:        Yellow
*spinBox5.xscHintMotifColorModel:   False
```

Margins

RESOURCE
xscHintMarginLeft
xscHintMarginRight
xscHintMarginWidth

Hints support all of the horizontal margin resources supported by the Motif label widget:

- `xscHintMarginLeft`
- `xscHintMarginRight`
- `xscHintMarginWidth`

The `xscHintMarginLeft` and `xscHintMarginRight` resources have a default value of 0. The `xscHintMarginWidth` resource has a default value of 2. There are no resource for vertical margins because the Hint text is always centered vertically in the Hint display widget.

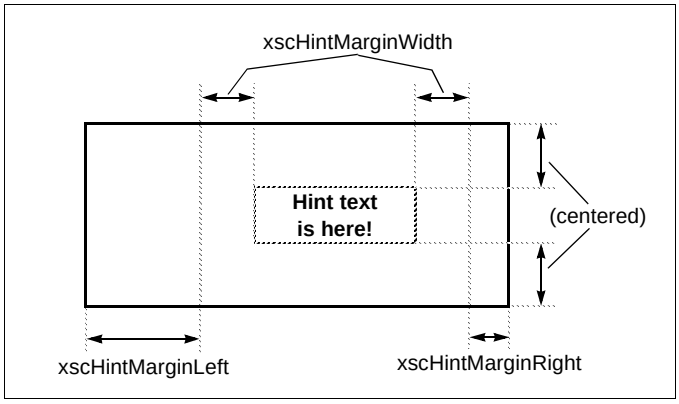


FIGURE 5-1: Hint margin resources

Control

As with Tips, there are some Hint resources that specify how Hints are controlled by the Help ToolKit.

RESOURCE
xscHintEnabled

The `xscHintEnabled` resource is used to enable and disable the Hint for a specific widget (or gadget). If value of the resource `xscHintEnabled` is set to True, then the Help ToolKit will display the Hint when the mouse cursor is in the widget; however, if `xscHintEnabled` is set to False, then the Hint is not displayed under any circumstance.

FUNCTION
XscHelpSetHints...
...EnabledGlobally()
...EnabledOnShell()

The `xscHintEnabled` resource can be overridden either globally or on a shell-by-shell basis. Using the function `XscHelpSetHintsEnabledOnShell()`, the programmer can prevent all Hints associated with widgets in a given shell from being displayed. Likewise, the function `XscHelpSetHintsEnabledGlobally()` can be used to prevent all Hints in the application from being displayed. For a given Hint to display, the global, shell, and widget "hint enable" resources must all be set to `True`.

RESOURCE
xscHintAutoDbReload

If the `xscHintAutoDbReload` resource is set to True, then the other Hint resources associated with the given widget are reloaded directly from the resource database each time the Hint is rendered. If `xscHintAutoDbReload` is set to False, then the initial values are never automatically reread from the database. This resource can be used as a poor man's mechanism to dynamically change Hint resources.

RESOURCE
xscHintCompound

In some cases it may be desirable to have a collection of widgets behave like a single primitive widget. For example, some widgets, such as the `XmScale` widget, are actually implemented as compound widgets containing one or more widget or gadget children. Normally, the Help ToolKit treats each child as a different widget instead of treating the `XmScale` as a singular widget.

The `xscHintCompound` resource is used to fix this problem. By default, the `xscHintCompound` resource is set to False. However, if it is set to True, then the associated widget is treated as if it contains no children.

CHAPTER 6

Context-Help

The Help ToolKit does not contain a built-in on-line context-sensitive help facility. Instead, XscHelp provides an infrastructure that allows a Motif application to be easily integrated with virtually any help facility designed for the X Window System. Once integrated, the help system can be completely configured through resource files.

In this manual, the terms Context-Help and context-sensitive help are used interchangeably

The Help ToolKit provides support for the CDE help system, so if CDE is your target environment, then no additional programming is required. The CDE help support is described in Chapter 7 starting on page 49. For other help systems, typically only a single function must be written to seamlessly bind the help facility to the application.

Installing Context-Help Support

FUNCTION
XscHelpContextInstall

In order to use the context-sensitive help capabilities of the Help ToolKit, the function `XscHelpContextInstall()` must be called once; ideally this function should be called before or immediately after the call to `XscHelpInstall()`.

The `XscHelpContextInstall()` function performs two basic functions: it records the resources that should be retrieved for a widget when context-sensitive help is requested and it registers the context-help application callback function. The callback function is the glue that connects the application to the help system.

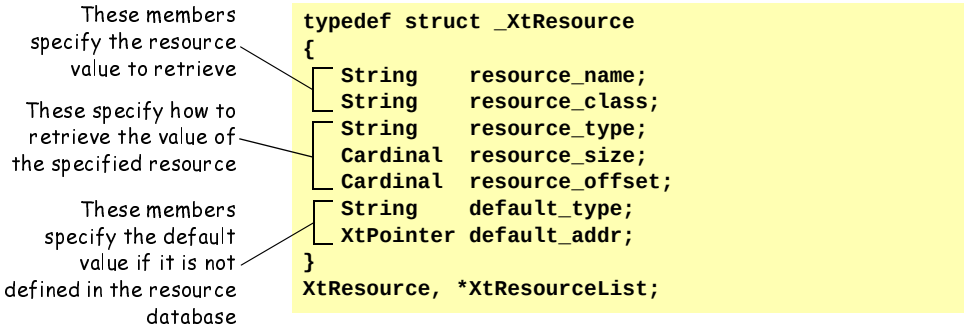
The examples in this chapter show how a simple interface to the CDE help facility could be set-up.

Specifying What to Extract from the Resource Database

The Help ToolKit uses the `XtResourceList` data type to define the data values to retrieve from the resource database when context-sensitive help is requested. By definition, an `XtResourceList` is an array of `XtResource` data types.

The `XtResource` data type is defined by the Intrinsics and is normally used to specify and retrieve widget and application resources from the resource database. The structure can be broken into three parts. The first part identifies how to find the desired resource. The second part specifies how to retrieve the resource. The third part specifies the value to use if the resource is not found. The definition of the `XtResource` structure is shown in Code Listing 6-1.

SOURCE CODE 6-1: Definition of the data type `XtResource`




```
typedef struct _XtResource
{
    String    resource_name;
    String    resource_class;
    String    resource_type;
    Cardinal  resource_size;
    Cardinal  resource_offset;
    String    default_type;
    XtPointer default_addr;
}
XtResource, *XtResourceList;
```

The data type `XtResource` is often used with the functions `XtGetSubresources()` and `XtGetApplicationResources()`. Both of these functions accept an array of `XtResource` types, one for each resource value to retrieve.

The members `resource_name` and `resource_class` are used to locate the desired resource. The `resource_name` is the proper name of the resource; for instance, many widget resource names are defined in the Intrinsics with constants beginning with "XtN". The 'N' stands for *name*. Likewise, the resource *class* is often defined by a constant starting with an "XtC". The name and class are assumed to be relative to a widget which is provided to the lookup function.

The member `resource_type` specifies the data type of the desired resource value. Generally, the value of this member

can be chosen from the list of constants beginning with "XtR" that are defined by the Intrinsics. The 'R' in "XtR" stands for *representation type*. If the desired type is an `int`, the constant `XtRInt` would be used; likewise, if the desired type is a string, the constant `XtRString` would be used.

The `resource_size` member specifies the number of bytes provided to store the retrieved data. The `resource_offset` specifies the byte offset from some base address where the retrieved data type should be stored. The base address is provided to the lookup function.

The XtResource data type can be difficult to understand at first; if you are confused about it, take a look a Volume 4 and 5 in the O'Reilly series on the X Window System.

If the desired resource value is not defined in the resource database, the `default_type` and `default_addr` members are used to supply a default value. The `default_type` is the representation type of the provided default value, while `default_addr` is the actual default value.

An array of `XtResource` data structures can be used to retrieve an arbitrary set of values associated with a widget. It is important to understand that the widget does not need to actually define or use any of these resources; they simply exist in the resource database and are associated with the widget.

For instance, a simple CDE help interface could require the help volume, the location identifier, and the number of columns to display. The data structure shown in Code Listing 6-2 can be used to maintain the values. Each time context-sensitive help is requested for a widget, the Help ToolKit needs to fill this data structure with the desired context-help values associated with the widget. After the values are retrieved and stored, the data structure should be passed to the application through a callback. The callback function, which is supplied by the application programmer, would then pass the values to the CDE help system.

SOURCE CODE 6-2: Example context-sensitive help data structure

```
typedef struct
{
   String    volume;
   Dimension columns;
   String    loc_id;
}
CdeDataStruct;
```

After the data structure used to carry the context-help data is defined, an array of `XtResource` data types must be created; this array is used by the Help ToolKit to find and load the correct context-help values. Code Listing 6-3 shows how this could be done for this simple CDE example.

The first element of the array *must* represent a pointer with a default value of `NULL`. If the target widget does not have the first resource defined in the resource database, then it is assumed that no help is defined for the widget and the widget's parent is checked instead. If the parent does not have any help defined, then its parent is checked. This will repeat until help data is obtained or a `Toplevel` shell is checked without success.

SOURCE CODE 6-3: Setting up an XtResource array

```
#  define OFFSET_OF( mem ) \
      XtOffsetOf( CdeDataStruct, mem )

   static XtResource resource[] =
   {
      {
         DtNhelpVolume, DtCHelpVolume,
         XmRString, sizeof( String ),
            OFFSET_OF( volume ),
         XtRImmediate, NULL
      },{
         DtNcolumns, DtCColumns,
         XmRDimension, sizeof( Dimension ),
            OFFSET_OF( columns ),
         XtRImmediate, (XtPointer) (Dimension) 70
      },{
         DtNlocationId, DtCLocationId,
         XmRString, sizeof( String ),
            OFFSET_OF( loc_id ),
         XtRImmediate, NULL
      }
   };
#  undef OFFSET_OF
```

Finally, the Help ToolKit must be instructed how to obtain all of this context-sensitive help information; this is done via the `XscHelpContextInstall()` function. An example of this function in use is shown in Code Listing 6-4.

The first argument to the `XscHelpContextInstall()` function is the array of `XtResource` used to extract the cor-

SOURCE CODE 6-4: Using `XscHelpContextInstall()`

```
...
   XscHelpInstall( shell );

   XscHelpContextInstall(
      resource, XtNumber( resource ),
      sizeof( CdeDataStruct ),
      CdeHelpCB, NULL );
...
```

rect values from the resource database; the second parameter is the number of elements in the array. The third parameter is the size of the actual data structure used to store the context-help values.

Finally, the fourth parameter is the callback function to call when context-help is requested and the fifth parameter is arbitrary client data that is passed to the callback.

The Context-Sensitive Help Callback

The Context-Help callback has the same function signature as widget callbacks: the first parameter is the widget associated with the help data, the second parameter is the client data, and the third parameter is callback specific data.

The callback specific data is a structure of type `XscHelpContextCallbackStruct`; the definition of this structure is shown in Code Listing 6-5.

SOURCE CODE 6-5: Definition of `XscHelpContextCallbackStruct`

```
typedef struct
{
   int       reason;
   XEvent*   event;
   int       depth;
   XtPointer data;
}
XscHelpContextCallbackStruct
```

The data field points to the data structure that contains all of the desired context-sensitive help data

The *reason* field indicates why the callback was invoked. The possible values are:

- `XmCR_XSC_HELP_CONTEXT_CALLBACK`
- `XmCR_XSC_HELP_CONTEXT_GRAB_SELECT`

The value XmCR_XSC_HELP_CONTEXT_CALLBACK is used to indicate that the standard Motif help callback was used to request context-sensitive help; this happens typically when the user presses the Help or F1 key.

CROSS-REFERENCE
Refer to "Picking Help with the Mouse Cursor" on page 48

The XmCR_XSC_HELP_CONTEXT_GRAB_SELECT reason is used when the widget is selected by the user through the function XscHelpContextPickAndActivate(). This function is similar to XmTrackingEvent() except after the widget is selected the help callback is automatically invoked.

The *event* field contains a pointer to the X event that caused the callback; the event field may contain a NULL pointer.

The *depth* field indicates the level in the widget hierarchy that the help was obtained; a value of 0 indicates that the help was defined for the target widget, a value of 1 indicates that the help was found with the target's parent, etc.

The *data* field is the application defined data structure used to store the help data. If no context-sensitive help could be found after the ancestor widgets are checked, then the *data* field is set to NULL.

Code Listing 6-6 show an example of a simple context-sensitive help callback. If the CDE help widget does not exist, then it is created. The help values provided in the callback data are given to the help widget using XtSetValues() and the widget is managed.

Defining Context Help Resources

The context help resources can be associated in the resource database with any widget or gadget. The resource file entries shown in Code Listing 6-7 would load the help volume "HelpToolKit-Demo", flip to topic "style", and adjust the width of the CDE help widget to view a maximum of 60 characters per line.

SOURCE CODE 6-6: Simple context help callback example

The help widget and its parent are declared as global variables for easy access. This is not a good idea in practise, but works for this simple example

Create the help widget if it does not already exist

Configure the help widget to correctly display the desired help information

```
Widget help_widget;
Widget help_parent;

static void CdeHelpCB(
   Widget obj, XtPointer cd, XtPointer cbd )
{
   XscHelpContextCallbackStruct* cb_data;
   CdeDataStruct*                cde_data;
   Arg                           argv[ 3 ];
   Cardinal                      argc = 0;

   cb_data = (XscHelpContextCallbackStruct*) cbd;
   cde_data = (CdeDataStruct*) cb_data->data;

   if (!help_widget)
   {
      help_widget = DtCreateHelpDialog(
         help_parent, "Help", NULL, 0 );
   }

   XtSetArg(
      argv[argc], DtNcolumns, cde_data->columns );
   argc++;

   XtSetArg(
      argv[argc],DtNhelpVolume,cde_data->volume );
   argc++;

   if (cde_data->location_id)
   {
      XtSetArg(
         argv[ argc ],
         DtNlocationId,
         cde_data->loc_id );
      argc++;
   }
   XtSetValues( help_widget, argv, argc );

   XtManageChild( help_widget );
}
```

SOURCE CODE 6-7: Example of context-sensitive help resource entries

```
*style.helpVolume: HelpToolKit-Demo
*style.locationId: style
*style.columns:    60
```

Picking Help with the Mouse Cursor

FUNCTION
xscHelpContextPick-AndActivate

The `XscHelpContextPickAndActivate()` function allows the user to pick a widget or gadget with the mouse cursor. After a widget is selected, the Help ToolKit help callback function is automatically called with the help data associated with the selected widget. If the `ESC` key is pressed before a widget is selected, then the function is canceled.

The function `XscHelpContextPickAndActivate()` takes three arguments. The first is a widget that contains the set of selectable widgets; a `Toplevel` shell is normally used for this parameter. The second parameter specifies the cursor to use. The third parameter indicates if the mouse cursor should be confined to the widget specified in the first parameter.

CHAPTER 7

CDE Context-Help

Context-sensitive help for CDE is provided automatically with the Help ToolKit. The CDE capability cannot actually be used unless the target environment is CDE compliant.

This manual does not attempt to describe the CDE Help System; you should refer to a CDE Help System programmer's guide for details about the CDE Help System and widgets.

Once the CDE Context-Help support is installed in an application built with the CDE object libraries, all of the application to help system coordination is handled by the Help ToolKit; in addition, all the help information needed for each widget can be configured via the resource database.

Each Toplevel shell in the application is associated with a single general and a single quick help CDE widget; a resource associated with each widget determines which type of help widget is used. The first time a help topic is requested within a given Toplevel shell, the appropriate help widget is created. This widget is reused for other requests within the given Toplevel shell.

Installing CDE Context-Help

FUNCTION
XscCdeHelpInstall

To install XscHelp CDE Context-Help, the application must include the file `<Xsc/CdeHelp.h>` and call the function `XscCdeHelpInstall()` instead of the standard XscHelp installation function `XscHelpInstall()`.

Code Listing 7-1 shows how to install XscHelp with CDE support.

The first argument to the `XscCdeHelpInstall()` function is the first shell widget created by the application — the same as the parameter specified with the standard install function `XscHelpInstall()`.

SOURCE CODE 7-1: Initializing with the CDE Context-Help support

The CDE Context-Help support is declared in the header file <Xsc/CdeHelp.h>

When installing CDE Context-Help, XscCdeHelpInstall() is called instead of XscHelpInstall() and XscHelpContext-Install().

```
...
#include <Xsc/CdeHelp.h>
...
int main( int argc, char** argv )
{
...
   toplevel = XtAppInitialize( &appContext,
      ApplicationClass, NULL, 0, &argc, argv,
      NULL, NULL, 0 );

   XscCdeHelpInstall( toplevel, NULL );
...
```

However, `XscCdeHelpInstall()` has a second parameter that specifies the name of the default help volume. If this parameter is `NULL`, then the default help volume name is retrieved from the `XSC_CDE_HELP_VOLUME` environment variable. If the environment variable is undefined, then the volume name must be specified via the resource database, otherwise the CDE Help System will not be able to locate the requested help text.

A Brief Look at the CDE Help System

The on-line text information associated with an application is normally completely contained within a help volume; the help volume, which is composed of a run-time help file and (possibly) multiple graphics files, is normally distributed and installed into the CDE environment with the application.

The simplest way to view the contents of a help volume is via the help viewer widgets supplied with CDE. There are two types of help viewers: general help and quick help. The general help widget has a menu bar, a topic tree, and a topic viewing area. The quick help widget is designed to display brief, self-contained help topics; since there is no topic tree, only the topic display area and one or more buttons are displayed.

The CDE help widgets are controlled using the function `XtVaSetValues()` or equivalent. Both widgets are controlled with the same resource.

The widgets can display five different types of help. The help widgets are placed in a *help mode* with the DtNhelpType resource. The allowed values of this resource are:

- DtHELP_TYPE_TOPIC
- DtHELP_TYPE_STRING
- DtHELP_TYPE_DYNAMIC_STRING
- DtHELP_TYPE_FILE
- DtHELP_TYPE_MAN_PAGE

DtHELP_TYPE_TOPIC displays help information from the application's help volume. The help volume is specified with the DtNhelpVolume resource; the specific entry to view from the help volume is specified with the DtNlocationId resource.

DtHELP_TYPE_STRING displays a string without any formatting. The string to display is retrieved from the DtNstringData resource. New lines in the string must be explicitly marked by a newline character.

DtHELP_TYPE_DYNAMIC_STRING displays a string with simple formatting — text is automatically word-wrapped and newline characters represent paragraph breaks. The string to display is specified with the DtNstringData resource.

DtHELP_TYPE_FILE displays the contents of a text file. The name of the text file must be specified in the DtNhelp-File resource.

DtHELP_TYPE_MAN_PAGE formats and displays an installed man page. The name of the man page to display is specified via the DtNmanPage resource.

Help ToolKit CDE Context-Help Resources

RESOURCE
xscCdeHelpTopic

The xscCdeHelpTopic resource is used to specify the data that will be displayed. If this resource is not defined or is set to NULL, then, by definition, no help data is defined for the widget. However, even though no context-help is defined for the widget, help could still be displayed if an ancestor had a non-NULL xscCdeHelpTopic resource.

If context-help is requested for a widget that does not have an `xscCdeHelpTopic` resource, then a search is made of its ancestors (starting at the most recent) until an `xscCdeHelpTopic` resource value is found. If no help is found, then the request is ignored. The search moves from parent to parent until either an `xscCdeHelpTopic` is found or a `WMShell` widget is reached.

Interpreting the Topic

RESOURCE
xscCdeHelpType

The `xscCdeHelpTopic` resource is interpreted several different ways based on the value of the `xscCdeHelpType` resource; this resource sets the mode for the CDE widget. The allowed values are the same as for the `DtNhelpType` resource, as discussed in the preceding section.

`DtHELP_TYPE_TOPIC` — The `xscCdeHelpTopic` resource is interpreted as a location identifier in the application's help volume. The name of the help volume is specified using the resource `xscCdeHelpVolume`; the default value for this resource is the value of the second parameter to the `XscCdeHelpInstall()` function. If the second parameter is equal to NULL, then the default help volume name is derived from the `XSC_CDE_HELP_VOLUME` environment variable. The example shown in Figure 7-1 contains a CDE general help widget with a missing help volume. The resource file entries for this example are shown below in Code Listing 7-2.

`DtHELP_TYPE_STRING` — The `XscCdeHelpTopic` resource is interpreted as a literal string for display without any implicit formatting. Line breaks must be physically specified in the string; if a line is too long to display, a horizontal scroll bar appears to handle horizontal scrolling. An example of this mode is shown in Figure 7-2 and the associated resource file is displayed in Code Listing 7-3.

`DtHELP_TYPE_DYNAMIC_STRING` — In contrast, dynamic strings word-wrap so that a horizontal scroll bar is never displayed. In this case, a single newline character represents a paragraph break. An example of a dynamic string is shown in Figure 7-3; the resource file entries for the example are shown in Code Listing 7-4.

CROSS-REFERENCE
The resource file for this example is shown in Code Listing 7-2.

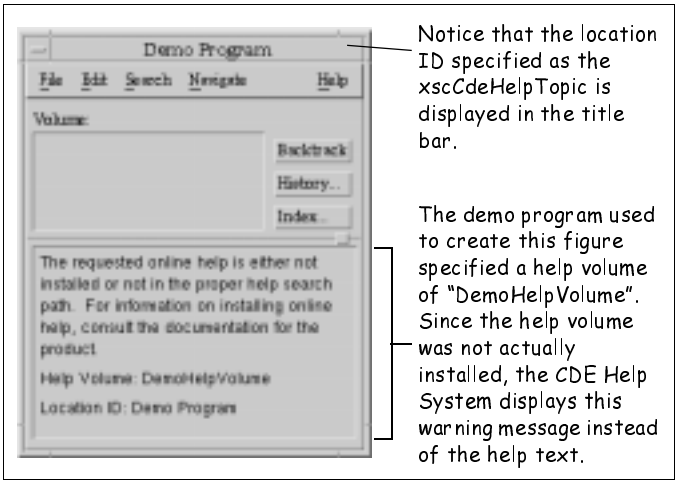


FIGURE 7-1: CDE help with type `HELP_TYPE_TOPIC`

SOURCE CODE 7-2: CDE help with type `HELP_TYPE_TOPIC`

```
*showTopic.xscCdeHelpType:   HELP_TYPE_TOPIC
*showTopic.xscCdeHelpTopic:  DemoProgram
*showTopic.xscCdeHelpVolume: DemoHelpVolume
```

CROSS-REFERENCE
The resource file for this example is shown in Code Listing 7-3.

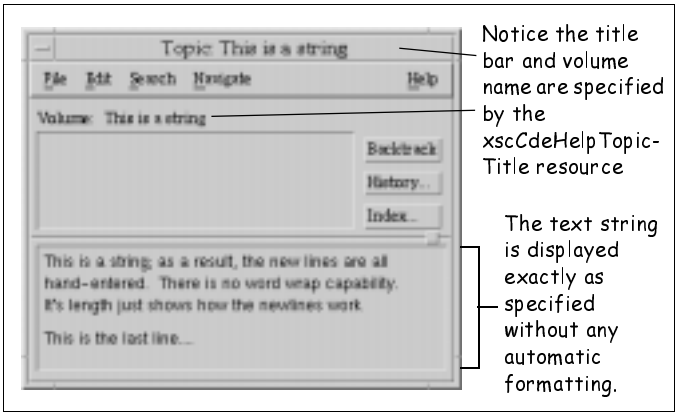


FIGURE 7-2: CDE help with type `HELP_TYPE_STRING`

`DtHELP_TYPE_FILE` — If the `xscCdeHelpType` is set to `DtHELP_TYPE_FILE`, then the `xscCdeHelpTopic` resource specifies the name of the file to display in the help widget. If the file is not in the application's current working directory, then a full pathname must be specified. Figure 7-4

SOURCE CODE 7-3: CDE help with type `HELP_TYPE_STRING`

```
*showStr.xscCdeHelpType:   HELP_TYPE_STRING
*showStr.xscCdeHelpTopic:      \
This is a string; as a result, \
the new lines are all \n\
hand-entered.  There is no word wrap capability.\n\
It's length just shows how the newlines work \n\n\
This is the last line....
*showStr.xscCdeHelpTopicTitle: This is a string
```

CROSS-REFERENCE
The resource file for this example is shown in Code Listing 7-4.

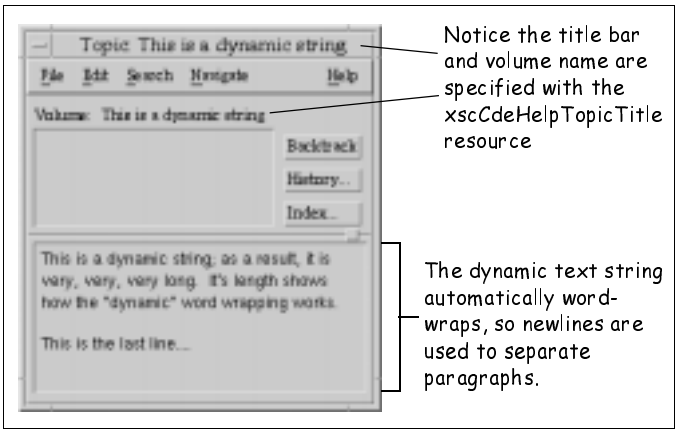


FIGURE 7-3: CDE help with type `HELP_TYPE_DYNAMIC_STRING`

SOURCE CODE 7-4: CDE help with type `HELP_TYPE_DYNAMIC_STRING`

```
*showDStr.xscCdeHelpType: HELP_TYPE_DYNAMIC_STRING
*showDStr.xscCdeHelpTopic:     \
This is a dynamic string; as a result, it \
is very, very, very long.  It's length \
shows how the "dynamic" word wrapping \
works.\n\nThis is the last line....
*showDStr.xscCdeHelpTopicTitle: \
This is a dynamic string
```

This double newline creates an empty paragraph between two paragraphs.

shows an example of this mode with the associated resource file shown in Code Listing 7-5.

SOURCE CODE 7-5: CDE help with type `HELP_TYPE_FILE`

```
*showFile.xscCdeHelpType:   HELP_TYPE_FILE
*showFile.xscCdeHelpTopic:  Demo
```

`DtHELP_TYPE_MAN_PAGE` — Finally, if the `xscCdeHelpType` is specified as `DtHELP_TYPE_MAN_PAGE`, then

CROSS-REFERENCE
The resource file for this example is shown in Code Listing 7-5.

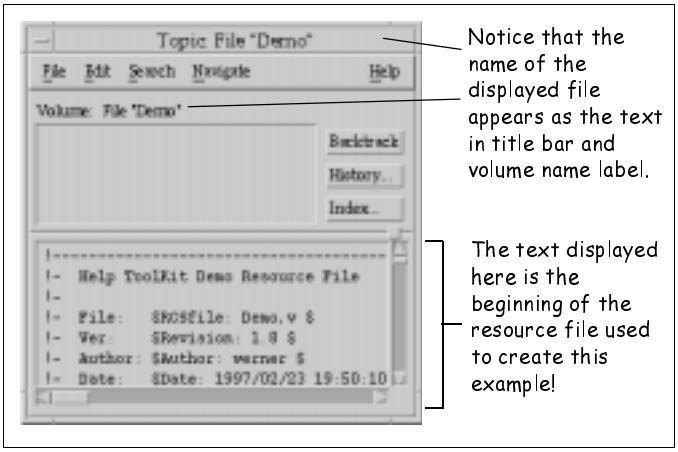


FIGURE 7-4: CDE help with type HELP_TYPE_FILE

the topic resource specifies the name of an installed man page to display. Figure 7-5 displays a *quick help* widget containing a man page for the program *man*; a portions of the resource file used to generate the example is shown in Code Listing 7-6.

CROSS-REFERENCE
The resource file for this example is shown in Code Listing 7-6.

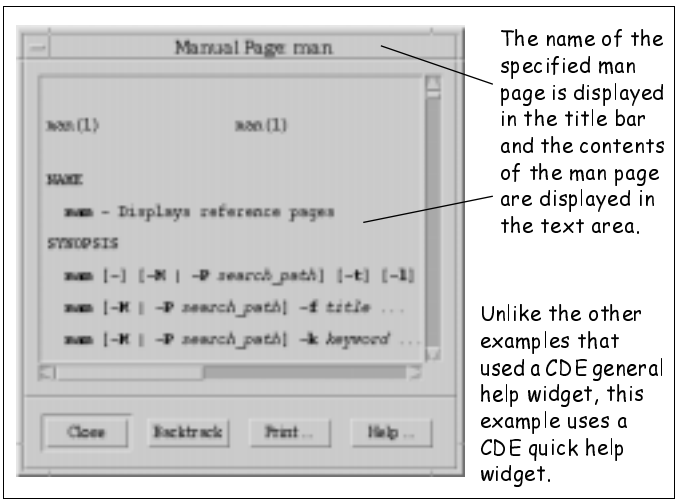


FIGURE 7-5: CDE help with type HELP_TYPE_MAN_PAGE

SOURCE CODE 7-6: CDE help with type HELP_TYPE_MAN_PAGE

```
*showMan.xscCdeHelpType:   HELP_TYPE_MAN_PAGE
*showMan.xscCdeHelpTopic:  man
```

Specifying a General or Quick Help Widget

RESOURCE
xscCdeHelpWidgetType

The `xscCdeHelpWidgetType` resource specifies the type of CDE widget to use when viewing the help information. The allowed values are

- `XmXSC_GENERAL_HELP_WIDGET`
- `XmXSC_QUICK_HELP_WIDGET`

The default value is `XmXSC_GENERAL_HELP_WIDGET`, which displays the full-featured CDE help widget. If the resource is set to `XmXSC_QUICK_HELP_WIDGET`, then the quick help widget is used instead.

Sizing the Help Widget

RESOURCE
xscCdeHelpColumns
xscCdeHelpRows

The size of the viewing area of the help widget can be specified with the `xscCdeHelpColumns` and `xscCdeHelpRows` resources.

The default value for the `xscCdeHelpColumns` is 70 characters. The number of lines visible is set with the `xscCdeHelpRows` resource; the default value is 25.

Controlling the Help Widget Titles

RESOURCE
xscCdeHelpTopicTitle

The name of the help topic is displayed in the help widget's volume field; it can be explicitly controlled by setting the `xscCdeHelpTopicTitle` resource to the desired string.

RESOURCE
xscCdeHelpDialogTitle

The dialog title displayed in the title bar, can also be explicitly set via the `xscCdeHelpDialogTitle` resource. If the dialog title resource is not set, then it is determined based on the value of the resource `xscCdeHelpType`.

If the value of the `xscCdeHelpType` resource is set to `DtHELP_TYPE_TOPIC`, then the default dialog title is the same as the name of the help volume, which is specified by the `xscCdeHelpVolume` resource.

If the value of the `xscCdeHelpType` resource is set to `DtHELP_TYPE_STRING`, then the default dialog title is "Topic: " followed by the value of the `xscCdeHelpTopicTitle` resource. This is also true if the help type resource is set to `DtHELP_TYPE_DYNAMIC_STRING`.

Likewise, the default for the dialog title when the resource `xscCdeHelpType` is set to `DtHELP_TYPE_FILE` is "Topic: " followed by the value of the topic title resource. However, if the `xscCdeHelpTopicTitle` resource is not specified, then the `xscCdeHelpDialogTitle` is set to the value "Topic: " followed by the name of the file being displayed.

Finally, if the `xscCdeHelpType` resource is set to the value `DtHELP_TYPE_MAN_PAGE`, then the default value of the dialog title is "Man Page: " followed by the name of the man page being displayed.

CHAPTER 8

Dynamic Updates

Although the Help ToolKit is highly configurable through resource files that are read as the application initializes, there is often a need to dynamically update Tip, Hint, Cues, and Context-Help.

Earlier chapters have eluded to a poor-man's approach to this problem. The idea is to dynamically update the resource database and update the various *help objects*[1] at run-time. Another approach is to use an API to directly manipulate help objects. Both of these approaches are discussed in this chapter.

Resource Database

Many developers and users think the resource database is synonymous with the files used to configure an X application during its initialization. The truth is that those flat, static *application-defaults* files are actually used to initialize the resource database. The resource database[2] can actually be quite dynamic. It is manipulated using the Resource Manager functions (Xrm) defined in Xlib.

This manual does not attempt to provide a complete description of the Resource Manager; a good discussion of the Resource Manager can be found in Volume 1 and 2 of the

1. Help object is a generic term for a Tip, Hint, or Cue. Although technically not an object, Context-Help is relevant to this discussion as well.s
2. A resource database is actually just a data structure; as a result, an application can maintain multiple resource databases simultaneously. In this discussion, the generic term "resource database" refers to the default resource database used to initialize widget and gadget resources.

X Window System documentation set published by O'Reilly & Associates, Inc. However, a brief discussion of some of the functions needed to update the resurce database is in order.

Getting a Reference to the Resource Database

FUNCTION
XrmGetDatabase

Each display has a different default resource database. In order to modify a default resource database, you must acquire a handle to it. The function `XrmGetDatabase()` is provided by Xlib to perform this service. The prototype for the function is shown below:

```
XrmDatabase XrmGetDatabase(
   Display*       display );
```

The function returns the default resource database associated with the specified display connection.

Modifying the Resource Database

There are many functions that can be used to modify a resource database. The functions referenced in this manual were chosen because they are easy to use, simple to understand, and similar to the conventional notation used in resource files.

FUNCTION
XrmPutLineResource

The first function is `XrmPutLineResource()`, which has the following prototype:

```
void XrmPutLineResource(
   XrmDatabase*   database,
   char*          line );
```

This function converts a string into an entry in a resource database. The resource database is specified with the *database* parameter — use the address of the value returned by `XrmGetDatabase()`. The *line* parameter is a string representing the resource value to add to the database; the string has the same syntax as a line used in a standard resource file. Code

Listing 8-1 shows how this function could be used to update the Tip topic for the widget “okIconButton” to the value “OK”.

SOURCE CODE 8-1: Using XrmPutLineResource()

```
...
extern Display* display;
XrmDatabase database;

database = XrmGetDatabase( display );

XrmPutLineResource(
   &database, “*okIconButton.xscTipTopic: OK” );
...
```

FUNCTION
XrmPutStringResource

The function `XrmPutStringResource()` is also used to update a resource database. It has the following prototype:

```
void XrmPutStringResource(
   XrmDatabase*   database,
   char*          specifier,
   char*          value );
```

This function specifies the resource entry and the resource value as two separate parameters; otherwise, it is quite similar to `XrmPutLineResource()`. Code Listing 8-2 shows how to update the Tip topic for the widget “okIconButton” to the value “OK”.

SOURCE CODE 8-2: Using XrmPutStringResource()

```
...
extern Display* display;
XrmDatabase database;

database = XrmGetDatabase( display );

XrmPutStringResource(
   &database, “*okIconButton.xscTipTopic”, “OK” );
...
```

Manually Rereading the Database

FUNCTION
XscHelpDbReload

Once the resource database has been modified, the Help Tool-Kit must be notified that the resource for a given widget have changed and need to be reloaded; this is done with the function `XscHelpDbReload()`. The only parameter to the function if the widget or gadget that needs to be updated.

FUNCTION
XscHelpUpdate

After the new resource values are loaded, they are used the next time the help object (Tip, Hint, etc.) is used; if it is currently displayed when `XscHelpDbReload()` is called, the help object is not re-rendered. If it needs to be re-rendered, call the function `XscHelpUpdate()`, providing the widget or gadget with the new XscHelp resources as the only parameter. There is no harm in calling this function if the help objects are not currently in use.

Code Listing 8-3 continues the previous example, showing how to make sure the screen is rendered immediately if XscHelp resources change.

SOURCE CODE 8-3: Re-rendering help objects

```
...
extern Display* display;
extern Widget okIcon;
XrmDatabase database;

database = XrmGetDatabase( display );

XrmPutStringResource(
   &database, "*okIconButton.xscTipTopic", "OK" );

XscHelpDbReload( okIcon );
XscHelpUpdate( okIcon );
...
```

Automatically Rereading the Database

Each help object has an automatic database reload resource. Normally, this resource is set to False; however, if it is set to True, then the resource database is automatically reloaded just before the help object is rendered. The table below shows the name of this resource for each help object.

Help Object	Automatic Database Reload Resource Name
Cue	`xscCueAutoDbReload`
Hint	`xscHintAutoDbReload`
Tip	`xscTipAutoDbReload`

Object API

The XscHelp Object API provides programmers with a convenient interface to perform simple Tip, Cue, and Hint functions. Each of these functions accepts a widget or gadget as the first parameter. There are four functions from this API set for each type of help object.

Tips

FUNCTION XscHelpTipExists

To determine if a widget or gadget has a Tip, use the function `XscHelpTipExists()`. Given a widget, the function returns True if the widget has a Tip; otherwise, False is returned.

FUNCTION XscHelpSetTipTopic

A Tip topic can easily be modified by calling the function `XscHelpSetTipTopic()`. This function accepts a widget and the new string to display as the Tip.

FUNCTION XscHelpTipUpdate

Once the Tip has been modified, the new topic is displayed the next time the Tip is popped-up. To update the Tip immediately (in case the Tip is currently displayed), call the function `XscHelpTipUpdate()`.

Code Listing 8-4 shows an example of how these functions can be used together.

SOURCE CODE 8-4: Changing Tip topic

```
...
extern Widget gadget;
...
if (XscHelpTipExists( gadget ))
{
   XscHelpSetTipTopic( gadget, "New Tip" );
   XscHelpTipUpdate( gadget );
}
...
```

This makes sure the new Tip topic is displayed even if the Tip is currently popped-up.

FUNCTION XscHelp-SetTipTopicDetails

The function `XscHelpSetTipTopicDetails()` is used to update a widget's Tip topic and core set of text rendering attributes. In particular, the function updates the Tip text, font

list, font list tag, alignment, direction, and string converter. Code Listing 8-5 shows an example of using this function.

SOURCE CODE 8-5: Changing Tip topic and text attributes

```
...
extern Widget gadget;
extern XmFontList font_list;
...
if (XscHelpTipExists( gadget ))
{
   XscHelpSetTipTopicDetails(
      gadget,
      "New @f[bold]Tip",
      font_list,
      "",
      XmALIGNMENT_CENTER,
      XmSTRING_DIRECTION_L_TO_R,
      XmXSC_STRING_CONVERTER_SEGMENTED );

   XscHelpTipUpdate( gadget );
}
...
```

The "Tip" in the specified text is rendered with the font tagged as "bold" in the specified font list since the string converter is segmented.

Cues

FUNCTION XscHelpCueExists

To determine if a widget or gadget has a Cue, use the function `XscHelpCueExists()`. Given a widget, the function returns True if the widget has a Cue; otherwise, False is returned.

FUNCTION XscHelpSetCueTopic

A Cue topic can easily be modified by calling the function `XscHelpSetCueTopic()`. This function accepts a widget and the new string to display as the Cue.

FUNCTION XscHelpCueUpdate

Once the Cue has been modified, the new topic is displayed the next time the Cue is popped-up. To update the Cue immediately (in case the Cue is currently displayed), call the function `XscHelpCueUpdate()`.

FUNCTION XscHelp-SetCueTopicDetails

The function `XscHelpSetCueTopicDetails()` is used to update a widget's Cue topic and core set of text rendering attributes. In particular, the function updates the Cue text, font list, font list tag, alignment, direction, and string converter.

Hints

FUNCTION XscHelpHintExists

To determine if a widget or gadget has a Hint, use the function `XscHelpHintExists()`. Given a widget, the function returns True if the widget has a Hint; otherwise, False is returned.

FUNCTION XscHelpSetHintTopic

A Hint topic can easily be modified by calling the function `XscHelpSetHintTopic()`. This function accepts a widget and the new string to display as the Hint.

FUNCTION XscHelpHintUpdate

Once the Hint has been modified, the new topic is displayed the next time the Hint is rendered. To update the Hint immediately (in case the Hint is currently displayed), call the function `XscHelpHintUpdate()`.

FUNCTION XscHelp-SetHintTopicDetails

The function `XscHelpSetHintTopicDetails()` is used to update a widget's Hint topic and core set of text rendering attributes. In particular, the function updates the Hint text, font list, font list tag, alignment, direction, and string converter.

Tip API

Using the Tip API, you can gain access to all the Tip resources of a widget or gadget. These functions directly interact with the internal Tip object associated with the widget rather than using the widget as a mediator. All of the members of this API begin with an "XscTip".

FUNCTION XscTip-DeriveFromWidget

In order to interact with the Tip object, you must first acquire a Tip handle. The function `XscTipDeriveFromWidget()` is used to retrieve the handle to a Tip object. The function accepts a widget or gadget as the only parameter and returns an opaque Tip handle. The return value equals `NULL` if there is no Tip object associated with the given widget or gadget.

`Function` XscTipHasValidTopic

Given a Tip handle, you can determine if the Tip has a valid topic by calling the function `XscTipHasValidTopic()`. Regardless of the state of the Tip or its attributes, a Tip that does not have a valid topic is never displayed. This function takes a Tip handle as the only parameter and returns a Boolean.

FUNCTION XscTipGet...
XscTipSet...

Each Tip resource has a pair of access functions; one access function is used to get the value and the other is used to set the value.

CROSS-REFERENCE
A complete listing of all the Tip resources is located in Chapter 14 on page 130. Likewise, a complete listing of all the Tip get/set functions can be found in Chapter 10 starting on page 103.

Each of the `XscTipGet`... functions takes a Tip handle as the only parameter; the return type value depends on the data type of the associated resource value.

The `XscTipSet`... functions have no return value; in addition they all accept a Tip handle as the first parameter. The second parameter is the value to assign to the resource; as a result, the second parameter has the same data type as the associated resource value.

Some of the `XscTipSet`... functions have a third Boolean parameter called *update*. These functions modify resources dealing with the actual string rendering of the Tip topic. If the *update* parameter is True, then the internal compound string used to render the Tip topic is reconstructed; if the *update* parameter is False, then the value is recorded and used the next time the compound string is built.

The internal compound string for the Tip topic is built only under the following circumstances:

- An `XscTipSet`... function is called to modify the alignment, font list, font list tag, string converter, or string direction resource with the *update* parameter set to True.
- The Tip topic is changed through any mechanism.

Cue API

Using the Cue API, you can gain access to all the Cue resources of a widget or gadget. These functions directly interact with the internal Cue object associated with the widget rather than using the widget as a mediator. All of the members of this API begin with an “XscCue”.

FUNCTION
XscCue-DeriveFromWidget

In order to interact with the Cue object, you must first acquire a Cue handle. The function `XscCueDeriveFromWidget()` is used to retrieve the handle to a Cue object. The function accepts a widget or gadget as the only parameter and returns an opaque Cue handle. The return value equals `NULL` if there is no Cue object associated with the given widget or gadget.

FUNCTION
XscCueHasValidTopic

Given a Cue handle, you can determine if the Cue has a valid topic by calling the function `XscCueHasValidTopic()`. Regardless of the state of the Cue or its attributes, a Cue that

does not have a valid topic is never displayed. This function takes a Cue handle as the only parameter and returns a Boolean.

FUNCTIONS
XscCueGet...
XscCueSet...

Each Cue resource has a pair of access functions; one access function is used to get the value and the other is used to set the value.

CROSS-REFERENCE
A complete listing of all the Cue resources is located in Chapter 14 on page 116. Likewise, a complete listing of all the Cue get/set functions can be found in Chapter 10 starting on page 79.

Each of the `XscCueGet`... functions takes a Cue handle as the only parameter; the return type value depends on the data type of the associated resource value.

The `XscCueSet`... functions have no return value; in addition they all accept a Cue handle as the first parameter. The second parameter is the value to assign to the resource; as a result, the second parameter has the same data type as the associated resource value.

Some of the `XscCueSet`... functions have a third Boolean parameter called *update*. These functions modify resources dealing with the actual string rendering of the Cue topic. If the *update* parameter is True, then the internal compound string used to render the Cue topic is reconstructed; if the *update* parameter is False, then the value is recorded and used the next time the compound string is built.

The internal compound string for the Cue topic is built only under the following circumstances:

- An `XscCueSet`... function is called to modify the alignment, font list, font list tag, string converter, or string direction resource with the *update* parameter set to True.
- The Cue topic is changed through any mechanism.

Hint API

Using the Hint API, you can gain access to all the Hint resources of a widget or gadget. These functions directly interact with the internal Hint object associated with the widget rather than using the widget as a mediator. All of the members of this API begin with an “XscHint”.

FUNCTION
XscHintDeriveFrom-Widget

In order to interact with the Hint object, you must first acquire a Hint handle. `XscHintDeriveFromWidget()` is used to retrieve the handle to a Hint object. The function accepts a

widget or gadget as the only parameter and returns an opaque Hint handle. The return value equals `NULL` if there is no Hint object associated with the given widget or gadget.

FUNCTION
XscHintHasValidTopic

Given a Hint handle, you can determine if the Hint has a valid topic by calling the function `XscHintHasValidTopic()`. Regardless of the state of the Hint or its attributes, a Hint that does not have a valid topic is never displayed. This function takes a Hint handle as the only parameter and returns a Boolean.

FUNCTION
XscHintGet...
XscHintSet...

Each Hint resource has a pair of access functions; one access function is used to get the value and the other is used to set the value.

CROSS-REFERENCE
A complete listing of all the Hint resources is located in Chapter 14 on page 123. Likewise, a complete listing of all the Hint get/set functions can be found in Chapter 10 starting on page 101.

Each of the `XscHintGet`... functions takes a Hint handle as the only parameter; the return type value depends on the data type of the associated resource value.

The `XscHintSet`... functions have no return value; in addition they all accept a Hint handle as the first parameter. The second parameter is the value to assign to the resource; as a result, the second parameter has the same data type as the associated resource value.

Some of the `XscHintSet`... functions have a third Boolean parameter called *update*. These functions modify resources dealing with the actual string rendering of the Hint topic. If the *update* parameter is True, then the internal compound string used to render the Hint topic is reconstructed; if the *update* parameter is False, then the value is recorded and used the next time the compound string is built.

The internal compound string for the Hint topic is built only under the following circumstances:

- An `XscHintSet`... function is called to modify the alignment, font list, font list tag, string converter, or string direction resource with the *update* parameter set to True.
- The Hint topic is changed through any mechanism.

Enabling and Disabling

Tip, hints, and cues can be disabled globally or on a per shell basis. For any given tip, hint, or cue to be rendered, it must be individually enabled, the shell must be enabled, and it must be globally enabled.

FUNCTIONS
XscHelpSetCuesEnabledGlobally
XscHelpSetHintsEnabledGlobally
XscHelpSetTipsEnabledGlobally

The following functions are used to globally enable or disable each type of help:

```
void XscHelpSetCuesEnabledGlobally(
     Boolean );
void XscHelpSetHintsEnabledGlobally(
     Boolean );
void XscHelpSetTipsEnabledGlobally(
     Boolean );
```

The Boolean specifies if the cues, hints, or tips should be enabled or disabled.

FUNCTIONS
XscHelpSetCuesEnabledOnShell
XscHelpSetHintsEnabledOnShell
XscHelpSetTipsEnabledOnShell

Likewise, the following functions are used to enable or disable help on a per shell basis:

```
void XscHelpSetCuesEnabledOnShell(
     Widget, Boolean );
void XscHelpSetHintsEnabledOnShell(
     Widget, Boolean );
void XscHelpSetTipsEnabledOnShell(
     Widget, Boolean );
```

The widget specifies the shell of interest and the Boolean specifies if the tips, hints, or cues should be enabled or disabled.

Tips for individual widgets are controlled with the resource `XmNxscTipEnabled` and the functions `XscTipGetEnabled` and `XscTipSetEnabled`. Similar resources and functions exist for hints and cues.

FUNCTIONS
XscHelpAreCuesEnabledGlobally
XscHelpAreHintsEnabledGlobally
XscHelpAreTipsEnabledGlobally

You can test if tips, hint, or cues are globally enabled with the following functions:

```
Boolean XscHelpAreCuesEnabledGlobally();
Boolean XscHelpAreHintsEnabledGlobally();
Boolean XscHelpAreTipsEnabledGlobally();
```

FUNCTIONS
XscHelpAreCues-EnabledOnShell
XscHelpAreHints-EnabledOnShell
XscHelpAreTips-EnabledOnShell

Likewise, you can test if tip, hints, or cues are enabled on a given shell with the following functions:

```
Boolean XscHelpAreCuesEnabledOnShell(
     Widget );

Boolean XscHelpAreHintsEnabledOnShell(
     Widget );

Boolean XscHelpAreTipsEnabledOnShell(
     Widget );
```

FUNCTIONS
XscHelpAreCues-Displayable
XscHelpAreHints-Displayable
XscHelpAreTips-Displayable

You can test is a tip, hint, or cue is enabled individually, with its shell, and globally be calling the following function and specifying the widget of interest:

```
Boolean XscHelpAreCuesDisplayable(
     Widget );

Boolean XscHelpAreHintsDisplayable(
     Widget );

Boolean XscHelpAreTipsDisplayable(
     Widget );
```

CHAPTER 9

Authoring Aids

The Help ToolKit for Motif provides software aids that simplify the development of the actual XscHelp resources.

The first aid allows the user to clearly identify (and optionally select) the name of a given widget; this makes it easier to associate help attributes with a given widget.

Another aid allows help text to be indirectly assigned to a widget — help text can be written once and assigned to multiple widgets. This allows the help text for all the widgets to be updated from a single resource entry.

Finally, help text can be specified in a format that is simpler to work with than a resource file. This help text file is than dynamically loaded and assigned (indirectly) to the appropriate widgets.

Identifying Widget Names

The Help ToolKit is designed so that it can be integrated into a Motif application without knowing exactly which widgets will activate what help text. The specific help attributes can be assigned after the application code has been finished through resource files.

A programmer can develop the application and hand it over to a technical writer who independently writes and configures the help. However, to bind help attributes to a specific widget, you must know the name (and sometimes the name hierarchy) of the widget. This can be frustrating thing to coordinate when the help author does not have access to the source code or does not know how to read Motif code.

To remedy this problem, the Help ToolKit can be configured (through resource files) to display the name of a given widget in a Tip window, a Hint area, or a Cue window.

RESOURCE
xscTipShowName

The resource name used to display the name of a widget in its Tip window is `xscTipShowName`. Likewise, the name of the resource used to show the name in a Hint or Cue is `xscHintShowName` and `xscCueShowName`, respectively.

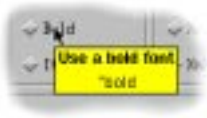
All three of these resources can have one of the following values:

- `XmXSC_SHOW_NAME_ALL`
- `XmXSC_SHOW_NAME_NONE`
- `XmXSC_SHOW_NAME_SELF`
- `XmXSC_SHOW_NAME_SHELL`

`XmXSC_SHOW_NAME_NONE` is the default value of the `XscTipShowName` resource, which results in the widget name not being shown. For comparison purposes, an example of a Tip with the show name resource set to `XmXSC_SHOW_NAME_NONE` is shown here. The relevant portion of the resource file associated with this example is shown in Code Listing 9-1.

SOURCE CODE 9-1: Tip with no widget name

```
...
*bold.xscTipTopic: Use a bold font
...
```



By adding a global `xscTipShowName` resource to the resource file with a value of `XmXSC_SHOW_NAME_SELF` (as shown in Code Listing 9-2), the widget name can be displayed in the Tip window without changing the application binary image.

SOURCE CODE 9-2: Tip with widget name

```
...
*xscTipShowName: XmXSC_SHOW_NAME_SELF
*bold.xscTipTopic: Use a bold font
...
```

The `xscTipShowName` resource value `XmXSC_SHOW_NAME_SHELL` specifies that the name of the widget and its nearest window manager shell parent need to be displaced. Notice that the shell name and widget name are displayed using the specific resource file syntax that could be used to qualify the widget as a child of the indicated shell widget.

Finally, by setting the value of the Help ToolKit resource `xscTipShowName` as `XmXSC_SHOW_NAME_ALL`, the complete widget hierarchy is displayed in the Tip window.

RESOURCE
xscCueShowName
xscHintShowName

The `xscCueShowName` and `xscHintShowName` resources work just like the `xscTipShowName` resource. Keep in mind that the Hint text displayed is vertically centered in the Hint display widget; if the hint display widget is not tall enough, then the widget name may be partially or completely clipped.

Selecting Widget Names

The name of the widget displayed in the Tip window via the `xscTipShowName` resource is written as a valid resource file widget specifier. This specifier can be literally copied by a help author into a resource file to bind help attributes to the widget.

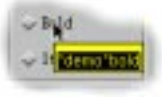
RESOURCE
xscTipSelectName-Interval

To make this process simpler, the name displayed in the Tip window can automatically be selected so that copy-and-paste can be used to copy the widget name into a text editor. The widget name in the Tip window is selected automatically after the Tip has been displayed for the number of milliseconds specified in the `xscTipSelectNameInterval` resource.

The default value for this resource is -1, which indicates that a selection should never be made.

NOTE
There is no way to select the widget name displayed in a Hint or Cue.

The resource file in Code Listing 9-3 shows how to specify that the widget name should be selected. In this example, two seconds after the tip is displayed the widget name is automatically selected. Once selected, it can be pasted into an



editor or used like any other selected text. This technique can be use to help create the Help ToolKit resource file entries when the name of a widget is not known.

SOURCE CODE 9-3: Widget name selection interval

After the tip is displayed for two seconds, the shell and widget name are selected.

```
...
*xscTipShowName: XmXSC_SHOW_NAME_SHELL
*xscTipSelectNameInterval: 2000
...
```

Help Topic Indirection

The Help ToolKit allows commonly used text to be defined once and referenced indirectly for display in Cues, Hints, and Tips. The actual text is defined under the resource name `_XscHelp.topic.`*`name`* where *`name`* is the specific name assigned to the topic text. The resource file Code Listing 9-4 shows an example of how to define an indirect help topic.

SOURCE CODE 9-4: Defining indirect help topic text

This indirect text topic has the name "dismiss button" which is used later to reference the text

```
...
_XscHelp.topic.dismiss button: \
Press this button to close the window;\n\
this button will not stop the application
...
```

The indirect text topic can be referenced from an `xscCue-Topic`, `xscHintTopic`, or `xscTipTopic` resource. The reference is specified by using the indirect topic's name as the value of the widget's topic resource; to indicate that this is an indirect reference, the name must be preceded by a period. An example of an indirect reference is shown in Code Listing 9-5.

SOURCE CODE 9-5: Defining indirect help topic text

All of these widgets will have a Tip with the text defined in Code Listing 9-4.

```
...
*exitButton.xscTipTopic: \
This button will stop the application

*dismissButton.xscTipTopic: .dismiss button
*closeButton.xscTipTopic: .dismiss button
*cancelButton.xscTipTopic: .dismiss button
...
```

Loading Indirect Help Topics

Like most other Help ToolKit resources, indirect topics can be loaded via resource files. However, the Help ToolKit defines a simple parsing format that allow indirect text topics to be easily created, assigned, and loaded.

FUNCTION
XscHelpLoadTopics

These indirect text topic files are loaded with the function `XscHelpLoadTopics`. The prototype for the function is shown below:

```
int XscHelpLoadTopics(
   Display*    aDisplay,
   const char* aFilename );
```

The function has two arguments: the first is the display handle and the second is the filename containing the topic information. Multiple topic files can be loaded by calling the function multiple times.

The function returns the error number associated with an errors that occurred while trying to read the specified filename. The function returns a value of 0 if there were no errors.

The format for the indirect text topic fie is simple and easy to use. First of all, any text that is not contained within a *text block* is ignored. A text block starts with the keyword `.TEXT` and ends with the keyword `.END`. A text block also automatically terminates if a new `.TEXT` keyword is found.

The `.TEXT` keyword has a single parameter contained in square brackets; this argument is the name of the text and is used as the resource name when the text is added to the resource database.

Any text can be added after the `.END` keyword.

All the text (including newline characters) inside a text block is assigned to the specified name in the resource database when the file is loaded.

As an example, the simple indirect topics defined in Code Listing 9-4 (directly in a resource file) could have been defined with the file shown in Code Listing 9-6.

SOURCE CODE 9-6: Indirect help topic file

This indirect text topic has the name "dismiss button" which can be reference indirectly for Cues, Hints, and Tips

```
...
This line is ignored since it is not in a TEXT block

.TEXT[dismiss button]
Press this button to close the window;\n\
this button will not stop the application
.END dismiss button text
...
```

CHAPTER 10

Functions

XscCdeHelpInstall

CROSS-REFERENCE
Refer to page 49 for additional instructions.

This function initializes the Help ToolKit and installs the CDE context-sensitive help infrastructure.

```
#include <Xsc/CdeHelp.h>

void XscCdeHelpInstall(
   Widget          shell,
   const char*     default_help_volume );
```

shell	(in)	The first shell created after the X Toolkit Intrinsics are initialized.
default_help_volume	(in)	The value to use as the default CDE help volume for the application.

NOTE
XscCdeHelpInstall can only be used in environments that support the CDE Help System

Normally, the shell parameter specifies the first application shell or session shell created (directly or indirectly) on the first display connection. Other display connections are handled with the function `XscHelpInstall()`.

If *default_help_volume* is set to NULL, then the default help volume name is derived from the environment variable `XSC_HELP_VOLUME_NAME`.

If `XscCdeHelpInstall()` is used to install CDE Context-Help, then the functions `XscHelpInstall()` and `XscHelpContextInstall()` should not be used to initialize the Help ToolKit in the application.

After the Help ToolKit is installed with this function, Tips, Cues, and CDE Context-Help support are completely enabled.

XscCueDeriveFromWidget

CROSS-REFERENCE
Refer to page 66 for additional instructions.

This function retrieves a handle to the Cue associated with an object.

```
#include <Xsc/Cue.h>

XscCue XscCueDeriveFromWidget(
   Widget object );
```

object	(in)	The widget (or gadget) of interest

A handle to the Cue is returned or `NULL` if there is no Cue for the specified *object*.

XscCueGet<name>

CROSS-REFERENCE
Refer to page 67 for additional instructions.

These functions extract the current resource values from a Cue handle.

#include <Xsc/Cue.h>

```
unsigned char XscCueGetAlignment      ( XscCue );
Boolean       XscCueGetAutoDbReload   ( XscCue );
Pixel         XscCueGetBackground     ( XscCue );
Pixel         XscCueGetBorderColor    ( XscCue );
Dimension     XscCueGetBorderWidth    ( XscCue );
Pixel         XscCueGetColorBase      ( XscCue );
Boolean       XscCueGetEnabled        ( XscCue );
XmFontList    XscCueGetFontList       ( XscCue );
String        XscCueGetFontListTag    ( XscCue );
Pixel         XscCueGetForeground     ( XscCue );
Dimension     XscCueGetMarginBottom   ( XscCue );
Dimension     XscCueGetMarginHeight   ( XscCue );
Dimension     XscCueGetMarginLeft     ( XscCue );
Dimension     XscCueGetMarginRight    ( XscCue );
Dimension     XscCueGetMarginTop      ( XscCue );
Dimension     XscCueGetMarginWidth    ( XscCue );
Boolean       XscCueGetMotifColorModel( XscCue );
unsigned char XscCueGetPosition       ( XscCue );
Dimension     XscCueGetShadowThickness( XscCue );
unsigned char XscCueGetShadowType     ( XscCue );
unsigned char XscCueGetStringConverter( XscCue );
unsigned char XscCueGetStringDirection( XscCue );
String        XscCueGetTopic          ( XscCue );
int           XscCueGetXOffset        ( XscCue );
int           XscCueGetYOffset        ( XscCue );
```

`XscCue`	(in)	The handle of the Cue to examine

Each function retrieves the value of the named Cue attribute.

The functions returning `String` data types return a copy of the Cue's string attributes. The returned `String` should be deallocated with `XtFree()`.

XscCueHasValidTopic

CROSS-REFERENCE
Refer to page 66 for additional instructions.

This function determines if a Cue contains a viewable topic.

```
#include <Xsc/Cue.h>
```

```
Boolean XscCueHasValidTopic( XscCue cue );
```

cue	(in)	The handle of the Cue to examine

A Cue without a valid topic is not displayable. Normally, a Cue will not be created for a widget unless there is a valid Cue topic. However, the Cue topic can be destroyed after the Cue is created.

XscCueSet<name>

CROSS-REFERENCE
Refer to page 67 for additional instructions.

These functions modify the current resource values of a Cue handle.

#include <Xsc/Cue.h>

```
/* Each of these begins with "void XscCueSet"... */

...Alignment      ( XscCue, unsigned char, Boolean );
...AutoDbReload   ( XscCue, Boolean       );
...Background     ( XscCue, Pixel         );
...BorderColor    ( XscCue, Pixel         );
...BorderWidth    ( XscCue, Dimension     );
...ColorBase      ( XscCue, Pixel         );
...Enabled        ( XscCue, Boolean       );
...FontList       ( XscCue, XmFontList   , Boolean );
...FontListTag    ( XscCue, String       , Boolean );
...Foreground     ( XscCue, Pixel         );
...MarginBottom   ( XscCue, Dimension     );
...MarginHeight   ( XscCue, Dimension     );
...MarginLeft     ( XscCue, Dimension     );
...MarginRight    ( XscCue, Dimension     );
...MarginTop      ( XscCue, Dimension     );
...MarginWidth    ( XscCue, Dimension     );
...MotifColorModel( XscCue, Boolean       );
...Position       ( XscCue, unsigned char );
...ShadowThickness( XscCue, Dimension     );
...ShadowType     ( XscCue, unsigned char );
...StringConverter( XscCue, unsigned char, Boolean );
...StringDirection( XscCue, unsigned char, Boolean );
...Topic          ( XscCue, String        );
...XOffset        ( XscCue, int           );
...YOffset        ( XscCue, int           );
```

`XscCue`	(i/o)	The handle of the Cue to modify

Each function sets the value of the named Cue attribute.

The functions setting a `String` data type internally make a copy of the string.

Some of the functions have a third, `Boolean` parameter; each of these functions relates to an attribute used to create a compound string for rendering. If the third parameter is set to `True`, then the internal compound string is created immediately. If you are setting more than one of these resources, set

the third parameter to `True` only for the last one. The function `XscCueSetTopic()` implicitly creates an internal compound string.

XscHelpAreCuesDisplayable

VERSION 1.1.1 This function indicates if Cues associated within a given shell are displayable.

```
#include <Xsc/Help.h>

Boolean XscHelpAreCuesDisplayable(
   Widget widget );
```

widget	(in)	The WMShell widget or any child of the WMShell widget being examined.

Cues must be enabled both globally and within a given shell before a Cue is displayable. The shell being tested is either the specified widget (if it is a shell widget) or the closest shell ancestor to the specified widget

XscHelpAreCuesEnabledGlobally

VERSION 1.1.1 This function tests if Cues have been globally enabled or disabled.

```
#include <Xsc/Help.h>

Boolean XscHelpAreCuesEnabledGlobally();
```

No Cue within the entire application can be displayed if this function returns `False`.

XscHelpAreCuesEnabledOnShell

VERSION 1.1.1 This function tests if Cues have been enabled or disabled with a given shell widget.

```
#include <Xsc/Help.h>

Boolean XscHelpAreCuesEnabledOnShell(
   Widget widget );
```

widget	(in)	The WMShell widget or any child of the WMShell widget being examined.

No Cue associated with a widget contained in the tested shell widget can be displayed if this function returns `False`. The shell being tested is either the specified widget (if it is a shell widget) or the closest shell ancestor to the specified widget.

XscHelpAreHintsDisplayable

VERSION 1.1.1 This function indicates if Hints associated within a given shell are displayable.

```
#include <Xsc/Help.h>

Boolean XscHelpAreHintsDisplayable(
   Widget widget );
```

widget	(in)	The WMShell widget or any child of the WMShell widget being examined.

Hints must be enabled both globally and within a given shell before a Hint is displayable. The shell being tested is either the specified widget (if it is a shell widget) or the closest shell ancestor to the specified widget

XscHelpAreHintsEnabledGlobally

VERSION 1.1.1 This function tests if Hints have been globally enabled or disabled.

```
#include <Xsc/Help.h>

Boolean XscHelpAreHintsEnabledGlobally();
```

No Hint within the entire application can be displayed if this function returns `False`.

XscHelpAreHintsEnabledOnShell

VERSION 1.1.1 This function tests if Hints have been enabled or disabled with a given shell widget.

```
#include <Xsc/Help.h>

Boolean XscHelpAreHintsEnabledOnShell(
   Widget widget );
```

widget	(in)	The WMShell widget or any child of the WMShell widget being examined.

No Hint associated with a widget contained in the tested shell widget can be displayed if this function returns `False`. The shell being tested is either the specified widget (if it is a shell widget) or the closest shell ancestor to the specified widget.

XscHelpAreTipsDisplayable

VERSION 1.1.1 This function indicates if Tips associated within a given shell are displayable.

```
#include <Xsc/Help.h>

Boolean XscHelpAreTipsDisplayable(
   Widget widget );
```

widget	(in)	The WMShell widget or any child of the WMShell widget being examined.

Tips must be enabled both globally and within a given shell before a Tip is displayable. The shell being tested is either the specified widget (if it is a shell widget) or the closest shell ancestor to the specified widget

XscHelpAreTipsEnabledGlobally

VERSION 1.1.1 This function tests if Tips have been globally enabled or disabled.

```
#include <Xsc/Help.h>

Boolean XscHelpAreTipsEnabledGlobally();
```

No Tip within the entire application can be displayed if this function returns `False`.

XscHelpAreTipsEnabledOnShell

VERSION 1.1.1 This function tests if Tips have been enabled or disabled with a given shell widget.

```
#include <Xsc/Help.h>

Boolean XscHelpAreTipsEnabledOnShell(
   Widget widget );
```

widget	(in)	The WMShell widget or any child of the WMShell widget being examined.

No Tip associated with a widget contained in the tested shell widget can be displayed if this function returns `False`. The shell being tested is either the specified widget (if it is a shell widget) or the closest shell ancestor to the specified widget.

XscHelpContextInstall

CROSS-REFERENCE
Refer to page 41 for additional instructions.

This function specifies what resource data should be retrieved, stored, and provided to the application when context-sensitive help is requested by the user.

```
#include <Xsc/Help.h>

void XscHelpContextInstall(
   XtResourceList resources,
   Cardinal       num_resources,
   int            sizeof_help_struct,
   XtCallbackProc context_callback,
   XtPointer      client_data );
```

resources	(in)	Specifies the resource list to retrieve when context-sensitive help is requested.
num_resources	(in)	Specifies the number of resources in the resource list.
sizeof_help_struct	(in)	Specifies the size of the data structure used to store the context-help data.
context_callback	(in)	Specifies the callback procedure to call when delivering context-sensitive help data to the application.
client_data	(in)	Specifies the data to be passed to *context_callback* when it is invoked.

Each entry in *resources* describes a resource that should be retrieved when context-sensitive help is requested. The retrieved values are associated with the widget (or one of the widget's ancestors) used as the target for the context-sensitive help request.

The first entry in the *resources* array must represent a pointer with a default value of NULL. If this resource is not defined in the resource database, then the closest ancestors are searched until a non-NULL resource value is found. Once the first entry

is found, all the other resource values are retrieved at that same level. The level is provided to the application in the *depth* data member in the callback data structure `XscHelpContext-CallbackStruct`. A depth of 0 indicates that the help data was found associated with the widget targeted for the context-sensitive help request; a depth of 1 indicates that the help was found with the widget's parent, etc.

The *sizeof_help_struct* parameter could be determined by using the `sizeof()` operator on the data structure used to compute the *resource_offset* values in the *resources* array.

The function pointer *context_callback* is called to provide the application with the requested context-sensitive data. The *client_data* parameter is not used by the Help ToolKit; it is an arbitrary value that is simply passed to the callback function via the callback *client_data* parameter.

The `XscHelpContextCallbackStruct` data structure is passed to the callback function via the third parameter. The *data* field in this structure contains a pointer to a structure containing the requested context-sensitive help data. If no context-sensitive help data could be found (at any level), then the *data* field is set to NULL.

Data Structures

`XscHelpContextCallbackStruct` has the following definition:

SOURCE CODE 10-1: Definition of XscHelpContextCallbackStruct

```
typedef struct
{
   int       reason;
   XEvent*   event;
   int       depth;
   XtPointer data;
}
XscHelpContextCallbackStruct;
```

`XtResource` has the following definition:

SOURCE CODE 10-2: Definition of XtResource

```
typedef struct _XtResource
{
   String    resource_name;
   String    resource_class;
   String    resource_type;
   Cardinal  resource_size;
   Cardinal  resource_offset;
   String    default_type;
   XtPointer default_addr;
}
XtResource, *XtResourceList;
```

XscHelpContextPickAndActivate

CROSS-REFERENCE
Refer to page 48 for additional instructions.

This function allows the user to request context-sensitive help for a widget by selecting the widget with the mouse.

```
#include <Xsc/Help.h>

void XscHelpContextPickAndActivate(
   Widget          widget,
   Cursor          cursor,
   Boolean         confine_to_widget );
```

widget	(in)	Specifies a widget (normally a `Toplevel` shell) that is the common ancestor of all widgets that can be picked using this function.
cursor	(in)	Specifies a standard X cursor that should be used as the pointer while the function is active.
confine_to_widget	(in)	Specifies if the cursor should be confined within the widget specified by the *widget* parameter.

This function grabs the mouse and keyboard and waits for a mouse key or a button to be pressed. If a mouse button or a key is pressed while the pointer is over a widget (or gadget)

contained by *widget*, then context-sensitive help is requested for the selected widget.

Clicking the mouse outside of *widget* or pressing the ESC key causes the function to return without selecting a widget.

XscHelpCueExists

CROSS-REFERENCE
Refer to page 64 for additional instructions.

This function indicates if the specified object has a Cue.

```
#include <Xsc/Help.h>

Boolean XscHelpCueExists( Widget object );
```

object	(in)	The widget of interest.

`True` is returned if and only if the `xscCueTopic` resource is defined for the specified widget.

XscHelpCueUpdate

CROSS-REFERENCE
Refer to page 64 for additional instructions.

This function updates the Cue of the specified widget, even if it is currently displayed.

```
#include <Xsc/Help.h>

void XscHelpCueUpdate( Widget object );
```

object	(in)	The target widget.

XscHelpDbReload

CROSS-REFERENCE
Refer to page 61 for additional instructions.

This function reloads from the resource database the help resources associated with a widget or gadget.

```
#include <Xsc/Help.h>

void XscHelpDbReload( Widget object );
```

object	(in)	The widget used as the target for the reload.

This function updates the Help ToolKit's internal data stores to reflect the current help resource values in the resource database

associated with *object*. This function can be used to change the appearance or behavior of the help associated with a widget by changing resource values in the resource database and then forcing a reload.

XscHelpHintExists

CROSS-REFERENCE
Refer to page 65 for additional instructions.

This function indicates if the specified object has a Hint.

```
#include <Xsc/Help.h>

Boolean XscHelpHintExists(
   Widget object );
```

object	(in)	The widget of interest.

`True` is returned if and only if the `xscHintTopic` resource is defined for the specified widget.

XscHelpHintInstall

CROSS-REFERENCE
Refer to page 34 for additional instructions.

This function enables Hint support for a window.

```
#include <Xsc/Help.h>

void XscHelpHintInstall(
   Widget hint_widget );
```

hint_widget	(in)	The widget to use to display Hint information in a window.

NOTE
An `XmLabel` with an `XmNlabelString` of "" works well as a hint display widget.

The specified *hint_widget* becomes the hint display widget for its closest shell ancestor. Any Hints generated by a widget within this shell are displayed on the window belonging to the *hint_widget*. As a result, *hint_widget* must not be a gadget and should not render any of its own text or graphics.

XscHelpHintUpdate

CROSS-REFERENCE
Refer to page 65 for additional instructions.

This function updates the Hint of the specified widget, even if it is currently displayed.

```
#include <Xsc/Help.h>

void XscHelpHintUpdate( Widget object );
```

object	(in)	The target widget.

XscHelpInstall

CROSS-REFERENCE
Refer to page 5 for additional instructions.

This function initializes the internal XscHelp data structures and binds to the initial X display connection.

```
#include <Xsc/Help.h>

void XscHelpInstall( Widget shell );
```

shell	(in)	The first shell created after the X Toolkit Intrinsics are initialized.

NOTE
This function enables Tip and Cue support.

The shell parameter specifies the first application shell or session shell created (directly or indirectly) on each display connection.

XscHelpIsDynamicTipGroupIdDefaultActive

VERSION 1.1.3

CROSS-REFERENCE
Refer to page 21 for additional details.

This function specifies if the default value for the resource `XmNtipGroupId` is derived with the old *static* method or the new *dynamic* method.

```
#include <Xsc/Help.h>

Boolean
XscHelpIsDynamicTipGroupIdDefaultActive();
```

`True` is returned if the new dynamic method is in use; `False` is returned if the old static method is in use.

This function can be called at any time, even before the toolkit is initialized.

XscHelpLoadTopics

VERSION 1.1

This function loads an indirect topic text file into the resource database for use with Cue, Hint, and Tip topics.

CROSS-REFERENCE
Refer to page 75 for more details about defining a loading indirect text topics.

```
#include <Xsc/Help.h>

void XscHelpLoadTopics(
   Display*       aDisplay,
   const char*    aFilename );
```

aDisplay	(in)	The display handle associated with the resource database to use.
aFilename	(in)	The complete name of the file containing the indirect topic text.

XscHelpSetCuesEnabledGlobally

VERSION 1.1.1

This function records if Cues should be enabled or disabled globally.

```
#include <Xsc/Help.h>

void XscHelpSetCuesEnabledGlobally(
   Boolean flag );
```

flag	(in)	The flag should be `True` to enable and `False` to disable Cues.

No Cue within the entire application can be displayed while Cues are globally disabled.

XscHelpSetCuesEnabledOnShell

VERSION 1.1.1

This function records if Cues should be enabled or disabled within a given shell.

```
#include <Xsc/Help.h>

void XscHelpSetCuesEnabledOnShell(
   Widget  widget,
   Boolean flag );
```

widget	(in)	The WMShell widget or any child of the WMShell widget being referenced.
flag	(in)	The flag should be `True` to enable and `False` to disable Cues.

No Cue associated with a widget contained in the specified shell widget can be displayed if this attribute is set to `False`. The shell being referenced is either the specified widget (if it is a shell widget) or the closest shell ancestor to the specified widget.

XscHelpSetCueTopic

CROSS-REFERENCE
Refer to page 64 for additional instructions.

This function updates the Cue topic for a given widget.

```
#include <Xsc/Help.h>

void XscHelpSetCueTopic(
   Widget          object,
   String          topic );
```

object	(in)	The widget of interest.
topic	(in)	The value for the `xscCueTopic` resource.

XscHelpSetCueTopicDetails

CROSS-REFERENCE
Refer to page 64 for additional instructions.

This function sets all of the Cue text rendering information for a given widget.

```
#include <Xsc/Help.h>

void XscHelpSetCueTopicDetails(
   Widget          object,
   String          topic,
   XmFontList      font_list,
   String          font_list_tag,
   unsigned char   alignment,
   unsigned char   direction,
   unsigned char   converter );
```

object	(in)	The widget of interest.
topic	(in)	The value for the `xscCueTopic` resource.
font_list	(in)	The value for the `xscCueFontList` resource.
font_list_tag	(in)	The value for the `xscCueFontListTag` resource.
alignment	(in)	The value for the `xscCueAlignment` resource.
direction	(in)	The value for the `xscCueStringDirection` resource.
converter	(in)	The value for the `xscCueStringConverter` resource.

XscHelpSetDynamicTipGroupDefault

VERSION 1.1.3

This function specifies the method used to derive the default value for the `XmNtipGroupId` resource.

CROSS-REFERENCE
Refer to page 21 for more details.

```
#include <Xsc/Help.h>

void XscHelpSetDynamicTipGroupDefault(
   Boolean flag )
```

flag	(in)	The flag should be `True` to use the new *dynamic* method and `False` to use the old *static* method.

This function can be called at any time, even before the toolkit is initialized.

XscHelpSetHintsEnabledGlobally

VERSION 1.1.1

This function records if Hints should be enabled or disabled globally.

```
#include <Xsc/Help.h>

void XscHelpSetHintsEnabledGlobally(
   Boolean flag );
```

flag	(in)	The flag should be `True` to enable and `False` to disable Hints.

No Hint within the entire application can be displayed while Hints are globally disabled.

XscHelpSetHintsEnabledOnShell

VERSION 1.1.1

This function records if Hints should be enabled or disabled within a given shell.

```
#include <Xsc/Help.h>

void XscHelpSetHintsEnabledOnShell(
   Widget  widget,
   Boolean flag );
```

widget	(in)	The WMShell widget or any child of the WMShell widget being referenced.
flag	(in)	The flag should be `True` to enable and `False` to disable Hints.

No Hint associated with a widget contained in the specified shell widget can be displayed if this attribute is set to `False`. The shell being referenced is either the specified widget (if it is a shell widget) or the closest shell ancestor to the specified widget.

XscHelpSetHintTopic

CROSS-REFERENCE
Refer to page 65 for additional instructions.

This function updates the Hint topic for a given widget.

```
#include <Xsc/Help.h>

void XscHelpSetHintTopic(
   Widget         object,
   String         topic );
```

object	(in)	The widget of interest.
topic	(in)	The value for the `xscHintTopic` resource.

XscHelpSetHintTopicDetails

CROSS-REFERENCE
Refer to page 65 for additional instructions.

This function sets all of the Hint text rendering information for a given widget.

```
#include <Xsc/Help.h>

void XscHelpSetHintTopicDetails(
   Widget          object,
   String          topic,
   XmFontList      font_list,
   String          font_list_tag,
   unsigned char   alignment,
   unsigned char   direction,
   unsigned char   converter );
```

object	(in)	The widget of interest.
topic	(in)	The value for `xscHintTopic`.
font_list	(in)	The value for `xscHintFontList`.
font_list_tag	(in)	The value for `xscHintFontLisTag`.
alignment	(in)	The value for `xscHintAlignment`.
direction	(in)	The value for `xscHintString-Direction`.
converter	(in)	The value for `xscHintString-Converter`.

XscHelpSetTipsEnabledGlobally

VERSION 1.1.1

This function records if Tips should be enabled or disabled globally.

```
#include <Xsc/Help.h>

void XscHelpSetTipsEnabledGlobally(
   Boolean flag );
```

flag	(in)	The flag should be `True` to enable and `False` to disable Tips.

No Tip within the entire application can be displayed while Tips are globally disabled.

XscHelpSetTipsEnabledOnShell

VERSION 1.1.1

This function records if Tips should be enabled or disabled within a given shell.

```
#include <Xsc/Help.h>

void XscHelpSetTipsEnabledOnShell(
   Widget  widget,
   Boolean flag );
```

widget	(in)	The WMShell widget or any child of the WMShell widget being referenced.
flag	(in)	The flag should be `True` to enable and `False` to disable Tips.

No Tip associated with a widget contained in the specified shell widget can be displayed if this attribute is set to `False`. The shell being referenced is either the specified widget (if it is a shell widget) or the closest shell ancestor to the specified widget.

XscHelpSetTipTopic

CROSS-REFERENCE
Refer to page 63 for additional instructions.

This function modifies the Tip topic of a given widget.

```
#include <Xsc/Help.h>

void XscHelpSetTipTopic(
   Widget         object,
   String         topic );
```

object	(in)	The widget of interest.
topic	(in)	The value for the `xscTipTopic` resource.

XscHelpSetTipTopicDetails

CROSS-REFERENCE
Refer to page 63 for additional instructions.

This function sets all of the Tip text rendering information for a given widget.

```
#include <Xsc/Help.h>

void XscHelpSetTipTopicDetails(
   Widget          object,
   String          topic,
   XmFontList      font_list,
   String          font_list_tag,
   unsigned char   alignment,
   unsigned char   direction,
   unsigned char   converter );
```

object	(in)	The widget of interest.
topic	(in)	The value for `xscTipTopic`.
font_list	(in)	The value for `xscTipFontList`.
font_list_tag	(in)	The value for `xscTipFontListTag`.
alignment	(in)	The value for `xscTipAlignment`.
direction	(in)	The value for `xscTipString-Direction`.
converter	(in)	The value for `xscTipString-Converter`.

XscHelpTipExists

CROSS-REFERENCE
Refer to page 63 for additional instructions.

This function indicates if the specified object has a Tip.

```
#include <Xsc/Help.h>

Boolean XscHelpTipExists( Widget object );
```

object	(in)	The widget of interest.

`True` is returned if and only if the `xscTipTopic` resource is defined for the specified widget.

XscHelpTipUpdate

CROSS-REFERENCE
Refer to page 63 for additional instructions.

This function updates the Tip of the specified widget, even if it is currently displayed.

```
#include <Xsc/Help.h>

void XscHelpTipUpdate( Widget object );
```

object (in) The target widget.

XscHelpUpdate

CROSS-REFERENCE
Refer to page 62 for additional instructions.

This function updates the screen representation of all of the XscHelp attributes for the specified widget.

```
#include <Xsc/Help.h>

void XscHelpUpdate( Widget object );
```

object (in) The target widget.

This function is generally called after the specified widget's XscHelp resources are modified.

XscHintDeriveFromWidget

CROSS-REFERENCE
Refer to page 67 for additional instructions.

This function retrieves a handle to the Hint associated with an object.

```
#include <Xsc/Hint.h>

XscHint XscHintDeriveFromWidget(
   Widget object );
```

object (in) The widget (or gadget) of interest

A handle to the Hint is returned or `NULL` if there is no Hint for the specified *object*.

XscHintGet<name>

CROSS-REFERENCE
Refer to page 68 for additional instructions.

These functions extract the current resource values from a Hint handle.

```
#include <Xsc/Hint.h>

unsigned char XscHintGetAlignment        ( XscHint );
Boolean       XscHintGetAutoDbReload     ( XscHint );
Pixel         XscHintGetBackground       ( XscHint );
Boolean       XscHintGetCompound         ( XscHint );
Boolean       XscHintGetEnabled          ( XscHint );
XmFontList    XscHintGetFontList         ( XscHint );
String        XscHintGetFontListTag      ( XscHint );
Pixel         XscHintGetForeground       ( XscHint );
Boolean       XscHintGetInheritBackground( XscHint );
Dimension     XscHintGetMarginLeft       ( XscHint );
Dimension     XscHintGetMarginRight      ( XscHint );
Dimension     XscHintGetMarginWidth      ( XscHint );
Boolean       XscHintGetMotifColorModel  ( XscHint );
unsigned char XscHintGetStringConverter  ( XscHint );
unsigned char XscHintGetStringDirection  ( XscHint );
String        XscHintGetTopic            ( XscHint );
```

`XscHint`	(in)	The handle of the Hint to examine

Each function retrieves the value of the named Hint attribute.

The functions returning `String` data types return a copy of the Hint's string attributes. The returned `String` should be deallocated with `XtFree()`.

XscHintHasValidTopic

CROSS-REFERENCE
Refer to page 68 for additional instructions.

This function determines if a Hint contains a viewable topic.

```
#include <Xsc/Hint.h>

Boolean XscHintHasValidTopic(
   XscHint hint );
```

hint	(in)	The handle of the Hint to examine

A Hint without a valid topic is not displayable. Normally, a Hint will not be created for a widget unless there is a valid Hint

topic. However, the Hint topic can be destroyed after the Hint is created.

XscHintSet<name>

CROSS-REFERENCE
Refer to page 68 for additional instructions.

These functions modify the current resource values of a Hint handle.

```
#include <Xsc/Hint.h>

/* Each of these begins with "void XscHintSet"... */

...Alignment        (XscHint,unsigned char,Boolean);
...AutoDbReload     (XscHint,Boolean      );
...Background       (XscHint,Pixel        );
...Compound         (XscHint,Boolean      );
...Enabled          (XscHint,Boolean      );
...FontList         (XscHint,XmFontList   ,Boolean);
...FontListTag      (XscHint,String       ,Boolean);
...Foreground       (XscHint,Pixel        );
...InheritBackground(XscHint,Boolean      );
...MarginLeft       (XscHint,Dimension    );
...MarginRight      (XscHint,Dimension    );
...MarginWidth      (XscHint,Dimension    );
...MotifColorModel  (XscHint,Boolean      );
...StringConverter  (XscHint,unsigned char,Boolean);
...StringDirection  (XscHint,unsigned char,Boolean);
...Topic            (XscHint,String       );
```

`XscHint`	(i/o)	The handle of the Hint to modify

Each function sets the value of the named Hint attribute.

The functions setting a `String` data type internally make a copy of the string.

Some of the functions have a third, `Boolean` parameter; each of these functions relates to an attribute used to create a compound string for rendering. If the third parameter is set to `True`, then the internal compound string is created immediately. If you are setting more than one of these resources, set the third parameter to `True` only for the last one. The function `XscHint.`

XscTipDeriveFromWidget

CROSS-REFERENCE
Refer to page 65 for additional instructions.

This function retrieves a handle to the Tip associated with an object.

```
#include <Xsc/Tip.h>

XscTip XscTipDeriveFromWidget(
   Widget object );
```

object	(in)	The widget (or gadget) of interest

A handle to the Tip is returned or NULL if there is no Tip for the specified *object*.

XscTipGet<name>

CROSS-REFERENCE
Refer to page 65 for additional instructions.

These functions extract the current resource values from a Tip handle.

```
#include <Xsc/Tip.h>

unsigned char XscTipGetAlignment      ( XscTip );
Boolean       XscTipGetAutoDbReload   ( XscTip );
Pixel         XscTipGetBackground     ( XscTip );
Pixel         XscTipGetBorderColor    ( XscTip );
Dimension     XscTipGetBorderWidth    ( XscTip );
Pixel         XscTipGetColorBase      ( XscTip );
Boolean       XscTipGetCompound       ( XscTip );
Boolean       XscTipGetEnabled        ( XscTip );
XmFontList    XscTipGetFontList       ( XscTip );
String        XscTipGetFontListTag    ( XscTip );
Pixel         XscTipGetForeground     ( XscTip );
int           XscTipGetGroupId        ( XscTip );
Dimension     XscTipGetMarginBottom   ( XscTip );
Dimension     XscTipGetMarginHeight   ( XscTip );
Dimension     XscTipGetMarginLeft     ( XscTip );
```

```
Dimension     XscTipGetMarginRight    ( XscTip );
Dimension     XscTipGetMarginTop      ( XscTip );
Dimension     XscTipGetMarginWidth    ( XscTip );
Boolean       XscTipGetMotifColorModel( XscTip );
unsigned long XscTipGetPopdownInterval( XscTip );
unsigned long XscTipGetPopupInterval  ( XscTip );
unsigned char XscTipGetPosition       ( XscTip );
Dimension     XscTipGetShadowThickness( XscTip );
unsigned char XscTipGetShadowType     ( XscTip );
unsigned char XscTipGetStringConverter( XscTip );
unsigned char XscTipGetStringDirection( XscTip );
String        XscTipGetTopic          ( XscTip );
int           XscTipGetXOffset        ( XscTip );
int           XscTipGetYOffset        ( XscTip );
```

XscTip	(in)	The handle of the Tip to examine

Each function retrieves the value of the named Tip attribute.

The functions returning `String` data types return a copy of the Tip's string attributes. The returned `String` should be deallocated with `XtFree()`.

XscTipHasValidTopic

CROSS-REFERENCE
Refer to page 65 for additional instructions.

This function determines if a Tip contains a viewable topic.

```
#include <Xsc/Tip.h>

Boolean XscTipHasValidTopic(
   XscTip tip );
```

tip	(in)	The handle of the Tip to examine

A Tip without a valid topic is not displayable. Normally, a Tip will not be created for a widget unless there is a valid Tip topic. However, the Tip topic can be destroyed after the Tip is created.

XscTipSet<name>

CROSS-REFERENCE
Refer to page 65 for additional instructions.

These functions modify the current resource values of a Tip handle.

```
#include <Xsc/Tip.h>

/* Each of these begins with "void XscTipSet"... */

...Alignment      ( XscTip, unsigned char, Boolean );
...AutoDbReload   ( XscTip, Boolean       );
...Background     ( XscTip, Pixel         );
...BorderColor    ( XscTip, Pixel         );
...BorderWidth    ( XscTip, Dimension     );
...ColorBase      ( XscTip, Pixel         );
...Compound       ( XscTip, Boolean       );
...Enabled        ( XscTip, Boolean       );
...FontList       ( XscTip, XmFontList   , Boolean );
...FontListTag    ( XscTip, String       , Boolean );
...Foreground     ( XscTip, Pixel         );
...GroupId        ( XscTip, int           );
...MarginBottom   ( XscTip, Dimension     );
...MarginHeight   ( XscTip, Dimension     );
...MarginLeft     ( XscTip, Dimension     );
...MarginRight    ( XscTip, Dimension     );
...MarginTop      ( XscTip, Dimension     );
...MarginWidth    ( XscTip, Dimension     );
...MotifColorModel( XscTip, Boolean       );
...PopdownInterval( XscTip, unsigned long );
...PopupInterval  ( XscTip, unsigned long );
...Position       ( XscTip, unsigned char );
...ShadowThickness( XscTip, Dimension     );
...ShadowType     ( XscTip, unsigned char );
...StringConverter( XscTip, unsigned char, Boolean );
...StringDirection( XscTip, unsigned char, Boolean );
...Topic          ( XscTip, String        );
...XOffset        ( XscTip, int           );
...YOffset        ( XscTip, int           );
```

`XscTip`	(i/o)	The handle of the Tip to modify

Each function sets the value of the named Tip attribute.

The functions setting a `String` data type internally make a copy of the string.

Some of the functions have a third, `Boolean` parameter; each of these functions relates to an attribute used to create a com-

pound string for rendering. If the third parameter is set to `True`, then the internal compound string is created immediately. If you are setting more than one of these resources, set the third parameter to `True` only for the last one. The function `XscTipSetTopic()` implicitly creates an internal compound string.

CHAPTER 11

Macros and Constants

XmCR_XSC_HELP_CONTEXT_CALLBACK

This macro specifies the callback reason code indicating that context-sensitive help was requested via the Help or F1 key.

```
#include <Xsc/Help.h>

#define XmCR_XSC_HELP_CONTEXT_CALLBACK ...
```

This macro returns the callback reason code used by the Help ToolKit when context-sensitive help is requested via the Help or F1 key. This value can be controlled to prevent XscHelp callback reason codes from conflicting with the callback reason codes defined by other third-party products.

Refer to the `_XscCROffset` external variable for information on changing the default reason codes.

XmCR_XSC_HELP_CONTEXT_GRAB_SELECT

This macro specifies the callback reason code indicating that context-sensitive help was requested by allowing the user to select the desired widget or gadget.

```
#include <Xsc/Help.h>

#define XmCR_XSC_HELP_CONTEXT_GRAB_SELECT
...
```

This macro returns the callback reason code used by the Help ToolKit when context-sensitive help is requested by allowing the user to select the desired widget or gadget. This can be done by using the `XscHelpContextPickAndActi-`

vate() function. This value can be controlled to prevent XscHelp callback reason codes from conflicting with the callback reason codes defined by other third-party products.

Refer to the _XscCROffset external variable for information on changing the default reason codes.

XscHelpREGISTERED

VERSION 1.1

This macro constant indicates if the Help ToolKit is the registered version of the unregistered "evaluation" version. The evaluation version contains a subset of the capabilities provided in the registered version.

```
#include <Xsc/Help.h>

#define XscHelpREGISTERED 1
```

The constant has the value 1 if it is the registered version and 0 if it is the evaluation version.

CHAPTER 12

Data Types

XscCue

This is an opaque handle to the Cue of a widget or gadget.

```
#include <Xsc/Cue.h>

typedef struct _XscCueRec *XscCue;
```

This type is retrieved using the function `XscCueDeriveFromWidget()` and is used to progamatically get and set the Cue values associated with a given widget (or gadget).

XscHelpContextCallbackStruct

This data type provides the retrieved context-sensitive help to the application via the context-help callback function.

```
#include <Xsc/Help.h>

typedef struct
{
   int       reason;
   XEvent*   event;
   int       depth;
   XtPointer data;
}
XscHelpContextCallbackStruct;
```

This data type is passed as the third parameter to the context-help callback function.

reason The *reason* field specifies why the callback function is called; it will be one of the following values:

- `XmCR_XSC_HELP_CONTEXT_CALLBACK`
- `XmCR_XSC_HELP_CONTEX_GRAB_SELECT`

event This field contains the X event that caused the context-help lookup; the *event* field may be NULL if there is no associated X event.

depth The *depth* member specifies how many ancestors where examined before the help data was located. A value of 0 indicates that the help data belongs to the target object.

data This member points to the data structure that contains the actual retrieved context-sensitive help data. If *data* is NULL, then no context-sensitive help could be found for the object or for any of its ancestors (up to and including the closest toplevel shell widget.)

XscHint

This is an opaque handle to the Hint of a widget or gadget.

```
#include <Xsc/Hint.h>
```

```
typedef struct _XscHintRec *XscHint;
```

This type is retrieved using the function `XscHintDeriveFromWidget()` and is used to progamatically get and set the Hint values associated with a given widget (or gadget).

XscTip

This is an opaque handle to the Tip of a widget or gadget.

```
#include <Xsc/Tip.h>
```

```
typedef struct _XscTipRec *XscTip;
```

This type is retrieved using the function `XscTipDeriveFromWidget()` and is used to progamatically get and set the Tip values associated with a given widget (or gadget).

CHAPTER 13

External Variables

_XscCROffset

This external variable is used to adjust the base value of the Help ToolKit callback reason codes.

```
#include <Xsc/Help.h>
```

```
extern int _XscCROffset;
```

`_XscCROffset` is used to modify the base value of the Help ToolKit callback reasons. All of the callback reasons are sequential offsets from the base value.

`_XscCROffset` is not used at run-time to determine the callback reason values; instead, it is used to initialize an internal variable when XscHelp is initialized. As a result, modifying `_XscCROffset` after XscHelp is initialized will have no impact on the callback reason values.

CHAPTER 14

Resources

This chapter describes all of the resource used by the Help ToolKit. The resources are divided into the following categories:

- CDE Context-Help resources
- Cue resources
- Hint resources
- Tip resources

CDE Context-Help Resources

Table 14-1 shows all of the defined CDE Context-Help resources. The literal resource names are used in the table. The header file `<Xsc/CdeHelp.h>` defines macros for the standard XmN/XmC name/class pairs for each resource.

TABLE 14-1: CDE Context-Help Resources

Name	Type	Default
xscCdeHelpColumns	Dimension	70
xscCdeHelpDialogTitle	String	*dynamic*
xscCdeHelpRows	Dimension	25
xscCdeHelpType	unsigned char	DtHELP_TYPE_TOPIC
xscCdeHelpTopic	String	NULL
xscCdeHelpTopicTitle	String	NULL
xscCdeHelpVolume	String	*dynamic*
xscCdeHelpWidgetType	unsigned char	XmXSC_GENERAL_-HELP_WIDGET

xscCdeHelpColumns

This is derived from the standard CDE Motif `DtNcolumns` resource that specifies, in characters, how wide the text display area on the Help widget should be. The default is 70.

xscCdeHelpDialogTitle

This is derived from the standard Motif dialog title resource used to specify the title of a dialog; the title is displayed in the window manager title bar decorations. If the dialog title resource is not explicitly set, then the dialog title is derived based on the value of the resource `xscCdeHelpType`.

`xscCdeHelpType` (`DtHELP_TYPE_`)	**Default for `xscCdeHelpDialogTitle`**
DYNAMIC_STRING	"Topic: " followed by the value of the `xscCdeHelpTopicTitle`
FILE	"Topic: " followed by the value of the `xscCdeHelpTopicTitle`; if a topic title is not specified, then "Topic: " followed by the name of the file being displayed (`xscCdeHelpTopic`)
MAN_PAGE	"Man Page: " followed by the name of the man page (`xscCdeHelpTopic`)
STRING	"Topic: " followed by the value of the `xscCdeHelpTopicTitle`
TOPIC	The name of the help volume as specified by the `xscCdeHelpVolume` resource

xscCdeHelpRows

This is derived from the standard CDE Motif `DtNrows` resource that specifies, in lines, how tall the text display area on the Help widget should be. The default is 25.

xscCdeHelpTopic

The `xscCdeHelpTopic` resource specifies what should be displayed for a widget (or gadget) when context-sensitive help is requested. By definition, if this resource is not defined for a

widget, then no CDE help information is defined and the widget's parent is examined. This process continues until CDE help data is located or all of the widget ancestors have been searched.

The value of the topic is interpreted differently based on the value of the `xscCdeHelpType` resource; refer to `xscCdeHelpType` for details.

The default value is NULL.

xscCdeHelpTopicTitle

This resource is displayed in the help widget as the name of the help topic being displayed. The default value is NULL.

xscCdeHelpType

The CDE help widgets can display five different types of help as specified by this resource.

Value of resource (`DtHELP_TYPE_`)	Type of help to display
`DYNAMIC_STRING`	Displays a string with simple formatting — text is automatically word-wrapped and newline characters represent paragraph breaks.
`FILE`	Displays the contents of a text file.
`MAN_PAGE`	Formats and displays an installed man page.
`STRING`	Displays a string without any formatting. New lines must be explicitly marked by a newline character.
`TOPIC`	Displays help information from the help volume specified by the `xscCdeHelpVolume` resource. This is the default value.

xscCdeHelpVolume

This resource is used to specify the name of the help volume to reference for a given widget or gadget. The default help vol-

ume is specified when the XscCdeHelpInstall() function is called. If a NULL is specified for the default volume name, then the value of the environment variable XSC_CDE_HELP_VOLUME is used. If the environment variable is undefined, then the CDE Help System will report an error.

xscCdeHelpWidgetType

The xscCdeHelpWidgetType resource specifies the type of CDE widget to use when viewing the help information. The allowed values are

- XmXSC_GENERAL_HELP_WIDGET
- XmXSC_QUICK_HELP_WIDGET

The default value is XmXSC_GENERAL_HELP_WIDGET, which displays the full-featured CDE help widget. If the resource is set to XmXSC_QUICK_HELP_WIDGET, then the quick help widget is used instead.

Cue Resources

Table 14-2 shows information about each Cue resource. The literal resource names are used in the table. The header file <Xsc/StrDef.h> defines macros for the standard XmN/XmC name/class pairs for each resource.

TABLE 14-2: Cue resources

Name	Type	Default
xscCueAlignment	unsigned char	XmALIGNMENT_-BEGINNING
xscCueAutoDbReload	Boolean	False
xscCueBackground	Pixel	Gold
xscCueBorderColor	Pixel	Black
xscCueBorderWidth	Dimension	1
xscCueColorBase	Pixel	Gold
xscCueEnabled	Boolean	True
xscCueFontList	XmFontList	*dynamic*
xscCueFontListTag	String	NULL
xscCueForeground	Pixel	Black

TABLE 14-2: Cue resources

	xscCueMarginBottom	Dimension	0
	xscCueMarginHeight	Dimension	1
	xscCueMarginLeft	Dimension	0
	xscCueMarginRight	Dimension	0
	xscCueMarginTop	Dimension	0
	xscCueMarginWidth	Dimension	2
	xscCueMotifColorModel	Boolean	False
	xscCuePosition	unsigned char	XmXSC_CUE_-POSITION_SHELL
	xscCueShadowThickness	Dimension	0
	xscCueShadowType	unsigned char	XmSHADOW_OUT
Version 1.1	xscCueShowName	unsigned char	XmXSC_SHOW_NAME_NONE
	xscCueStringConverter	unsigned char	XmXSC_STRING_CON-VERTER_STANDARD
	xscCueStringDirection	unsigned char	XmSTRING_DIREC-TION_L_TO_R
	xscCueTopic	String	NULL
	xscCueXOffset	int	0
	xscCueYOffset	int	*dynamic*

xscCueAlignment

The standard Motif text alignment values are used to specify the text alignment of the Cue. The values that can be specified are:

- `XmALIGNMENT_BEGINNING`
- `XmALIGNMENT_CENTER`
- `XmALIGNMENT_END`.

The default is `XmALIGNMENT_BEGINNING`.

xscCueAutoDbReload

If this resource is set to `True`, then all of a widget's Cue resources are reloaded each time the Cue is displayed. This allows Cue related changes in the resource database to be automatically realized. The default value is `False`.

xscCueBackground

This resource specifies the background color of the Cue window. The default is “Gold”.

xscCueBorderColor

This resource specifies the border color of the Cue window. The default is “Black”.

xscCueBorderWidth

This resource specifies the width of the Cue window border. The default is 1 pixel.

xscCueColorBase

This resource specifies the color to use when deriving the proper shadow border colors. The color specified in this resource is never directly displayed, but rather, is used in the shadow border color calculations. This resource is used only if `xscCueMotifColorModel` is set to `False`. The default is “Gold”.

xscCueEnabled

If this resource is set to `False`, the associated Cue is not displayed under any circumstance. The default is `True`.

xscCueFontList

This resource specifies the font list to use when rendering the Cue text. The default is implementation dependent.

xscCueFontListTag

This resource specifies the font list tag to use when building the compound string for the Cue text. It is referenced only if the resource `xscCueStringConverter` is set to the value `XmSTRING_CONVERTER_FONT_TAG`. The default is NULL, indicating that the first entry in the font list should be used.

xscCueForeground

This resource specifies the foreground color of the text rendered for the Cue. The default is "Black". This value is referenced only if `xscCueMotifColorModel` is set to `False`.

xscCueMarginBottom

This resource specifies the number of extra pixels below the rendered Cue text. The default is 0.

xscCueMarginHeight

This resource specifies the number of extra pixels above and below the rendered Cue text. The default is 1.

xscCueMarginLeft

This resource specifies the number of extra pixels to the left of the rendered Cue text. The default is 0.

xscCueMarginRight

This resource specifies the number of extra pixels to the right of the rendered Cue text. The default is 0.

xscCueMarginTop

This resource specifies the number of extra pixels above the rendered Cue text. The default is 0.

xscCueMarginWidth

This resource specifies the number of extra pixels to the right and left of the rendered Cue text. The default is 2.

xscCueMotifColorModel

This resource controls if the Cue colors are specified separately or derived from the actual background color. If this resource is set to `True`, then the foreground and shadow border colors are derived from the value of the `xscCueBackground`

resource. If this resource is set to `False`, then the foreground color is specified by the `xscCueForeground` resource and the shadow border colors are derived from the value of the `xscCueColorBase` resource. The default is `False`.

xscCuePosition

This resource controls where the Cue window is displayed. The possible values are:

Value[a]	Description of compound string converters
`BOTTOM_-BEGINNING`	Place the Cue below the object, flush with the beginning edge of the Cue window.
`BOTTOM_END`	Place the Cue below the object, flush with the ending edge of the Cue window.
`BOTTOM_LEFT`	Place the Cue below the object, flush left
`BOTTOM_RIGHT`	Place the Cue below the object, flush right.
`SHELL`	Place the Cue above the shell.
`TOP_BEGIN-NING`	Place the Cue above the object, flush with the beginning edge of the Cue window.
`TOP_END`	Place the Cue above the object, flush with the ending edge of the Cue window.
`TOP_LEFT`	Place the Cue above the object, flush left
`TOP_RIGHT`	Place the Cue above the object, flush right.

a. Each value actually begins with the prefix "`XmXSC_CUE_POSITION_`"

`XmXSC_CUE_POSITION_SHELL` is the default value for this resource.

xscCueShadowThickness

This resource specifies the shadow width of the Cue window shadow borders. The default is 0.

xscCueShadowType

This resources specifies the appearance of the shadow borders. The values are:

- XmSHADOW_IN
- XmSHADOW_OUT
- XmSHADOW_ETCHED_IN
- XmSHADOW_ETCHED_OUT.

The default is XmSHADOW_OUT.

xscCueShowName

VERSION 1.1

Display the name of a given widget in its Cue window. If there is cue text, the widget name appears below the text; otherwise, the Cue window contains only the widget name. This resource can be used to help determine the name to use in a resource file when specifying resources for a widget. The value of this resource indicates how the widget's name is displayed, as indicated below:

- XmXSC_SHOW_NAME_ALL
- XmXSC_SHOW_NAME_NONE
- XmXSC_SHOW_NAME_SELF
- XmXSC_SHOW_NAME_SHELL

The default value is XmXSC_SHOW_NAME_NONE. Each is described in the following list:

Value[a]	Description of show name values
ALL	The names of all widget ancestors and the widget itself are displayed, e.g., *top.one.two.shell.three.name
NONE	The widget name is not displayed
SELF	Only the name of the widget is displayed, e.g., *name
SHELL	The names of the widget's closest window manager shell and the widget itself are displayed, e.g., *shell.name

a. Each value actually begins with the prefix "XmXSC_SHOW_NAME_"

xscCueStringConverter

This resource specifies how the Cue text should be converted into a compound string. The values defined for this resource are:

- XmSTRING_CONVERTER_STANDARD
- XmSTRING_CONVERTER_FONT_TAG
- XmSTRING_CONVERTER_SEGMENTED

The default is XmSTRING_CONVERTER_STANDARD. Each is described in the following list:

Value[a]	Description of compound string converters
FONT_TAG	The entire compound string is assigned the font tag defined via the xscCueFontListTag resource. Newline characters are recognized and handled.
SEGMENTED	The converter checks for newlines, as well as, *escape sequences* that adjust how the text is rendered. An escape sequence always begins with the '@' character. Three escape sequences are currently defined: @@ This escape sequence indicates that a single '@' character is to be rendered. @d This escape sequence changes the rendering direction for the remainder of the text or until changed. The sequence "@d<" changes direction to right-to-left and "@d>" changes the direction to left-to-right. @f This escape sequence changes the font tag for the remainder of the text or until changed. The syntax is "@f[tag]" where "tag" is the value of the desired tag. A null-tag (e.g., "@f[]") forces the tag to have the value default value.
STANDARD	The standard Motif converter using the default font tag.

a. Each value actually begins with the prefix "XmSTRING_CONVERTER"

xscCueStringDirection

This resource specifies the direction that the Cue text should be rendered. The values are:

- `XmSTRING_DIRECTION_L_TO_R`
- `XmSTRING_DIRECTION_R_TO_L`

`XmSTRING_DIRECTION_L_TO_R`. is the default value.

xscCueTopic

This resource specifies the text to render in the Cue pop-up. If no text is specified, then no Cue management data structures are allocated for this object and no Cue text is displayed.

xscCueXOffset

This resource specifies the X offset to add to the calculated location of the Cue window. The default is 0.

xscCueYOffset

This resource specifies the Y offset to add to the calculated location of the Cue window. The default value for this resource is -10 pixels if the `xscCuePosition` resource equals `XmCUE_POSITION_SHELL`. and 0 pixels otherwise.

Hint Resources

Table 14-3 shows information about each Hint resource. The literal resource names are used in the table. The header file `<Xsc/StrDef.h>` defines macros for the standard XmN/XmC name/class pairs for each resource.

TABLE 14-3: Hint Resources

Name	Type	Default
xscHintAlignment	unsigned char	XmALIGNMENT_-BEGINNING
xscHintAutoDbReload	Boolean	False
xscHintBackground	Pixel	*dynamic*
xscHintCompound	Boolean	False

TABLE 14-3: Hint Resources

	xscHintEnabled	Booelan	True
	xscHintFontList	XmFontList	*dynamic*
	xscHintFontListTag	String	NULL
	xscHintForeground	Pixel	Black
	xscHintInheritBack-ground	Boolean	True
	xscHintMarginLeft	Dimension	0
	xscHintMarginRight	Dimension	0
	xscHintMarginWidth	Dimension	2
	xscHintMotifColorModel	Boolean	True
Version 1.1	xscHintShowName	unsigned char	XmXSC_SHOW_NAME_NONE
	xscHintStringConverter	unsigned char	XmXSC_STRING_CONVERTER_STANDARD
	xscHintStringDirection	unsigned char	XmSTRING_DIRECTION_L_TO_R
	xscHintTopic	String	NULL

xscHintAlignment

The standard Motif text alignment values are used to specify the text alignment of the Hint. The values that can be specified are:

- `XmALIGNMENT_BEGINNING`
- `XmALIGNMENT_CENTER`
- `XmALIGNMENT_END`

The default is `XmALIGNMENT_BEGINNING`.

xscHintAutoDbReload

If this resource is set to `True`, then all of a widget's Hint resources are reloaded each time the Hint is displayed. This allows Hint related changes in the resource database to be automatically realized. The default value is `False`.

xscHintBackground

This resource specifies the background color of the hint display widget. This resource is only referenced if the value of the `xscHintInheritBackground` resource is set to `False`. The default is “Grey”.

xscHintCompound

This resource is used to make a collection of widgets and gadgets act like a single object. If the `xscHintCompound` resource is set to `True`, then all children (not including shells) of this widget are ignored with regards to Hint processing. The default is `False`. This resource can be used to make composite widgets act like a single widget.

xscHintEnabled

If this resource is set to `False`, the associated hint is not displayed under any circumstance. The default is `True`.

xscHintFontList

This resource specifies the font list to use when rendering the Hint text. The default is implementation dependent.

xscHintFontListTag

This resource specifies the font list tag to use when building the compound string for the Hint text. It is referenced only if the value of the resource `xscHintStringConverter` is set to `XmSTRING_CONVERTER_FONT_TAG`. The default is NULL, indicating that the first entry in the font list should be used.

xscHintForeground

This resource specifies the foreground color of the text rendered in the hint display widget. The default is “Black”. This value is referenced only if `xscHintMotifColorModel` is set to `False`.

xscHintInheritBackground

If this resource is set to `True`, then the background color of the Hint will be the same as the background color of the hint display widget's immediate parent. If this resource is set to `False`, then the background color of the Hint is specified by the `xscHintBackground` resource.

xscHintMarginLeft

This resource specifies the number of extra pixels to the left of the rendered text. The default is 0.

xscHintMarginRight

This resource specifies the number of extra pixels to the right of the rendered text. The default is 0.

xscHintMarginWidth

This resource specifies the number of extra pixels to the right and left of the rendered text. The default is 2.

xscHintMotifColorModel

This resource controls if the Hint colors are specified separately or derived from the background color. If this resource is set to `True`, then the foreground color is derived from the either the value of the `xscHintBackground` resource or the background color of the widget's immediate parent (if `xscHintInheritBackground` is set to `True`.). If the `xscHintMotifColorModel` resource is set to `False`, then the foreground color is specified explicitly via the `xscHintForeground` resource. The default is `True`.

xscHintShowName

VERSION 1.1 Display the name of a given widget in its the Hint area. If there is hint text, the widget name appears below the text; otherwise, the Hint area contains only the widget name. This resource can be used to help determine the name to use in a resource file

when specifying resources for a widget. The value of this resource indicates how the widget's name is displayed, as indicated below:

- XmXSC_SHOW_NAME_ALL
- XmXSC_SHOW_NAME_NONE
- XmXSC_SHOW_NAME_SELF
- XmXSC_SHOW_NAME_SHELL

The default value is `XmXSC_SHOW_NAME_NONE`. Each is described in the following list:

Value[a]	Description of show name values
`ALL`	The names of all widget ancestors and the widget itself are displayed, e.g., `*top.one.two.shell.three.name`
`NONE`	The widget name is not displayed
`SELF`	Only the name of the widget is displayed, e.g., `*name`
`SHELL`	The names of the widget's closest window manager shell and the widget itself are displayed, e.g., `*shell.name`

a. Each value actually begins with the prefix "`XmXSC_SHOW_NAME_`"

xscHintStringConverter

This resource specifies how the Hint text should be converted into a compound string. The values defined for this resource are:

- `XmSTRING_CONVERTER_STANDARD`
- `XmSTRING_CONVERTER_FONT_TAG`
- `XmSTRING_CONVERTER_SEGMENTED`

The default value is XmSTRING_CONVERTER_STANDARD. Each is described in the following list:

Value[a]	Description of compound string converters
FONT_TAG	The entire compound string is assigned the font tag defined via the xscHintFontListTag resource. Newline characters are recognized and handled.
SEGMENTED	The converter checks for newlines, as well as, *escape sequences* that adjust how the text is rendered. An escape sequence always begins with the '@' character. Three escape sequences are currently defined: @@ This escape sequence indicates that a single '@' character is to be rendered. @d This escape sequence changes the rendering direction for the remainder of the text or until changed. The sequence "@d<" changes direction to right-to-left and "@d>" changes the direction to left-to-right. @f This escape sequence changes the font tag for the remainder of the text or until changed. The syntax is "@f[tag]" where "tag" is the value of the desired tag. A null-tag (e.g., "@f[]") forces the tag to have the value default value.
STANDARD	The standard Motif converter using the default font tag.

a. Each value actually begins with the prefix "XmSTRING_CONVERTER"

xscHintStringDirection

This resource specifies the direction that the Hint text should be rendered. The values are:

- XmSTRING_DIRECTION_L_TO_R
- XmSTRING_DIRECTION_R_TO_L

XmSTRING_DIRECTION_L_TO_R. is the default value.

xscHintTopic

This resource specifies the text to be rendered in the Hint display area. If no text is specified, then no Hint management data structures are allocated for this object and no hint text can be displayed.

Tip Resources

Table 14-4 shows information about each Tip resource. The literal resource names are used in the table. The header file `<Xsc/StrDef.h>` defines macros for the standard XmN/XmC name/class pairs for each resource.

TABLE 14-4: Tip resources

Name	Type	Default
xscTipAlignment	unsigned char	XmALIGNMENT_CENTER
xscTipAutoDbReload	Boolean	False
xscTipBackground	Pixel	Yellow
xscTipBorderColor	Pixel	Black
xscTipBorderWidth	Dimension	1
xscTipColorBase	Pixel	Yellow
xscTipCompound	Boolean	False
xscTipEnabled	Boolean	True
xscTipFontList	XmFontList	*dynamic*
xscTipFontListTag	String	NULL
xscTipForeground	Pixel	Black
xscTipGroupId	int	*dynamic*
xscTipGroupOverride (Deprecated)	*special*	XmXSC_TIP_GROUP_NULL
xscTipMarginBottom	Dimension	0
xscTipMarginHeight	Dimension	1
xscTipMarginLeft	Dimension	0
xscTipMarginRight	Dimension	0
xscTipMarginTop	Dimension	0
xscTipMarginWidth	Dimension	2
xscTipMotifColorModel	Boolean	True
xscTipPopdownInterval	int	0
xscTipPopupInterval	int	1000
xscTipPosition	unsigned char	XmXSC_TIP_POSITION_POINTER
xscTipSelectNameInterval (Version 1.1)	unsigned long	-1
xscTipShadowThickness	Dimension	0
xscTipShadowType	unsigned char	XmSHADOW_OUT

TABLE 14-4: Tip resources

Version 1.1

xscTipShowName	unsigned char	XmXSC_SHOW_NAME_NONE
xscTipStringConverter	unsigned char	XmXSC_STRING_CONVERTER_-STANDARD
xscTipStringDirection	unsigned char	XmSTRING_DIRECTION_L_TO_R
xscTipTopic	String	NULL
xscTipXOffset	int	0
xscTipYOffset	int	*dynamic*

xscTipAlignment

The standard Motif text alignment values are used to specify the text alignment of the Tip. The values that can be specified are:

- `XmALIGNMENT_BEGINNING`
- `XmALIGNMENT_CENTER`
- `XmALIGNMENT_END`.

The default is `XmALIGNMENT_CENTER`.

xscTipAutoDbReload

If this resource is set to `True`, then all of a widget's Tip resources are reloaded each time the Tip is displayed. This allows Tip related changes in the resource database to be automatically realized. The default value is `False`.

xscTipBackground

This resource specifies the background color of the Tip window. The default is "Yellow".

xscTipBorderColor

This resource specifies the border color of the Tip window. The default is "Black".

xscTipBorderWidth

This resource specifies the width of the Tip window border. The default is 1 pixel.

xscTipColorBase

This resource specifies the color to use when deriving the proper shadow border colors. The color specified in this resource is never directly displayed, but rather, is used in the shadow border color calculations. This resource is used only if `xscTipMotifColorModel` is set to `False`. The default is "Yellow".

xscTipCompound

This resource is used to make a collection of widgets and gadgets act like a single object. If the `xscTipCompound` resource is set to `True`, then all children (not including shells) of this widget are ignored with regards to Tip processing. The default is `False`. This resource can be used to make composite widgets act like a single widget.

xscTipEnabled

If this resource is set to `False`, the associated Tip is not displayed under any circumstance. The default is `True`.

xscTipFontList

This resource specifies the font list to use when rendering the Tip text. The default is implementation dependent.

xscTipFontListTag

This resource specifies the font list tag to use when building the compound string for the Tip text. It is referenced only if the resource `xscTipStringConverter` is set to the value `XmSTRING_CONVERTER_FONT_TAG`. The default is NULL, indicating that the first entry in the font list should be used.

xscTipForeground

This resource specifies the foreground color of the text rendered for the Tip. The default is "Black". This value is referenced only if `xscTipMotifColorModel` is set to `False`.

xscTipGroupId

This resource associates a widget with a Tip group. Tips in a Tip group are displayed immediately (without waiting the time specified via the `xscTipPopupInterval` resource) if a member of the Tip group was previously displayed.

The `xscTipGroupId` resource can be assigned an integer value between 1 and 10000, inclusive. There are two special values that can also be used:

- `XmXSC_TIP_GROUP_PARENT`
- `XmXSC_TIP_GROUP_SELF`

Setting the `xscTipGroupId` resource to the value `XmXSC_TIP_GROUP_PARENT` forces the widget to have the same group as its parent.

Using `XmXSC_TIP_GROUP_SELF` assigns a unique value to the widget's `xscTipGroupId` resource.

Starting with Version 1.1.2, there are two algorithms used to determine the default value: the old *static* method and the new *dynamic* method.

Beginning with version 1.1.2, there are two methods for deriving the default for this resource. The *static* method was the only method prior to version 1.1.2. In the static method, the default is always `XmXSC_TIP_GROUP_PARENT`. The new *dynamic* method makes the default value equal to `XmXSC_TIP_GROUP_SELF` as long as all of the widget's ancestors also use the default; if an ancestor has an explicit value for this resource (even if that value is the symbolic value `XmXSC_TIP_GROUP_SELF`), then the default value is `XmXSC_TIP_GROUP_PARENT`. The dynamic algorithm is used by default.

Version 1.1.2 note:

The symbolic values `XmXSC_TIP_GROUP_PARENT` and `XmXSC_TIP_GROUP_SELF` can be directly used for this resource, making the `xscTipGroupOverride` resource functionally obsolete.

xscTipGroupOverride

VERSION 1.1
Deprecated

This resource is used to set the value of the `xscTipGroupId` resource to a symbolic value. The available symbols are:

- `XmXSC_TIP_GROUP_NULL` (default)
- `XmXSC_TIP_GROUP_PARENT`
- `XmXSC_TIP_GROUP_SELF`

If this resource is set to `XmXSC_TIP_GROUP_NULL`, then the resource `xscTipGroupId` is handled normally. However, if this resource is set to `XmXSC_TIP_GROUP_PARENT` or `XmXSC_TIP_GROUP_SELF`, then the `xscTipGroupId` resource is given a value as described in the section for resource `xscTipGroupId`.

Version 1.1.2 note:

This resource is functionally obsolete due to changes in the `xscTipGroupId` resource. As a result, this resource has been deprecated.

xscTipMarginBottom

This resource specifies the number of extra pixels below the rendered Tip text. The default is 0.

xscTipMarginHeight

This resource specifies the number of extra pixels above and below the rendered Tip text. The default is 1.

xscTipMarginLeft

This resource specifies the number of extra pixels to the left of the rendered Tip text. The default is 0.

xscTipMarginRight

This resource specifies the number of extra pixels to the right of the rendered Tip text. The default is 0.

xscTipMarginTop

This resource specifies the number of extra pixels above the rendered Tip text. The default is 0.

xscTipMarginWidth

This resource specifies the number of extra pixels to the right and left of the rendered Tip text. The default is 2.

xscTipMotifColorModel

This resource controls if the Tip colors are specified separately or derived from the actual background color. If this resource is set to `True`, then the foreground and shadow border colors are derived from the value of the `xscTipBackground` resource. If this resource is set to `False`, then the foreground color is specified by the `xscTipForeground` resource and the shadow border colors are derived from the value of the `xscTipColorBase` resource. The default is `True`.

xscTipPopdownInterval

This resource specified the amount of time, in milliseconds, that the Tip should be displayed before it is automatically popped down. The default is 0, which indicates that the Tip window should not be automatically popped down.

xscTipPopupInterval

This resource specified the amount of time, in milliseconds, that the mouse should stay at rest over the object before the associated Tip is displayed. The default is 1000.

xscTipPosition

This resource controls where the Tip window is displayed. The possible values are:

Value[a]	Description of compound string converters
BOTTOM_-BEGINNING	Place the Tip below the object, flush with the beginning edge of the Tip window.
BOTTOM_END	Place the Tip below the object, flush with the ending edge of the Tip window.
BOTTOM_LEFT	Place the Tip below the object, flush left
BOTTOM_RIGHT	Place the Tip below the object, flush right.
POINTER	Place the Tip relative to the current location of the mouse pointer.
TOP_BEGIN-NING	Place the Tip above the object, flush with the beginning edge of the Tip window.
TOP_END	Place the Tip above the object, flush with the ending edge of the Tip window.
TOP_LEFT	Place the Tip above the object, flush left
TOP_RIGHT	Place the Tip above the object, flush right.

a. Each value actually begins with the prefix "XmXSC_TIP_POSITION_"

XmXSC_TIP_POSITION_POINTER is the default value for this resource.

xscTipSelectNameInterval

VERSION 1.1

This is used to control if and when the widget name displayed in the Tip window (see xscTipShowName) is selected. This resource specifies the number of milliseconds to wait before selecting the name. Once selected, the name can be copied just like and other text selection. There is no other way to select the displayed name in the Tip window. If the interval is set to -1, then the widget name is not selected at all. This resource is only used if the xscTipShowName resource is not set to XmXSC_SHOW_NAME_NONE.

xscTipShadowThickness

This resource specifies the shadow width of the Tip window shadow borders. The default is 0.

xscTipShadowType

This resources specifies the appearance of the shadow borders. The values are:

- XmSHADOW_IN
- XmSHADOW_OUT
- XmSHADOW_ETCHED_IN
- XmSHADOW_ETCHED_OUT.

The default is XmSHADOW_OUT.

xscTipShowName

VERSION 1.1

Display the name of a given widget in its Tip window. If there is tip text, the widget name appears below the text; otherwise, the Tip window contains only the widget name. This resource can be used to help determine the name to use in a resource file when specifying resources for a widget. The value of this resource indicates how the widget's name is displayed, as indicated below:

- XmXSC_SHOW_NAME_ALL
- XmXSC_SHOW_NAME_NONE
- XmXSC_SHOW_NAME_SELF
- XmXSC_SHOW_NAME_SHELL

The default value is XmXSC_SHOW_NAME_NONE. Each is described in the following list:

Value[a]	Description of show name values
ALL	The names of all widget ancestors and the widget itself are displayed, e.g., *top.one.two.shell.three.name
NONE	The widget name is not displayed

Value[a]	Description of show name values
SELF	Only the name of the widget is displayed, e.g., *name
SHELL	The names of the widget's closest window manager shell and the widget itself are displayed, e.g., *shell.name

a. Each value actually begins with the prefix "XmXSC_SHOW_NAME_"

xscTipStringConverter

This resource specifies how the Tip text should be converted into a compound string. The values defined for this resource are:

- XmSTRING_CONVERTER_STANDARD
- XmSTRING_CONVERTER_FONT_TAG
- XmSTRING_CONVERTER_SEGMENTED

The default is XmSTRING_CONVERTER_STANDARD. Each is described in the following list:

Value[a]	Description of compound string converters
FONT_TAG	The entire compound string is assigned the font tag defined via the xscTipFontListTag resource. Newline characters are recognized and handled.

Value[a]	Description of compound string converters
SEGMENTED	The converter checks for newlines, as well as, *escape sequences* that adjust how the text is rendered. An escape sequence always begins with the '@' character. Three escape sequences are currently defined: @@ This escape sequence indicates that a single '@' character is to be rendered. @d This escape sequence changes the rendering direction for the remainder of the text or until changed. The sequence "@d<" changes direction to right-to-left and "@d>" changes the direction to left-to-right. @f This escape sequence changes the font tag for the remainder of the text or until changed. The syntax is "@f[tag]" where "tag" is the value of the desired tag. A null-tag (e.g., "@f[]") forces the tag to have the value default value.
STANDARD	The standard Motif converter using the default font tag.

a. Each value actually begins with the prefix "XmSTRING_CONVERTER"

xscTipStringDirection

This resource specifies the direction that the Tip text should be rendered. The values are:

- XmSTRING_DIRECTION_L_TO_R
- XmSTRING_DIRECTION_R_TO_L

XmSTRING_DIRECTION_L_TO_R. is the default value.

xscTipTopic

This resource specifies the text to render in the Tip pop-up. If no text is specified, then no Tip management data structures are allocated for this object and no Tip text is displayed.

xscTipXOffset

This resource specifies the X offset to add to the calculated location of the Tip window. The default is 0.

xscTipYOffset

This resource specifies the Y offset to add to the calculated location of the Tip window. If this resource is set to the value 0 and the `xscTipPosition` resource has the value `XmTIP_POSITION_POINTER`, then this resource is quietly set to the default. The default is 15 if `xscTipPosition` equals `XmTIP_POSITION_POINTER` and 0 otherwise.

APPENDIX A

Unregistered Differences

The Help ToolKit for Motif is distributed in source code in two versions: registered and unregistered.

The unregistered version is a stripped-down version of the registered version. The unregistered version can be dowloaded from the Software Components, Inc. web site and used for evaluation purposes. The unregistered version can also be used without fee on Linux system for non-commerical purposes.

The registered version can only be obtained by purchasing one or more licenses.

Functions in Registered and Unregistered

This table shows all of the public functions available in the Help ToolKit API. The table also contains a brief description and an indication if the function is avalable in the unregistered (U) or registered (R) version.

U	R	Function Name and Description
	✓	`XscCdeHelpInstall` — Initializes the Help ToolKit and installs the CDE context-sensitive help infrastructure
✓	✓	`XscCueDeriveFromWidget` — Retrieves a handle to the Cue associated with an object
	✓	`XscCueGetAlignment` — Get the current text alignment
✓	✓	`XscCueGetAlignment` — Get the current database automatic reload setting
	✓	`XscCueGetAlignment` — Get the current background color
	✓	`XscCueGetBorderColor` — Get the current border color

- ✓ `XscCueGetBorderWidth` — Get the current border width
- ✓ `XscCueGetColorBase` — Get the current color base (used to automatically calculate foreground and shadow colors)
- ✓ ✓ `XscCueGetEnabled` — Determine if the Cue is enabled
- ✓ `XscCueGetFontList` — Get the current font list
- ✓ `XscCueGetFontListTag` — Get the current font list tag
- ✓ `XscCueGetForeground` — Get the current foreground color
- ✓ `XscCueGetMarginBottom` — Get the current bottom margin value
- ✓ `XscCueGetMarginHeight` — Get the current margin height value
- ✓ `XscCueGetMarginLeft` — Get the current left margin value
- ✓ `XscCueGetMarginRight` — Get the current right margin value
- ✓ `XscCueGetMarginTop` — Get the current top margin value
- ✓ `XscCueGetMarginWidth` — Get the current margin width value
- ✓ `XscCueGetMotifColorModel` — Determine if automatic color selection is active
- ✓ `XscCueGetPosition` — Determine the positioning algorithm
- ✓ `XscCueGetShadowThickness` — Get the shadow thickness
- ✓ `XscCueGetShadowType` — Get the shadow type
- ✓ `XscCueGetStringConverter` — Determine the type of string converter used
- ✓ `XscCueGetStringDirection` — Determine the default string rendering direction
- ✓ `XscCueGetTopic` — Get the topic text
- ✓ `XscCueGetXOffset` — Get the X offset
- ✓ `XscCueGetYOffset` — Get the Y offset
- ✓ ✓ `XscCueHasValidTopic` — Determines if a Cue contains a viewable topic
- ✓ `XscCueSetAlignment` — Set the current text alignment

- ✓ ✓ `XscCueSetAutoDbReload` — Set the current database automatic reload setting
- ✓ `XscCueSetBackground` — Set the current background color
- ✓ `XscCueSetBorderColor` — Set the current border color
- ✓ `XscCueSetBorderWidth` — Set the current border width
- ✓ `XscCueSetColorBase` — Set the current color base (used to automatically calculate foreground and shadow colors)
- ✓ ✓ `XscCueSetEnabled` — Specify if the Cue is enabled
- ✓ `XscCueSetFontList` — Set the current font list
- ✓ `XscCueSetFontListTag` — Set the current font list tag
- ✓ `XscCueSetForeground` — Set the current foreground color
- ✓ `XscCueSetMarginBottom` — Set the current bottom margin value
- ✓ `XscCueSetMarginHeight` — Set the current margin height value
- ✓ `XscCueSetMarginLeft` — Set the current left margin value
- ✓ `XscCueSetMarginRight` — Set the current right margin value
- ✓ `XscCueSetMarginTop` — Set the current top margin value
- ✓ `XscCueSetMarginWidth` — Set the current margin width value
- ✓ `XscCueSetMotifColorModel` — Specify if automatic color selection is active
- ✓ `XscCueSetPosition` — Specify the positioning algorithm
- ✓ `XscCueSetShadowThickness` — Set the shadow thickness
- ✓ `XscCueSetShadowType` — Set the shadow type
- ✓ `XscCueSetStringConverter` — Specify the type of string converter used
- ✓ `XscCueSetStringDirection` — Specify the default string rendering direction
- ✓ `XscCueSetTopic` — Set the topic text
- ✓ `XscCueSetXOffset` — Set the X offset

	✓	`XscCueSetYOffset` — Set the Y offset
	✓	`XscHelpAreCuesDisplayable` — Indicates if Cues associated within a given shell are displayable
	✓	`XscHelpAreCuesEnabledGlobally` — Tests if Cues have been globally enabled or disabled
	✓	`XscHelpAreCuesEnabledOnShell` — Tests if Cues have been enabled or disabled with a given shell widget
	✓	`XscHelpAreHintsDisplayable` — Indicates if Hints associated within a given shell are displayable
	✓	`XscHelpAreHintsEnabledGlobally` — Tests if Hints have been globally enabled or disabled
	✓	`XscHelpAreHintsEnabledOnShell` — Tests if Hints have been enabled or disabled with a given shell widget
	✓	`XscHelpAreTipsDisplayable` — Indicates if Tips associated within a given shell are displayable
	✓	`XscHelpAreTipsEnabledGlobally` — Tests if Tips have been globally enabled or disabled
	✓	`XscHelpAreTipsEnabledOnShell` — Tests if Tips have been enabled or disabled with a given shell widget
✓	✓	`XscHelpContextInstall` — Specifies what resource data should be retrieved, stored, and provided to the application when context-sensitive help is requested by the user
✓	✓	`XscHelpContextPickAndActivate` — Allows the user to request context-sensitive help for a widget by selecting the widget with the mouse
✓	✓	`XscHelpCueExists` — Indicates if the specified object has a Cue
✓	✓	`XscHelpCueUpdate` — Updates the Cue of the specified widget, even if it is currently displayed
✓	✓	`XscHelpDbReload` — Reloads from the resource database the help resources associated with a widget or gadget
✓	✓	`XscHelpHintExists` — Indicates if the specified object has a Hint
✓	✓	`XscHelpHintInstall` — Enables Hint support for a window
✓	✓	`XscHelpHintUpdate` — Updates the Hint of the specified widget, even if it is currently displayed
✓	✓	`XscHelpInstall` — Initializes the internal XscHelp data structures and binds to the initial X display connection
	✓	`XscHelpIsDynamicTipGroupIdDefault-Active` — Specifies if the default value for the resource XmNtipGroupId is derived with the old *static* method or the new *dynamic* method

	✓	`XscHelpLoadTopics` — Loads an indirect topic text file into the resource database for use with Cue, Hint, and Tip topics
	✓	`XscHelpSetCuesEnabledGlobally` — Records if Cues should be enabled or disabled globally
	✓	`XscHelpSetCuesEnabledOnShell` — Records if Cues should be enabled or disabled within a given shell
	✓	`XscHelpSetCueTopic` — Updates the Cue topic for a given widget
	✓	`XscHelpSetCueTopicDetails` — Sets all of the Cue text rendering information for a given widget
	✓	`XscHelpSetDynamicTipGroupDefault` — Specifies the method used to derive the default value for the XmNtipGroupId resource
	✓	`XscHelpSetHintsEnabledGlobally` — Records if Hints should be enabled or disabled globally
	✓	`XscHelpSetHintsEnabledOnShell` — Records if Hints should be enabled or disabled within a given shell
	✓	`XscHelpSetHintTopic` — Updates the Hint topic for a given widget
	✓	`XscHelpSetHintTopicDetails` — Sets all of the Hint text rendering information for a given widget
	✓	`XscHelpSetTipsEnabledGlobally` — Records if Tips should be enabled or disabled globally
	✓	`XscHelpSetTipsEnabledOnShell` — Records if Tips should be enabled or disabled within a given shell
	✓	`XscHelpSetTipTopic` — Modifies the Tip topic of a given widget
	✓	`XscHelpSetTipTopicDetails` — Sets all of the Tip text rendering information for a given widget
✓	✓	`XscHelpTipExists` — Indicates if the specified object has a Tip
✓	✓	`XscHelpTipUpdate` — Updates the Tip of the specified widget, even if it is currently displayed
✓	✓	`XscHelpUpdate` — Updates the screen representation of all of the XscHelp attributes for the specified widget
✓	✓	`XscHintDeriveFromWidget` — Retrieves a handle to the Hint associated with an object
	✓	`XscHintGetAlignment` — Get the current text alignment
✓	✓	`XscHintGetAutoDbReload` — Get the current database automatic reload setting
	✓	`XscHintGetBackground` — Get the current background color
✓	✓	`XscHintGetCompound` — Determine if children should be ignored

- ✓ ✓ `XscHintGetEnabled` — Determine if the Hint is enabled
- ✓ `XscHintGetFontList` — Get the current font list
- ✓ `XscHintGetFontListTag` — Get the current font list tag
- ✓ `XscHintGetForeground` — Get the current foreground color
- ✓ `XscHintGetInheritBackground` — Determine if the bacground color is inherited
- ✓ `XscHintGetMarginLeft` — Get the current left margin value
- ✓ `XscHintGetMarginRight` — Get the current right margin value
- ✓ `XscHintGetMarginWidth` — Get the current margin width value
- ✓ `XscHintGetMotifColorModel` — Determine if automatic color selection is active
- ✓ `XscHintGetStringConverter` — Determine the type of string converter used
- ✓ `XscHintGetStringDirection` — Determine the default string rendering direction
- ✓ `XscHintGetTopic` — Get the topic text
- ✓ ✓ `XscHintHasValidTopic` — Determines if a Hint contains a viewable topic
- ✓ `XscHintSetAlignment` — Set the current text alignment
- ✓ ✓ `XscHintSetAutoDbReload` — Set the current database automatic reload setting
- ✓ `XscHintSetBackground` — Set the current background color
- ✓ ✓ `XscHintSetCompound` — Specify if children should be ignored
- ✓ ✓ `XscHintSetEnabled` — Specify if the Hint is enabled
- ✓ `XscHintSetFontList` — Set the current font list
- ✓ `XscHintSetFontListTag` — Set the current font list tag
- ✓ `XscHintSetForeground` — Set the current foreground color
- ✓ `XscHintSetInheritBackground` — Specify if the bacground color is inherited
- ✓ `XscHintSetMarginLeft` — Set the current left margin value

- ✓ `XscHintSetMarginRight` — Set the current right margin value
- ✓ `XscHintSetMarginWidth` — Set the current margin width value
- ✓ `XscHintSetMotifColorModel` — Specify if automatic color selection is active
- ✓ `XscHintSetStringConverter` — Specify the type of string converter used
- ✓ `XscHintSetStringDirection` — Specify the default string rendering direction
- ✓ `XscHintSetTopic` — Set the topic text
- ✓ ✓ `XscTipDeriveFromWidget` — Retrieves a handle to the Tip associated with an object
- ✓ `XscTipGetAlignment` — Get the current text alignment
- ✓ ✓ `XscTipGetAutoDbReload` — Get the current database automatic reload setting
- ✓ `XscTipGetBackground` — Get the current background color
- ✓ `XscTipGetBorderColor` — Get the current border color
- ✓ `XscTipGetBorderWidth` — Get the current border width
- ✓ `XscTipGetColorBase` — Get the current color base (used to automatically calculate foreground and shadow colors)
- ✓ ✓ `XscTipGetCompound` — Determine if children should be ignored
- ✓ ✓ `XscTipGetEnabled` — Determine if the Tip is enabled
- ✓ `XscTipGetFontList` — Get the current font list
- ✓ `XscTipGetFontListTag` — Get the current font list tag
- ✓ `XscTipGetForeground` — Get the current foreground color
- ✓ ✓ `XscTipGetGroupId` — Get the group identifier
- ✓ `XscTipGetMarginBottom` — Get the current bottom margin value
- ✓ `XscTipGetMarginHeight` — Get the current margin height value
- ✓ `XscTipGetMarginLeft` — Get the current left margin value
- ✓ `XscTipGetMarginRight` — Get the current right margin value

- ✓ `XscTipGetMarginTop` — Get the current top margin value
- ✓ `XscTipGetMarginWidth` — Get the current margin width value
- ✓ `XscTipGetMotifColorModel` — Determine if automatic color selection is active
- ✓ `XscTipGetPopdownInterval` — Determine how long Tip should be displayed
- ✓ `XscTipGetPopupInterval` — Determine how long before Tip is displayed
- ✓ `XscTipGetPosition` — Determine the positioning algorithm
- ✓ `XscTipGetSelectNameInterval` — Determine how long before the displayed name is selected
- ✓ `XscTipGetShadowThickness` — Get the shadow thickness
- ✓ `XscTipGetShadowType` — Get the shadow type
- ✓ `XscTipGetStringConverter` — Determine the type of string converter used
- ✓ `XscTipGetStringDirection` — Determine the default string rendering direction
- ✓ `XscTipGetTopic` — Get the topic text
- ✓ `XscTipGetXOffset` — Get the X offset
- ✓ `XscTipGetYOffset` — Get the Y offset
- ✓ ✓ `XscTipHasValidTopic` — Determines if a Tip contains a viewable topic
- ✓ `XscTipSetAlignment` — Set the current text alignment
- ✓ ✓ `XscTipSetAutoDbReload` — Set the current database automatic reload setting
- ✓ `XscTipSetBackground` — Set the current background color
- ✓ `XscTipSetBorderColor` — Set the current border color
- ✓ `XscTipSetBorderWidth` — Set the current border width
- ✓ `XscTipSetColorBase` — Set the current color base (used to automatically calculate foreground and shadow colors)
- ✓ ✓ `XscTipSetCompound` — Specify if children should be ignored
- ✓ ✓ `XscTipSetEnabled` — Specify if the Tip is enabled

- ✓ `XscTipSetFontList` — Set the current font list
- ✓ `XscTipSetFontListTag` — Set the current font list tag
- ✓ `XscTipSetForeground` — Set the current foreground color
- ✓ ✓ `XscTipSetGroupId` — Set the group identifier
- ✓ `XscTipSetMarginBottom` — Set the current bottom margin value
- ✓ `XscTipSetMarginHeight` — Set the current margin height value
- ✓ `XscTipSetMarginLeft` — Set the current left margin value
- ✓ `XscTipSetMarginRight` — Set the current right margin value
- ✓ `XscTipSetMarginTop` — Set the current top margin value
- ✓ `XscTipSetMarginWidth` — Set the current margin width value
- ✓ `XscTipSetMotifColorModel` — Specify if automatic color selection is active
- ✓ `XscTipSetPopdownInterval` — Specify how long Tip should be displayed
- ✓ `XscTipSetPopupInterval` — Specify how long before Tip is displayed
- ✓ `XscTipSetPosition` — Specify the positioning algorithm
- ✓ `XscTipSetSelectNameInterval` — Specify how long before the displayed name is selected
- ✓ `XscTipSetShadowThickness` — Set the shadow thickness
- ✓ `XscTipSetShadowType` — Set the shadow type
- ✓ `XscTipSetStringConverter` — Specify the type of string converter used
- ✓ `XscTipSetStringDirection` — Specify the default string rendering direction
- ✓ `XscTipSetTopic` — Set the topic text
- ✓ `XscTipSetXOffset` — Set the X offset
- ✓ `XscTipSetYOffset` — Set the Y offset

Resources in Registered and Unregistered

This table shows all of the resource values available with the Help ToolKit. The table also contains a brief description and an indication if each resource is avalable in the unregistered (U) or registered (R) version.

U	R	Resource Name and Description
	✓	`XmNxscCdeHelpColumns` — Specifies, in characters, how wide the text display area on the Help widget should be
	✓	`XmNxscCdeHelpDialogTitle` — Specify the title of a dialog
	✓	`XmNxscCdeHelpRows` — Specifies, in lines, how tall the text display area on the Help widget should be
	✓	`XmNxscCdeHelpTopic` — Specifies what should be displayed for a widget (or gadget) when context-sensitive help is requested
	✓	`XmNxscCdeHelpTopicTitle` — Displayed in the help widget as the name of the help topic being displayed
	✓	`XmNxscCdeHelpType` — The type of help to use
	✓	`XmNxscCdeHelpVolume` — Specifies the name of the help volume to reference
	✓	`XmNxscCdeHelpWidgetType` — Specifies the type of CDE widget to use when viewing the help information
	✓	`XmNxscCueAlignment` — Specify the text alignment of the Cue
✓	✓	`XmNxscCueAutoDbReload` — Indicates if resource values should be automatically reloaded
	✓	`XmNxscCueBackground` — Specifies the background color of the Cue window
	✓	`XmNxscCueBorderColor` — Specifies the border color of the Cue window
	✓	`XmNxscCueBorderWidth` — Specifies the width of the Cue window border
	✓	`XmNxscCueColorBase` — Specifies the color to use when deriving the proper shadow border colors
✓	✓	`XmNxscCueEnabled` — Specifies if the given Cue is enabled

- ✓ ✓ `XmNxscCueFontList` — Specifies the font list to use when rendering the Cue text
- ✓ `XmNxscCueFontListTag` — Specifies the font list tag to use when building the compound string for the Cue text
- ✓ `XmNxscCueForeground` — Specifies the foreground color of the text rendered for the Cue
- ✓ `XmNxscCueMarginBottom` — Specifies the bottom margin
- ✓ `XmNxscCueMarginHeight` — Specifies the margin height
- ✓ `XmNxscCueMarginLeft` — Specifies the left margin
- ✓ `XmNxscCueMarginRight` — Specifies the right margin
- ✓ `XmNxscCueMarginTop` — Specifies the top margin
- ✓ `XmNxscCueMarginWidth` — Specifies the margin width
- ✓ `XmNxscCueMotifColorModel` — Specifies if the colors are derived or specified
- ✓ `XmNxscCuePosition` — Specifies where the Cue window is displayed
- ✓ `XmNxscCueShadowThickness` — Specifies the shadow width of the Cue window shadow borders
- ✓ `XmNxscCueShadowType` — Specifies the appearance of the shadow borders
- ✓ `XmNxscCueShowName` — Displays the name of a given widget in its Cue window
- ✓ `XmNxscCueStringConverter` — Specifies how the Cue text should be converted into a compound string
- ✓ ✓ `XmNxscCueStringDirection` — Specifies the direction that the Cue text should be rendered
- ✓ ✓ `XmNxscCueTopic` — Specifies the text to render in the Cue pop-up
- ✓ `XmNxscCueXOffset` — Specifies the X offset to add to the calculated location of the Cue window
- ✓ `XmNxscCueYOffset` — Specifies the Y offset to add to the calculated location of the Cue window
- ✓ `XmNxscGadgetProcessing` — Specifies if gadgets should be ignored

	✓	XmNxscHintAlignment — Specify the text alignment of the Hint
✓	✓	XmNxscHintAutoDbReload — Indicates if resource values should be automatically reloaded
	✓	XmNxscHintBackground — Specifies the background color of the hint display widget
✓	✓	XmNxscHintCompound — Makes a collection of widgets and gadgets act like a single object
✓	✓	XmNxscHintEnabled — Specifies if the given Hint is enabled
✓	✓	XmNxscHintFontList — Specifies the font list to use when rendering the Hint text
	✓	XmNxscHintFontListTag — Specifies the font list tag to use when building the compound string for the Hint text
	✓	XmNxscHintForeground — Specifies the foreground color of the text rendered in the hint display widget
	✓	XmNxscHintInheritBackground — Specifies if the background color of the Hint will be the same as the background color of the hint display widget's immediate parent
	✓	XmNxscHintMarginLeft — Specifies the left margin
	✓	XmNxscHintMarginRight — Specifies the right margin
	✓	XmNxscHintMarginWidth — Specifies the margin width
	✓	XmNxscHintMotifColorModel — Specifies if the colors are derived or specified
	✓	XmNxscHintShowName — Displays the name of a given widget in its the Hint area
	✓	XmNxscHintStringConverter — Specifies how the Hint text should be converted into a compound string
✓	✓	XmNxscHintStringDirection — Specifies the direction that the Hint text should be rendered
✓	✓	XmNxscHintTopic — Specifies the text to be rendered in the Hint display area
	✓	XmNxscTipAlignment — Specify the text alignment of the Tip

- ✓ ✓ `XmNxscTipAutoDbReload` — Indicates if resource values should be automatically reloaded
- ✓ `XmNxscTipBackground` — Specifies the background color of the Tip window
- ✓ `XmNxscTipBorderColor` — Specifies the border color of the Tip window
- ✓ `XmNxscTipBorderWidth` — Specifies the width of the Tip window border
- ✓ `XmNxscTipColorBase` — Specifies the color to use when deriving the proper shadow border colors
- ✓ ✓ `XmNxscTipCompound` — Makes a collection of widgets and gadgets act like a single object
- ✓ ✓ `XmNxscTipEnabled` — Specifies if the given Tip is enabled
- ✓ ✓ `XmNxscTipFontList` — Specifies the font list to use when rendering the Tip text
- ✓ `XmNxscTipFontListTag` — Specifies the font list tag to use when building the compound string for the Tip text
- ✓ `XmNxscTipForeground` — Specifies the foreground color of the text rendered for the Tip
- ✓ ✓ `XmNxscTipGroupId` — Associates a widget with a Tip group
- ✓ `XmNxscTipGroupOverride` — *deprecated*
- ✓ `XmNxscTipMarginBottom` — Specifies the bottom margin
- ✓ `XmNxscTipMarginHeight` — Specifies the margin height
- ✓ `XmNxscTipMarginLeft` — Specifies the left margin
- ✓ `XmNxscTipMarginRight` — Specifies the right margin
- ✓ `XmNxscTipMarginTop` — Specifies the top margin
- ✓ `XmNxscTipMarginWidth` — Specifies the margin width
- ✓ `XmNxscTipMotifColorModel` — Specifies if the colors are derived or specified
- ✓ `XmNxscTipPopdownInterval` — Specifies the amount of time that the Tip should be displayed before it is automatically popped down

- ✓ `XmNxscTipPopupInterval` — Specifies the amount of time, in milliseconds, that the mouse should stay at rest over the object before the associated Tip is displayed
- ✓ `XmNxscTipPosition` — Controls where the Tip window is displayed
- ✓ `XmNxscTipSelectNameInterval` — Control if and when the widget name displayed in the Tip window is selected
- ✓ `XmNxscTipShadowThickness` — Specifies the shadow width of the Tip window shadow borders
- ✓ `XmNxscTipShadowType` — Specifies the appearance of the shadow borders
- ✓ `XmNxscTipShowName` — Displays the name of a given widget in its Tip window
- ✓ `XmNxscTipStringConverter` — Specifies how the Tip text should be converted into a compound string
- ✓ ✓ `XmNxscTipStringDirection` — Specifies the direction that the Tip text should be rendered
- ✓ ✓ `XmNxscTipTopic` — Specifies the text to render in the Tip pop-up
- ✓ `XmNxscTipXOffset` — Specifies the X offset to add to the calculated location of the Tip window
- ✓ `XmNxscTipYOffset` — Specifies the Y offset to add to the calculated location of the Tip window

APPENDIX B

Release Notes

This section discuss the major changes made to the Help ToolKit from one version to the next.

Version 1.1.2

Symbolic values with tip groups

CROSS-REFERENCE
For more information, refer to page 21.

The symbolic values `XmXSC_TIP_GROUP_SELF` and `XmXSC_TIP_GROUP_PARENT` can now be specified in a resource file for the resource `XmNxscTipGroupId`. Previously, only a numeric identifier in the range of 1 to 10000 could be specified for the `XmNxscTipGroupId`. If you wanted to use the symbolic values, you had to use the virtual resource `XmNxscTipGroupOverride`, which is now functionally obsolete and deprecated.

Dynamic default for tip groups

CROSS-REFERENCE
For more information, refer to page 21.

Earlier versions of the ToolKit had a static default of `XmXSC_TIP_GROUP_PARENT` for the value of the resource `XmNxscTipGroupId`. A new dynamic algorithm has been added with this version.

The new dynamic algorithm has a default of `XmXSC_TIP_GROUP_SELF` unless one of the object's ancestors has an explicit (i.e., non-defaulted) value, in which case the default is `XmXSC_TIP_GROUP_PARENT`. This is true even if the ancestor has an explicit value of `XmXSC_TIP_GROUP_SELF`.

The new algorithm is used by default, but the old static algorithm can be installed by calling the function

```
void XscHelpSetDynamicTipGroupDefault(
     Boolean )
```

If the `Boolean` is `True`, then the new dynamic algorithm is used; otherwise, the old static default is used. This function can be called at any time, even before the ToolKit is initialized.

You can also test to see if the new dynamic algorithm is being used by calling the function

```
Boolean
XscHelpIsDynamicTipGroupIdDefaultActive()
```

Version 1.1.1

Enable and disable tip, hints, and cues

CROSS-REFERENCE
For for information, refer to “Enabling and Disabling” on page 69.

A set of functions has been added that allow tips, hints, and cues to be enabled globally and per shell. For a given tip, hint, or cue to render, it now must be enabled individually, on its shell, and globally.

Version 1.1.0

Tip Group Override

NOTE
This capability is now functionally obsolete and deprecated

The virtual resource `XmNtipGroupOverride` was added to allow symbolic values to be specified and override the value specified for the `XmNtipGroupId` resource.

Displaying widget name

CROSS-REFERENCE
For more information, refer to “Identifying Widget Names” on page 71.

A new capability was added allowing the widget name (and possibly a portion of its hierarchy) to be displayed in an associated tip, hint, or cue. The name is on a new line below the normal topic text. The widget name can be specified even if there is no topic text. This capability is activated based on the value

of the resources: XmNxscTipShowName, XmNhint-ShowName, and XmNcueShowName. The possible values are:

XmXSC_SHOW_NAME_NONE — (default) capability is not active

XmXSC_SHOW_NAME_SELF — only the name of the widget is displayed, e.g., "*name"

XmXSC_SHOW_NAME_SHELL — the names of the widget's closets WM shell and the widget itself are displayed, e.g., "*shellName*name"

XmXSC_SHOW_NAME_ALL — the names of all widget ancestors and the widget itself are displayed, e.g., "*top.one.two.shell.three.four.five.name"

Select displayed widget name

CROSS-REFERENCE
For more information, refer to “Selecting Widget Names” on page 73.

The ability to automatically select the widget name shown in a tip has been added. The widget name is shown via the resource XmNxscTipShowName. By default, the name is not selected. However, if the XmNtipSelectNameInterval is set to a value other than -1, then the absolute value is used as the number of milliseconds to wait before selecting the name. Once the name is automatically selected, the selection can be pasted into into an editor to assist with resource file creation.

Text topic indirection

CROSS-REFERENCE
For more information, refer to “Help Topic Indirection” on page 74.

Text topic indirection has been added which allows text to be specified in one place and referenced by multiple tips, hints, or cues. If the text for a topic begins with a period (.) the text (following the period) is assumed to be a key which is used to lookup the real text in a "global" area of the resource database. The global area has the binding "_XscHelp.topic". Therefore, if a tip topic has the value ".Tip one", then the real text would be retrieved from the resource "_XscHelp.topic.Tip one".

Read topics from a file

CROSS-REFERENCE
For more information, refer to “Loading Indirect Help Topics” on page 75.

The new function XscHelpLoadTopics() was added which allows text to be read from a file and associated with an XscHelp key in the resource database. This key can then be used to indirectly specify the text of a topic.

Index