

VAX LISP/VMS DECwindows Programming Guide

Order Number: AA-MK71A-TE

This manual is a guide to programming with the DECwindows Toolkit and Common LISP X in VAX LISP.

Revision/Update Information: This is a new manual.

Operating System and Version: VMS 5.1

Software Version: VAX LISP 3.0

July 1989

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

© Digital Equipment Corporation, Massachusetts Institute of Technology, and Texas Instruments Incorporated. 1989.

All rights reserved.
Printed in U.S.A.

The postpaid Reader's Comments form at the end of this document requests your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

AI VAXstation	PDP	VAX LISP/ULTRIX
DEC	ULTRIX	VAXstation
DECnet	ULTRIX-11	VAXstation II
DECUS	ULTRIX-32	VMS
DECwindows	UNIBUS	XUI
MicroVAX	VAX	
MicroVAX II	VAX LISP	
MicroVMS	VAX LISP/VMS	digital™

X Window System and its derivations (X, X11, X Version 11, X Window System) are trademarks of the Massachusetts Institute of Technology.

MLO-S837

This document was prepared using VAX DOCUMENT, Version 1.2.

Contents

Preface	xxiii
---------------	-------

Part I Guide to the DECwindows ToolKit

Chapter 1 Overview of the DECwindows Toolkit

1.1 DECwindows Toolkit Components	1-2
1.2 User Interface Widgets	1-3
1.3 Creating a User Interface	1-7
1.4 Calling Back to Application Functions	1-7

Chapter 2 Creating a User Interface

2.1 A Sample DECwindows Application	2-1
2.2 The User Interface Language	2-3
2.3 High-Level Functions	2-7
2.4 Low-Level Functions	2-9

Chapter 3 Building a DECwindows Application

3.1 Accessing the User Interface	3-1
3.1.1 Initialization	3-1
3.1.2 Creating Widgets	3-2
3.1.3 Realizing the Top-Level Widget	3-3
3.1.4 Calling the Main Loop	3-4
3.2 Writing a Callback Function	3-4
3.2.1 Declaring Callback Functions	3-4
3.2.2 Accessing the Callback Argument List	3-6
3.2.2.1 Accessing the Widget ID	3-6
3.2.2.2 Accessing the User-Defined Data (Tag)	3-7
3.2.2.3 Accessing the Callback Structure	3-9

Part II Reference to the DECwindows ToolKit

DWT:ADD-ACTIONS INTRINSIC FUNCTION	DWT-1
DWT:ADD-CALLBACK INTRINSIC FUNCTION	DWT-1
DWT:ADD-CALLBACKS INTRINSIC FUNCTION	DWT-1
DWT:ADD-CONVERTER INTRINSIC FUNCTION	DWT-2
DWT:ADD-EVENT-HANDLER INTRINSIC FUNCTION	DWT-2
DWT:ADD-EXPOSURE-TO-REGION INTRINSIC FUNCTION	DWT-2
DWT:ADD-FONT-LIST COMPOUND STRING FUNCTION.	DWT-3
DWT:ADD-GRAB INTRINSIC FUNCTION	DWT-3
DWT:ADD-INPUT INTRINSIC FUNCTION	DWT-3
DWT:ADD-RAW-EVENT-HANDLER INTRINSIC FUNCTION	DWT-4
DWT:ADD-TIME-OUT INTRINSIC FUNCTION	DWT-4
DWT:ADD-WORK-PROC INTRINSIC FUNCTION	DWT-4
DWT:APP-ADD-INPUT INTRINSIC FUNCTION	DWT-5
DWT:APP-ADD-TIME-OUT INTRINSIC FUNCTION	DWT-5
DWT:APP-ADD-WORK-PROC INTRINSIC FUNCTION	DWT-5
DWT:APP-CREATE-SHELL INTRINSIC FUNCTION	DWT-6
DWT:APP-MAIN-LOOP INTRINSIC FUNCTION	DWT-6
DWT:APP-NEXT-EVENT INTRINSIC FUNCTION	DWT-6
DWT:APP-PEEK-EVENT INTRINSIC FUNCTION	DWT-7
DWT:APP-PENDING INTRINSIC FUNCTION	DWT-7
DWT:APP-PROCESS-EVENT INTRINSIC FUNCTION	DWT-7
DWT:ATTACHED-DB HIGH-LEVEL FUNCTION	DWT-8
DWT:ATTACHED-DB-CREATE LOW-LEVEL FUNCTION	DWT-8
DWT:ATTACHED-DB-POPUP-CREATE LOW-LEVEL FUNCTION	DWT-8
DWT:AUGMENT-TRANSLATIONS INTRINSIC FUNCTION	DWT-9
DWT:BEGIN-COPY-TO-CLIPBOARD CUT-AND-PASTE FUNCTION	DWT-9
DWT:BUILD-EVENT-MASK INTRINSIC FUNCTION	DWT-10
DWT:CALL-ACCEPT-FOCUS INTRINSIC FUNCTION	DWT-10
DWT:CALL-CALLBACKS INTRINSIC FUNCTION	DWT-10
DWT:CALLBACK-EXCLUSIVE INTRINSIC FUNCTION	DWT-11
DWT:CALLBACK-NONE INTRINSIC FUNCTION	DWT-11
DWT:CALLBACK-NONEXCLUSIVE CONVENIENCE FUNCTION	DWT-11
DWT:CALLBACK-POPDOWN INTRINSIC FUNCTION	DWT-12
DWT:CALLOC INTRINSIC FUNCTION	DWT-12
DWT:CANCEL-COPY-FORMAT CUT-AND-PASTE FUNCTION	DWT-12
DWT:CANCEL-COPY-TO-CLIPBOARD CUT-AND-PASTE FUNCTION	DWT-13
DWT:CAUTION-BOX HIGH-LEVEL FUNCTION	DWT-13
DWT:CAUTION-BOX-CREATE LOW-LEVEL FUNCTION	DWT-14
DWT:CHILDREN CONVENIENCE FUNCTION	DWT-14
DWT:CLASS INTRINSIC FUNCTION	DWT-14
DWT:CLIPBOARD-LOCK CUT-AND-PASTE FUNCTION	DWT-15
DWT:CLIPBOARD-UNLOCK CUT-AND-PASTE FUNCTION	DWT-15
DWT:CLOSE-DISPLAY INTRINSIC FUNCTION	DWT-15
DWT:CLOSE-HIERARCHY DRM FUNCTION	DWT-16
DWT:COMMAND-APPEND HIGH-LEVEL FUNCTION	DWT-16
DWT:COMMAND-ERROR-MESSAGE HIGH-LEVEL FUNCTION	DWT-16
DWT:COMMAND-SET HIGH-LEVEL FUNCTION	DWT-17
DWT:COMMAND-WINDOW HIGH-LEVEL FUNCTION	DWT-17
DWT:COMMAND-WINDOW-CREATE LOW-LEVEL FUNCTION	DWT-17
DWT:CONFIGURE-WIDGET INTRINSIC FUNCTION	DWT-18
DWT:CONVERT INTRINSIC FUNCTION	DWT-18
DWT:CONVERT-CASE INTRINSIC FUNCTION	DWT-18

DWT:COPY-FROM-CLIPBOARD CUT-AND-PASTE FUNCTION	DWT-19
DWT:COPY-TO-CLIPBOARD CUT-AND-PASTE FUNCTION	DWT-19
DWT:CREATE-APPLICATION-CONTEXT INTRINSIC FUNCTION.....	DWT-20
DWT:CREATE-APPLICATION-SHELL INTRINSIC FUNCTION	DWT-20
DWT:CREATE-FONT-LIST COMPOUND STRING FUNCTION	DWT-21
DWT:CREATE-MANAGED-WIDGET INTRINSIC FUNCTION	DWT-21
DWT:CREATE-POPUP-SHELL INTRINSIC FUNCTION	DWT-21
DWT:CREATE-WIDGET INTRINSIC FUNCTION	DWT-22
DWT:CREATE-WINDOW INTRINSIC FUNCTION	DWT-22
DWT:CSBYTECMP COMPOUND STRING FUNCTION.....	DWT-22
DWT:CSEMPTRY COMPOUND STRING FUNCTION	DWT-23
DWT:CS-STRING COMPOUND STRING FUNCTION.....	DWT-23
DWT:CSTRCAT COMPOUND STRING FUNCTION	DWT-23
DWT:CSTRCPY COMPOUND STRING FUNCTION	DWT-24
DWT:CSTRLEN COMPOUND STRING FUNCTION	DWT-24
DWT:CSTRNCAT COMPOUND STRING FUNCTION	DWT-24
DWT:CSTRNCPY COMPOUND STRING FUNCTION	DWT-25
DWT:DATABASE INTRINSIC FUNCTION	DWT-25
DWT:DESTROY-APPLICATION-CONTEXT INTRINSIC FUNCTION	DWT-25
DWT:DESTROY-GC INTRINSIC FUNCTION	DWT-26
DWT:DESTROY-WIDGET INTRINSIC FUNCTION	DWT-26
DWT:DIALOG-BOX HIGH-LEVEL FUNCTION	DWT-26
DWT:DIALOG-BOX-CREATE LOW-LEVEL FUNCTION	DWT-27
DWT:DIALOG-BOX-POPUP-CREATE LOW-LEVEL FUNCTION	DWT-27
DWT:DIRECT-CONVERT INTRINSIC FUNCTION	DWT-27
DWT:DISOWN-SELECTION INTRINSIC FUNCTION	DWT-28
DWT:DISPATCH-EVENT INTRINSIC FUNCTION	DWT-28
DWT:DISPLAY INTRINSIC FUNCTION	DWT-28
DWT:DISPLAY-CS-MESSAGE CONVENIENCE FUNCTION	DWT-29
DWT:DISPLAY-INITIALIZE INTRINSIC FUNCTION	DWT-29
DWT:DISPLAY-VMS-MESSAGE CONVENIENCE FUNCTION	DWT-30
DWT:DRM-FREE-RESOURCE-CONTEXT DRM FUNCTION	DWT-30
DWT:DRM-GET-RESOURCE-CONTEXT DRM FUNCTION	DWT-31
DWT:DRM-HGET-INDEXED-LITERAL DRM FUNCTION	DWT-31
DWT:DRM-RC-BUFFER DRM FUNCTION	DWT-32
DWT:DRM-RC-SET-TYPE DRM FUNCTION	DWT-32
DWT:DRM-RC-SIZE DRM FUNCTION	DWT-32
DWT:DRM-RC-TYPE DRM FUNCTION	DWT-33
DWT:DWTHELP HIGH-LEVEL FUNCTION	DWT-33
DWT:DWTWINDOW INTRINSIC FUNCTION	DWT-33
DWT:END-COPY-TO-CLIPBOARD CUT-AND-PASTE FUNCTION	DWT-34
DWT:ERROR-MSG INTRINSIC FUNCTION	DWT-34
DWT:FETCH-INTERFACE-MODULE DRM FUNCTION	DWT-35
DWT:FETCH-SET-VALUES DRM FUNCTION	DWT-35
DWT:FETCH-WIDGET DRM FUNCTION	DWT-36
DWT:FETCH-WIDGET-OVERRIDE DRM FUNCTION	DWT-36
DWT:FILE-SELECTION HIGH-LEVEL FUNCTION	DWT-37
DWT:FILE-SELECTION-CREATE LOW-LEVEL FUNCTION	DWT-37
DWT:FILE-SELECTION-DO-SEARCH HIGH-LEVEL FUNCTION	DWT-38
DWT:FREE INTRINSIC FUNCTION	DWT-38

DWT:GET-APPLICATION-RESOURCES INTRINSIC FUNCTION	DWT-38
DWT:GET-DISPLAY CONVENIENCE FUNCTION	DWT-39
DWT:GET-ERROR-DATABASE INTRINSIC FUNCTION	DWT-39
DWT:GET-ERROR-DATABASE-TEXT INTRINSIC FUNCTION	DWT-39
DWT:GET-GC INTRINSIC FUNCTION	DWT-40
DWT:GET-NEXT-SEGMENT COMPOUND STRING FUNCTION	DWT-40
DWT:GET-RESOURCE-LIST INTRINSIC FUNCTION	DWT-41
DWT:GET-SCREEN CONVENIENCE FUNCTION	DWT-41
DWT:GET-SELECTION-TIMEOUT INTRINSIC FUNCTION	DWT-41
DWT:GET-SELECTION-VALUE INTRINSIC FUNCTION	DWT-41
DWT:GET-SELECTION-VALUE-INCR INTRINSIC FUNCTION	DWT-42
DWT:GET-SELECTION-VALUES INTRINSIC FUNCTION	DWT-42
DWT:GET-SELECTION-VALUES-INCR INTRINSIC FUNCTION	DWT-43
DWT:GET-SUBRESOURCES INTRINSIC FUNCTION	DWT-43
DWT:GET-SUBVALUES INTRINSIC FUNCTION	DWT-44
DWT:GET-VALUES INTRINSIC FUNCTION	DWT-44
DWT:HAS-CALLBACKS INTRINSIC FUNCTION	DWT-44
DWT:HELP HIGH-LEVEL FUNCTION	DWT-45
DWT:HELP-CREATE LOW-LEVEL FUNCTION	DWT-45
DWT:INIT-GET-SEGMENT COMPOUND STRING FUNCTION	DWT-45
DWT:INITIALIZE INTRINSIC FUNCTION	DWT-46
DWT:INITIALIZE-DRM DRM FUNCTION	DWT-46
DWT:INQUIRE-NEXT-PASTE-COUNT CUT-AND-PASTE FUNCTION	DWT-46
DWT:INQUIRE-NEXT-PASTE-FORMAT CUT-AND-PASTE FUNCTION	DWT-47
DWT:INQUIRE-NEXT-PASTE-LENGTH CUT-AND-PASTE FUNCTION	DWT-47
DWT:INSTALL-ACCELERATORS INTRINSIC FUNCTION	DWT-48
DWT:INSTALL-ALL-ACCELERATORS INTRINSIC FUNCTION	DWT-48
DWT:IS-COMPOSITE INTRINSIC FUNCTION	DWT-49
DWT:IS-MANAGED INTRINSIC FUNCTION	DWT-49
DWT:IS-REALIZED INTRINSIC FUNCTION	DWT-49
DWT:IS-SENSITIVE INTRINSIC FUNCTION	DWT-49
DWT:IS-SUBCLASS INTRINSIC FUNCTION	DWT-50
DWT:LABEL HIGH-LEVEL FUNCTION	DWT-50
DWT:LABEL-CREATE LOW-LEVEL FUNCTION	DWT-50
DWT:LABEL-GADGET-CREATE GADGET FUNCTION	DWT-51
DWT:LATIN1-STRING COMPOUND STRING FUNCTION	DWT-51
DWT:LIST-BOX HIGH-LEVEL FUNCTION	DWT-51
DWT:LIST-BOX-ADD-ITEM HIGH-LEVEL FUNCTION	DWT-52
DWT:LIST-BOX-CREATE LOW-LEVEL FUNCTION	DWT-52
DWT:LIST-BOX-DELETE-ITEM HIGH-LEVEL FUNCTION	DWT-53
DWT:LIST-BOX-DELETE-POS HIGH-LEVEL FUNCTION	DWT-53
DWT:LIST-BOX-DESELECT-ALL-ITEMS HIGH-LEVEL FUNCTION	DWT-53
DWT:LIST-BOX-DESELECT-ITEM HIGH-LEVEL FUNCTION	DWT-54
DWT:LIST-BOX-DESELECT-POS HIGH-LEVEL FUNCTION	DWT-54
DWT:LIST-BOX-ITEM-EXISTS HIGH-LEVEL FUNCTION	DWT-54
DWT:LIST-BOX-SELECT-ITEM HIGH-LEVEL FUNCTION	DWT-55
DWT:LIST-BOX-SELECT-POS HIGH-LEVEL FUNCTION	DWT-55
DWT:LIST-BOX-SET-HORIZ-POS HIGH-LEVEL FUNCTION	DWT-55
DWT:LIST-BOX-SET-ITEM HIGH-LEVEL FUNCTION	DWT-56
DWT:LIST-BOX-SET-POS HIGH-LEVEL FUNCTION	DWT-56
DWT:LIST-PENDING-ITEMS CUT-AND-PASTE FUNCTION	DWT-56
DWT:MAIN-LOOP INTRINSIC FUNCTION	DWT-57
DWT:MAIN-WINDOW HIGH-LEVEL FUNCTION	DWT-57
DWT:MAIN-WINDOW-CREATE LOW-LEVEL FUNCTION	DWT-57
DWT:MAIN-WINDOW-SET-AREAS HIGH-LEVEL FUNCTION	DWT-58

DWT:MAKE-GEOMETRY-REQUEST INTRINSIC FUNCTION	DWT-58
DWT:MAKE-RESIZE-REQUEST INTRINSIC FUNCTION	DWT-59
DWT:MALLOC INTRINSIC FUNCTION	DWT-59
DWT:MANAGE-CHILD INTRINSIC FUNCTION	DWT-59
DWT:MANAGE-CHILDREN INTRINSIC FUNCTION	DWT-60
DWT:MAP-WIDGET INTRINSIC FUNCTION	DWT-60
DWT:MENU HIGH-LEVEL FUNCTION	DWT-60
DWT:MENU-BAR HIGH-LEVEL FUNCTION	DWT-61
DWT:MENU-BAR-CREATE LOW-LEVEL FUNCTION	DWT-61
DWT:MENU-CREATE LOW-LEVEL FUNCTION	DWT-61
DWT:MENU-POPUP-CREATE LOW-LEVEL FUNCTION	DWT-62
DWT:MENU-POSITION HIGH-LEVEL FUNCTION	DWT-62
DWT:MENU-PULLDOWN-CREATE LOW-LEVEL FUNCTION	DWT-62
DWT:MERGE-ARG-LISTS INTRINSIC FUNCTION	DWT-63
DWT:MESSAGE-BOX HIGH-LEVEL FUNCTION	DWT-63
DWT:MESSAGE-BOX-CREATE LOW-LEVEL FUNCTION	DWT-64
DWT:MOVE-WIDGET INTRINSIC FUNCTION	DWT-64
DWT:NAME-TO-WIDGET INTRINSIC FUNCTION	DWT-64
DWT:NEXT-EVENT INTRINSIC FUNCTION	DWT-65
DWT:NUMBER-CHILDREN CONVENIENCE FUNCTION	DWT-65
DWT:OPEN-DISPLAY INTRINSIC FUNCTION	DWT-65
DWT:OPEN-HIERARCHY DRM FUNCTION	DWT-66
DWT:OPTION-MENU HIGH-LEVEL FUNCTION	DWT-66
DWT:OPTION-MENU-CREATE LOW-LEVEL FUNCTION	DWT-66
DWT:OVERRIDE-TRANSLATIONS INTRINSIC FUNCTION	DWT-67
DWT:OWN-SELECTION INTRINSIC FUNCTION	DWT-67
DWT:OWN-SELECTION-INCREMENTAL INTRINSIC FUNCTION	DWT-68
DWT:PARENT INTRINSIC FUNCTION	DWT-68
DWT:PARSE-ACCELERATOR-TABLE INTRINSIC FUNCTION	DWT-68
DWT:PARSE-TRANSLATION-TABLE INTRINSIC FUNCTION	DWT-69
DWT:PEEK-EVENT INTRINSIC FUNCTION	DWT-69
DWT:PENDING INTRINSIC FUNCTION	DWT-69
DWT:POPDOWN INTRINSIC FUNCTION	DWT-70
DWT:POPUP INTRINSIC FUNCTION	DWT-70
DWT:PROCESS-EVENT INTRINSIC FUNCTION	DWT-70
DWT:PULL-DOWN-MENU-ENTRY HIGH-LEVEL FUNCTION	DWT-71
DWT:PULL-DOWN-MENU-ENTRY-CREATE LOW-LEVEL FUNCTION	DWT-71
DWT:PULL-DOWN-MENU-ENTRY-HILITE HIGH-LEVEL FUNCTION	DWT-71
DWT:PUSH-BUTTON HIGH-LEVEL FUNCTION	DWT-72
DWT:PUSH-BUTTON-CREATE LOW-LEVEL FUNCTION	DWT-72
DWT:PUSH-BUTTON-GADGET-CREATE GADGET FUNCTION	DWT-73
DWT:QUERY-GEOMETRY INTRINSIC FUNCTION	DWT-73
DWT:RADIO-BOX HIGH-LEVEL FUNCTION	DWT-73
DWT:RADIO-BOX-CREATE LOW-LEVEL FUNCTION	DWT-74
DWT:REALIZE-WIDGET INTRINSIC FUNCTION	DWT-74
DWT:REALLOC INTRINSIC FUNCTION	DWT-74
DWT:RECOPY-TO-CLIPBOARD CUT-AND-PASTE FUNCTION	DWT-75
DWT:REGISTER-CASE-CONVERTER INTRINSIC FUNCTION	DWT-75
DWT:REGISTER-CLASS DRM FUNCTION	DWT-76
DWT:REGISTER-DRM-NAMES DRM FUNCTION	DWT-76
DWT:REMOVE-ALL-CALLBACKS INTRINSIC FUNCTION	DWT-76
DWT:REMOVE-CALLBACK INTRINSIC FUNCTION	DWT-77
DWT:REMOVE-CALLBACKS INTRINSIC FUNCTION	DWT-77

DWT:REMOVE-EVENT-HANDLER INTRINSIC FUNCTION	DWT-77
DWT:REMOVE-GRAB INTRINSIC FUNCTION	DWT-78
DWT:REMOVE-INPUT INTRINSIC FUNCTION	DWT-78
DWT:REMOVE-RAW-EVENT-HANDLER INTRINSIC FUNCTION	DWT-78
DWT:REMOVE-TIME-OUT INTRINSIC FUNCTION	DWT-79
DWT:REMOVE-WORK-PROC INTRINSIC FUNCTION	DWT-79
DWT:RESIZE-WIDGET INTRINSIC FUNCTION	DWT-79
DWT:RESIZE-WINDOW INTRINSIC FUNCTION	DWT-80
DWT:SCALE HIGH-LEVEL FUNCTION	DWT-80
DWT:SCALE-CREATE LOW-LEVEL FUNCTION	DWT-81
DWT:SCALE-GET-SLIDER HIGH-LEVEL FUNCTION	DWT-81
DWT:SCALE-SET-SLIDER HIGH-LEVEL FUNCTION	DWT-81
DWT:SCREEN INTRINSIC FUNCTION	DWT-82
DWT:SCROLL-BAR HIGH-LEVEL FUNCTION	DWT-82
DWT:SCROLL-BAR-CREATE LOW-LEVEL FUNCTION	DWT-83
DWT:SCROLL-BAR-GET-SLIDER HIGH-LEVEL FUNCTION	DWT-83
DWT:SCROLL-BAR-SET-SLIDER HIGH-LEVEL FUNCTION	DWT-84
DWT:SCROLL-WINDOW HIGH-LEVEL FUNCTION	DWT-84
DWT:SCROLL-WINDOW-CREATE LOW-LEVEL FUNCTION	DWT-84
DWT:SCROLL-WINDOW-SET-AREAS HIGH-LEVEL FUNCTION	DWT-85
DWT:SELECTION HIGH-LEVEL FUNCTION	DWT-85
DWT:SELECTION-CREATE LOW-LEVEL FUNCTION	DWT-86
DWT:SEPARATOR HIGH-LEVEL FUNCTION	DWT-86
DWT:SEPARATOR-CREATE LOW-LEVEL FUNCTION	DWT-86
DWT:SEPARATOR-GADGET-CREATE GADGET FUNCTION	DWT-87
DWT:SET-ERROR-HANDLER INTRINSIC FUNCTION	DWT-87
DWT:SET-ERROR-MSG-HANDLER INTRINSIC FUNCTION	DWT-87
DWT:SET-KEYBOARD-FOCUS INTRINSIC FUNCTION	DWT-88
DWT:SET-KEY-TRANSLATOR INTRINSIC FUNCTION	DWT-88
DWT:SET-MAPPED-WHEN-MANAGED INTRINSIC FUNCTION	DWT-88
DWT:SET-SELECTION-TIMEOUT INTRINSIC FUNCTION	DWT-89
DWT:SET-SENSITIVE INTRINSIC FUNCTION	DWT-89
DWT:SET-SUBVALUES INTRINSIC FUNCTION	DWT-89
DWT:SET-VALUES INTRINSIC FUNCTION	DWT-90
DWT:SET-WARNING-HANDLER INTRINSIC FUNCTION	DWT-90
DWT:SET-WARNING-MSG-HANDLER INTRINSIC FUNCTION	DWT-90
DWT:S-TEXT HIGH-LEVEL FUNCTION	DWT-91
DWT:S-TEXT-CLEAR-SELECTION HIGH-LEVEL FUNCTION	DWT-91
DWT:S-TEXT-CREATE LOW-LEVEL FUNCTION	DWT-91
DWT:S-TEXT-GET-EDITABLE HIGH-LEVEL FUNCTION	DWT-92
DWT:S-TEXT-GET-MAX-LENGTH HIGH-LEVEL FUNCTION	DWT-92
DWT:S-TEXT-GET-SELECTION HIGH-LEVEL FUNCTION	DWT-92
DWT:S-TEXT-GET-STRING HIGH-LEVEL FUNCTION	DWT-93
DWT:S-TEXT-REPLACE HIGH-LEVEL FUNCTION	DWT-93
DWT:S-TEXT-SET-EDITABLE HIGH-LEVEL FUNCTION	DWT-93
DWT:S-TEXT-SET-MAX-LENGTH HIGH-LEVEL FUNCTION	DWT-94
DWT:S-TEXT-SET-SELECTION HIGH-LEVEL FUNCTION	DWT-94
DWT:S-TEXT-SET-STRING HIGH-LEVEL FUNCTION	DWT-94
DWT:STRING COMPOUND STRING FUNCTION	DWT-95
DWT:SUPERCLASS INTRINSIC FUNCTION	DWT-95
DWT:TOGGLE-BUTTON HIGH-LEVEL FUNCTION	DWT-95
DWT:TOGGLE-BUTTON-CREATE LOW-LEVEL FUNCTION	DWT-96
DWT:TOGGLE-BUTTON-GADGET-CREATE GADGET FUNCTION	DWT-96
DWT:TOGGLE-BUTTON-GET-STATE HIGH-LEVEL FUNCTION	DWT-96

DWT:TOGGLE-BUTTON-SET-STATE HIGH-LEVEL FUNCTION	DWT-97
DWT:TOOLKIT-INITIALIZE INTRINSIC FUNCTION	DWT-97
DWT:TRANSLATE-COORDS INTRINSIC FUNCTION	DWT-97
DWT:TRANSLATE-KEYCODE INTRINSIC FUNCTION	DWT-98
DWT:UNDO-COPY-TO-CLIPBOARD CUT-AND-PASTE FUNCTION	DWT-98
DWT:UNINSTALL-TRANSLATIONS INTRINSIC FUNCTION	DWT-98
DWT:UNMANAGE-CHILD INTRINSIC FUNCTION	DWT-99
DWT:UNMANAGE-CHILDREN INTRINSIC FUNCTION	DWT-99
DWT:UNREALIZE-WIDGET INTRINSIC FUNCTION	DWT-99
DWT:VMS-CLEAR-STRING CONVENIENCE FUNCTION	DWT-100
DWT:VMS-FREE-ARGNAMES CONVENIENCE FUNCTION	DWT-100
DWT:VMS-SET-ARG CONVENIENCE FUNCTION	DWT-100
DWT:VMS-SET-CALLBACK-ARG CONVENIENCE FUNCTION	DWT-101
DWT:VMS-SET-DESC-ARG CONVENIENCE FUNCTION	DWT-101
DWT:WARNING INTRINSIC FUNCTION	DWT-101
DWT:WARNING-MSG INTRINSIC FUNCTION	DWT-101
DWT:WIDGET-TO-APPLICATION-CONTEXT INTRINSIC FUNCTION	DWT-102
DWT:WINDOW INTRINSIC FUNCTION	DWT-102
DWT:WINDOW-CREATE LOW-LEVEL FUNCTION	DWT-102
DWT:WINDOW-TO-WIDGET INTRINSIC FUNCTION	DWT-103
DWT:WORK-BOX HIGH-LEVEL FUNCTION	DWT-103
DWT:WORK-BOX-CREATE LOW-LEVEL FUNCTION	DWT-104
DWT:XERROR INTRINSIC FUNCTION	DWT-104
DWT:XTSTRING COMPOUND STRING FUNCTION	DWT-104
DWT:XTWARNING INTRINSIC FUNCTION	DWT-105

Part III Guide to Programming CLX

Chapter 4 Programming Overview of CLX

4.1 Overview of CLX	4-1
4.2 CLX Data Types	4-2
4.3 A Sample CLX Program	4-5

Chapter 5 Managing the Client-Server Connection

5.1 Opening the Display	5-1
5.2 Getting Information About the Display and Screen	5-2
5.3 Sending Requests to the Server	5-4
5.4 Closing the Display	5-4

Chapter 6	Working with Windows	
6.1	Creating Windows	6-1
6.2	Destroying Windows	6-3
6.3	Mapping and Unmapping Windows	6-4
6.4	Associating Properties with Windows	6-5
6.4.1	Communicating with the Window Manager	6-7
6.4.2	Exchanging Properties Between Clients	6-9
6.5	Stacking Windows	6-10
6.6	Changing Window Attributes	6-10
Chapter 7	Defining Graphics Characteristics	
7.1	Creating a Graphics Context	7-1
7.2	Querying and Setting GContext Components	7-4
7.3	Copying and Freeing GContexts	7-5
7.4	Using GContexts Efficiently	7-6
Chapter 8	Using Color	
8.1	Pixels and Colormaps	8-1
8.2	Matching Color Requirements to Screen Types	8-2
8.3	Sharing Color Resources	8-3
8.3.1	Using Named Colors	8-3
8.3.2	Specifying Exact Color Values	8-4
8.4	Allocating Colors for Exclusive Use	8-4
8.4.1	Specifying a Colormap	8-5
8.4.2	Allocating Color Cells	8-5
8.5	Storing Color Values	8-6
8.6	Freeing Color Resources	8-7
8.7	Querying Colormap Entries	8-8
Chapter 9	Graphics Functions	
9.1	Drawing Points	9-1
9.2	Drawing Lines	9-4
9.3	Drawing and Filling Rectangles	9-6

9.4	Drawing and Filling Arcs	9–7
9.5	Clearing and Copying Areas	9–9
9.6	Creating Cursors	9–10

Chapter 10 Using Pixmaps and Images

10.1	Creating and Freeing Pixmaps	10–1
10.2	Creating and Managing Bitmap Files	10–2
10.3	Working with Images	10–3
10.3.1	Getting Images from the Server	10–6
10.3.2	Displaying Images	10–6

Chapter 11 Writing Text

11.1	Characters and Fonts	11–1
11.2	Specifying Fonts	11–3
11.3	Computing Text Size	11–5
11.4	Drawing Text on the Screen	11–6

Chapter 12 Event Functions

12.1	Selecting Events	12–1
12.1.1	Constructing Event Masks	12–2
12.1.2	Specifying an Event Mask	12–4
12.1.3	Changing an Event Mask	12–4
12.2	Processing Events	12–5
12.2.1	Locking the Event Queue	12–6
12.2.2	Finding the Length of the Event Queue	12–6
12.2.3	Handling Events in the Queue	12–6
12.2.4	Adding an Event to the Queue	12–9
12.2.5	Removing an Event from the Queue	12–9
12.3	Controlling Events	12–9
12.3.1	Grabbing the Pointer	12–11
12.3.2	Grabbing the Keyboard	12–12
12.3.3	Grabbing the Server	12–14
12.3.4	Allowing Events	12–14
12.4	Sending Events	12–15
12.5	Event Keys	12–16

Chapter 13 Window and Session Management

13.1	Reparenting Windows	13-1
13.2	Customizing the Keyboard and the Pointer	13-1
13.2.1	Ringing the Bell	13-2
13.2.2	Keyboard and Pointer Mappings	13-2
13.2.3	Keycode Mapping	13-3
13.2.4	Keyboard and Pointer Controls	13-5
13.2.5	Setting Pointer Controls	13-6
13.3	Using the Screen Saver	13-7
13.3.1	Querying the Screen Saver	13-7
13.3.2	Setting the Screen Saver	13-7
13.3.3	Enabling the Screen Saver	13-8
13.4	Controlling Network Access	13-8
13.4.1	Adding and Removing Hosts	13-9
13.4.2	Getting Information About Hosts	13-9
13.5	Closing the Connection	13-10
13.5.1	Deallocating Resources	13-10
13.5.2	Disconnecting Other Clients	13-10
13.5.3	Saving Windows	13-11
13.6	Finding Extensions	13-11

Part IV Reference to Common LISP X

CLX:ACCESS-CONTROL FUNCTION	CLX-1
CLX:ACCESS-HOSTS FUNCTION	CLX-1
CLX:ACTIVATE-SCREEN-SAVER FUNCTION	CLX-2
CLX:ADD-ACCESS-HOST FUNCTION	CLX-2
CLX:ADD-TO-SAVE-SET FUNCTION	CLX-3
CLX:ALIST TYPE SPECIFIER	CLX-3
CLX:ALLOC-COLOR FUNCTION	CLX-3
CLX:ALLOC-COLOR-CELLS FUNCTION	CLX-4
CLX:ALLOC-COLOR-PLANES FUNCTION	CLX-5
CLX:ALLOW-EVENTS FUNCTION	CLX-6
CLX:ANGLE TYPE SPECIFIER	CLX-7
CLX:ARC-SEQ TYPE SPECIFIER	CLX-7
CLX:ARRAY-INDEX TYPE SPECIFIER	CLX-7
CLX:ATOM-NAME FUNCTION	CLX-7
CLX:BELL FUNCTION	CLX-8
CLX:BIT-GRAVITY TYPE SPECIFIER	CLX-8
CLX:BITMAP TYPE SPECIFIER	CLX-9
CLX:BITMAP-FORMAT STRUCTURE	CLX-9
CLX:BITMAP-FORMAT-LSB-FIRST-P FUNCTION	CLX-9
CLX:BITMAP-FORMAT-P FUNCTION	CLX-10
CLX:BITMAP-FORMAT-PAD FUNCTION	CLX-10
CLX:BITMAP-FORMAT-UNIT FUNCTION	CLX-11
CLX:BOOLEAN TYPE SPECIFIER	CLX-11
CLX:CARD8 TYPE SPECIFIER	CLX-11
CLX:CARD16 TYPE SPECIFIER	CLX-11
CLX:CARD29 TYPE SPECIFIER	CLX-12
CLX:CHANGE-ACTIVE-POINTER-GRAB FUNCTION	CLX-12

CLX:CHANGE-KEYBOARD-CONTROL FUNCTION	CLX-12
CLX:CHANGE-KEYBOARD-MAPPING FUNCTION	CLX-13
CLX:CHANGE-POINTER-CONTROL FUNCTION	CLX-14
CLX:CHANGE-PROPERTY FUNCTION	CLX-15
CLX:CHARACTER->KEYSYMS FUNCTION	CLX-16
CLX:CHAR-ASCENT FUNCTION	CLX-16
CLX:CHAR-ATTRIBUTES FUNCTION	CLX-17
CLX:CHAR-DESCENT FUNCTION	CLX-17
CLX:CHAR-LEFT-BEARING FUNCTION	CLX-18
CLX:CHAR-RIGHT-BEARING FUNCTION	CLX-18
CLX:CHAR-WIDTH FUNCTION	CLX-19
CLX:CIRCULATE-WINDOW-DOWN FUNCTION	CLX-19
CLX:CIRCULATE-WINDOW-UP FUNCTION	CLX-20
CLX:CLEAR-AREA FUNCTION	CLX-20
CLX:CLOSE-DISPLAY FUNCTION	CLX-21
CLX:CLOSE-DOWN-MODE FUNCTION	CLX-21
CLX:CLOSE-FONT FUNCTION	CLX-22
CLX:COLOR STRUCTURE	CLX-22
CLX:COLOR-BLUE FUNCTION	CLX-23
CLX:COLOR-GREEN FUNCTION	CLX-23
CLX:COLORMAP STRUCTURE	CLX-23
CLX:COLORMAP-DISPLAY FUNCTION	CLX-24
CLX:COLORMAP-EQUAL FUNCTION	CLX-24
CLX:COLORMAP-ID FUNCTION	CLX-25
CLX:COLORMAP-P FUNCTION	CLX-25
CLX:COLOR-P FUNCTION	CLX-26
CLX:COLOR-RED FUNCTION	CLX-26
CLX:COLOR-RGB FUNCTION	CLX-26
CLX:CONVERT-SELECTION FUNCTION	CLX-27
CLX:COPY-AREA FUNCTION	CLX-28
CLX:COPY-COLORMAP-AND-FREE FUNCTION	CLX-28
CLX:COPY-GCONTEXT FUNCTION	CLX-29
CLX:COPY-GCONTEXT-COMPONENTS FUNCTION	CLX-29
CLX:COPY-IMAGE FUNCTION	CLX-30
CLX:COPY-PLANE FUNCTION	CLX-30
CLX:CREATE-COLORMAP FUNCTION	CLX-31
CLX:CREATE-CURSOR FUNCTION	CLX-32
CLX:CREATE-GCONTEXT FUNCTION	CLX-32
CLX:CREATE-GLYPH-CURSOR FUNCTION	CLX-34
CLX:CREATE-IMAGE FUNCTION	CLX-35
CLX:CREATE-PIXMAP FUNCTION	CLX-36
CLX:CREATE-WINDOW FUNCTION	CLX-36
CLX:CURSOR-DISPLAY FUNCTION	CLX-38
CLX:CURSOR-EQUAL FUNCTION	CLX-39
CLX:CURSOR-ID FUNCTION	CLX-39
CLX:CURSOR-P FUNCTION	CLX-40
CLX:DELETE-PROPERTY FUNCTION	CLX-40
CLX:DESTROY-SUBWINDOWS FUNCTION	CLX-41
CLX:DESTROY-WINDOW FUNCTION	CLX-41
CLX:DEVICE-EVENT-MASK TYPE SPECIFIER	CLX-42
CLX:DEVICE-EVENT-MASK-CLASS TYPE SPECIFIER	CLX-42
CLX:DISCARD-CURRENT-EVENT FUNCTION	CLX-42
CLX:DISCARD-FONT-INFO FUNCTION	CLX-43
CLX:DISPLAY STRUCTURE	CLX-43
CLX:DISPLAY-AFTER-FUNCTION FUNCTION	CLX-44
CLX:DISPLAY-BITMAP-FORMAT FUNCTION	CLX-44
CLX:DISPLAY-DEFAULT-SCREEN FUNCTION	CLX-45

CLX:DISPLAY-FINISH-OUTPUT FUNCTION	CLX-45
CLX:DISPLAY-FORCE-OUTPUT FUNCTION	CLX-46
CLX:DISPLAY-IMAGE-LSB-FIRST-P FUNCTION	CLX-46
CLX:DISPLAY-MAX-KEYCODE FUNCTION	CLX-47
CLX:DISPLAY-MAX-REQUEST-LENGTH FUNCTION	CLX-47
CLX:DISPLAY-MIN-KEYCODE FUNCTION	CLX-48
CLX:DISPLAY-MOTION-BUFFER-SIZE FUNCTION	CLX-48
CLX:DISPLAY-P FUNCTION	CLX-49
CLX:DISPLAY-PIXMAP-FORMATS FUNCTION	CLX-49
CLX:DISPLAY-PROTOCOL-MAJOR-VERSION FUNCTION	CLX-49
CLX:DISPLAY-PROTOCOL-MINOR-VERSION FUNCTION	CLX-50
CLX:DISPLAY-RELEASE-NUMBER FUNCTION	CLX-50
CLX:DISPLAY-ROOTS FUNCTION	CLX-51
CLX:DISPLAY-VENDOR-NAME FUNCTION	CLX-51
CLX:DRAWABLE TYPE SPECIFIER	CLX-52
CLX:DRAWABLE-BORDER-WIDTH FUNCTION	CLX-52
CLX:DRAWABLE-DEPTH FUNCTION	CLX-52
CLX:DRAWABLE-DISPLAY FUNCTION	CLX-53
CLX:DRAWABLE-EQUAL FUNCTION	CLX-53
CLX:DRAWABLE-HEIGHT FUNCTION	CLX-54
CLX:DRAWABLE-ID FUNCTION	CLX-54
CLX:DRAWABLE-P FUNCTION	CLX-55
CLX:DRAWABLE-ROOT FUNCTION	CLX-55
CLX:DRAWABLE-WIDTH FUNCTION	CLX-56
CLX:DRAWABLE-X FUNCTION	CLX-56
CLX:DRAWABLE-Y FUNCTION	CLX-57
CLX:DRAW-ARC FUNCTION	CLX-57
CLX:DRAW-ARCS FUNCTION	CLX-58
CLX:DRAW-DIRECTION TYPE SPECIFIER	CLX-59
CLX:DRAW-GLYPH FUNCTION	CLX-59
CLX:DRAW-GLYPHS FUNCTION	CLX-60
CLX:DRAW-IMAGE-GLYPH FUNCTION	CLX-61
CLX:DRAW-IMAGE-GLYPHS FUNCTION	CLX-62
CLX:DRAW-LINE FUNCTION	CLX-63
CLX:DRAW-LINES FUNCTION	CLX-64
CLX:DRAW-POINT FUNCTION	CLX-64
CLX:DRAW-POINTS FUNCTION	CLX-65
CLX:DRAW-RECTANGLE FUNCTION	CLX-66
CLX:DRAW-RECTANGLES FUNCTION	CLX-66
CLX:DRAW-SEGMENTS FUNCTION	CLX-67
CLX:EVENT-CASE MACRO	CLX-68
CLX:EVENT-KEY TYPE SPECIFIER	CLX-69
CLX:EVENT-LISTEN FUNCTION	CLX-69
CLX:EVENT-MASK TYPE SPECIFIER	CLX-70
CLX:EVENT-MASK-CLASS TYPE SPECIFIER	CLX-70
CLX:FIND-ATOM FUNCTION	CLX-70
CLX:FONT STRUCTURE	CLX-71
CLX:FONTABLE TYPE SPECIFIER	CLX-71
CLX:FONT-ALL-CHARS-EXIST-P FUNCTION	CLX-72
CLX:FONT-ASCENT FUNCTION	CLX-72
CLX:FONT-DEFAULT-CHAR FUNCTION	CLX-72
CLX:FONT-DESCENT FUNCTION	CLX-73
CLX:FONT-DIRECTION FUNCTION	CLX-73
CLX:FONT-DISPLAY FUNCTION	CLX-74
CLX:FONT-EQUAL FUNCTION	CLX-74
CLX:FONT-ID FUNCTION	CLX-75
CLX:FONT-MAX-BYTE1 FUNCTION	CLX-75

CLX:FONT-MAX-BYTE2 FUNCTION	CLX-75
CLX:FONT-MAX-CHAR FUNCTION	CLX-76
CLX:FONT-MIN-BYTE1 FUNCTION	CLX-76
CLX:FONT-MIN-BYTE2 FUNCTION	CLX-77
CLX:FONT-MIN-CHAR FUNCTION	CLX-77
CLX:FONT-NAME FUNCTION	CLX-78
CLX:FONT-P FUNCTION	CLX-78
CLX:FONT-PATH FUNCTION	CLX-79
CLX:FONT-PROPERTIES FUNCTION	CLX-79
CLX:FONT-PROPERTY FUNCTION	CLX-80
CLX:FONT-PROPS TYPE SPECIFIER	CLX-80
CLX:FORCE-GCONTEXT-CHANGES FUNCTION	CLX-80
CLX:FREE-COLORMAP FUNCTION	CLX-81
CLX:FREE-COLORS FUNCTION	CLX-81
CLX:FREE-CURSOR FUNCTION	CLX-82
CLX:FREE-GCONTEXT FUNCTION	CLX-82
CLX:FREE-PIXMAP FUNCTION	CLX-83
CLX:GCONTEXT STRUCTURE	CLX-83
CLX:GCONTEXT-ARC-MODE FUNCTION	CLX-84
CLX:GCONTEXT-BACKGROUND FUNCTION	CLX-85
CLX:GCONTEXT-CACHE-P FUNCTION	CLX-85
CLX:GCONTEXT-CAP-STYLE FUNCTION	CLX-86
CLX:GCONTEXT-CLIP-MASK FUNCTION	CLX-86
CLX:GCONTEXT-CLIP-X FUNCTION	CLX-87
CLX:GCONTEXT-CLIP-Y FUNCTION	CLX-87
CLX:GCONTEXT-DASHES FUNCTION	CLX-88
CLX:GCONTEXT-DASH-OFFSET FUNCTION	CLX-88
CLX:GCONTEXT-DISPLAY FUNCTION	CLX-89
CLX:GCONTEXT-EQUAL FUNCTION	CLX-89
CLX:GCONTEXT-EXPOSURES FUNCTION	CLX-90
CLX:GCONTEXT-FILL-RULE FUNCTION	CLX-90
CLX:GCONTEXT-FILL-STYLE FUNCTION	CLX-91
CLX:GCONTEXT-FONT FUNCTION	CLX-91
CLX:GCONTEXT-FOREGROUND FUNCTION	CLX-92
CLX:GCONTEXT-FUNCTION FUNCTION	CLX-92
CLX:GCONTEXT-ID FUNCTION	CLX-93
CLX:GCONTEXT-JOIN-STYLE FUNCTION	CLX-93
CLX:GCONTEXT-KEY TYPE SPECIFIER	CLX-94
CLX:GCONTEXT-LINE-STYLE FUNCTION	CLX-94
CLX:GCONTEXT-LINE-WIDTH FUNCTION	CLX-94
CLX:GCONTEXT-P FUNCTION	CLX-95
CLX:GCONTEXT-PLANE-MASK FUNCTION	CLX-95
CLX:GCONTEXT-STIPPLE FUNCTION	CLX-96
CLX:GCONTEXT-SUBWINDOW-MODE FUNCTION	CLX-96
CLX:GCONTEXT-TILE FUNCTION	CLX-97
CLX:GCONTEXT-TS-X FUNCTION	CLX-97
CLX:GCONTEXT-TS-Y FUNCTION	CLX-98
CLX:GET-IMAGE FUNCTION	CLX-98
CLX:GET-PROPERTY FUNCTION	CLX-99
CLX:GET-RAW-IMAGE FUNCTION	CLX-100
CLX:GET-WM-CLASS FUNCTION	CLX-101
CLX:GLOBAL-POINTER-POSITION FUNCTION	CLX-101
CLX:GRAB-BUTTON FUNCTION	CLX-102
CLX:GRAB-KEY FUNCTION	CLX-103
CLX:GRAB-KEYBOARD FUNCTION	CLX-104
CLX:GRAB-POINTER FUNCTION	CLX-104
CLX:GRAB-SERVER FUNCTION	CLX-105

CLX:GRAB-STATUS TYPE SPECIFIER	CLX-106
CLX:ICON-SIZES FUNCTION	CLX-106
CLX:IMAGE STRUCTURE	CLX-106
CLX:IMAGE-BIT-LSB-FIRST-P FUNCTION	CLX-107
CLX:IMAGE-BITS-PER-PIXEL FUNCTION	CLX-108
CLX:IMAGE-BLUE-MASK FUNCTION	CLX-108
CLX:IMAGE-BYTE-LSB-FIRST-P FUNCTION	CLX-109
CLX:IMAGE-BYTES-PER-LINE FUNCTION	CLX-109
CLX:IMAGE-DEPTH FUNCTION	CLX-110
CLX:IMAGE-DEPTH TYPE SPECIFIER	CLX-110
CLX:IMAGE-FORMAT FUNCTION	CLX-110
CLX:IMAGE-GREEN-MASK FUNCTION	CLX-111
CLX:IMAGE-HEIGHT FUNCTION	CLX-111
CLX:IMAGE-P FUNCTION	CLX-112
CLX:IMAGE-RED-MASK FUNCTION	CLX-112
CLX:IMAGE-SCANLINE-PAD FUNCTION	CLX-112
CLX:IMAGE-WIDTH FUNCTION	CLX-113
CLX:INDEX-SIZE TYPE SPECIFIER	CLX-113
CLX:INPUT-FOCUS FUNCTION	CLX-114
CLX:INSTALL-COLORMAP FUNCTION	CLX-114
CLX:INSTALLED-COLORMAPS FUNCTION	CLX-115
CLX:INTERN-ATOM FUNCTION	CLX-115
CLX:KEYBOARD-CONTROL FUNCTION	CLX-116
CLX:KEYBOARD-MAPPING FUNCTION	CLX-116
CLX:KEYCODE->CHARACTER FUNCTION	CLX-117
CLX:KEYCODE->KEYSYM FUNCTION	CLX-118
CLX:KEYSYM TYPE SPECIFIER	CLX-119
CLX:KEYSYM->CHARACTER FUNCTION	CLX-119
CLX:KEYSYM->KEYCODES FUNCTION	CLX-120
CLX:KILL-CLIENT FUNCTION	CLX-120
CLX:KILL-Temporary-CLIENTS FUNCTION	CLX-121
CLX:LIST-EXTENSIONS FUNCTION	CLX-121
CLX:LIST-FONT-NAMES FUNCTION	CLX-122
CLX:LIST-FONTS FUNCTION	CLX-122
CLX:LIST-PROPERTIES FUNCTION	CLX-123
CLX:LOGICAL-OP TYPE SPECIFIER	CLX-124
CLX:LOOKUP-COLOR FUNCTION	CLX-124
CLX:MAKE-COLOR FUNCTION	CLX-125
CLX:MAKE-COLORMAP FUNCTION	CLX-125
CLX:MAKE-CURSOR FUNCTION	CLX-126
CLX:MAKE-DRAWABLE FUNCTION	CLX-126
CLX:MAKE-EVENT-KEYS FUNCTION	CLX-127
CLX:MAKE-EVENT-MASK FUNCTION	CLX-127
CLX:MAKE-FONT FUNCTION	CLX-128
CLX:MAKE-GCONTEXT FUNCTION	CLX-128
CLX:MAKE-PIXMAP FUNCTION	CLX-129
CLX:MAKE-STATE-KEYS FUNCTION	CLX-129
CLX:MAKE-STATE-MASK FUNCTION	CLX-130
CLX:MAKE-VISUAL-INFO FUNCTION	CLX-130
CLX:MAKE-WINDOW FUNCTION	CLX-130
CLX:MAKE-WM-HINTS FUNCTION	CLX-131
CLX:MAKE-WM-SIZE-HINTS FUNCTION	CLX-131
CLX:MAP-SUBWINDOWS FUNCTION	CLX-132
CLX:MAP-WINDOW FUNCTION	CLX-132
CLX:MASK16 TYPE SPECIFIER	CLX-133
CLX:MASK32 TYPE SPECIFIER	CLX-133
CLX:MAX-CHAR-ASCENT FUNCTION	CLX-133

CLX:MAX-CHAR-ATTRIBUTES FUNCTION	CLX-134
CLX:MAX-CHAR-DESCENT FUNCTION	CLX-134
CLX:MAX-CHAR-LEFT-BEARING FUNCTION	CLX-135
CLX:MAX-CHAR-RIGHT-BEARING FUNCTION	CLX-135
CLX:MAX-CHAR-WIDTH FUNCTION	CLX-135
CLX:MIN-CHAR-ASCENT FUNCTION	CLX-136
CLX:MIN-CHAR-ATTRIBUTES FUNCTION	CLX-136
CLX:MIN-CHAR-DESCENT FUNCTION	CLX-137
CLX:MIN-CHAR-LEFT-BEARING FUNCTION	CLX-137
CLX:MIN-CHAR-RIGHT-BEARING FUNCTION	CLX-138
CLX:MIN-CHAR-WIDTH FUNCTION	CLX-138
CLX:MODIFIER-KEY TYPE SPECIFIER	CLX-138
CLX:MODIFIER-MAPPING FUNCTION	CLX-139
CLX:MODIFIER-MASK TYPE SPECIFIER	CLX-139
CLX:MOTION-EVENTS FUNCTION	CLX-140
CLX:OPEN-DISPLAY FUNCTION	CLX-140
CLX:OPEN-FONT FUNCTION	CLX-141
CLX:PIXARRAY TYPE SPECIFIER	CLX-141
CLX:PIXEL TYPE SPECIFIER	CLX-142
CLX:PIXMAP STRUCTURE	CLX-142
CLX:PIXMAP-DISPLAY FUNCTION	CLX-142
CLX:PIXMAP-EQUAL FUNCTION	CLX-143
CLX:PIXMAP-FORMAT STRUCTURE	CLX-143
CLX:PIXMAP-FORMAT-BITS-PER-PIXEL FUNCTION	CLX-144
CLX:PIXMAP-FORMAT-DEPTH FUNCTION	CLX-144
CLX:PIXMAP-FORMAT-P FUNCTION	CLX-145
CLX:PIXMAP-FORMAT-PAD FUNCTION	CLX-145
CLX:PIXMAP-ID FUNCTION	CLX-146
CLX:PIXMAP-P FUNCTION	CLX-146
CLX:POINTER-CONTROL FUNCTION	CLX-146
CLX:POINTER-EVENT-MASK TYPE SPECIFIER	CLX-147
CLX:POINTER-EVENT-MASK-CLASS TYPE SPECIFIER	CLX-147
CLX:POINTER-MAPPING FUNCTION	CLX-147
CLX:POINTER-POSITION FUNCTION	CLX-148
CLX:POINT-SEQ TYPE SPECIFIER	CLX-149
CLX:PROCESS-EVENT FUNCTION	CLX-149
CLX:PUT-IMAGE FUNCTION	CLX-150
CLX:PUT-RAW-IMAGE FUNCTION	CLX-151
CLX:QUERY-BEST-CURSOR FUNCTION	CLX-152
CLX:QUERY-BEST-STIPPLE FUNCTION	CLX-152
CLX:QUERY-BEST-TILE FUNCTION	CLX-153
CLX:QUERY-COLORS FUNCTION	CLX-153
CLX:QUERY-EXTENSION FUNCTION	CLX-154
CLX:QUERY-KEYMAP FUNCTION	CLX-154
CLX:QUERY-POINTER FUNCTION	CLX-155
CLX:QUERY-TREE FUNCTION	CLX-156
CLX:QUEUE-EVENT FUNCTION	CLX-156
CLX:READ-BITMAP-FILE FUNCTION	CLX-157
CLX:RECOLOR-CURSOR FUNCTION	CLX-158
CLX:RECT-SEQ TYPE SPECIFIER	CLX-158
CLX:REMOVE-ACCESS-HOST FUNCTION	CLX-158
CLX:REMOVE-FROM-SAVE-SET FUNCTION	CLX-159
CLX:REARENT-WINDOW FUNCTION	CLX-159
CLX:REPEAT-SEQ TYPE SPECIFIER	CLX-160
CLX:RESET-SCREEN-SAVER FUNCTION	CLX-160
CLX:RESOURCE-ID TYPE SPECIFIER	CLX-160
CLX:RGB-VAL TYPE SPECIFIER	CLX-161

CLX:SCREEN STRUCTURE	CLX-161
CLX:SCREEN-BACKING-STORES FUNCTION	CLX-162
CLX:SCREEN-BLACK-PIXEL FUNCTION	CLX-162
CLX:SCREEN-DEFAULT-COLORMAP FUNCTION	CLX-163
CLX:SCREEN-DEPTHES FUNCTION	CLX-163
CLX:SCREEN-DISPLAY FUNCTION	CLX-164
CLX:SCREEN-EVENT-MASK-AT-OPEN FUNCTION	CLX-164
CLX:SCREEN-HEIGHT FUNCTION	CLX-165
CLX:SCREEN-HEIGHT-IN-MILLIMETERS FUNCTION	CLX-165
CLX:SCREEN-MAX-INSTALLED-MAPS FUNCTION	CLX-165
CLX:SCREEN-MIN-INSTALLED-MAPS FUNCTION	CLX-166
CLX:SCREEN-P FUNCTION	CLX-166
CLX:SCREEN-ROOT FUNCTION	CLX-167
CLX:SCREEN-ROOT-DEPTH FUNCTION	CLX-167
CLX:SCREEN-ROOT-VISUAL FUNCTION	CLX-168
CLX:SCREEN-SAVER FUNCTION	CLX-168
CLX:SCREEN-SAVE-UNDERS-P FUNCTION	CLX-169
CLX:SCREEN-WHITE-PIXEL FUNCTION	CLX-169
CLX:SCREEN-WIDTH FUNCTION	CLX-170
CLX:SCREEN-WIDTH-IN-MILLIMETERS FUNCTION	CLX-170
CLX:SEG-SEQ TYPE SPECIFIER	CLX-171
CLX:SELECTION-OWNER FUNCTION	CLX-171
CLX:SEND-EVENT FUNCTION	CLX-171
CLX:SET-INPUT-FOCUS FUNCTION	CLX-173
CLX:SET-MODIFIER-MAPPING FUNCTION	CLX-173
CLX:SET-SCREEN-SAVER FUNCTION	CLX-174
CLX:SET-WM-CLASS FUNCTION	CLX-174
CLX:STATE-MASK-KEY TYPE SPECIFIER	CLX-175
CLX:STORE-COLOR FUNCTION	CLX-175
CLX:STORE-COLORS FUNCTION	CLX-176
CLX:STRINGABLE TYPE SPECIFIER	CLX-176
CLX:TEXT-EXTENTS FUNCTION	CLX-177
CLX:TEXT-WIDTH FUNCTION	CLX-178
CLX:TIMESTAMP TYPE SPECIFIER	CLX-178
CLX:TRANSLATE-COORDINATES FUNCTION	CLX-179
CLX:TRANSLATE-DEFAULT FUNCTION	CLX-179
CLX:UNGRAB-BUTTON FUNCTION	CLX-180
CLX:UNGRAB-KEY FUNCTION	CLX-181
CLX:UNGRAB-KEYBOARD FUNCTION	CLX-181
CLX:UNGRAB-POINTER FUNCTION	CLX-182
CLX:UNGRAB-SERVER FUNCTION	CLX-182
CLX:UNINSTALL-COLORMAP FUNCTION	CLX-183
CLX:UNMAP-SUBWINDOWS FUNCTION	CLX-183
CLX:UNMAP-WINDOW FUNCTION	CLX-184
CLX:VISUAL TYPE SPECIFIER	CLX-184
CLX:VISUAL-INFO STRUCTURE	CLX-184
CLX:VISUAL-INFO-BITS-PER-RGB FUNCTION	CLX-185
CLX:VISUAL-INFO-BLUE-MASK FUNCTION	CLX-185
CLX:VISUAL-INFO-CLASS FUNCTION	CLX-186
CLX:VISUAL-INFO-COLORMAP-ENTRIES FUNCTION	CLX-186
CLX:VISUAL-INFO-GREEN-MASK FUNCTION	CLX-187
CLX:VISUAL-INFO-ID FUNCTION	CLX-187
CLX:VISUAL-INFO-P FUNCTION	CLX-188
CLX:VISUAL-INFO-RED-MASK FUNCTION	CLX-188
CLX:WARP-POINTER FUNCTION	CLX-188
CLX:WARP-POINTER-IF-INSIDE FUNCTION	CLX-189
CLX:WARP-POINTER-RELATIVE FUNCTION	CLX-190

CLX:WARP-POINTER-RELATIVE-IF-INSIDE FUNCTION	CLX-190
CLX:WINDOW-STRUCTURE	CLX-191
CLX:WINDOW-ALL-EVENT-MASKS FUNCTION	CLX-193
CLX:WINDOW-BACKING-PIXEL FUNCTION	CLX-194
CLX:WINDOW-BACKING-PLANES FUNCTION	CLX-194
CLX:WINDOW-BACKING-STORE FUNCTION	CLX-195
CLX:WINDOW-BIT-GRAVITY FUNCTION	CLX-195
CLX:WINDOW-CLASS FUNCTION	CLX-196
CLX:WINDOW-COLORMAP FUNCTION	CLX-196
CLX:WINDOW-COLORMAP-INSTALLED-P FUNCTION	CLX-197
CLX:WINDOW-DISPLAY FUNCTION	CLX-197
CLX:WINDOW-DO-NOT-PROPAGATE-MASK FUNCTION	CLX-198
CLX:WINDOW-ID FUNCTION	CLX-198
CLX:WINDOW-EQUAL FUNCTION	CLX-198
CLX:WINDOW-EVENT-MASK FUNCTION	CLX-199
CLX:WINDOW-GRAVITY FUNCTION	CLX-199
CLX:WINDOW-ID FUNCTION	CLX-200
CLX:WINDOW-MAP-STATE FUNCTION	CLX-200
CLX:WINDOW-OVERRIDE-REDIRECT FUNCTION	CLX-201
CLX:WINDOW-SAVE-UNDER FUNCTION	CLX-201
CLX:WINDOW-VISUAL FUNCTION	CLX-202
CLX:WIN-GRAVITY TYPE SPECIFIER	CLX-203
CLX:WITH-DISPLAY MACRO	CLX-203
CLX:WITH-EVENT-QUEUE MACRO	CLX-204
CLX:WITH-GCONTEXT MACRO	CLX-204
CLX:WITH-SERVER-GRABBED MACRO	CLX-206
CLX:WITH-STATE MACRO	CLX-207
CLX:WM-HINTS FUNCTION	CLX-208
CLX:WM-HINTS STRUCTURE	CLX-209
CLX:WM-HINTS-FLAGS FUNCTION	CLX-209
CLX:WM-HINTS-ICON-MASK FUNCTION	CLX-210
CLX:WM-HINTS-ICON-PIXMAP FUNCTION	CLX-210
CLX:WM-HINTS-ICON-WINDOW FUNCTION	CLX-211
CLX:WM-HINTS-ICON-X FUNCTION	CLX-211
CLX:WM-HINTS-ICON-Y FUNCTION	CLX-212
CLX:WM-HINTS-INITIAL-STATE FUNCTION	CLX-212
CLX:WM-HINTS-INPUT FUNCTION	CLX-213
CLX:WM-HINTS-P FUNCTION	CLX-213
CLX:WM-HINTS-WINDOW-GROUP FUNCTION	CLX-214
CLX:WM-ICON-NAME FUNCTION	CLX-214
CLX:WM-NORMAL-HINTS FUNCTION	CLX-215
CLX:WM-SIZE-HINTS STRUCTURE	CLX-215
CLX:WM-SIZE-HINTS-HEIGHT-INC FUNCTION	CLX-216
CLX:WM-SIZE-HINTS-MAX-ASPECT FUNCTION	CLX-217
CLX:WM-SIZE-HINTS-MAX-HEIGHT FUNCTION	CLX-217
CLX:WM-SIZE-HINTS-MAX-WIDTH FUNCTION	CLX-218
CLX:WM-SIZE-HINTS-MIN-HEIGHT FUNCTION	CLX-219
CLX:WM-SIZE-HINTS-USER-SPECIFIED-SIZE-P FUNCTION	CLX-219
CLX:WM-SIZE-HINTS-WIDTH FUNCTION	CLX-220
CLX:WM-SIZE-HINTS-P FUNCTION	CLX-220
CLX:WM-SIZE-HINTS-USER-SPECIFIED-POSITION-P FUNCTION	CLX-221
CLX:WM-SIZE-HINTS-USER-SPECIFIED-SIZE-P FUNCTION	CLX-221
CLX:WM-SIZE-HINTS-WIDTH-INC FUNCTION	CLX-222
CLX:WM-SIZE-HINTS-X FUNCTION	CLX-223

CLX:WM-SIZE-HINTS-Y FUNCTION	CLX-223
CLX:WM-ZOOM-HINTS FUNCTION	CLX-224
CLX:WRITE-BITMAP-FILE FUNCTION	CLX-224
CLX:XATOM TYPE SPECIFIER	CLX-225

Appendix A DECwindows Constants

Appendix B CLX to X11 Mappings

Index

Examples

2-1 Defining the UIL Module	2-4
2-2 Defining the Main Window	2-5
2-3 Defining a Simple Text Widget	2-5
2-4 Defining a Menu Bar	2-6
2-5 Defining Pull-Down Menu Entries	2-6
2-6 Defining Widget Variables	2-8
2-7 Defining Widgets with High-Level Functions	2-8
2-8 Creating a Message Box	2-10
3-1 Initializing a UIL Application	3-2
3-2 Creating a Widget Hierarchy	3-3
3-3 Realizing the User Interface	3-4
3-4 Calling the Main Loop	3-4
3-5 Creating a List of Callback Names	3-5
3-6 Creating a List of Callback Entry Points	3-5
3-7 Accessing the Callback Widget	3-6
3-8 Writing to a Text Widget	3-7
3-9 Pretty-Printing a Recipe	3-7
3-10 Accessing Callback Arguments	3-8
3-11 Printing Calorie Information	3-8
4-1 Sample CLX Program	4-5
6-1 Creating a Window	6-3
6-2 Changing Window Attributes	6-12
9-1 Drawing Points	9-2
9-2 Drawing Lines and Line Segments	9-6
9-3 Drawing an Arc	9-8
9-4 Creating a Cursor	9-11
11-1 Specifying Fonts	11-4
11-2 Drawing Text	11-8
12-1 Using the CLX:PROCESS-EVENT Function	12-7
12-2 Using the CLX:EVENT-CASE Macro	12-8

Figures

1–1	DECwindows Layered Architecture	1–3
1–2	Hello World! Application	1–6
1–3	Hello World! Widget Hierarchy	1–7
2–1	Choosing from a Pull-Down Menu	2–2
2–2	A Widget Hierarchy	2–3
2–3	A Message Box	2–11
3–1	DECwindows Callback Argument List	3–6
9–1	Drawing Points	9–4
9–2	Drawing Lines and Line Segments	9–6
9–3	Drawing an Arc	9–8
11–1	Drawing Text	11–9
12–1	The Event Queue	12–5

Tables

1–1	Types of DECwindows Widgets	1–4
4–1	CLX Data Types	4–2
5–1	Information Functions for DISPLAY Objects	5–2
5–2	Information Functions for CLX:SCREEN Objects	5–3
5–3	Output Buffer Routines	5–4
6–1	CLX:CREATE-WINDOW Keywords	6–2
6–2	Information Functions for CLX:WM-HINTS Objects	6–7
6–3	Information Functions for CLX:WM-SIZE-HINTS Objects	6–8
6–4	Additional Window Manager Property Functions	6–9
6–5	Information Functions for CLX:WINDOW Objects	6–11
6–6	SETF Functions for CLX:WINDOW Objects	6–12
6–7	Information Functions for CLX:DRAWABLE Objects	6–12
7–1	CLX:CREATE-GCONTEXT Keywords	7–2
7–2	CLX:LOGICAL-OP Values	7–3
7–3	Information Functions for CLX:GCONTEXT Objects	7–4
8–1	Information Functions for CLX:VISUAL-INFO Objects	8–2
8–2	Information Functions for CLX:COLOR Objects	8–4
8–3	Information Functions for CLX:COLORMAP Objects	8–5
10–1	Information Functions for Pixmaps	10–2
10–2	Information Functions for Pixmap Formats	10–2
10–3	Information Functions for Bitmap-Formats	10–3
10–4	Information Functions for CLX:IMAGE Objects	10–4
10–5	Information Functions for :XY-PIXMAP Images	10–5
10–6	Information Functions for :Z-PIXMAP Images	10–5
11–1	Information Functions for Characters	11–1
11–2	Information Functions for CLX:FONT Objects	11–2
12–1	Event Keys	12–2
12–2	Event Masks	12–2
12–3	Pointer Event Masks	12–10
12–4	State Masks	12–10
12–5	Modifier Masks	12–10

12-6	:BUTTON-PRESS Event-Key	12-16
12-7	:BUTTON-RELEASE Event-Key	12-16
12-8	:CIRCULATE-NOTIFY Event-Key	12-17
12-9	:CIRCULATE-REQUEST Event-Key	12-17
12-10	:CLIENT-MESSAGE Event-Key	12-17
12-11	:COLORMAP-NOTIFY Event-Key	12-18
12-12	:CONFIGURE-NOTIFY Event-Key	12-18
12-13	:CONFIGURE-REQUEST Event-Key	12-19
12-14	:CREATE-NOTIFY Event-Key	12-19
12-15	:DESTROY-NOTIFY Event-Key	12-20
12-16	:ENTER-NOTIFY Event-Key	12-20
12-17	:EXPOSURE Event-Key	12-21
12-18	:FOCUS-IN Event-Key	12-21
12-19	:FOCUS-OUT Event-Key	12-21
12-20	:GRAPHICS-EXPOSURE Event-Key	12-22
12-21	:GRAVITY-NOTIFY Event-Key	12-22
12-22	:KEYMAP-NOTIFY Event-Key	12-23
12-23	:KEY-PRESS Event-Key	12-23
12-24	:KEY-RELEASE Event-Key	12-24
12-25	:LEAVE-NOTIFY Event-Key	12-24
12-26	:MAPPING-NOTIFY Event-Key	12-25
12-27	:MAP-NOTIFY Event-Key	12-25
12-28	:MAP-REQUEST Event-Key	12-26
12-29	:MOTION-NOTIFY Event-Key	12-26
12-30	:NO-EXPOSURE Event-Key	12-26
12-31	:PROPERTY-NOTIFY Event-Key	12-27
12-32	:REARENT-NOTIFY Event-Key	12-27
12-33	:RESIZE-REQUEST Event-Key	12-27
12-34	:SELECTION-CLEAR Event-Key	12-28
12-35	:SELECTION-NOTIFY Event-Key	12-28
12-36	:SELECTION-REQUEST Event-Key	12-28
12-37	:UNMAP-NOTIFY Event-Key	12-29
12-38	:VISIBILITY-NOTIFY Event-Key	12-29
CLX-1	WITH-STATE Accessor and SETF Groups	CLX-207
A-1	DECwindows Constants Defined by VAX LISP	A-1
A-2	DRM Constants	A-3
A-3	Callback Reasons	A-5
B-1	Mapping X11 Requests to CLX Functions	B-2
B-2	Mapping CLX Functions and Macros to X11 Requests	B-10

Preface

This manual provides information on writing VAX LISP programs that use the DECwindows Toolkit or Common LISP X (CLX).

Intended Audience

This manual is intended for programmers with a good knowledge of VAX LISP and the DECwindows Toolkit or the X Window System.TM

Structure

The DECwindows Toolkit and CLX are discussed in separate parts of this manual, as follows:

- Part I explains how to use the DECwindows Toolkit with VAX LISP.
- Part II provides brief descriptions of the DECwindows Toolkit functions.
- Part III is a guide to programming CLX.
- Part IV contains reference material on all functions, macros, data types, and structures in CLX.

If you are programming strictly with the DECwindows Toolkit and not calling any CLX functions, you should read Parts I and II only. Similarly, if you want to program in CLX only, you need to read Parts III and IV only.

Appendix A lists the constants associated with the DECwindows Toolkit which are defined in VAX LISP.

Appendix B contains two tables that show the mapping among X Protocol requests, Xlib routines, and CLX routines.

Associated Documents

You will need the following documents to do DECwindows Toolkit programming in VAX LISP/VMS:

- *VMS DECwindows User Interface Language Reference Manual*
- *VMS DECwindows User's Guide*
- *XUI Style Guide*

TM The X Window System is a trademark of the Massachusetts Institute of Technology.

You will need the following documents to do CLX programming in VAX LISP/VMS:

- *VMS DECwindows Guide to Xlib Programming: MIT C Binding*
- *VMS DECwindows Xlib Routines Reference Manual*
- *X Window System: C Library and Protocol Reference*¹

The following documents may also be helpful:

- *VAX LISP/VMS Program Development Guide*
- *VAX LISP/VMS System Access Programming Guide*
- *Common LISP: The Language*

Conventions

The following conventions are used in this manual:

Convention	Meaning
UPPERCASE	DCL commands and qualifiers and VMS file names are printed in uppercase characters; however, you can enter them in uppercase, lowercase, or a combination of uppercase and lowercase characters. For example: The examples directory (SYS\$SYSROOT:[VAXLISP.EXAMPLES] by default) contains sample LISP source files.
UPPERCASE TYPEWRITER	Defined LISP functions, macros, variables, constants, and other symbol names are printed in uppercase TYPEWRITER characters; however, you can enter them in uppercase, lowercase, or a combination of uppercase and lowercase characters. For example: The CALL-OUT macro calls a defined external routine. . . .
lowercase typewriter	LISP forms are printed in the text in lowercase typewriter characters; however, you can enter them in uppercase, lowercase, or a combination of uppercase and lowercase characters. For example: <code>(setf example-1 (make-space))</code>
SANS SERIF	Format specifications of LISP functions and macros are printed in a sans serif typeface. For example: <code>CALL-OUT <i>external-routine</i> &REST <i>routine-arguments</i></code>
<i>italics</i>	Lowercase <i>italics</i> in format specifications and in text indicate arguments that you supply; however, you can enter them in lowercase, uppercase, or a combination of lowercase and uppercase characters. For example: The <i>routine-arguments</i> must be compatible with the arguments defined in the call to the DEFINE-EXTERNAL-ROUTINE macro.
()	Parentheses used in examples of LISP code and in format specifications indicate the beginning and end of a LISP form. For example: <code>(setq name lisp)</code>

¹ Robert W. Scheifler, James Gettys, and Ron Newman, *X Window System: C Library and Protocol Reference*, Digital Press (1988), Burlington, Massachusetts.

Convention	Meaning
[]	<p>Square brackets in format specifications enclose optional elements. For example:</p> <p><i>[doc-string]</i></p> <p>Square brackets do not indicate optional elements when they are used in the syntax of a directory name in a VMS file specification. Here, the square bracket characters must be included in the syntax. For example:</p> <p>(pathname "MIAMI::DBA1:[SMITH]LOGIN.COM;4")</p>
{ }	<p>In function and macro format specifications, braces enclose elements that are considered one unit of code. For example:</p> <p>{<i>keyword value</i>}</p>
{ }*	<p>In function and macro format specifications, braces followed by an asterisk enclose elements that are considered one unit of code, which can be repeated zero or more times. For example:</p> <p>{<i>keyword value</i>}*</p>
&OPTIONAL	<p>In function and macro format specifications, the word &OPTIONAL indicates that the arguments that follow it are optional. For example:</p> <p>PPRINT <i>object</i> &OPTIONAL <i>stream</i></p> <p>Do not specify &OPTIONAL when you invoke a function or macro whose definition includes &OPTIONAL.</p>
&REST	<p>In function and macro format specifications, the word &REST indicates that an indefinite number of arguments may appear. For example:</p> <p>CALL-OUT <i>external-routine</i> &REST <i>routine-arguments</i></p> <p>Do not specify &REST when you invoke a function or macro whose definition includes &REST.</p>
&KEY	<p>In function and macro format specifications, the word &KEY indicates that keyword arguments are accepted. For example:</p> <pre>COMPILE-FILE <i>input-pathname</i> &KEY :LISTING :MACHINE-CODE :OPTIMIZE :OUTPUT-FILE :VERBOSE :WARNINGS</pre> <p>Do not specify &KEY when you invoke a function or macro whose definition includes &KEY.</p>
...	<p>A horizontal ellipsis in a format specification means that the element preceding the ellipsis can be repeated. For example:</p> <p><i>function-name</i> ...</p> <p>A vertical ellipsis in a code example indicates that all the information that the system would display in response to the function call is not shown; or that all the information a user is to enter is not shown.</p>

Convention	Meaning
[Return]	A word inside a box indicates that you press a key on the keyboard. For example:
	[Return] or [Tab]
	In code examples, carriage returns are implied at the end of each line. However, [Return] is used in some examples to emphasize carriage returns.
[Ctrl/x]	Two key names enclosed in a box indicate a control key sequence in which you hold down Ctrl while you press another key. For example:
	[Ctrl/C] or [Ctrl/S]
[PF1] [x]	A sequence such as [PF1] [x] indicates that you must first press and release the key labeled PF1, then press and release another key.
mouse	The term <i>mouse</i> refers to any pointing device, such as a mouse, a puck, or a stylus.
MB1, MB2, MB3	By default, MB1 indicates the left mouse button, MB2 indicates the middle mouse button, and MB3 indicates the right mouse button. You can rebind the mouse buttons.
Red print	In interactive examples, user input is shown in red. For example:
	Lisp> (cdr '(a b c)) (B C) Lisp>

Part I
Guide to the DECwindows ToolKit

196
of the same 300 individuals

Chapter 1

Overview of the DECwindows Toolkit

The DECwindows Toolkit is a set of application development tools and run-time functions that help you implement DECwindows applications. A DECwindows application has the "look and feel" described in the XUI Style Guide. It is characterized by a direct-manipulation interface rather than a command-line interface. For example, the user selects options not by issuing commands but by pointing with a mouse at an object such as a menu choice.

The DECwindows Toolkit is layered on top of Xlib, the routines library that implements the X Windows System, Version 11, protocol. The routines in Xlib enable application programs to communicate with a server to perform the following tasks:

- Open the connection to a workstation display
- Create windows on the display
- Perform output operations to the windows
- Notify the application of pointer or keyboard input through the windows

The DECwindows Toolkit does not hide Xlib from applications. You can call Xlib routines directly via CLX (described in Parts III and IV of this book). For example, an application can call Xlib routines directly to perform drawing operations in a window.

Programming with the DECwindows Toolkit lets you separate form from function. That is, you can consider the form your application takes—its user interface—apart from the functions the application performs. The user interface is a collection of widgets. Widgets are objects like menus, windows, dialog boxes, and so on, through which the user interacts with the application. The application functions are defined as attributes of the widgets. They are called by the widgets under predefined circumstances, or reasons.

Because form and function are separate, programming with the DECwindows Toolkit differs somewhat from traditional programming techniques. The application cannot be thought of as a set of procedures to be performed in a predefined order. Rather, the application is a collection of widgets and a collection of functions that can be invoked by those widgets. The behavior of the application is determined by the widgets that the user selects.

Before you begin to program with the DECwindows Toolkit in LISP, you should be familiar with the following DECwindows documents:

- *XUI Style Guide*
- *VMS DECwindows User Interface Language Reference Manual*
- *VMS DECwindows Toolkit Routines Reference Manual*

This chapter provides an overview of the components of the DECwindows Toolkit. It describes the types of widgets that can be used by a DECwindows application. It describes how the widgets make up a user interface and how they call back to your application functions.

1.1 DECwindows Toolkit Components

The DECwindows Toolkit consists of the following components:

- Low-level functions, which let you access all the attributes associated with a widget. You assign values to widget attributes in an argument list that is passed to the function as an alien structure. A low-level function accepts this argument list as one of its standard formal parameters.
- High-level functions, which are easier to program than low-level functions but let you access only some of the widget attributes. Instead of using an argument list to specify widget attributes, you pass attribute values as formal parameters to the high-level function.
- Alternatively, you can define widgets with the User Interface Language (UIL) and the DECwindows Resource Manager (DRM).
 - The UIL lets you define widgets in a text file called a UIL specification file. You compile this specification file with the UIL compiler. However, instead of creating an object module, as other compilers do, the UIL compiler creates a user-interface database (UID). Your application accesses this database at run time with the functions of the DRM.
 - The DRM is a database manager designed to operate on UID files. DRM functions can open the UID, retrieve widget specifications, create widgets, and build the user interface at run time. DRM functions optimize the initialization and startup of a DECwindows application.

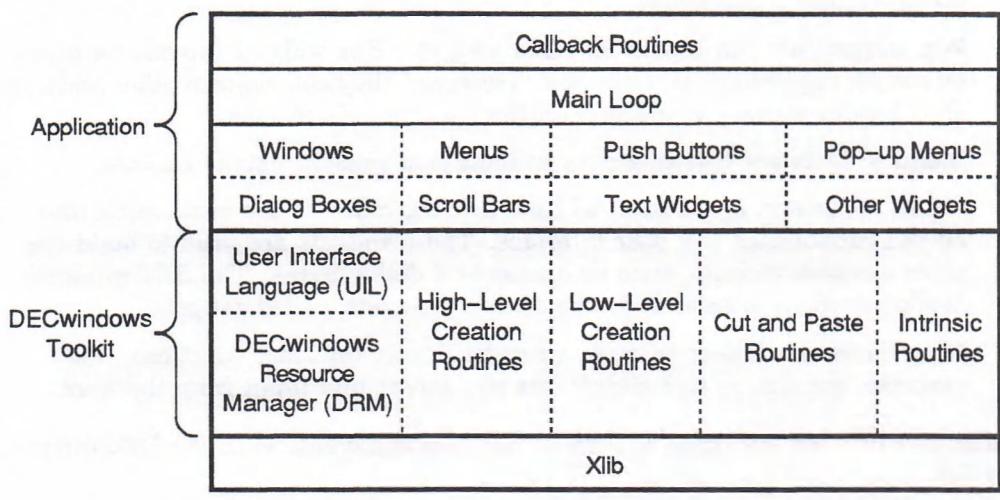
Because the UIL and DRM let you separate widget definitions from the operations performed on those widgets, they can make it easier to translate an application into other languages.

When you define widgets in a UIL specification file, you can access all widget attributes. The UIL compiler checks that the values you assign to attributes are of the data type expected by the widget. The high-level and low-level functions do not check the data types of attribute values.

When deciding how to define your application's widgets, you must choose the method that lets you access the widget attributes you need to set. Although you can access all attributes after the widget has been created, it is more efficient to assign the widget's attribute values when you create it.

- Intrinsic functions are the backbone of every DECwindows application. They provide general application support services. For example, a main loop intrinsic function controls the process of determining which widgets have been selected and which actions to take when they have been selected. After you create widgets, intrinsic functions make them appear on the display. These functions also provide the facilities you need to build your own widgets.
- Cut-and-paste functions let your application exchange data with other DECwindows applications through the clipboard.

Figure 1–1: DECwindows Layered Architecture



MLO-003384

Figure 1–1 illustrates the components of the DECwindows Toolkit and their relationship to both the application and Xlib.

VAX LISP has its own interface to the Toolkit because it supports special LISP objects that are not VAX standard data types. Therefore, when calling a Toolkit function, VAX LISP must convert data from LISP types to VAX types. When the Toolkit calls back to LISP, VAX LISP must convert data from VAX types to LISP types.

The VAX LISP interface to the Toolkit provides the following source files in LISP\$EXAMPLES to help you pass data between LISP and the Toolkit functions:

- DWTAPPL.LSP defines Toolkit constants and some alien structures.
- DECW-CALLBACK-STRUCTURES.LSP defines callback structures.
- DECW-PUBLIC.LSP defines additional structures and many functions to construct and dereference those structures.

The LISP functions are defined in the DWT+ package. All references to the DWT+ package in this manual refer to the functions defined in LISP\$EXAMPLES.

1.2 User Interface Widgets

A DECwindows user interface is made up of widgets. A widget can be a menu, a push button, a scroll bar, and so on. Some widgets display information, such as text or graphics. Others, like menu bars and dialog boxes, simply contain other widgets. Still others change their display in response to user input and can invoke functions.

Each widget in a DECwindows user interface has a set of attributes, such as width, height, font, color, and border. The DECwindows Toolkit assigns default values for these attributes. The default attributes conform to the recommendations of the *XUI Style Guide*. However, you can customize the widget's appearance and behavior by specifying the desired attributes.

The DECwindows Toolkit widgets fall into five general categories:

- Window widgets enable you to open windows. They include a title bar and other window components.
- Box widgets are containers for other widgets. Box widgets provide no input or output capabilities of their own. However, they can contain other widgets that display messages and prompt the user for information.
- Menu widgets are containers for widgets that present lists of choices.
- Label, separator, and button widgets provide many of the basic input and output capabilities of a user interface. These widgets are used to build the more complex widgets, such as menus and dialog boxes. The DECwindows Toolkit provides a second version of these widgets, called gadgets.
- Miscellaneous widgets perform specialized user interface functions. For example, the simple text widget lets you accept text input from the user.

Table 1-1 lists all the types of widgets that you can create with the DECwindows Toolkit.

Table 1-1: Types of DECwindows Widgets

Type	Purpose
Window Widgets	
Command window	A command entry area in a main window widget.
Main window	A window with a title bar, and, optionally, a menu bar, command window, and scroll bars.
Box Widgets	
Attached dialog box	A dialog box that changes the position of its subwidgets when resized.
Caution box	A dialog box that presents a warning message before an irreversible action is executed.
Dialog box	A container for other widgets. It has no input capabilities apart from those of the subwidgets it contains.
File selection box	A dialog box that queries the user for a file specification.
Help box	A widget that presents help text.
Message box	A dialog box that displays a message.
Pop-up attached dialog box	An attached dialog box that cannot extend beyond the edges of its parent.
Pop-up dialog box	A dialog box that cannot extend beyond the edges of its parent.
Selection box	A dialog box that presents the user with a list of choices, only one of which can be selected at a time.
Work-in-progress box	A dialog box that displays a "Work in Progress" message.

(continued on next page)

Table 1–1 (Cont.): Types of DECwindows Widgets

Type	Purpose
Menu Widgets	
Menu bar	A list of choices, each of which is a pull-down menu entry.
Option menu	A list of choices, from which one may be chosen. It always displays the current choice.
Pop-up menu	A menu that appears when the user presses MB2. It can extend beyond the borders of its parent.
Pull-down menu	A menu that appears when the user presses MB1 or MB2. It can extend beyond the borders of its parents.
Pull-down menu entry	A button-like widget that causes a pull-down menu to appear.
Radio box	A work area menu in which a list of choices is presented, only one of which can be selected at a time.
Work area menu	A container for menu items.
Label, Separator, and Button Widgets	
Label	Read-only, constant text.
Push button	A label that invokes a callback routine when selected.
Separator	A line or bar that separates items in a list or menu.
Toggle button	A label that controls an on and off state.
Miscellaneous Widgets	
List box	A list of choices, typically a long list. It contains a scroll bar for viewing all list items.
Scale	A pointer that moves within a range of values. The user selects a value by sliding the pointer to the desired position within the range.
Scroll bar	A bar with a sliding scale at the side of a window. It shows the user's current position within a body of data that is larger than the window. The user can reposition the sliding scale to display a different portion of the data.
Scroll window	A window with a built-in scroll bar.
Simple text	A window for reading and writing text.
Window	An area on the screen in which to perform graphics operations.

Some widgets have variants, called gadgets. A gadget has the same appearance as its widget counterpart but uses fewer resources. Gadgets do not support all the attributes supported by their widget counterparts. Thus, gadgets offer improved performance but are not as customizable.

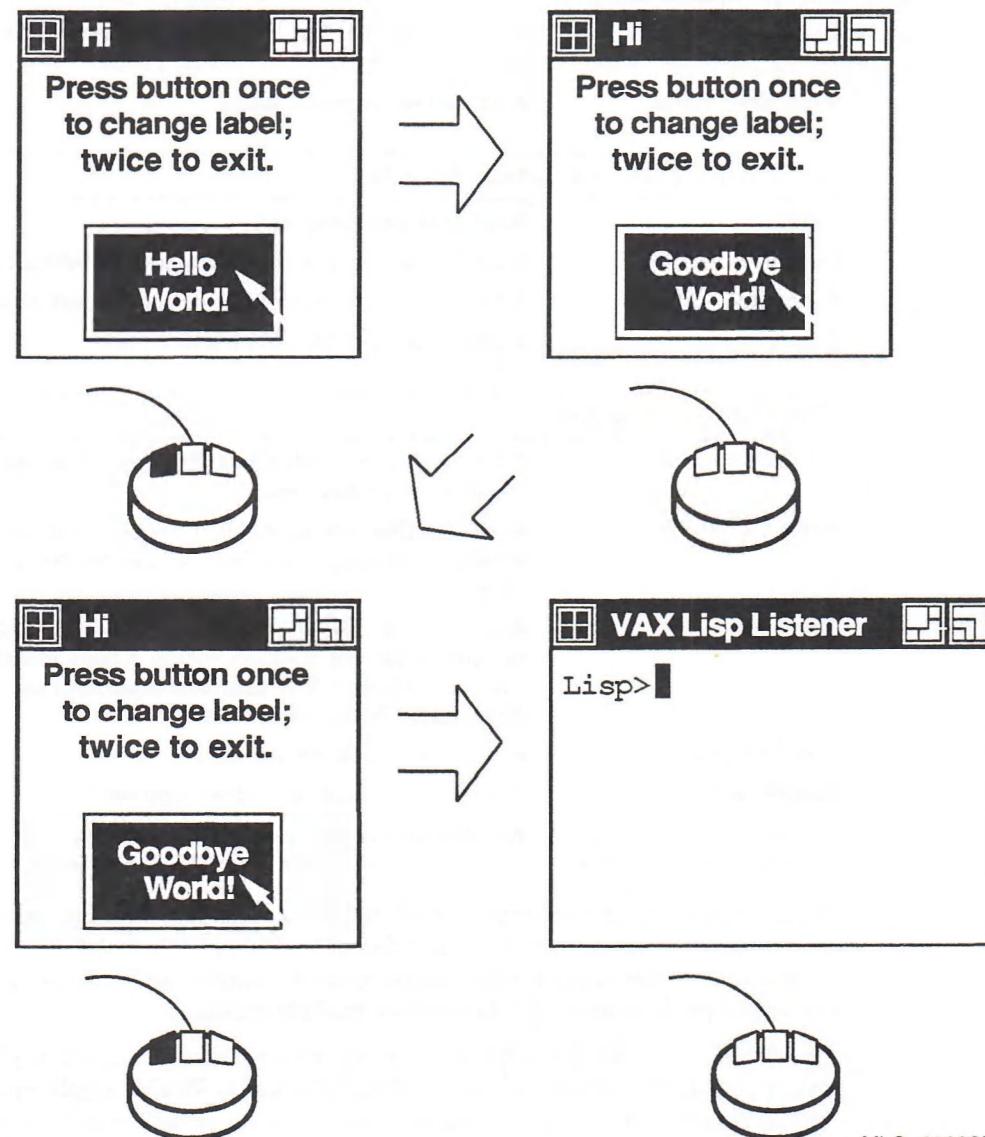
The distribution kit for VAX LISP contains some examples of applications written with the DECwindows Toolkit. The Hello World! application is a simple DECwindows application of four widgets: a main window, a dialog box, a push button, and a label containing the text "Hello World!" When you click MB1 on the push button once, it changes the label to "Goodbye World!" When you click a second time, the application exits. You can run Hello World! to see an example of these common types of widgets and to become familiar with the look and feel of a DECwindows application.

The source files for the Hello World! application can be installed by the LISP distribution kit in the directory defined by the logical name LISP\$EXAMPLES. Copy the files from LISP\$EXAMPLES to your working directory, then execute the command procedure as follows:

```
$ COPY LISP$EXAMPLES:HELLOWORLD.* *.*  
$ @HELLOWORLD
```

Figure 1-2 shows the user interface to Hello World! as it appears initially and as it changes when you interact with the application by clicking MB1.

Figure 1-2: Hello World! Application



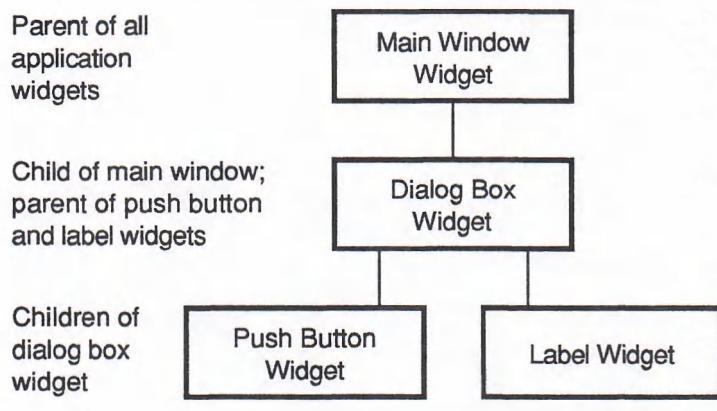
1.3 Creating a User Interface

You create a user interface by establishing parent/child relationships among the widgets. The resulting family tree is called a widget hierarchy. Widgets that can only be children of other widgets are called primitive widgets. Widgets that can be either parents or children are called composite widgets. Labels, separators, and buttons are primitive widgets; all others are composite widgets.

The user interface of the Hello World! application has a simple widget hierarchy made up of four widgets. At the top of the hierarchy is the application main window widget. The main window acts as the mediator between the application program and the workstation environment. In all DECwindows applications, a main window widget is at the top of the hierarchy. In the Hello World! application, the dialog box widget is the only child of the main window widget. The dialog box is the parent of both a push button widget and a label widget.

The main window and dialog box widgets are examples of composite widgets; they contain other widgets. The push button and label widgets are examples of primitive widgets; they do not have children. Figure 1-3 shows the hierarchy formed by the user interface of the Hello World! application.

Figure 1-3: Hello World! Widget Hierarchy



MLO-003386

1.4 Calling Back to Application Functions

The user interface interacts with your application functions through the VAX LISP callback facility. Callback functions are defined as attributes of widgets. These attributes define what functions to call under specific circumstances, or reasons. Each widget supports one or more reasons. The following are some common reasons:

- The widget has been created.
- The user has activated the widget.
- The user has requested help.
- The user has changed the value of the widget attribute.

The reasons each widget supports are detailed in the *VMS DECwindows Toolkit Routines Reference Manual*.

You can specify different functions to call for different reasons. For example, a widget may define one callback function for the "help" reason and another for the "activated" reason. You can also associate more than one callback function with a single reason. When you do, the functions are executed in the order you specify.

In the Hello World! application, only the push button is associated with a callback function. This function is invoked when the widget is activated (the "activate" reason). The callback function checks to see how many times the push button has been activated. The first time, the callback function changes the text of the label widget. The second time, the function exits from the application.

Creating a User Interface

A DECwindows application is a collection of user interface widgets and the functions they invoke. This chapter describes how to create a DECwindows user interface; Chapter 3 describes how to access that user interface from LISP.

There are three ways to define the widgets that make up an application's user interface.

- Low-level functions let you access all the attributes associated with a widget but present a more complex programming task.
- High-level functions are easier to program but let you access only some of the widget attributes.
- The User Interface Language (UIL) lets you define widgets in a text file, which you then compile with the UIL compiler to create a user-interface database (UID). Your program can access the UID at run time using DRM functions. When you define widgets in this way, you can access all widget attributes.

The UIL compiler checks that the values you assign to widget attributes are of the data type expected by the widget; the high-level and low-level functions do not.

The UIL is perhaps the simplest method to use because it is a high-level language that provides many default attributes to make the widgets conform to the DECwindows style. Furthermore, the user interface specification is separate from the actions that the application performs. This makes the UIL a good choice for applications that will be translated into other languages. High-level and low-level functions are better for creating widgets or changing their attributes at run time.

2.1 A Sample DECwindows Application

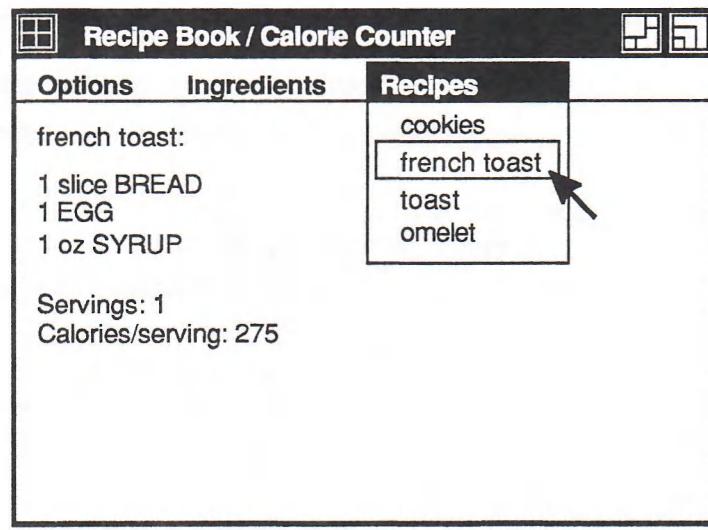
The VAX LISP/VMS Program Development Guide shows a LISP application that calculates the number of calories per serving for a few recipes. This chapter explains how to write a DECwindows interface for the recipe application.

The source files for the recipe application are included in the LISP\$EXAMPLES directory, created when VAX LISP is installed on your system. To run the application, copy the files to your working directory and invoke the command file, as follows:

```
$ COPY LISP$EXAMPLES:RECIPE.* *.*  
$ @RECIPE
```

The recipe application displays a main window, a menu bar, and a simple text widget. If you press MB1 on the Ingredients pull-down menu entry, a menu appears. You can choose from a list of ingredients. When you choose an ingredient, the application displays the number of calories per some unit of measure. If you press MB1 on the Recipes pull-down menu entry, a menu of recipes appears. When you choose a recipe, the application displays the ingredients, the number of servings, and the number of calories per serving for that recipe, based on the number of calories in each ingredient. Figure 2-1 shows what the application displays if you choose french toast from the Recipe menu.

Figure 2-1: Choosing from a Pull-Down Menu



MLO-003387

NOTE

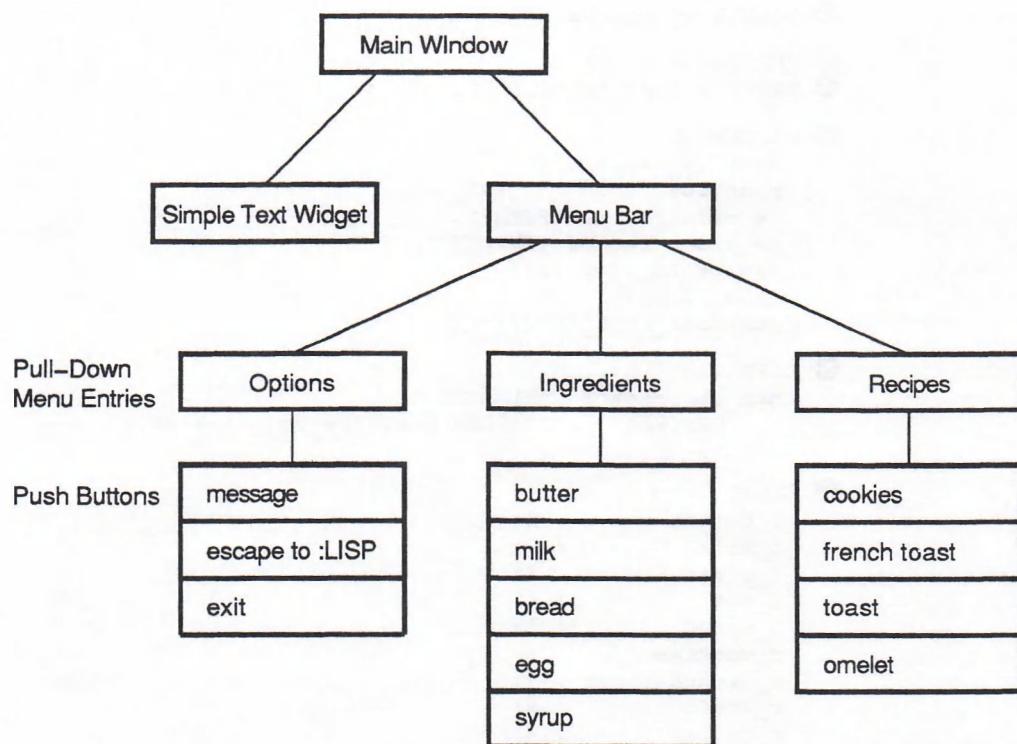
In order to choose french toast or toast from the Recipes menu, you must enter the following form at the `Lisp>` prompt or add it to the `RECIPE.LSP` source file:

```
(setf (get 'bread 'calories) 100)
```

See Chapter 5 of your *Program Development Guide* for more information about the recipe application.

The recipe application has one main window. This is the top-level widget. All other widgets are children of the top-level widget or its children. A simple text widget and a menu bar are the immediate children of the main window. The menu bar is parent to the pull-down menu entries, which are each parents to one pull-down menu. Each pull-down menu is parent to a series of push buttons, and associated with each push button is a callback function. When the user clicks on a push button, it invokes the callback function associated with that push button. Figure 2-2 shows the hierarchical relationship among the widgets that make up the application.

Figure 2–2: A Widget Hierarchy



MLO-003388

2.2 The User Interface Language

The UIL lets you describe widgets with UIL statements stored in one or more source files. UIL statements can define any widgets required by the application and their position in a widget hierarchy. The UIL is described in detail in *VMS DECwindows User Interface Language Reference Manual*.

The widget hierarchy for the recipe application is defined in the file named RECIPE.UIL. Example 2–1 shows the portion of the source file that sets up the user interface module. That is, it gives the module a name, defines some of its characteristics, and defines the procedure names and constants that will be referred to throughout the module. All UIL keywords are shown in uppercase letters. Note that the names of procedures and widgets cannot contain hyphens; you may use underscores instead.

Example 2–1: Defining the UIL Module

```
❶ MODULE my_module
❷ VERSION = 'v1.0'
❸ NAMES = CASE_INSENSITIVE

❹ PROCEDURE
    display_message ();
    practice_callback (INTEGER);
    dw_calories (INTEGER);
    dw_print_recipe (INTEGER);
    escape_to_lisp ();
    exit_lisp ();
    remember_text_id ();

❺ LIST
    nop_callback : CALLBACKS {
        DESTROY = PROCEDURE practice_callback (1);
    };

❻ VALUE
    i_butter      : 1;
    i_milk        : 2;
    i_bread       : 3;
    i_egg         : 4;
    i_syrup       : 5;
    r_cookies    : 6;
    r_french_toast : 7;
    r_toast       : 8;
    r_omelet      : 9;
```

- ❶ The MODULE statement allows you to divide the user interface definitions into separate areas called modules. A UIL module consists of a single module block, which contains any number of procedure, list, value, identifier, and object sections. Modules let you share UIL code between applications. That is, you can place commonly used definitions in their own modules, which many applications can access.
- ❷ You can assign a version number to the application's user interface with the VERSION statement.
- ❸ The NAMES statement determines whether the UIL module is case-sensitive. In this example, the UIL is case-insensitive, which means that it will treat uppercase and lowercase letters the same.
- ❹ The PROCEDURE section declares the callback procedures referenced by the widgets. You must supply the names of the callback procedures and the data types of their arguments, if any.
- ❺ Like a macro expansion in other programming languages, the LIST section lets you give a name to an argument, controls, or callback list to which you can refer anywhere in the UIL. This example defines a callback list that defines the callback function to call when an object is destroyed. This callback list can be referred to by the name NOP_CALLBACK.
- ❻ The VALUE section defines symbolic constants. The symbolic constants defined in this example are used as arguments to the callback functions.

Example 2–2 shows the definition of the application’s main window.

Example 2–2: Defining the Main Window

```
OBJECT app_window : PRIVATE MAIN_WINDOW {
    ①   ARGUMENTS {
        X = 10;
        Y = 20;
        HEIGHT = 300;
        WIDTH = 300;
    };
    ②   CALLBACKS nop_callback;
    ③   CONTROLS {
        MENU_BAR main_menubar;
        SIMPLE_TEXT text_object;
    };
}
```

The main window is named APP_WINDOW. It is defined with the PRIVATE keyword, indicating that the window cannot be referenced from outside this module. APP_WINDOW has the following attributes:

- ① The ARGUMENTS section defines the X and Y coordinates and the height and width of the window in pixels.
- ② The CALLBACKS section refers to the callback list defined in Example 2–1. It calls the PRACTICE_CALLBACK function when the main window widget is destroyed.
- ③ The CONTROLS section defines the names and types of widgets that are the children of the main window.

Example 2–3 shows the definition of a simple text widget that is a child of the main window.

Example 2–3: Defining a Simple Text Widget

```
OBJECT text_object : PRIVATE SIMPLE_TEXT {
    ①   ARGUMENTS {
        Y = 40;
        HEIGHT = 220;
        WIDTH = 300;
    };
    ②   CALLBACKS {
        CREATE = PROCEDURE remember_text_id;
    };
    ③   CONTROLS {
    };
}
```

The TEXT_OBJECT has the following attributes:

- ① The ARGUMENTS section defines the window’s position and size.
- ② The CALLBACKS section states that the function REMEMBER_TEXT_ID should be called when this widget is created.
- ③ The CONTROLS section does not contain any widget names. Therefore, this widget does not have children. (This section could be omitted because it is empty.)

Example 2-4 shows the definition of the main menu bar (the horizontal bar at the top of the main window showing the names of the pull-down menus).

Example 2-4: Defining a Menu Bar

```
OBJECT main_menu_bar : PRIVATE MENU_BAR {
①   ARGUMENTS {
      ORIENTATION = HORIZONTAL;
    };
②   CONTROLS {
      PULLDOWN_ENTRY option_menu_entry;
      PULLDOWN_ENTRY ingredients_menu_entry;
      PULLDOWN_ENTRY recipes_menu_entry;
    };
}
```

The MAIN_MENUBAR has the following attributes:

- ① The ARGUMENTS section sets the ORIENTATION attribute to HORIZONTAL.
- ② The CONTROLS keyword lists the names and types of the menu bar's children, which are all pull-down menu entries.

Example 2-5 shows the definition of a pull-down menu. It defines the label that appears in the menu bar, the pull-down menu, and the push buttons that are the pull-down menu entries.

Example 2-5: Defining Pull-Down Menu Entries

```
OBJECT option_menu_entry : PRIVATE PULLDOWN_ENTRY {
①   ARGUMENTS {
      LABEL_LABEL = 'Options';
    };
②   CONTROLS {
      PULLDOWN_MENU options_menu;
    };
③ OBJECT options_menu : PRIVATE PULLDOWN_MENU {
      CALLBACKS nop_callback;
      CONTROLS {
        PUSH_BUTTON message_button;
        PUSH_BUTTON escape_button;
        PUSH_BUTTON exit_button;
      };
    };
④ OBJECT message_button : PRIVATE PUSH_BUTTON {
      ARGUMENTS {
        LABEL_LABEL = 'Message';
      };
      CALLBACKS {
        ACTIVATE = PROCEDURE display_message;
      };
}
```

(continued on next page)

Example 2-5 (Cont.): Defining Pull-Down Menu Entries

```
⑤ OBJECT escape_button : PRIVATE PUSH_BUTTON {
    ARGUMENTS {
        LABEL_LABEL = 'Escape to Lisp';
    };
    CALLBACKS {
        ACTIVATE = PROCEDURE escape_to_lisp;
    };
};

⑥ OBJECT exit_button : PRIVATE PUSH_BUTTON {
    ARGUMENTS {
        LABEL_LABEL = 'Exit';
    };
    CALLBACKS {
        ACTIVATE = PROCEDURE exit_lisp;
    };
};
```

- ① The LABEL_LABEL keyword defines the text of this menu entry, which is 'Options'. (Note that character strings are enclosed in single quotes.)
- ② The OPTION_MENU_ENTRY has one pull-down menu as its child.
- ③ The OPTIONS_MENU pull-down menu has two children. They are push buttons.
- ④ The first push button, called MESSAGE_BUTTON, has the label 'Message'. It calls the LISP procedure DISPLAY_MESSAGE when activated. ACTIVATE is one of the reasons with which a push button can call back to a LISP function.
- ⑤ The ESCAPE_BUTTON has the label 'Escape to Lisp'. It defines ESCAPE_TO_LISP as the function to call when the button is activated.
- ⑥ The EXIT_BUTTON push button has the label 'Exit'. It defines EXIT_LISP as the LISP function to call when the button is activated.

The rest of the widget hierarchy consists of two more pull-down menus, which are constructed in the same way as the Options menu in Example 2-5. Refer to the RECIPE.LSP source file to see how these menus are defined. An END_MODULE statement marks the end of the UIL module.

The UIL file contains only source code. It must be compiled into a UID file before LISP can access the user interface at run time. To compile the UIL file for the recipe application, issue the following UIL command:

```
$ UIL RECIPE
```

In this command line, the UIL compiler assumes that the file has the default file type (.UIL). It creates a file with the same name and the default file type (.UID).

2.3 High-Level Functions

When you use high-level functions to define widgets, their attributes are specified as formal parameters. The widgets you define with high-level functions are not a separate part of the application, as is the UID file, but are part of the LISP source code for the application.

The example in this section uses high-level functions to define a portion of the widget hierarchy for the recipe application. In this example, each widget is represented by a LISP variable. Thus, the widgets can easily be passed as arguments to Toolkit functions at run time. Initially, the value of the variables is set to NIL, as shown in Example 2-6.

Example 2-6: Defining Widget Variables

```
(defvar *top-level* nil)
(defvar *app-window* nil)
(defvar *text-widget* nil)
(defvar *main-menu-bar* nil)
(defvar *option-menu-entry* nil)
(defvar *options-menu* nil)
(defvar *message-button* nil)
(defvar *exit-button* nil)
```

When the widget is created, its variable is set to the widget's ID using SETF. The high-level functions are called within a LISP function named MAKE-USER-INTERFACE, shown in Example 2-7.

Example 2-7: Defining Widgets with High-Level Functions

```
(defun make-user-interface ()
  (setf *app-window*
    ①   (dwt:main-window *top-level* "APP_WINDOW" 10 20 300 300))
    (setf *text-object*
      ②   (dwt:s-text *app-window* "TEXT_OBJECT" 0 24 40 22 ""))
    (setf *main-menu-bar*
      ③   (dwt:menu-bar *app-window* "MAIN_MENU_BAR" nil nil))
    (setf *option-menu-entry*
      ④   (dwt:pull-down-menu-entry
            *main-menu-bar* "OPTION_MENU_ENTRY" 0 0
            (dwt:latin1-string "Options")
            (setf *options-menu*
              ⑤   (dwt:menu
                    *main-menu-bar*
                    "OPTIONS_MENU"
                    0 0
                    DwtMenuPulldown
                    DwtOrientationVertical
                    nil nil nil)
                  nil nil))
    (setf *exit-button*
      ⑥   (dwt:push-button
            *options-menu* "MESSAGE_BUTTON" 0 0
            (dwt:latin1-string "Message")
            (dwt:+:make-decw-callback-point 'display-message)
            nil))
    (setf *message-button*
      (dwt:push-button
        *options-menu* "EXIT_BUTTON" 0 0
        (dwt:latin1-string "Exit")
        (dwt:+:make-decw-callback-point 'exit-lisp)
        nil))))
```

- ① DWT:MAIN-WINDOW creates the application's main window. All widgets are children of the main window or its children. When calling DWT:MAIN-WINDOW, you supply the name of the main window's parent widget, the name of the main window, its X and Y coordinates, and its length and width in pixels. The

parent of the main window is created when the Toolkit is initialized by the application at run time. Initialization is described in Section 3.1.1.

- ② DWT:S-TEXT creates the simple text widget, which is a child of the main window. Formal parameters name the parent widget, the window's name, its position and size, and any initial text to be displayed in the window.
- ③ DWT:MENU-BAR creates the main menu bar. This menu bar is the child of the application main window, and its name is MAIN_MENUBAR. The last two parameters allow you to name two callback functions: an application function and a help function. Because these parameters are specified as NIL, this menu bar does not call back to either type of function.
- ④ A pull-down menu entry is created with the DWT:PULL-DOWN-MENU-ENTRY function. It defines the attributes of the labels that appear in the menu bar. In this example, there is only one menu entry, the Options entry. Its parent is the main menu bar. Its label, "Options", appears at offset 0,0 within the menu bar. Its name is OPTION_MENU_ENTRY and it has one child—the Options menu.

The text of the labels is converted to compound strings by the DWT:LATIN1-STRING function. All text that is displayed in a window must be converted to compound strings. A compound string describes not only the text string to be displayed, but also the character set to be used and the writing direction.

- ⑤ DWT:MENU creates the Options menu. This menu is defined as follows:
 - Its parent is the main menu bar.
 - Its name is OPTIONS_MENU.
 - It is at offset 0,0 from the upper left corner of the menu entry.
 - It is a pull-down menu, defined with the symbolic constant DwtMenuPulldown, defined in LISP\$EXAMPLES:DWTAPPL.LSP.
 - It is a vertical pull-down menu, defined with the symbolic constant DwtOrientationVertical, defined in LISP\$EXAMPLES:DWTAPPL.LSP.
 - There are three possible callback functions for this menu—a map callback, an application callback, and a help callback. All three are defined as NIL.
- ⑥ DWT:PUSH-BUTTON defines the Message and Exit push buttons. Like the other calls to high-level functions, the calls to DWT:PUSH-BUTTON define the parent widget, the widget name, the location, and the label. Unlike the other widgets, these push buttons define callback functions, using the DWT+:MAKE-DECW-CALLBACK-POINT function in LISP\$EXAMPLES:DECW-PUBLIC.LSP. DWT+:MAKE-DECW-CALLBACK-POINT creates a structure with the address of the callback function.

2.4 Low-Level Functions

Like high-level functions, low-level Toolkit functions are called directly from LISP. However, you supply widget attributes in an argument list, not as formal parameters. You create an argument list with the DWT+:CREATE-ARGLIST function, defined in the LISP\$EXAMPLES:DECW-PUBLIC.LSP source file.

You define the arguments as a list of triples of the form

(attribute-name value data-type ...).

Attribute-name is a string; *value* is an arbitrary type; *data-type* is a keyword indicating the C type to use. Symbolic constants for the attribute names known to standard Toolkit widgets are defined in LISP\$EXAMPLES:DWTAPPL.LSP.

The DWT+:CREATE-ARGLIST function returns two values: the argument list and the length of the list. You can call DWT+:CREATE-ARGLIST within the context of the MULTIPLE-VALUE-BIND function to assign the return values to named variables.

For example, the recipe application displays a message box when the user activates the Message push button. This message box is created at run time by the DWT:MESSAGE-BOX-CREATE low-level function, as shown in Example 2-8.

Example 2-8: Creating a Message Box

```
1 (defun display-message (ap)
2   (dwt:manage-child
3    (multiple-value-bind (arglist argcnt)
4      (dwt+:create-arglist
5       (list DwtNLabel "This is a message." :compound-string
6             DwtNokLabel "Acknowledged" :compound-string)))
7      (dwt:message-box-create *app-window* "MY_MESSAGE"
8                               arglist argcnt))))
```

- ① The DISPLAY-MESSAGE function takes an argument pointer as its argument, the value passed to this callback function from DECwindows. See Chapter 3 for information on the argument pointer.
- ② DWT+:CREATE-ARGLIST is passed a list of triples defining the DwtNlabel and the DwtNokLabel attributes of the message box. DwtNlabel is set to the string "This is a message." and DwtNokLabel is set to the string "Acknowledged". Each one is a compound string.
- ③ MULTIPLE-VALUE-BIND assigns the two return values to the variables ARGLIST and ARGCOUNT.
- ④ The message box is created by calling DWT:MESSAGE-BOX-CREATE, which takes the parent widget, the widget name, the argument list, and the argument count as its parameters.
- ⑤ The message box is not displayed until it is managed. This is done with DWT:MANAGE-CHILD, described in Chapter 3.

When the application calls back to DISPLAY-MESSAGE, a message box like the one shown in Figure 2-3 is displayed.

Figure 2–3: A Message Box





Chapter 3

Building a DECwindows Application

Every DECwindows application has a similar structure, no matter what function the application performs. That is, a main loop determines which widget the user has selected. As a result of being selected, a widget may invoke a callback function to perform some function. When the callback function has completed, the user interface returns control to the main loop.

Chapter 2 describes how to define the user interface. This chapter describes how to access that user interface from a DECwindows application and how to write the callback functions that the user interface invokes.

3.1 Accessing the User Interface

To access the user interface, the application must perform the following steps:

1. Initialize DECwindows
2. Create the application's widgets
3. Realize the top-level widget
4. Invoke the application's main loop

3.1.1 Initialization

Initialization performs several essential startup functions, including establishing the connection between the client application and the server. It also creates a widget that is the parent of the application's main window. Simple initialization is performed by the DWT:INITIALIZE function. More flexible but more complicated initialization can be done using the functions DWT:TOOLKIT-INITIALIZE, DWT:CREATE-APPLICATION-CONTEXT, DWT:OPEN-DISPLAY, and DWT:APP-CREATE-SHELL.

If the user interface for the recipe example were defined by high-level functions, you would initialize DECwindows as follows:

```
(setf *top-level*
      (dwt:initialize "Recipe Book / Calorie Counter"
                     "Lispclass" 0 0 0 0))
```

When DWT:INITIALIZE returns, it sets the variable called *TOP-LEVEL* to the top-level widget for the application. Your application should call DWT:INITIALIZE only once.

If you define the user interface with the UIL, as in the source files supplied in LISP\$EXAMPLES, you must also initialize the UID and prepare it for use by DECwindows. In this case, you have to call the Toolkit functions in the following order:

1. The DWT:INITIALIZE-DRM function prepares the application to use the DRM functions.
2. The DWT:INITIALIZE function initializes the DECwindows Toolkit.
3. The DRM hierarchy is the set of UID files containing the widget definitions. The DWT:OPEN-HIERARCHY function opens these UID files.
4. The DWT:REGISTER-DRM-NAMES function registers the names and associated values for access by DRM. The values can be callback functions, pointers to user-defined data, or other values. DRM uses this information to resolve symbolic references in UID files to their run-time values.

Because the recipe application defines its user interface in a UID file, it must call DRM functions in addition to DWT:INITIALIZE, as shown in Example 3-1.

Example 3-1: Initializing a UIL Application

```
(dwt:initialize-drm)
(unless *top-level*
  (setf *top-level*
    (dwt:initialize "Recipe Book / Calorie Counter"
      "Lispclass" 0 0 0 0)))
(unless *drm-hierarchy*
  (setf *drm-hierarchy*
    (dwt:open-hierarchy 1 (dwt+:make-asciz-array *uil-file*) nil))
  (dwt:register-drm-names
    (dwt+:make-register-callback-list *uil-callback-names*
      (uil-callback-points))
    (length *uil-callback-names*)))
```

In Example 3-1, *uil-file* is a variable that has been initialized to the name of the application's UIL file; *uil-callback-names* is a variable that has been initialized to the names of the application's callback functions, as described in Section 3.2.1. UIL-CALLBACK-POINTS is a function that creates a list of callback entry points, as described in Example 3-5.

3.1.2 Creating Widgets

Widgets are created with Toolkit intrinsic or DRM functions. The intrinsic functions create widgets on the basis of definitions made by high-level or low-level functions. The DRM functions create widgets on the basis of definitions in UID files.

In Section 2.3, the MAKE-USER-INTERFACE function defined a portion of the recipe widget hierarchy using high-level functions. These widgets are not created until the MAKE-USER-INTERFACE function is invoked and the main window's children are managed. One of the intrinsic functions, either DWT:MANAGE-CHILD or DWT:MANAGE-CHILDREN, is used to manage children.

DWT:MANAGE-CHILD manages one widget at a time, as in:

```
(dwt:manage-child *text-widget*)
(dwt:manage-child *main-menubar*)
(dwt:manage-child *option-menu-entry*)
(dwt:manage-child *exit-button*)
(dwt:manage-child *message-button*)
```

DWT:MANAGE-CHILDREN manages every widget in a list. You can construct this list with a function called DWT:CREATE-WIDGET-LIST, provided in LISP\$EXAMPLES:DECW-PUBLIC.LSP. This function returns the widget list and the length of the list. You can pass both these return values to DWT:MANAGE-CHILDREN as follows:

```
(multiple-value-bind (widget-list widget-count)
  (dwt+:create-widget-list
    (list *text-widget*
          *main-menubar*
          *option-menu-entry*
          *exit-button*))
  (dwt:manage-children widget-list widget-count))
```

Similarly, the widgets defined in a UID file are not created or managed until you call the DRM function DWT:FETCH-WIDGET at run time. The term "fetch" means to both create the widget and manage its children.

You specify the main window widget of the application and its parent (the top-level widget created by the call to DWT:INITIALIZE) in the call to DWT:FETCH-WIDGET. As a result of this single call, DRM fetches all the widgets in the hierarchy.

Example 3-2: Creating a Widget Hierarchy

```
(setf *app-window*
      (dwt:fetch-widget *drm-hierarchy* "APP_WINDOW" *top-level*))
```

3.1.3 Realizing the Top-Level Widget

The term "realize" means to display the widget hierarchy on the workstation screen. During creation, only the child widgets are created. During realization, the main window widget is managed and the top-level widget is realized.

The steps for managing and realizing a user interface are the same whether the user interface has been created with the UIL, the high-level functions, or the low-level functions:

1. Manage the main window widget. The intrinsic function DWT:MANAGE-CHILD makes the main window a child of the top-level widget returned by the call to DWT:INITIALIZE. The entire widget hierarchy is managed as a result of this call.
2. Realize the top-level widget. The intrinsic function DWT:REALIZE-WIDGET displays the entire widget hierarchy.

Example 3-3 shows how these functions are called to realize the recipe application's user interface.

Example 3-3: Realizing the User Interface

```
(dwt:manage-child *app-window*)
(dwt:realize-widget *top-level*)
```

3.1.4 Calling the Main Loop

The Toolkit provides an intrinsic function to act as the main loop of any DECwindows application. The main loop handles callback events; that is, it determines which widgets are selected and which callback functions to invoke. Example 3-4 shows how the recipe application invokes the DWT:MAIN-LOOP function. This step is the same regardless of how you create the user interface.

Example 3-4: Calling the Main Loop

```
(catch 'original-lisp (dwt:main-loop))
```

In this example, DWT:MAIN-LOOP is called within the context of the CATCH special form. A callback routine can return to LISP by issuing a call to the THROW special form and naming the ORIGINAL-LISP label. Once you return to LISP in this way, you cannot restart this or any other DECwindows application from the same LISP session; you must first exit and restart LISP. To write an application that can be restarted, you must explicitly create an application context and handle the DECwindows display rather than letting DWT:INITIALIZE and DWT:MAIN-LOOP handle these things for you.

3.2 Writing a Callback Function

Although the main loop controls the DECwindows application, it is your LISP callback functions that perform the functions of the application. Writing LISP functions that operate within a DECwindows application requires special considerations:

- Defining the callback functions in LISP

You use the callback facility to let LISP know which functions DECwindows will call back to.

- Accessing the callback structure

DECwindows widgets always call back with an argument list containing three arguments: the widget ID, user-defined data (called a tag), and a callback structure that differs, depending on the type of widget that is calling back. Your callback function must be written in a way that correctly accesses these arguments.

3.2.1 Declaring Callback Functions

Callback functions must be declared in LISP before the application invokes the main loop. These entry points must also be registered with DECwindows. You can use the MAKE-CALL-BACK-ROUTINE function to declare the callback functions for LISP and the DWT:REGISTER-DRM-NAMES function to register them with DECwindows. (See *VAX LISP/VMS System Access Programming Guide* for more information about MAKE-CALL-BACK-ROUTINE.)

To declare the callback points for both LISP and DECwindows, you must first create a list of callback names. The list contains all the callback function names as they are known to DECwindows. They must be in uppercase letters and appear in alphabetical order within the list. Each name is written as a separate string. Example 3-5 shows the list of callback names for the recipe application. The list is stored in the variable called *UIL-CALLBACK-NAMES*.

Example 3-5: Creating a List of Callback Names

```
(defvar *uil-callback-names* '("DISPLAY_MESSAGE" "DW_CALORIES"
                               "DW_PRINT_RECIPE" "ESCAPE_TO_LISP"
                               "EXIT_LISP" "PRACTICE_CALLBACK"
                               "REMEMBER_TEXT_ID"))
```

You then define the callback entry points, using the MAKE-CALL-BACK-ROUTINE function. The names for MAKE-CALL-BACK-ROUTINE are written as they appear in LISP. That is, they can contain lowercase letters and hyphens. These callback points are also placed in a list in the same order as they appear in the list of names; otherwise, the function's external name will not be correctly paired with its LISP name. Example 3-6 shows how the entry points for the recipe application are created by the UIL-CALLBACK-POINTS function:

Example 3-6: Creating a List of Callback Entry Points

```
(defun uil-callback-points ()
  (list (make-call-back-routine
          #'(lambda (ap) (display-message ap)))
        (make-call-back-routine
          #'(lambda (i) (dw-calories i)))
        (make-call-back-routine
          #'(lambda (r) (dw-print-recipe r)))
        (make-call-back-routine
          #'(lambda (ap) (escape-to-lisp ap)))
        (make-call-back-routine
          #'(lambda (i) (declare (ignore i)) (exit)))
        (make-call-back-routine
          #'(lambda (ap) (practice-callback ap)))
        (make-call-back-routine
          #'(lambda (ap) (remember-text-id ap)))))
```

After you create the lists, you create a register list of the callback points with the DWT+:MAKE-REGISTER-LIST function. This function is in the file called LISP\$EXAMPLES:DECW-PUBLIC.LSP, which you can load into your LISP application. For example, you can create the callback register list for the recipe application as follows:

```
(dwt+:make-register-callback-list *uil-callback-names*
                                    (uil-callback-points))
```

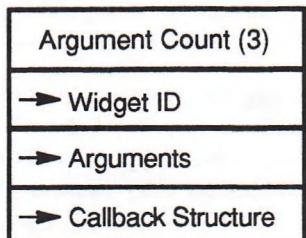
This register list is of the appropriate form to pass to the DWT:REGISTER-DRM-NAMES function, as follows:

```
(dwt:register-drm-names
  (dwt+:make-register-callback-list *uil-callback-names*
                                    (uil-callback-points))
  (length *uil-callback-names*))
```

3.2.2 Accessing the Callback Argument List

When DECwindows calls back to LISP, it always passes a pointer to an argument list containing the widget ID, the user-defined data (tag), and a callback structure. Figure 3-1 shows the form of the argument list. You can either define your callback function to explicitly access the three arguments, or you can access them indirectly through an argument pointer (AP) mechanism. All of the examples in this section are written in terms of the latter approach.

Figure 3-1: DECwindows Callback Argument List



MLO-003390

Your callback function can access any of the items on the argument list, using the DWT+:GET-CALLBACK-WIDGET, DWT+:GET-CALLBACK-TAG, and DWT+:GET-CALLBACK-REASON routines, defined in LISP\$EXAMPLES:DECW-PUBLIC.LSP.

3.2.2.1 Accessing the Widget ID

The DWT+:GET-CALLBACK-WIDGET function is in LISP\$EXAMPLES:DECW-PUBLIC.LSP, a file that you can load into your LISP application. It returns the widget that called back to LISP. In the recipe application, the text widget is defined to call back to a function called REMEMBER-TEXT-ID when it is created, as follows:

```
CALLBACKS {
    CREATE = PROCEDURE remember_text_id;
};
```

REMEMBER-TEXT-ID is a LISP function. It uses DWT+:GET-CALLBACK-WIDGET to return the ID of the text widget. It then stores that ID in the variable called *TEXT-WIDGET*.

Example 3-7 shows how the REMEMBER-TEXT-ID function accesses the callback widget.

Example 3-7: Accessing the Callback Widget

```
(defun remember-text-id (ap)
  (declare (special *text-widget*))
  (format t "The text widget called back~%")
  (setf *text-widget* (dwt+:get-callback-widget ap)))
```

Other LISP functions can then use this widget to read from or write to the text widget. For instance, the PRINT-TO-WINDOW function in Example 3-8 references the *TEXT-WIDGET* variable to write to the window. It first checks to make sure that the variable contains a widget ID. If not, it displays an error message. Otherwise, it calls DWT:S-TEXT-SET-STRING to write to the text widget. The function DWT:S-TEXT-SET-STRING is high level. See Part II for more information about DWT:S-TEXT-SET-STRING.

Example 3-8: Writing to a Text Widget

```
(defun print-to-window (the-string)
  (if (null *text-widget*)
      (error "The text widget never called back with its ID."))
      (dwt:s-text-set-string *text-widget* the-string)))
```

In the recipe application, the DW-PRINT-RECIPE function uses PRINT-TO-WINDOW to pretty-print a recipe. DW-PRINT-RECIPE is shown in Example 3-9.

Example 3-9: Pretty-Printing a Recipe

```
(defun dw-print-recipe (the-recipe)
  "DW-PRINT-RECIPE recipe
Pretty print a recipe, number of servings, and calories/serving,
using the DECwindows Toolkit."
  (let ((recipe (convert-arg the-recipe)))
    (print-lists recipe)
    (print-to-window
      (concatenate 'string
        (format nil "~A:" (recipe-name recipe))
        (princ-to-string #\linefeed)
        (princ-to-string #\linefeed)
        (format nil "~A *part-of-recipe*")
        (princ-to-string #\linefeed)
        (format nil "Servings: ~D" (recipe-servings recipe))
        (princ-to-string #\linefeed)
        (format nil "Calories/serving: ~D"
          (dw-calories the-recipe)))))

  (defun print-lists (recipe)
    (setf *part-of-recipe* nil)
    (mapc #'(lambda (a u i)
              (setf *part-of-recipe*
                (concatenate 'string
                  *part-of-recipe*
                  (format nil "~A ~D ~A" (rationalize a) u i)
                  (princ-to-string #\linefeed))))
      (recipe-amount recipe))
    (mapcar
      #'(lambda (i) (get i 'units))
      (recipe-ingredients recipe))
    (recipe-ingredients recipe)))
```

3.2.2.2 Accessing the User-Defined Data (Tag)

The DWT+:GET-CALLBACK-TAG function, supplied in LISP\$EXAMPLES:DECW-PUBLIC.LSP, lets you access the user-defined data with which the callback function was invoked. For example, the COOKIES_BUTTON is defined to call back to DW-PRINT-RECIPE as follows:

```

CALLBACKS {
    ACTIVATE = PROCEDURE dw_print_recipe (r_cookies);
}

```

The R_COOKIES argument, in turn, is defined in the VALUE section of the UIL as follows:

```
r_cookies : 6;
```

In order for LISP to determine what value has been passed to the DW-PRINT-RECIPE function, it must call DWT+:GET-CALLBACK-TAG. Example 3-10 shows how the recipe application defines the CONVERT-ARG function, which uses DWT+:GET-CALLBACK-TAG to locate the arguments passed back by DECwindows and translate them into the LISP representation of those arguments. A CASE statement converts the argument to the appropriate LISP representation—either a symbol or a structure. In the cookies example, DECwindows passes back the number 6 to LISP and CONVERT-ARG translates it into the structure COOKIES.

Example 3-10: Accessing Callback Arguments

```
(defun convert-arg (the-recipe)
  (let ((tag (dwt+:get-callback-tag the-recipe)))
    (declare (fixnum tag))
    (case tag
      (1 'butter)
      (2 'milk)
      (3 'bread)
      (4 'egg)
      (5 'syrup)
      (6 cookies)
      (7 french-toast)
      (8 toast)
      (9 omelet))))
```

The DW-CALORIES function in Example 3-11 uses CONVERT-ARG to determine whether DECwindows has passed back an ingredient or a recipe.

Example 3-11: Printing Calorie Information

```
(defun dw-calories (the-recipe)
  "DW-CALORIES the-recipe
Calculate the number of calories for an ingredient/unit-of-measure
or for a recipe/serving, using the DECwindows Toolkit."
① (let ((x (convert-arg the-recipe)))
  (cond ((symbolp x)
        ②   (print-to-window
              (format nil "~A: ~D calories/~A"
                     x (get x 'calories) (get x 'units))))
        ((recipe-p x)
         (numerator
          (floor
           (reduce #'+
                  (mapcar #'*
                          (mapcar #'(lambda (i) (get i 'calories))
                                  (recipe-ingredients x))
                          (recipe-amount x)))
          (recipe-servings x)))))))
```

- ① DW-CALORIES uses CONVERT-ARG to determine what item was selected—either an ingredient or a recipe.

- ② PRINT-TO-WINDOW displays the number of calories and the unit of measure if the argument is an ingredient (represented as a symbol).

3.2.2.3 Accessing the Callback Structure

The DWT+:GET-CALLBACK-STRUCTURE-ADDRESS function, supplied in LISP\$EXAMPLES:DECW-PUBLIC.LSP, returns the address of the callback structure with which the callback function was invoked. You can use this address as the :DATA argument to the constructor function for any of the alien structures defined in LISP\$EXAMPLES:DECW-CALLBACK-STRUCTURES.LSP.

For example, you could define a function to get a DwtScaleCallbackStruct structure from an argument pointer. (This type of callback structure is passed by Scale widgets.)

```
(defun get-callback-DwtScaleCallbackStruct (ap)
  (make-DwtScaleCallbackStruct
    :data (dwt+:get-callback-structure-address ap)))
```

You could then use the alien structure accessor functions to get the value now indicated by the scale widget, as in MY-SCALE-WIDGET-CALLBACK, shown below.

```
(defun my-scale-widget-callback (ap)
  (setf *my-scale-value*
    (DwtScaleCallbackStruct-value
      (get-callback-DwtScaleCallbackStruct ap))))
```

LISP\$EXAMPLES:DECW-PUBLIC.LSP also includes two functions for getting the values of the reason and the event fields that are present in any callback structure. Both DWT+:GET-CALLBACK-REASON and DWT+:GET-CALLBACK-EVENT take an argument pointer as their single argument.

DWT+:GET-CALLBACK-REASON returns a callback reason as an integer. Symbolic constants for the standard DECwindows Toolkit reasons are defined in LISP\$EXAMPLES:DWTAPPL.LSP. For example, if the callback is due to widget activation (such as pushing a push button), the reason value is 10, which is the value of the constant DwtCRAActivate.

DWT+:GET-CALLBACK-EVENT returns an X event structure. You can pass this structure as the :EVENT argument to the CLX:PROCESS-EVENT routine in order to use the CLX event-processing mechanisms to access the fields of this structure. See Chapter 12 of this manual for more information about CLX event processing or Part IV for more specific information about the CLX:PROCESS-EVENT function.

the same time, the following morning, I had a long talk with Mr. G. about the whole question of the proposed new building.

He was very friendly and I think he is really interested in the project.

He said that he would like to have a meeting with me at his office next week to discuss the details of the proposed new building.

I told him that I would be happy to meet with him at his office next week to discuss the details of the proposed new building.

I also told him that I would be happy to meet with him at his office next week to discuss the details of the proposed new building.

I also told him that I would be happy to meet with him at his office next week to discuss the details of the proposed new building.

I also told him that I would be happy to meet with him at his office next week to discuss the details of the proposed new building.

I also told him that I would be happy to meet with him at his office next week to discuss the details of the proposed new building.

I also told him that I would be happy to meet with him at his office next week to discuss the details of the proposed new building.

I also told him that I would be happy to meet with him at his office next week to discuss the details of the proposed new building.

I also told him that I would be happy to meet with him at his office next week to discuss the details of the proposed new building.

Part II

Reference to the DECwindows ToolKit

All DECwindows Toolkit routines can be called from VAX LISP. They are in the LISP package named DWT:. In most cases, the number of arguments and return values is the same in VAX LISP as in the MIT C and VAX versions of the routine. In some cases, the LISP routine differs slightly.

VAX LISP requires special LISP versions of the Toolkit routines because of the way LISP represents data structures. For example, in the MIT C version of these routines, objects such as widgets and argument lists are represented as C typedefs (user-defined data types). In VAX LISP, they must be represented as alien structures.

The Toolkit manages so many kinds of data structures that it is impractical for VAX LISP to provide alien structure definitions for all Toolkit objects. However, VAX LISP does provide the following source files in LISP\$EXAMPLES. When your application must access a specific type of structure, you can include the alien structure definition from LISP\$EXAMPLES.

- DWTAPPL.LSP defines Toolkit constants and some alien structures.
- DECW-CALLBACK-STRUCTURES.LSP defines callback structures.
- DECW-PUBLIC.LSP defines more structures and many functions to construct and dereference those structures.

Therefore, to call a DECwindows routine, you need the following things:

- VMS DECwindows Toolkit Routines Reference Manual
- The routine description in Part II of this book
- DWTAPPL.LSP, DECW-CALLBACK-STRUCTURES.LSP, and DECW-PUBLIC.LSP, found in LISP\$EXAMPLES

Determine how to call the routine as follows:

1. Look up the complete routine description in the *VMS DECwindows Toolkit Routines Reference Manual*.
2. Locate the description of the routine in Part II of this book. It describes the argument list and return values for each Toolkit routine that you can call from VAX LISP. Note the names of any alien structures used as either arguments or return values. An argument or return value is either a Common LISP type such as INTEGER, a CLX type such as CLX:WINDOW, a DWT type such as DWT:WIDGET, or an MIT C structure such as XtInputId.
3. If the object is a Common LISP, CLX, or DWT type, the data type is already defined by VAX LISP. You need do nothing more.

If the object's MIT C name is given, find its alien structure or constructor definition in LISP\$EXAMPLES and copy it into your LISP source file. If the MIT C name is not there, you will need to build the structure yourself. Look in DECW\$INCLUDE:*.H for the C definition of these objects.

1932.

Geological Survey of India

The Geological Survey of India is a Governmental department which
is engaged in the study of the geological structure of the country.

The Survey has been established in 1851 by Sir Charles Lyell, a
well-known English geologist, who was appointed as the first Director of
the Survey.

The Survey has been engaged in the study of the geological structure
of the country since its establishment. It has made many important
discoveries, such as the discovery of coal deposits in Bihar, Jharkhand,
and West Bengal, and the discovery of oil fields in Assam and
Bihar.

The Survey has also been engaged in the study of the geological
structure of the country, and has made many important
discoveries, such as the discovery of coal deposits in Bihar, Jharkhand,
and West Bengal, and the discovery of oil fields in Assam and
Bihar.

The Survey has also been engaged in the study of the geological
structure of the country, and has made many important
discoveries, such as the discovery of coal deposits in Bihar, Jharkhand,
and West Bengal, and the discovery of oil fields in Assam and
Bihar.

The Survey has also been engaged in the study of the geological
structure of the country, and has made many important
discoveries, such as the discovery of coal deposits in Bihar, Jharkhand,
and West Bengal, and the discovery of oil fields in Assam and
Bihar.

The Survey has also been engaged in the study of the geological
structure of the country, and has made many important
discoveries, such as the discovery of coal deposits in Bihar, Jharkhand,
and West Bengal, and the discovery of oil fields in Assam and
Bihar.

The Survey has also been engaged in the study of the geological
structure of the country, and has made many important
discoveries, such as the discovery of coal deposits in Bihar, Jharkhand,
and West Bengal, and the discovery of oil fields in Assam and
Bihar.

The Survey has also been engaged in the study of the geological
structure of the country, and has made many important
discoveries, such as the discovery of coal deposits in Bihar, Jharkhand,
and West Bengal, and the discovery of oil fields in Assam and
Bihar.

DWT:ADD-ACTIONS Intrinsic Function

Declares an action table and registers it with the translation manager.

Format

DWT:ADD-ACTIONS *action num-actions*

<i>action</i>	XtActionList
<i>num-actions</i>	INTEGER

Return Value

Unspecified.

DWT:ADD-CALLBACK Intrinsic Function

Adds a callback procedure to a callback list.

Format

DWT:ADD-CALLBACK *widget callback-name callback client-data*

<i>widget</i>	DWT:WIDGET
<i>callback-name</i>	STRING
<i>callback</i>	XtCallbackProc
<i>client-data</i>	INTEGER or alien structure

Return Value

Unspecified.

DWT:ADD-CALLBACKS Intrinsic Function

Adds a list of callback procedures to a callback list.

Format

DWT:ADD-CALLBACKS *widget callback-name newcallbacks*

<i>widget</i>	DWT:WIDGET
<i>callback-name</i>	STRING
<i>newcallbacks</i>	XtCallbackList

Return Value

Unspecified.

DWT:ADD-CONVERTER Intrinsic Function

DWT:ADD-CONVERTER Intrinsic Function

Registers a new resource converter.

Format

DWT:ADD-CONVERTER *from-type to-type converter convert-arglist argcount*

<i>from-type</i>	STRING
<i>to-type</i>	STRING
<i>converter</i>	XtConverter
<i>convert-arglist</i>	XtConvertArgList
<i>argcount</i>	INTEGER

Return Value

Unspecified.

DWT:ADD-EVENT-HANDLER Intrinsic Function

Registers an event handler procedure with the dispatch mechanism.

Format

DWT:ADD-EVENT-HANDLER *widget event-mask other client-proc client-data*

<i>widget</i>	DWT:WIDGET
<i>event-mask</i>	XtEventMask
<i>other</i>	INTEGER
<i>client-proc</i>	XtEventHandler
<i>client-data</i>	INTEGER or alien structure

Return Value

Unspecified.

DWT:ADD-EXPOSURE-TO-REGION Intrinsic Function

Merges Expose and Graphics Expose events into a region.

Format

DWT:ADD-EXPOSURE-TO-REGION *event region*

<i>event</i>	DWT:EVENT
<i>region</i>	INTEGER

Return Value

Unspecified.

DWT:ADD-FONT-LIST Compound String Function

DWT:ADD-FONT-LIST Compound String Function

Adds an entry to a font list.

Format

DWT:ADD-FONT-LIST *font-list font charset*

<i>font-list</i>	DwtFontList
<i>font</i>	INTEGER
<i>charset</i>	INTEGER

Return Value

A DwtFontList alien structure representing the new font list.

DWT:ADD-GRAB Intrinsic Function

Redirects user input to a modal widget.

Format

DWT:ADD-GRAB *widget exclusive spring-loaded*

<i>widget</i>	DWT:WIDGET
<i>exclusive</i>	INTEGER
<i>spring-loaded</i>	INTEGER

Return Value

Unspecified.

DWT:ADD-INPUT Intrinsic Function

Registers a new file for input.

Format

DWT:ADD-INPUT *source condition client-proc client-data*

<i>source</i>	INTEGER
<i>condition</i>	INTEGER or alien structure
<i>client-proc</i>	XtInputCallbackProc
<i>client-data</i>	INTEGER or alien structure

Return Value

An XtInputId alien structure representing the input source identifier.

DWT:ADD-RAW-EVENT-HANDLER Intrinsic Function

DWT:ADD-RAW-EVENT-HANDLER Intrinsic Function

Registers an event handler procedure with the dispatch mechanism without causing the server to select that event.

Format

DWT:ADD-RAW-EVENT-HANDLER *widget event-mask other client-proc client-data*

<i>widget</i>	DWT:WIDGET
<i>event-mask</i>	DWT:EVENT
<i>other</i>	INTEGER
<i>client-proc</i>	XtEventHandler
<i>client-data</i>	INTEGER or alien structure

Return Value

Unspecified.

DWT:ADD-TIME-OUT Intrinsic Function

Creates a timeout.

Format

DWT:ADD-TIME-OUT *interval client-proc client-data*

<i>interval</i>	INTEGER
<i>client-proc</i>	XtTimerCallbackProc
<i>client-data</i>	INTEGER or alien structure

Return Value

An XtIntervalId alien structure representing the interval identifier.

DWT:ADD-WORK-PROC Intrinsic Function

Registers a work procedure in the default application context.

Format

DWT:ADD-WORK-PROC *work-proc client-data*

<i>work-proc</i>	XtWorkProc
<i>client-data</i>	INTEGER or alien structure

Return Value

An XtWorkProcId alien structure representing the work procedure identifier.

DWT:APP-ADD-INPUT Intrinsic Function

Registers a new file for input.

Format

DWT:APP-ADD-INPUT *context source condition client-proc client-data*

<i>context</i>	XtApplicationContext
<i>source</i>	INTEGER
<i>condition</i>	INTEGER or alien structure
<i>client-proc</i>	XtInputCallbackProc
<i>client-data</i>	INTEGER or alien structure

Return Value

An XtInputId alien structure representing the input file identifier.

DWT:APP-ADD-TIME-OUT Intrinsic Function

Creates a timeout value.

Format

DWT:APP-ADD-TIME-OUT *context interval client-proc client-data*

<i>context</i>	XtApplicationContext
<i>interval</i>	INTEGER
<i>client-proc</i>	XtTimerCallbackProc
<i>client-data</i>	INTEGER or alien structure

Return Value

An XtIntervalId alien structure representing the timeout value identifier.

DWT:APP-ADD-WORK-PROC Intrinsic Function

Registers a work procedure.

Format

DWT:APP-ADD-WORK-PROC *context client-proc client-data*

<i>context</i>	XtApplicationContext
<i>client-proc</i>	XtWorkProc
<i>client-data</i>	INTEGER or alien structure

Return Value

An XtWorkProcId alien structure representing the work procedure identifier.

DWT:APP-CREATE-SHELL Intrinsic Function

DWT:APP-CREATE-SHELL Intrinsic Function

Creates a new top-level application widget that is the root of the widget hierarchy.

Format

DWT:APP-CREATE-SHELL *name class widget-class display arglist argcoun*t

<i>name</i>	STRING
<i>class</i>	STRING
<i>widget-class</i>	INTEGER
<i>display</i>	CLX:DISPLAY
<i>arglist</i>	ArgList
<i>argcoun</i> t	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:APP-MAIN-LOOP Intrinsic Function

Processes input.

Format

DWT:APP-MAIN-LOOP *context*

<i>context</i>	INTEGER
----------------	---------

Return Value

Unspecified.

DWT:APP-NEXT-EVENT Intrinsic Function

Returns the value from the head of an application's input queue, or waits for an event if none are queued.

Format

DWT:APP-NEXT-EVENT *context &OPTIONAL event*

<i>context</i>	INTEGER
<i>event</i>	DWT:EVENT

Return Value

An object of type DWT:EVENT representing the next event in the queue.

DWT:APP-PEEK-EVENT Intrinsic Function

DWT:APP-PEEK-EVENT Intrinsic Function

Returns the value from the head of a given application's input queue without removing input from the queue.

Format

DWT:APP-PEEK-EVENT *context* &**OPTIONAL** *event*

<i>context</i>	INTEGER
<i>event</i>	DWT:EVENT

Return Value

Unspecified.

DWT:APP-PENDING Intrinsic Function

Determines if there are any events on the input queue for a given application.

Format

DWT:APP-PENDING *context*

<i>context</i>	INTEGER
----------------	---------

Return Value

An XtInputMask integer bit mask representing the OR of XtIMXEvent, XtIMTimer, and XtIMAlternateInput. A nonzero value means there are events pending from the X server, timer, or other input sources.

DWT:APP-PROCESS-EVENT Intrinsic Function

Provides direct control of the processing of different types of input.

Format

DWT:APP-PROCESS-EVENT *context* *input-mask*

<i>context</i>	XtAppContext
<i>input-mask</i>	XtInputMask

Return Value

Unspecified.

DWT:ATTACHED-DB High-Level Function

DWT:ATTACHED-DB High-Level Function

Creates an attached dialog box widget.

Format

DWT:ATTACHED-DB *parent-widget name default-position x y title style map-callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>default-position</i>	INTEGER
<i>x y</i>	INTEGER
<i>title</i>	COMPOUND-STRING
<i>style</i>	INTEGER
<i>map-callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:ATTACHED-DB-CREATE Low-Level Function

Creates an attached dialog box widget.

Format

DWT:ATTACHED-DB-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:ATTACHED-DB-POPUP-CREATE Low-Level Function

Creates an attached pop-up dialog box widget.

Format

DWT:ATTACHED-DB-POPUP-CREATE *parent-widget name override-arglist override-argcount*

DWT:ATTACHED-DB-POPUP-CREATE Low-Level Function

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcoun</i> t	INTEGER

Return Value

An object of type DWT: IDGET representing the new widget.

DWT:AUGMENT-TRANSLATIONS Intrinsic Function

Merges new translations into an existing translation table.

Format

DWT:AUGMENT-TRANSLATIONS *widget translation-table*

<i>widget</i>	DWT:WIDGET
<i>translation-table</i>	XtTranslations

Return Value

Unspecified.

DWT:BEGIN-COPY-TO-CLIPBOARD Cut-and-Paste Function

Sets up storage and data structures to receive clipboard data.

Format

DWT:BEGIN-COPY-TO-CLIPBOARD *display window clip-label widget callback*
&OPTIONAL *item-id*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>clip-label</i>	COMPOUND-STRING
<i>widget</i>	DWT:WIDGET
<i>callback</i>	VoidProc
<i>item-id</i>	INTEGER

Return Values

Two values:

- An integer representing the *item-id*.
- One of two possible constants whose integer values represent the clipboard status:

DWT:BEGIN-COPY-TO-CLIPBOARD Cut-and-Paste Function

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:BUILD-EVENT-MASK Intrinsic Function

Retrieves the event mask for a widget.

Format

DWT:BUILD-EVENT-MASK *widget*

widget DWT:WIDGET

Return Value

An EventMask alien structure representing the event mask.

DWT:CALL-ACCEPT-FOCUS Intrinsic Function

Calls a widget's accept input focus procedure.

Format

DWT:CALL-ACCEPT-FOCUS *widget time*

widget DWT:WIDGET
time INTEGER

Return Value

The accept focus procedure (XtAcceptFocusProc) return, which is false if accept focus is nil.

DWT:CALL-CALLBACKS Intrinsic Function

Executes the procedures in a callback list.

Format

DWT:CALL-CALLBACKS *widget callback-name calldata*

widget DWT:WIDGET
callback-name STRING
calldata INTEGER or alien structure

Return Value

Unspecified.

DWT:CALLBACK-EXCLUSIVE Intrinsic Function

DWT:CALLBACK-EXCLUSIVE Intrinsic Function

Maps a pop-up shell from a given widget's callback list.

Format

DWT:CALLBACK-EXCLUSIVE *widget client-data callback-data*

<i>widget</i>	DWT:WIDGET
<i>client-data</i>	XtClientData
<i>callback-data</i>	INTEGER or alien structure

Return Value

Unspecified.

DWT:CALLBACK-NONE Intrinsic Function

Maps a pop-up shell from a given widget's callback list.

Format

DWT:CALLBACK-NONE *widget client-data callback-data*

<i>widget</i>	DWT:WIDGET
<i>client-data</i>	XtClientData
<i>callback-data</i>	XtClientData

Return Value

Unspecified.

DWT:CALLBACK-NONEXCLUSIVE Convenience Function

Maps a pop-up shell from a given widget's callback list.

Format

DWT:CALLBACK-NONEXCLUSIVE *widget client-data callback-data*

<i>widget</i>	DWT:WIDGET
<i>client-data</i>	XtClientData
<i>callback-data</i>	INTEGER or alien structure

Return Value

Unspecified.

DWT:CALLBACK-POPDOWN Intrinsic Function

DWT:CALLBACK-POPDOWN Intrinsic Function

Brings down a widget that was popped up with one of the callback routines DWT:CALLBACK-NONE, DWT:CALLBACK-NONEXCLUSIVE, or DWT:CALLBACK-EXCLUSIVE.

Format

DWT:CALLBACK-POPDOWN *widget client-data callback-data*

<i>widget</i>	DWT:WIDGET
<i>client-data</i>	INTEGER or alien structure
<i>callback-data</i>	INTEGER or alien structure

Return Value

Unspecified.

DWT:CALLOC Intrinsic Function

Allocates and initializes an array.

Format

DWT:CALLOC *num-elements element-size*

<i>num-elements</i>	INTEGER
<i>element-size</i>	INTEGER

Return Value

An integer representing the storage address.

DWT:CANCEL-COPY-FORMAT Cut-and-Paste Function

Indicates that the application will no longer supply a data item to the clipboard that the application had previously passed by name.

Format

DWT:CANCEL-COPY-FORMAT *display window data-id*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>data-id</i>	INTEGER

DWT:CANCEL-COPY-FORMAT Cut-and-Paste Function

Return Value

One of two possible constants whose integer values represent the clipboard status:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:CANCEL-COPY-TO-CLIPBOARD Cut-and-Paste Function

Cancels the copy to clipboard that is in progress.

Format

DWT:CANCEL-COPY-TO-CLIPBOARD *display window item-id*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>item-id</i>	INTEGER

Return Value

Unspecified.

DWT:CAUTION-BOX High-Level Function

Creates a caution box widget.

Format

DWT:CAUTION-BOX *parent-widget name default-position x y format
label yes-label no-label cancel-label default-push-button
callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>default-position</i>	INTEGER
<i>x y</i>	INTEGER
<i>format</i>	INTEGER
<i>label</i>	COMPOUND-STRING
<i>yes-label</i>	COMPOUND-STRING
<i>no-label</i>	COMPOUND-STRING
<i>cancel-label</i>	COMPOUND-STRING
<i>default-push-button</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:CAUTION-BOX-CREATE Low-Level Function

DWT:CAUTION-BOX-CREATE Low-Level Function

Creates a caution box widget.

Format

DWT:CAUTION-BOX-CREATE *parent-widget* *name* *override-arglist* *override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:CHILDREN Convenience Function

Returns a list of the widget's children.

Format

DWT:CHILDREN *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

A list of the widget's children.

DWT:CLASS Intrinsic Function

Identifies the widget's class structure.

Format

DWT:CLASS *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

A WidgetClass alien structure representing a pointer to the widget's class structure.

DWT:CLIPBOARD-LOCK Cut-and-Paste Function

DWT:CLIPBOARD-LOCK Cut-and-Paste Function

Locks the clipboard from access by other applications.

Format

DWT:CLIPBOARD-LOCK *display window*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW

Return Value

One of two possible constants whose integer values represent the clipboard status:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:CLIPBOARD-UNLOCK Cut-and-Paste Function

Unlocks the clipboard, enabling it to be accessed by other applications.

Format

DWT:CLIPBOARD-UNLOCK *display window remove-locks*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>remove-locks</i>	INTEGER

Return Value

One of two possible constants whose integer values represent the clipboard status:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:CLOSE-DISPLAY Intrinsic Function

Closes a display and removes it from an application context.

Format

DWT:CLOSE-DISPLAY *display*

<i>display</i>	CLX:DISPLAY
----------------	-------------

Return Value

Unspecified.

DWT:CLOSE-HIERARCHY DRM Function

DWT:CLOSE-HIERARCHY DRM Function

Closes a DRM hierarchy.

Format

DWT:CLOSE-HIERARCHY *hierarchy-id*

hierarchy-id INTEGER

Return Value

One of two possible constants whose integer values represent the DRM status:

DWT:DRM-SUCCESS	1
DWT:DRM-FAILURE	0

DWT:COMMAND-APPEND High-Level Function

Appends the passed string to the current command line and executes it if required.

Format

DWT:COMMAND-APPEND *widget command*

widget DWT:WIDGET
command STRING

Return Value

Unspecified.

DWT:COMMAND-ERROR-MESSAGE High-Level Function

Writes an error message in the command window and refreshes the command line.

Format

DWT:COMMAND-ERROR-MESSAGE *widget error*

widget DWT:WIDGET
error STRING

Return Value

Unspecified.

DWT:COMMAND-SET High-Level Function

DWT:COMMAND-SET High-Level Function

Replaces the current command string with the one passed and executes it if required.

Format

DWT:COMMAND-SET *widget command*

<i>widget</i>	DWT:WIDGET
<i>command</i>	STRING

Return Value

Unspecified.

DWT:COMMAND-WINDOW High-Level Function

Creates a command window widget.

Format

DWT:COMMAND-WINDOW *parent-widget name prompt lines callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>prompt</i>	COMPOUND-STRING
<i>lines</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:COMMAND-WINDOW-CREATE Low-Level Function

Creates a command window widget.

Format

DWT:COMMAND-WINDOW-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

DWT:COMMAND-WINDOW-CREATE Low-Level Function

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:CONFIGURE-WIDGET Intrinsic Function

Moves and resizes the sibling widget of the child widget making the geometry request.

Format

DWT:CONFIGURE-WIDGET *widget x y width height borderwidth*

<i>widget</i>	DWT:WIDGET
<i>x y</i>	INTEGER
<i>width</i>	INTEGER
<i>height</i>	INTEGER
<i>borderwidth</i>	INTEGER

Return Value

Unspecified.

DWT:CONVERT Intrinsic Function

Invokes resource conversions.

Format

DWT:CONVERT *widget src-type src-value dst-type dst-value*

<i>widget</i>	DWT:WIDGET
<i>src-type</i>	STRING
<i>src-value</i>	XrmValuePtr
<i>dst-type</i>	STRING
<i>dst-value</i>	XrmValuePtr

Return Value

Unspecified.

DWT:CONVERT-CASE Intrinsic Function

Determines the uppercase and lowercase equivalents for a key symbol.

Format

DWT:CONVERT-CASE *display keysym lower-return upper-return*

<i>display</i>	CLX:DISPLAY
----------------	-------------

DWT:CONVERT-CASE Intrinsic Function

<i>keysym</i>	KeySym
<i>lower-return</i>	KeySym
<i>upper-return</i>	KeySym

Return Value

Unspecified.

DWT:COPY-FROM-CLIPBOARD Cut-and-Paste Function

Retrieves a data item from the clipboard.

Format

DWT:COPY-FROM-CLIPBOARD *display window format-name buffer length*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>format-name</i>	STRING
<i>buffer</i>	STRING
<i>length</i>	INTEGER

Return Values

Four values:

- A string representing the *buffer*.
- An integer representing *num-bytes*.
- An integer representing the *private-id*.
- One of four possible constants whose integer values represent the clipboard status:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-TRUNCATE	2
DWT:CLIPBOARD-LOCKED	4
DWT:CLIPBOARD-NON-DATA	5

DWT:COPY-TO-CLIPBOARD Cut-and-Paste Function

Copies a data item to the clipboard.

Format

DWT:COPY-TO-CLIPBOARD *display window item-id format-name buffer
length private-id
&OPTIONAL data-id*

DWT:COPY-TO-CLIPBOARD Cut-and-Paste Function

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>item-id</i>	INTEGER
<i>format-name</i>	STRING
<i>buffer</i>	STRING
<i>length</i>	INTEGER
<i>private-id</i>	INTEGER
<i>data-id</i>	INTEGER

Return Values

Two values:

- An integer representing the *data-id*.
- One of two possible constants whose integers represent the status as follows:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:CREATE-APPLICATION-CONTEXT Intrinsic Function

Creates an application context.

Format

DWT:CREATE-APPLICATION-CONTEXT

Return Value

XtApplicationContext, an integer representing the application context.

DWT:CREATE-APPLICATION-SHELL Intrinsic Function

Creates a top-level widget that is the root of another widget hierarchy.

Format

DWT:CREATE-APPLICATION-SHELL *name widget-class arglist argcnt*

<i>name</i>	STRING
<i>widget-class</i>	INTEGER
<i>arglist</i>	ArgList
<i>argcnt</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:CREATE-FONT-LIST Compound String Function

DWT:CREATE-FONT-LIST Compound String Function

Creates a new font list.

Format

DWT:CREATE-FONT-LIST *font charset*

<i>font</i>	CLX:FONT
<i>charset</i>	INTEGER

Return Value

DwtFontList, an integer representing the new font list.

DWT:CREATE-MANAGED-WIDGET Intrinsic Function

Creates and manages a widget.

Format

DWT:CREATE-MANAGED-WIDGET *name widget-class parent-widget arglist argcount*

<i>name</i>	STRING
<i>widget-class</i>	INTEGER
<i>parent-widget</i>	DWT:WIDGET
<i>arglist</i>	ArgList
<i>argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:CREATE-POPUP-SHELL Intrinsic Function

Creates a pop-up shell.

Format

DWT:CREATE-POPUP-SHELL *name widget-class parent-widget arglist argcount*

<i>name</i>	STRING
<i>widget-class</i>	INTEGER
<i>parent-widget</i>	DWT:WIDGET
<i>arglist</i>	ArgList
<i>argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:CREATE-WIDGET Intrinsic Function

DWT:CREATE-WIDGET Intrinsic Function

Creates an instance of a widget.

Format

DWT:CREATE-WIDGET *name* *widget-class* *parent-widget* *arglist* *argcount*

<i>name</i>	STRING
<i>widget-class</i>	INTEGER
<i>parent-widget</i>	DWT:WIDGET
<i>arglist</i>	ArgList
<i>argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:CREATE-WINDOW Intrinsic Function

Creates a window for a widget.

Format

DWT:CREATE-WINDOW *widget* *window-class* *visual* *value-mask* *attributes*

<i>widget</i>	DWT:WIDGET
<i>window-class</i>	INTEGER
<i>visual</i>	Visual
<i>value-mask</i>	XtValueMask
<i>attributes</i>	XSetWindowAttributes

Return Value

An object of type CLX:WINDOW representing the new window.

DWT:CSBYTECMP Compound String Function

Compares two compound strings to determine if they are identical.

Format

DWT:CSBYTECMP *compound-string1* *compound-string2*

<i>compound-string1</i>	COMPOUND-STRING
<i>compound-string2</i>	COMPOUND-STRING

DWT:CSBYTECMP Compound String Function

Return Value

One of two integers representing the result of the comparison:

- | | |
|---|---|
| 0 | <i>compound-string1</i> and <i>compound-string2</i> are identical |
| 1 | <i>compound-string1</i> and <i>compound-string2</i> are not identical |

DWT:CSEMPY Compound String Function

Determines if the compound string contains any text segments.

Format

DWT:CSEMPY *compound-string*

compound-string COMPOUND-STRING

Return Value

A boolean value that is true if the compound string is empty.

DWT:CS-STRING Compound String Function

Creates a compound string.

Format

DWT:CS-STRING *text charset language dir-r-to-l rend*

<i>text</i>	STRING
<i>charset</i>	INTEGER
<i>language</i>	INTEGER
<i>dir-r-to-l</i>	INTEGER
<i>rend</i>	DwtRendMask

Return Value

An object of type COMPOUND-STRING representing the new compound string, or NIL if the input string is null.

DWT:CSTRCAT Compound String Function

Appends a copy of a compound string to the end of another compound string.

Format

DWT:CSTRCAT *compound-string1 compound-string2*

<i>compound-string1</i>	COMPOUND-STRING
<i>compound-string2</i>	COMPOUND-STRING

DWT:CSTRCAT Compound String Function

Return Value

An object of type COMPOUND-STRING.

DWT:CSTRCPY Compound String Function

Copies a compound string.

Format

DWT:CSTRCPY *compound-string*

compound-string COMPOUND-STRING

Return Value

An object of type COMPOUND-STRING.

DWT:CSTRLEN Compound String Function

Returns the number of bytes in a compound string.

Format

DWT:CSTRLEN *compound-string*

compound-string COMPOUND-STRING

Return Value

An integer representing the number of bytes in the compound string, including compound string terminators for headers and trailers. If the compound string has an invalid structure, zero is returned.

DWT:CSTRNCAT Compound String Function

Appends a copy of a compound string to the end of another compound string.

Format

DWT:CSTRNCAT *compound-string1* *compound-string2* *num-chars*

compound-string1 COMPOUND-STRING

compound-string2 COMPOUND-STRING

num-chars INTEGER

Return Value

An object of type COMPOUND-STRING.

DWT:CSTRNCPY Compound String Function

DWT:CSTRNCPY Compound String Function

Copies a compound string.

Format

DWT:CSTRNCPY *compound-string num-chars*

<i>compound-string</i>	COMPOUND-STRING
<i>num-chars</i>	INTEGER

Return Value

An object of type COMPOUND-STRING.

DWT:DATABASE Intrinsic Function

Returns the resource database for a display.

Format

DWT:DATABASE *display*

<i>display</i>	CLX:DISPLAY
----------------	-------------

Return Value

An integer representing the resource database.

DWT:DESTROY-APPLICATION-CONTEXT Intrinsic Function

Destroys an application context and closes any displays within it.

Format

DWT:DESTROY-APPLICATION-CONTEXT *context*

<i>context</i>	INTEGER
----------------	---------

Return Value

Unspecified.

DWT:DESTROY-GC Intrinsic Function

DWT:DESTROY-GC Intrinsic Function

Deallocates a graphics context.

Format

DWT:DESTROY-GC *widget gc*

<i>widget</i>	DWT:WIDGET
<i>gc</i>	CLX:GCONTEXT

Return Value

Unspecified.

DWT:DESTROY-WIDGET Intrinsic Function

Destroys an instance of a widget.

Format

DWT:DESTROY-WIDGET *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

Unspecified.

DWT:DIALOG-BOX High-Level Function

Creates a dialog box widget.

Format

DWT:DIALOG-BOX *parent-widget name default-position x y title style map-callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>default-position</i>	INTEGER
<i>x y</i>	INTEGER
<i>title</i>	COMPOUND-STRING
<i>style</i>	INTEGER
<i>map-callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:DIALOG-BOX-CREATE Low-Level Function

DWT:DIALOG-BOX-CREATE Low-Level Function

Creates a dialog box widget.

Format

DWT:DIALOG-BOX-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:DIALOG-BOX-POPUP-CREATE Low-Level Function

Creates a pop-up dialog box widget.

Format

DWT:DIALOG-BOX-POPUP-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:DIRECT-CONVERT Intrinsic Function

Invokes resource conversions.

Format

DWT:DIRECT-CONVERT *converter arglist argcount src-value dst-value*

<i>converter</i>	XtConverter
<i>arglist</i>	ArgList
<i>argcount</i>	INTEGER

DWT:DIRECT-CONVERT Intrinsic Function

src-value XrmValuePtr
dst-value XrmValuePtr

Return Value

Unspecified.

DWT:DISOWN-SELECTION Intrinsic Function

Informs the intrinsics global selection mechanism that a widget is no longer the selection owner.

Format

DWT:DISOWN-SELECTION *widget selection time*

widget DWT:WIDGET
selection INTEGER
time Time

Return Value

Unspecified.

DWT:DISPATCH-EVENT Intrinsic Function

Sends events to registered functions and widgets.

Format

DWT:DISPATCH-EVENT *event*

event DWT:EVENT

Return Value

Unspecified.

DWT:DISPLAY Intrinsic Function

Returns the display pointer for the specified widget.

Format

DWT:DISPLAY *widget*

widget DWT:WIDGET

Return Value

An object of type CLX:DISPLAY.

DWT:DISPLAY-CS-MESSAGE Convenience Function

DWT:DISPLAY-CS-MESSAGE Convenience Function

Displays a compound string message.

Format

```
DWT:DISPLAY-CS-MESSAGE parent-widget name pos x y style message-vector
widget convert-proc ok-callback help-callback
```

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>pos</i>	INTEGER
<i>x y</i>	INTEGER
<i>style</i>	INTEGER
<i>message-vector</i>	INTEGER
<i>widget</i>	DWT:WIDGET
<i>convert-proc</i>	CALL-BACK-ROUTINE
<i>ok-callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

Unspecified.

DWT:DISPLAY-INITIALIZE Intrinsic Function

Initializes a display and adds it to an application context.

Format

```
DWT:DISPLAY-INITIALIZE context display-name application-name class-name
urlist urcount argcount argvalue
```

<i>context</i>	INTEGER
<i>display-name</i>	STRING
<i>application-name</i>	STRING
<i>class-name</i>	STRING
<i>urlist</i>	XrmOptionDescRec
<i>urcount</i>	INTEGER
<i>argcount</i>	INTEGER
<i>argvalue</i>	ArgList

Return Value

Unspecified.

DWT:DISPLAY-VMS-MESSAGE Convenience Function

DWT:DISPLAY-VMS-MESSAGE Convenience Function

Accepts and displays a VMS message.

Format

DWT:DISPLAY-VMS-MESSAGE *parent-widget name pos x y style message-vector
widget convert-proc ok-callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>pos</i>	INTEGER
<i>x y</i>	INTEGER
<i>style</i>	INTEGER
<i>message-vector</i>	INTEGER
<i>widget</i>	DWT:WIDGET
<i>convert-proc</i>	CALL-BACK-ROUTINE
<i>ok-callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET.

DWT:DRM-FREE-RESOURCE-CONTEXT DRM Function

Frees a resource context.

Format

DWT:DRM-FREE-RESOURCE-CONTEXT *context-id*

<i>context-id</i>	DRMResourceContextPtr
-------------------	-----------------------

Return Value

One of two possible constants whose integer values represent the DRM status:

DWT:DRM-SUCCESS	1
DWT:DRM-BAD-CONTEXT	24

DWT:DRM-GET-RESOURCE-CONTEXT DRM Function

DWT:DRM-GET-RESOURCE-CONTEXT DRM Function

Gets a resource context.

Format

DWT:DRM-GET-RESOURCE-CONTEXT *alloc-func free-func size*

<i>alloc-func</i>	CALL-BACK-ROUTINE or NULL
<i>free-func</i>	CALL-BACK-ROUTINE or NULL
<i>size</i>	DRMSize

Return Values

Two values:

- DRMRessourceContextPtr, an integer representing the *context-id-return*.
- One of two possible constants whose integer values represent the DRM status:

DWT:DRM-FAILURE	0
DWT:DRM-SUCCESS	1

DWT:DRM-HGET-INDEXED-LITERAL DRM Function

Fetches indexed literals from a DRM hierarchy.

Format

DWT:DRM-HGET-INDEXED-LITERAL *hierarchy-id index context-id*

<i>hierarchy-id</i>	INTEGER
<i>index</i>	STRING
<i>context-id</i>	DRMRessourceContextPtr

Return Value

One of three possible constants whose integer values represent the DRM status:

DWT:DRM-SUCCESS	1
DWT:DRM-FAILURE	0
DWT:DRM-NOT-FOUND	2

DWT:DRM-RC-BUFFER DRM Function

DWT:DRM-RC-BUFFER DRM Function

Returns a pointer to the resource context buffer.

Format

DWT:DRM-RC-BUFFER *context-id*

context-id DRMResourceContextPtr

Return Value

An integer representing a pointer to the resource context buffer.

DWT:DRM-RC-SET-TYPE DRM Function

Modifies the type in the resource context.

Format

DWT:DRM-RC-SET-TYPE *context-id type-value*

context-id DRMResourceContextPtr

type-value DRMTType

Return Value

Unspecified.

DWT:DRM-RC-SIZE DRM Function

Returns the size of the value in the resource context buffer.

Format

DWT:DRM-RC-SIZE *context-id*

context-id DRMResourceContextPtr

Return Value

An integer representing the buffer size.

DWT:DRM-RC-TYPE DRM Function

Returns the type of the value in the resource context buffer.

Format

DWT:DRM-RC-TYPE *context-id*

context-id DRMResourceContextPtr

Return Value

An integer representing the type.

DWT:DWTHELP High-Level Function

Creates a help widget.

Format

DWT:DWTHELP *parent-widget name default-position x y application-name library-type
library-spec first-topic overview-topic glossary-topic unmap-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	COMPOUND-STRING
<i>default-position</i>	INTEGER
<i>x y</i>	INTEGER
<i>application-name</i>	COMPOUND-STRING
<i>library-type</i>	INTEGER
<i>library-spec</i>	COMPOUND-STRING
<i>first-topic</i>	COMPOUND-STRING
<i>overview-topic</i>	COMPOUND-STRING
<i>glossary-topic</i>	COMPOUND-STRING
<i>unmap-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET.

DWT:DWTWINDOW Intrinsic Function

Creates a window widget for simple applications to display in the main window widget work area.

Format

DWT:DWTWINDOW *parent-widget name x y width height callback*

parent-widget DWT:WIDGET

DWT:DWTWINDOW Intrinsic Function

<i>name</i>	STRING
<i>x</i> <i>y</i>	INTEGER
<i>width</i>	INTEGER
<i>height</i>	INTEGER

callback CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the window widget.

DWT:END-COPY-TO-CLIPBOARD Cut-and-Paste Function

Places data in the clipboard data structure.

Format

DWT:END-COPY-TO-CLIPBOARD *display* *window* *item-id*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>item-id</i>	INTEGER

Return Value

One of two possible constants whose integer values represent the clipboard status:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:ERROR-MSG Intrinsic Function

Calls the high-level error handler and passes the information specified.

Format

DWT:ERROR-MSG *resource-name* *resource-type* *resource-class*
 default-string *params* *num-params*

<i>resource-name</i>	STRING
<i>resource-type</i>	STRING
<i>resource-class</i>	STRING
<i>default-string</i>	STRING
<i>params</i>	STRING
<i>num-params</i>	INTEGER

Return Value

Unspecified.

DWT:FETCH-INTERFACE-MODULE DRM Function

DWT:FETCH-INTERFACE-MODULE DRM Function

Fetches all the widgets defined in an interface module in the DRM hierarchy.

Format

DWT:FETCH-INTERFACE-MODULE *hierarchy-id* *module-name* *parent*
&OPTIONAL *widget-return*

<i>hierarchy-id</i>	INTEGER
<i>module-name</i>	STRING
<i>parent</i>	DWT:WIDGET
<i>widget-return</i>	DWT:WIDGET

Return Values

Two values:

- An object of type DWT:WIDGET representing *widget-return*.
- One of three possible constants whose integer values represent the DRM status:

DWT:DRM-FAILURE	0
DWT:DRM-SUCCESS	1
DWT:DRM-NOT-FOUND	2

DWT:FETCH-SET-VALUES DRM Function

Fetches the values to be set from literals stored in UID files.

Format

DWT:FETCH-SET-VALUES *hierarchy-id* *widget* *args* *num-args*

<i>hierarchy-id</i>	INTEGER
<i>widget</i>	DWT:WIDGET
<i>args</i>	ArgList
<i>num-args</i>	INTEGER

Return Value

One of two possible constants whose integer values represent the DRM status:

DWT:DRM-SUCCESS	1
DWT:DRM-FAILURE	0

DWT:FETCH-WIDGET DRM Function

DWT:FETCH-WIDGET DRM Function

Fetches any indexed widget.

Format

DWT:FETCH-WIDGET *hierarchy-id index parent*

<i>hierarchy-id</i>	INTEGER
<i>index</i>	STRING
<i>parent</i>	DWT:WIDGET

Return Values

Two values:

- An object of type DWT:WIDGET representing the widget.
- A string representing the widget class.

DWT:FETCH-WIDGET-OVERRIDE DRM Function

Fetches any indexed widget. Overrides DWT:FETCH-WIDGET arguments.

Format

DWT:FETCH-WIDGET-OVERRIDE *hierarchy-id index parent override-name
override-args override-num-args*

<i>hierarchy-id</i>	INTEGER
<i>index</i>	STRING
<i>parent</i>	DWT:WIDGET
<i>override-name</i>	STRING
<i>override-args</i>	ArgList
<i>override-num-args</i>	INTEGER

Return Values

Two values:

- An object of type DWT:WIDGET representing the widget.
- A string representing the class.

DWT:FILE-SELECTION High-Level Function

DWT:FILE-SELECTION High-Level Function

Creates a file selection box widget.

Format

```
DWT:FILE-SELECTION parent-widget name x y title value dirmask visible-items-count  
                  style default-position callback help-callback
```

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>title</i>	COMPOUND-STRING
<i>value</i>	COMPOUND-STRING
<i>dirmask</i>	COMPOUND-STRING
<i>visible-items-count</i>	INTEGER
<i>style</i>	INTEGER
<i>default-position</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:FILE-SELECTION-CREATE Low-Level Function

Creates a file selection widget.

Format

```
DWT:FILE-SELECTION-CREATE parent-widget name override-arglist override-argcount
```

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:FILE-SELECTION-DO-SEARCH High-Level Function

DWT:FILE-SELECTION-DO-SEARCH High-Level Function

Initiates a search with a directory mask option. Otherwise, the current directory mask is used.

Format

DWT:FILE-SELECTION-DO-SEARCH *widget dirmask*

<i>widget</i>	DWT:WIDGET
<i>dirmask</i>	COMPOUND-STRING

Return Value

Unspecified.

DWT:FREE Intrinsic Function

Frees an allocated block of storage.

Format

DWT:FREE *address*

<i>address</i>	INTEGER or any alien structure
----------------	--------------------------------

Return Value

Unspecified.

DWT:GET-APPLICATION-RESOURCES Intrinsic Function

Retrieves resources that are not specific to a widget but apply to the overall application.

Format

DWT:GET-APPLICATION-RESOURCES *widget base-addr resource-list arglist
resource-count argcnt*

<i>widget</i>	DWT:WIDGET
<i>base-addr</i>	INTEGER
<i>resource-list</i>	XtResourceList
<i>arglist</i>	ArgList
<i>resource-count</i>	INTEGER
<i>argcnt</i>	INTEGER

DWT:GET-APPLICATION-RESOURCES Intrinsic Function

Return Value

Unspecified.

DWT:GET-DISPLAY Convenience Function

See DWT:DISPLAY.

DWT:GET-ERROR-DATABASE Intrinsic Function

Obtains the error database.

Format

DWT:GET-ERROR-DATABASE

Return Value

An integer representing the address of the error database.

DWT:GET-ERROR-DATABASE-TEXT Intrinsic Function

Obtains the error database text for an error or a warning.

Format

DWT:GET-ERROR-DATABASE-TEXT *resource-name resource-type resource-class default-string buffer size*

<i>resource-name</i>	STRING
<i>resource-type</i>	STRING
<i>resource-class</i>	STRING
<i>default-string</i>	STRING
<i>buffer</i>	STRING
<i>size</i>	INTEGER

Return Value

Unspecified.

DWT:GET-GC Intrinsic Function

DWT:GET-GC Intrinsic Function

Returns a read-only, shareable graphics context.

Format

DWT:GET-GC *widget value-mask values*

<i>widget</i>	DWT:WIDGET
<i>value-mask</i>	XtGCMask
<i>values</i>	XtGCValues

Return Value

An object of type CLX:GCONTEXT representing the graphics context.

DWT:GET-NEXT-SEGMENT Compound String Function

Gets information about the next segment in the compound string.

Format

DWT:GET-NEXT-SEGMENT *context*
 &OPTIONAL *string-text charset dir-r-to-l language rend*

<i>context</i>	DwtCompStringContext
<i>string-text</i>	STRING
<i>charset</i>	INTEGER
<i>dir-r-to-l</i>	INTEGER
<i>language</i>	INTEGER
<i>rend</i>	INTEGER

Return Values

Six values:

- A string representing the *string-text*.
- An integer representing the *charset*.
- An integer representing the *dir-r-to-l*.
- An integer representing the *language*.
- An integer representing *rend*.
- One of three possible constants whose integer values represent the status as follows:

DWT:FAIL	0
DWT:SUCCESS	1
DWT:END-CS	3

DWT:GET-RESOURCE-LIST Intrinsic Function

DWT:GET-RESOURCE-LIST Intrinsic Function

Obtains the resource list structure for a particular class of widget.

Format

DWT:GET-RESOURCE-LIST *class resources-return num-resources-return*

<i>class</i>	WidgetClass
<i>resources-return</i>	XtResourceList
<i>num-resources-return</i>	INTEGER

Return Value

Unspecified.

DWT:GET-SCREEN Convenience Function

Returns the widget screen.

Format

DWT:GET-SCREEN *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

An object of type CLX:SCREEN representing the widget screen.

DWT:GET-SELECTION-TIMEOUT Intrinsic Function

Returns the current value of the intrinsics selection timeout interval.

Format

DWT:GET-SELECTION-TIMEOUT

Return Value

An integer representing the current value of the timeout interval in milliseconds.

DWT:GET-SELECTION-VALUE Intrinsic Function

Obtains the selection value in a single, logical unit.

Format

DWT:GET-SELECTION-VALUE *widget selection target callback client-data time*

DWT:GET-SELECTION-VALUE Intrinsic Function

<i>widget</i>	DWT:WIDGET
<i>selection</i>	INTEGER
<i>target</i>	INTEGER
<i>callback</i>	XtSelectionCallbackProc
<i>client-data</i>	INTEGER or alien structure
<i>time</i>	Time

Return Value

Unspecified.

DWT:GET-SELECTION-VALUE-INCR Intrinsic Function

Obtains the selection data to the specified widget when using incremental data transfers.

Format

DWT:GET-SELECTION-VALUE-INCR *widget selection target callback
cancel client-data time*

<i>widget</i>	DWT:WIDGET
<i>selection</i>	INTEGER
<i>target</i>	INTEGER
<i>callback</i>	XtSelectionIncrCallbackProc
<i>cancel</i>	XtCancelConvertSelectionProc
<i>client-data</i>	INTEGER or alien structure
<i>time</i>	Time

Return Value

Unspecified.

DWT:GET-SELECTION-VALUES Intrinsic Function

Obtains the selection value in a single, logical unit.

Format

DWT:GET-SELECTION-VALUES *widget selection targets count callback
client-datas time*

<i>widget</i>	DWT:WIDGET
<i>selection</i>	INTEGER
<i>targets</i>	Alien structure
<i>count</i>	INTEGER

DWT:GET-SELECTION-VALUES Intrinsic Function

<i>callback</i>	XtSelectionCallbackProc
<i>client-data</i>	INTEGER or alien structure
<i>time</i>	Time

Return Value

Unspecified.

DWT:GET-SELECTION-VALUES-INCR Intrinsic Function

Obtains the selection data to the specified widget when using incremental data transfers.

Format

DWT:GET-SELECTION-VALUES-INCR *widget selection targets count callback cancel client-data time*

<i>widget</i>	DWT:WIDGET
<i>selection</i>	INTEGER
<i>targets</i>	Alien structure
<i>count</i>	INTEGER
<i>callback</i>	XtSelectionIncrCallbackProc
<i>cancel</i>	XtCancelConvertSelectionProc
<i>client-data</i>	INTEGER or alien structure
<i>time</i>	Time

Return Value

Unspecified.

DWT:GET-SUBRESOURCES Intrinsic Function

Retrieves resources for nonwidget subparts.

Format

DWT:GET-SUBRESOURCES *widget base-addr resource-name resource-class resource-list resource-count arglist argcnt*

<i>widget</i>	DWT:WIDGET
<i>base-addr</i>	INTEGER
<i>resource-name</i>	STRING
<i>resource-class</i>	STRING
<i>resource-list</i>	XtResourceList
<i>resource-count</i>	INTEGER

DWT:GET-SUBRESOURCES Intrinsic Function

arglist ArgList
argcoun INTEGER

Return Value

Unspecified.

DWT:GET-SUBVALUES Intrinsic Function

Retrieves the value of nonwidget resource data associated with a widget instance.

Format

DWT:GET-SUBVALUES *base-addr* *resource-list* *resource-count* *arglist* *argcoun*

base-addr INTEGER
resource-list XtResourceList
resource-count INTEGER
arglist ArgList
argcoun INTEGER

Return Value

Unspecified.

DWT:GET-VALUES Intrinsic Function

Retrieves the current value of resource data associated with a widget instance.

Format

DWT:GET-VALUES *widget* *arglist* *argcoun*

widget DWT:WIDGET
arglist ArgList
argcoun INTEGER

Return Value

The argument list that was passed to the function.

DWT:HAS-CALLBACKS Intrinsic Function

Determines the status of a callback list.

Format

DWT:HAS-CALLBACKS *widget* *callback-name*

DWT:HAS-CALLBACKS Intrinsic Function

<i>widget</i>	DWT:WIDGET
<i>callback-name</i>	INTEGER

Return Value

One of the following constants whose integer values represent the status:

DWT:XTCALLBACKNOLIST	0
DWT:XTCALLBACKHASNONE	1
DWT:XTCALLBACKHAS SOME	2

DWT:HELP High-Level Function

See DWT:DWTHELP high-level function.

DWT:HELP-CREATE Low-Level Function

Creates a help widget.

Format

DWT:HELP-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:INIT-GET-SEGMENT Compound String Function

Initializes the context needed by DWT:GET-NEXT-SEGMENT.

Format

DWT:INIT-GET-SEGMENT *context compound-string*

<i>context</i>	DwtCompStringContext
<i>compound-string</i>	COMPOUND-STRING

Return Values

Two values:

- A compound string representing the *context*.

DWT:INIT-GET-SEGMENT Compound String Function

- One of three possible constants whose integer values represent the status:

DWT:FAIL	0
DWT:SUCCESS	1
DWT:END-CS	3

DWT:INITIALIZE Intrinsic Function

Initializes the DECwindows Toolkit.

Format

DWT:INITIALIZE *name class-name urlist num-urlist argcount argvalue*

<i>name</i>	STRING
<i>class-name</i>	STRING
<i>urlist</i>	XrmOptionDescRec
<i>num-urlist</i>	INTEGER
<i>argcount</i>	INTEGER
<i>argvalue</i>	ArgList

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:INITIALIZE-DRM DRM Function

Prepares an application to use DRM facilities.

Format

DWT:INITIALIZE-DRM

Return Value

Unspecified.

DWT:INQUIRE-NEXT-PASTE-COUNT Cut-and-Paste Function

Opens all the UID files in the DRM hierarchy.

Format

DWT:INQUIRE-NEXT-PASTE-COUNT *display window*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW

DWT:INQUIRE-NEXT-PASTE-COUNT Cut-and-Paste Function

Return Values

Two values:

- An integer representing the *count*.
- An integer representing *max-format-name-length*.

DWT:INQUIRE-NEXT-PASTE-FORMAT Cut-and-Paste Function

Returns a specified format name for the next-paste item in the clipboard.

Format

DWT:INQUIRE-NEXT-PASTE-FORMAT *display window number*
format-name-buf buffer-len

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>number</i>	INTEGER
<i>format-name-buf</i>	STRING
<i>buffer-len</i>	INTEGER

Return Values

Three values:

- A string representing the *format-name-buf*.
- An integer representing the *copied-length*.
- One of three possible constants whose integer values represent the clipboard status:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-TRUNCATE	2
DWT:CLIPBOARD-LOCKED	4

DWT:INQUIRE-NEXT-PASTE-LENGTH Cut-and-Paste Function

Returns the length of the data stored under a specified format name for the next-paste item in the clipboard.

Format

DWT:INQUIRE-NEXT-PASTE-LENGTH *display window format-name*

<i>display</i>	CLX:DISPLAY
----------------	-------------

DWT:INQUIRE-NEXT-PASTE-LENGTH Cut-and-Paste Function

<i>window</i>	CLX:WINDOW
<i>format-name</i>	STRING

Return Values

Two values:

- An integer representing the length.
- One of two possible constants whose integer values represent the clipboard status:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:INSTALL-ACCELERATORS Intrinsic Function

Installs accelerators from a widget on another widget.

Format

DWT:INSTALL-ACCELERATORS *dst-widget src-widget*

<i>dst-widget</i>	DWT:WIDGET
<i>src-widget</i>	DWT:WIDGET

Return Value

Unspecified.

DWT:INSTALL-ALL-ACCELERATORS Intrinsic Function

Installs all accelerators from a widget and all its descendants onto one destination.

Format

DWT:INSTALL-ALL-ACCELERATORS *dst-widget src-widget*

<i>dst-widget</i>	DWT:WIDGET
<i>src-widget</i>	DWT:WIDGET

Return Value

Unspecified.

DWT:IS-COMPOSITE Intrinsic Function

Determines if a widget is a subclass of Composite.

Format

DWT:IS-COMPOSITE *widget*

widget DWT:WIDGET

Return Value

A boolean value that is true if the widget is a subclass of Composite.

DWT:IS-MANAGED Intrinsic Function

Determines the managed state of a given child widget.

Format

DWT:IS-MANAGED *widget*

widget DWT:WIDGET

Return Value

A boolean value that is true if the widget is managed.

DWT:IS-REALIZED Intrinsic Function

Determines if the widget is realized.

Format

DWT:IS-REALIZED *widget*

widget DWT:WIDGET

Return Value

A boolean value that is true if the widget is realized.

DWT:IS-SENSITIVE Intrinsic Function

Determines the current sensitivity state of a widget.

Format

DWT:IS-SENSITIVE *widget*

widget DWT:WIDGET

DWT:IS-SENSITIVE Intrinsic Function

Return Value

True or false to indicate whether user input events are being dispatched.

DWT:IS-SUBCLASS Intrinsic Function

Determines the subclass that the specified widget belongs to.

Format

DWT:IS-SUBCLASS *widget* *widget-class*

<i>widget</i>	DWT:WIDGET
<i>widget-class</i>	INTEGER

Return Value

A boolean value that is true if the widget is a subclass of *widget-class*.

DWT:LABEL High-Level Function

Creates a label widget for the application to display identification information on the screen.

Format

DWT:LABEL *parent-widget* *name* *x* *y* *label* *help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x</i> <i>y</i>	INTEGER
<i>label</i>	COMPOUND-STRING
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:LABEL-CREATE Low-Level Function

Creates a label widget for the application to display identification information on the screen.

Format

DWT:LABEL-CREATE *parent-widget* *name* *override-arglist* *override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING

DWT:LABEL-CREATE Low-Level Function

<i>override-arglist</i>	ArgList
<i>override-argcoun</i> t	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:LABEL-GADGET-CREATE Gadget Function

Creates a label gadget.

Format

DWT:LABEL-GADGET-CREATE *parent-widget name override-arglist override-argcoun*t

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcoun</i> t	INTEGER

Return Value

An object of type DWT:WIDGET representing the newly created widget.

DWT:LATIN1-STRING Compound String Function

Creates a compound string for the LATIN1 character set.

Format

DWT:LATIN1-STRING *text-string*

<i>text-string</i>	STRING
--------------------	--------

Return Value

An object of type COMPOUND-STRING representing the resulting compound string.

DWT:LIST-BOX High-Level Function

Creates a list box widget for the application to display large numbers of item choices or entries in a list format.

Format

DWT:LIST-BOX *parent-widget name x y items item-count visible-items-count callback help-callback resize horiz*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING

DWT:LIST-BOX High-Level Function

<i>x</i>	INTEGER
<i>items</i>	COMPOUND-STRING
<i>item-count</i>	INTEGER
<i>visible-items-count</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE
<i>resize</i>	INTEGER
<i>horiz</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:LIST-BOX-ADD-ITEM High-Level Function

Adds an item to the list within a list box widget.

Format

DWT:LIST-BOX-ADD-ITEM *widget item pos*

<i>widget</i>	DWT:WIDGET
<i>item</i>	COMPOUND-STRING
<i>pos</i>	INTEGER

Return Value

Unspecified.

DWT:LIST-BOX-CREATE Low-Level Function

Creates a list box widget.

Format

DWT:LIST-BOX-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:LIST-BOX-DELETE-ITEM High-Level Function

DWT:LIST-BOX-DELETE-ITEM High-Level Function

Deletes an item from the list within a list box widget.

Format

DWT:LIST-BOX-DELETE-ITEM *widget item*

widget

DWT:WIDGET

item

COMPOUND-STRING

Return Value

Unspecified.

DWT:LIST-BOX-DELETE-POS High-Level Function

Deletes an item identified by its position from the list within a list box widget.

Format

DWT:LIST-BOX-DELETE-POS *widget position*

widget

DWT:WIDGET

position

INTEGER

Return Value

Unspecified.

DWT:LIST-BOX-DESELECT-ALL-ITEMS High-Level Function

Deselects all of the previously selected items in a list box.

Format

DWT:LIST-BOX-DESELECT-ALL-ITEMS *widget*

widget

DWT:WIDGET

Return Value

Unspecified.

DWT:LIST-BOX-DESELECT-ITEM High-Level Function

DWT:LIST-BOX-DESELECT-ITEM High-Level Function

Deselects a previously selected item in a list box.

Format

DWT:LIST-BOX-DESELECT-ITEM *widget item*

<i>widget</i>	DWT:WIDGET
<i>item</i>	COMPOUND-STRING

Return Value

Unspecified.

DWT:LIST-BOX-DESELECT-POS High-Level Function

Deselects an item identified by its position in a list box.

Format

DWT:LIST-BOX-DESELECT-POS *widget position*

<i>widget</i>	DWT:WIDGET
<i>position</i>	INTEGER

Return Value

Unspecified.

DWT:LIST-BOX-ITEM-EXISTS High-Level Function

Verifies the existence of a particular item in a list box.

Format

DWT:LIST-BOX-ITEM-EXISTS *widget item*

<i>widget</i>	DWT:WIDGET
<i>item</i>	COMPOUND-STRING

Return Value

A boolean value that is true if the list box item exists.

DWT:LIST-BOX-SELECT-ITEM High-Level Function

DWT:LIST-BOX-SELECT-ITEM High-Level Function

Selects an item in the list box.

Format

DWT:LIST-BOX-SELECT-ITEM *widget item notify*

<i>widget</i>	DWT:WIDGET
<i>item</i>	COMPOUND-STRING
<i>notify</i>	INTEGER

Return Value

Unspecified.

DWT:LIST-BOX-SELECT-POS High-Level Function

Selects an item identified by its position in the list box.

Format

DWT:LIST-BOX-SELECT-POS *widget position notify*

<i>widget</i>	DWT:WIDGET
<i>position</i>	INTEGER
<i>notify</i>	INTEGER

Return Value

Unspecified.

DWT:LIST-BOX-SET-HORIZ-POS High-Level Function

Sets the horizontal position to a specified position.

Format

DWT:LIST-BOX-SET-HORIZ-POS *widget position*

<i>widget</i>	DWT:WIDGET
<i>position</i>	INTEGER

Return Value

Unspecified.

DWT:LIST-BOX-SET-ITEM High-Level Function

DWT:LIST-BOX-SET-ITEM High-Level Function

Makes a specified item (if it exists) the first visible item in a list box, or as close to the top as possible. The item always becomes visible.

Format

DWT:LIST-BOX-SET-ITEM *widget item*

<i>widget</i>	DWT:WIDGET
<i>item</i>	COMPOUND-STRING

Return Value

Unspecified.

DWT:LIST-BOX-SET-POS High-Level Function

Makes a specified position (item number in list) the top visible position in a list box, or as close to the top as possible.

Format

DWT:LIST-BOX-SET-POS *widget position*

<i>widget</i>	DWT:WIDGET
<i>position</i>	INTEGER

Return Value

Unspecified.

DWT:LIST-PENDING-ITEMS Cut-and-Paste Function

Returns a list of data id/private id pairs for a specified format name.

Format

DWT:LIST-PENDING-ITEMS *display window format-name item-list count*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>format-name</i>	STRING
<i>item-list</i>	DwtClipboardPendingList
<i>count</i>	INTEGER

DWT:LIST-PENDING-ITEMS Cut-and-Paste Function

Return Values

Three values:

- An alien structure representing the *item-list*.
- An integer representing the *count*.
- One of two possible constants whose integer values represent the clipboard status:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:MAIN-LOOP Intrinsic Function

Processes input data.

Format

DWT:MAIN-LOOP

Return Value

Unspecified.

DWT:MAIN-WINDOW High-Level Function

Creates the main window widget.

Format

DWT:MAIN-WINDOW *parent-widget name x y width height*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>width</i>	INTEGER
<i>height</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MAIN-WINDOW-CREATE Low-Level Function

Creates a main window widget that is the parent of all widgets in an application that conforms to the DECwindows style.

Format

DWT:MAIN-WINDOW-CREATE *parent-widget name override-arglist override-argc*

DWT:MAIN-WINDOW-CREATE Low-Level Function

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argc</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MAIN-WINDOW-SET-AREAS High-Level Function

Sets up or adds the menu bar, work window, command window, and scroll bar widgets to the main window widget of the application.

Format

DWT:MAIN-WINDOW-SET-AREAS *widget menu-bar work-window command-window h-scroll v-scroll*

<i>widget</i>	DWT:WIDGET
<i>menu-bar</i>	DWT:WIDGET
<i>work-window</i>	DWT:WIDGET
<i>command-window</i>	DWT:WIDGET
<i>h-scroll</i>	DWT:WIDGET
<i>v-scroll</i>	DWT:WIDGET

Return Value

Unspecified.

DWT:MAKE-GEOMETRY-REQUEST Intrinsic Function

Makes a geometry manager request from a widget.

Format

DWT:MAKE-GEOMETRY-REQUEST *widget requested-size approved-size*

<i>widget</i>	DWT:WIDGET
<i>requested-size</i>	XtWidgetGeometry
<i>approved-size</i>	XtWidgetGeometry

Return Value

An XtGeometryResult alien structure representing the return code from the Geometry Manager.

DWT:MAKE-RESIZE-REQUEST Intrinsic Function

DWT:MAKE-RESIZE-REQUEST Intrinsic Function

Makes a resize request from a widget.

Format

DWT:MAKE-RESIZE-REQUEST *widget requested-width requested-height*

<i>widget</i>	DWT:WIDGET
<i>requested-width</i>	INTEGER
<i>requested-height</i>	INTEGER

Return Values

Three values:

- An integer representing the *approved-width*.
- An integer representing the *approved-height*.
- One of the following constants representing the status:

DWT:FAIL	0
DWT:SUCCESS	1
DWT:GEOMETRY-ALMOST	3

DWT:ALLOC Intrinsic Function

Allocates storage.

Format

DWT:ALLOC *size*

<i>size</i>	INTEGER
-------------	---------

Return Value

An integer representing a pointer to a block of storage.

DWT:MANAGE-CHILD Intrinsic Function

Adds a single child to the managed children of the parent widget.

Format

DWT:MANAGE-CHILD *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

Unspecified.

DWT:MANAGE-CHILDREN Intrinsic Function

DWT:MANAGE-CHILDREN Intrinsic Function

Performs the basic functions of managing children.

Format

DWT:MANAGE-CHILDREN *widget-list num-children*

<i>widget-list</i>	WidgetList
<i>num-children</i>	INTEGER

Return Value

Unspecified.

DWT:MAP-WIDGET Intrinsic Function

Maps a widget explicitly.

Format

DWT:MAP-WIDGET *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

Unspecified.

DWT:MENU High-Level Function

Creates a menu widget to contain other menu items (subwidgets) for the display of application menus.

Format

DWT:MENU *parent-widget name x y format orientation callback map-callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>format</i>	INTEGER
<i>orientation</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>map-callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MENU-BAR High-Level Function

DWT:MENU-BAR High-Level Function

Creates a menu bar widget to contain other menu items (subwidgets) for the application in menu bar displays.

Format

DWT:MENU-BAR *parent-widget name callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MENU-BAR-CREATE Low-Level Function

Creates a menu bar widget.

Format

DWT:MENU-BAR-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MENU-CREATE Low-Level Function

Creates a menu widget to contain other menu items (subwidgets) for the display of application menus.

Format

DWT:MENU-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

DWT:MENU-CREATE Low-Level Function

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MENU-POPUP-CREATE Low-Level Function

Creates a pop-up menu (MB2 only).

Format

DWT:MENU-POPUP-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MENU-POSITION High-Level Function

Positions the pop-up menu when user presses MB2.

Format

DWT:MENU-POSITION *widget event*

<i>widget</i>	DWT:WIDGET
<i>event</i>	DWT:EVENT

Return Value

Unspecified.

DWT:MENU-PULLDOWN-CREATE Low-Level Function

Creates a pull-down menu.

Format

DWT:MENU-PULLDOWN-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

DWT:MENU-PULLDOWN-CREATE Low-Level Function

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MERGE-ARG-LISTS Intrinsic Function

Merges two argument lists.

Format

DWT:MERGE-ARG-LISTS *arglist1 argcount1 arglist2 argcount2*

<i>arglist1</i>	ArgList
<i>argcount1</i>	INTEGER
<i>arglist2</i>	ArgList
<i>argcount2</i>	INTEGER

Return Value

An object of type ArgList representing the combined argument list.

DWT:MESSAGE-BOX High-Level Function

Creates a message box widget for the application to display text to the user.

Format

DWT:MESSAGE-BOX *parent-widget name default-position x y style label ok-label callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>default-position</i>	INTEGER
<i>x</i>	INTEGER
<i>y</i>	INTEGER
<i>style</i>	INTEGER
<i>label</i>	COMPOUND-STRING
<i>ok-label</i>	COMPOUND-STRING
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MESSAGE-BOX-CREATE Low-Level Function

DWT:MESSAGE-BOX-CREATE Low-Level Function

Creates a message box widget for the application to display text to the user.

Format

DWT:MESSAGE-BOX-CREATE *parent-widget name override-arglist override-argcoun*t

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcoun</i> t	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:MOVE-WIDGET Intrinsic Function

Moves widget children around.

Format

DWT:MOVE-WIDGET *widget x y*

<i>widget</i>	DWT:WIDGET
<i>x y</i>	INTEGER

Return Value

Unspecified.

DWT:NAME-TO-WIDGET Intrinsic Function

Translates a widget name to a widget instance.

Format

DWT:NAME-TO-WIDGET *root name*

<i>root</i>	DWT:WIDGET
<i>name</i>	STRING

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:NEXT-EVENT Intrinsic Function

Reads the next input available, or waits for the next event if none is queued.

Format

DWT:NEXT-EVENT &*OPTIONAL event*

event DWT:EVENT

Return Value

An object of type DWT:EVENT representing the next event in the queue.

DWT:NUMBER-CHILDREN Convenience Function

Determines how many widgets are children of a given widget.

Format

DWT:NUMBER-CHILDREN *widget*

widget DWT:WIDGET

Return Value

A positive integer representing the number of children.

DWT:OPEN-DISPLAY Intrinsic Function

Opens a display, initializes it, and adds it to an application context.

Format

DWT:OPEN-DISPLAY *context display-name application-name class-name urlist urcount argcoun argvalue*

<i>context</i>	INTEGER
<i>display-name</i>	STRING
<i>application-name</i>	STRING
<i>class-name</i>	STRING
<i>urlist</i>	XrmOptionDescRec
<i>urcount</i>	INTEGER
<i>argcoun</i>	INTEGER
<i>argvalue</i>	ArgList

Return Value

An object of type CLX:DISPLAY representing the CLX display.

DWT:OPEN-HIERARCHY DRM Function

DWT:OPEN-HIERARCHY DRM Function

Opens all the UID files in the DRM hierarchy.

Format

DWT:OPEN-HIERARCHY *num-files file-names-list ancillary-structures-list*
&OPTIONAL *hierarchy-id-return*

<i>num-files</i>	INTEGER
<i>file-names-list</i>	STRING
<i>ancillary-structures-list</i>	IDBOSOpenParamPtr
<i>hierarchy-id-return</i>	DRMHierarchy

Return Value

An alien structure representing the *hierarchy-id-return*.

DWT:OPTION-MENU High-Level Function

Creates an option menu widget to display and handle an application option list of attributes or modes of the menu topic. It allows just one option selected from the list in the menu.

Format

DWT:OPTION-MENU *parent-widget name x y label submenu callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>label</i>	COMPOUND-STRING
<i>submenu</i>	DWT:WIDGET
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:OPTION-MENU-CREATE Low-Level Function

Creates an option menu widget.

Format

DWT:OPTION-MENU-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
----------------------	------------

DWT:OPTION-MENU-CREATE Low-Level Function

<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:OVERRIDE-TRANSLATIONS Intrinsic Function

Overwrites existing translations with new translations.

Format

DWT:OVERRIDE-TRANSLATIONS *widget translation-table*

<i>widget</i>	DWT:WIDGET
<i>translation-table</i>	XtTranslations

Return Value

Unspecified.

DWT:OWN-SELECTION Intrinsic Function

Sets the selection owner when using atomic data transfers.

Format

DWT:OWN-SELECTION *widget selection time convert-proc loss-proc notify-proc*

<i>widget</i>	DWT:WIDGET
<i>selection</i>	INTEGER
<i>time</i>	Time
<i>convert-proc</i>	XtConvertSelectionProc
<i>loss-proc</i>	XtLoseSelectionProc
<i>notify-proc</i>	XtSelectionDoneProc

Return Value

Unspecified.

DWT:OWN-SELECTION-INCREMENTAL Intrinsic Function

DWT:OWN-SELECTION-INCREMENTAL Intrinsic Function

Sets the selection owner when using incremental transfers.

Format

DWT:OWN-SELECTION-INCREMENTAL *widget selection time convert-proc loss-proc
notify-proc cancel-proc client-datas*

<i>widget</i>	DWT:WIDGET
<i>selection</i>	INTEGER
<i>time</i>	Time
<i>convert-proc</i>	XtConvertSelectionIncrProc
<i>loss-proc</i>	XtLoseSelectionIncrProc
<i>notify-proc</i>	XtSelectionDoneIncrProc
<i>cancel-proc</i>	XtCancelSelectionCallbackProc
<i>client-datas</i>	INTEGER or alien structure

Return Value

A boolean value that is true if the widget is successful in becoming the owner.

DWT:PARENT Intrinsic Function

Returns the parent widget for a specified widget.

Format

DWT:PARENT *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

An object of type DWT:WIDGET that is the parent widget.

DWT:PARSE-ACCELERATOR-TABLE Intrinsic Function

Parses an accelerator table.

Format

DWT:PARSE-ACCELERATOR-TABLE *source*

<i>source</i>	STRING
---------------	--------

DWT:PARSE-ACCELERATOR-TABLE Intrinsic Function

Return Value

An XtAccelerators alien structure pointer representing the parsed translation table.

DWT:PARSE-TRANSLATION-TABLE Intrinsic Function

Parses a translation table.

Format

DWT:PARSE-TRANSLATION-TABLE *source*

source STRING

Return Value

XtTranslations, a pointer to the parsed translation table.

DWT:PEEK-EVENT Intrinsic Function

Returns the value from the head of the input queue without removing input from the queue.

Format

DWT:PEEK-EVENT &OPTIONAL *event*

event DWT:EVENT

Return Value

An object of type DWT:EVENT.

DWT:PENDING Intrinsic Function

Determines if any events are pending on the input queue.

Format

DWT:PENDING

Return Value

A boolean value that is true if events are pending on the input queue.

DWT:POPDOWN Intrinsic Function

DWT:POPDOWN Intrinsic Function

Unmaps a pop-up from within an application.

Format

DWT:POPDOWN *widget*

widget DWT:WIDGET

Return Value

Unspecified.

DWT:POPUP Intrinsic Function

Maps (pops up) a pop-up from within an application.

Format

DWT:POPUP *widget pgrab-kind*

widget DWT:WIDGET

pgrab-kind XtGrabKind

Return Value

Unspecified.

DWT:PROCESS-EVENT Intrinsic Function

Processes one input event, timeout, or alternate input source.

Format

DWT:PROCESS-EVENT *input-mask*

input-mask XtInputMask

Return Value

Unspecified.

DWT:PULL-DOWN-MENU-ENTRY High-Level Function

DWT:PULL-DOWN-MENU-ENTRY High-Level Function

Creates an instance of the pull-down menu entry widget.

Format

DWT:PULL-DOWN-MENU-ENTRY *parent-widget name x y label menu-id
callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>label</i>	COMPOUND-STRING
<i>menu-id</i>	DWT:WIDGET
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:PULL-DOWN-MENU-ENTRY-CREATE Low-Level Function

Creates a pull-down menu entry widget.

Format

DWT:PULL-DOWN-MENU-ENTRY-CREATE *parent-widget name
override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:PULL-DOWN-MENU-ENTRY-HILITE High-Level Function

Highlights a menu entry.

Format

DWT:PULL-DOWN-MENU-ENTRY-HILITE *menu hilite*

DWT:PULL-DOWN-MENU-ENTRY-HILITE High-Level Function

menu DWT:WIDGET
hilite INTEGER

Return Value

Unspecified.

DWT:PUSH-BUTTON High-Level Function

Creates a push button widget for the application.

Format

DWT:PUSH-BUTTON *parent-widget name x y label callback help-callback*

parent-widget DWT:WIDGET
name STRING
x y INTEGER
label COMPOUND-STRING
callback CALL-BACK-ROUTINE
help-callback CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:PUSH-BUTTON-CREATE Low-Level Function

Creates a push button widget.

Format

DWT:PUSH-BUTTON-CREATE *parent-widget name override-arglist override-argcount*

parent-widget DWT:WIDGET
name STRING
override-arglist ArgList
override-argcount INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:PUSH-BUTTON-GADGET-CREATE Gadget Function

DWT:PUSH-BUTTON-GADGET-CREATE Gadget Function

Creates a push button gadget.

Format

DWT:PUSH-BUTTON-GADGET-CREATE *parent-widget name*
 override-arglist override-argcount

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:QUERY-GEOMETRY Intrinsic Function

Obtains a widget's preferred geometry.

Format

DWT:QUERY-GEOMETRY *widget requested approved*

<i>widget</i>	DWT:WIDGET
<i>requested</i>	XtWidgetGeometry
<i>approved</i>	XtWidgetGeometry

Return Value

An XtGeometry alien structure pointer representing the widget's preferred geometry.

DWT:RADIO-BOX High-Level Function

Creates a radio box widget for the application to display multiple toggle buttons.

Format

DWT:RADIO-BOX *parent-widget name x y callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER

DWT:RADIO-BOX High-Level Function

<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:RADIO-BOX-CREATE Low-Level Function

Creates a radio box widget.

Format

DWT:RADIO-BOX-CREATE *parent-widget name override-arglist override-argc*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argc</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:REALIZE-WIDGET Intrinsic Function

Performs widget realization.

Format

DWT:REALIZE-WIDGET *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

Unspecified.

DWT:REALLOC Intrinsic Function

Changes the size of an allocated block of storage.

Format

DWT:REALLOC *addr size*

<i>addr</i>	INTEGER
<i>size</i>	INTEGER

Return Value

An integer representing the new storage address.

DWT:RECOPY-TO-CLIPBOARD Cut-and-Paste Function

DWT:RECOPY-TO-CLIPBOARD Cut-and-Paste Function

Copies a data item that had been passed by name to the clipboard.

Format

DWT:RECOPY-TO-CLIPBOARD *display window data buffer private-id*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW
<i>data</i>	INTEGER
<i>buffer</i>	STRING
<i>private-id</i>	INTEGER

Return Value

One of two possible constants whose integer values represent the clipboard status:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:REGISTER-CASE-CONVERTER Intrinsic Function

Registers a case converter.

Format

DWT:REGISTER-CASE-CONVERTER *display converter start stop*

<i>display</i>	CLX:DISPLAY
<i>converter</i>	XtCaseProc
<i>start</i>	KeySym
<i>stop</i>	KeySym

Return Value

Unspecified.

DWT:REGISTER-CLASS DRM Function

DWT:REGISTER-CLASS DRM Function

Provides DRM with information about a user-defined widget class, such as the class record and class name, that is necessary to create widgets of this class.

Format

DWT:REGISTER-CLASS *class-code* *class-name* *create-name* *creator-proc* *class-record*

<i>class-code</i>	DRMType
<i>class-name</i>	STRING
<i>create-name</i>	STRING
<i>creator-proc</i>	DWT:WIDGET
<i>class-record</i>	WidgetClass

Return Value

One of two possible constants whose integer values represent the DRM status:

DWT:DRM-SUCCESS	1
DWT:DRM-FAILURE	0

DWT:REGISTER-DRM-NAMES DRM Function

Registers a vector of names of identifiers or callback routines for access in DRM.

Format

DWT:REGISTER-DRM-NAMES *register-list* *register-count*

<i>register-list</i>	DRMRegisterArgList
<i>register-count</i>	DRMCount

Return Value

One of two possible constants whose integers represent the status as follows:

DWT:DRM-SUCCESS	1
DWT:DRM-FAILURE	0

DWT:REMOVE-ALL-CALLBACKS Intrinsic Function

Removes all callback procedures from a callback list.

Format

DWT:REMOVE-ALL-CALLBACKS *widget* *callback-name*

<i>widget</i>	DWT:WIDGET
---------------	------------

DWT:REMOVE-ALL-CALLBACKS Intrinsic Function

callback-name STRING

Return Value

Unspecified.

DWT:REMOVE-CALLBACK Intrinsic Function

Removes a callback procedure from a callback list.

Format

DWT:REMOVE-CALLBACK *widget callback-name callback client-data*

widget DWT:WIDGET

callback-name STRING

callback XtCallbackProc

client-data INTEGER or alien structure

Return Value

Unspecified.

DWT:REMOVE-CALLBACKS Intrinsic Function

Removes a list of callback procedures from a callback list.

Format

DWT:REMOVE-CALLBACKS *widget callback-name callbacks*

widget DWT:WIDGET

callback-name STRING

callbacks XtCallbackList

Return Value

Unspecified.

DWT:REMOVE-EVENT-HANDLER Intrinsic Function

Removes a previously registered event handler.

Format

DWT:REMOVE-EVENT-HANDLER *widget event-mask other client-proc client-data*

widget DWT:WIDGET

DWT:REMOVE-EVENT-HANDLER Intrinsic Function

<i>event-mask</i>	XtEventMask
<i>other</i>	INTEGER
<i>client-proc</i>	XtEventHandler
<i>client-data</i>	INTEGER or alien structure

Return Value

Unspecified.

DWT:REMOVE-GRAB Intrinsic Function

Removes the redirection of user input to a modal widget.

Format

DWT:REMOVE-GRAB *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

Unspecified.

DWT:REMOVE-INPUT Intrinsic Function

Discontinues a source of input.

Format

DWT:REMOVE-INPUT *input-id*

<i>input-id</i>	XtInputId
-----------------	-----------

Return Value

Unspecified.

DWT:REMOVE-RAW-EVENT-HANDLER Intrinsic Function

Removes a previously registered raw event handler.

Format

DWT:REMOVE-RAW-EVENT-HANDLER *widget event-mask other client-proc client-data*

<i>widget</i>	DWT:WIDGET
<i>event-mask</i>	XtEventMask
<i>other</i>	INTEGER

DWT:REMOVE-RAW-EVENT-HANDLER Intrinsic Function

<i>client-proc</i>	XtEventHandler
<i>client-data</i>	INTEGER or alien structure
Return Value	
Unspecified.	

DWT:REMOVE-TIME-OUT Intrinsic Function

Clears a timeout value.

Format

DWT:REMOVE-TIME-OUT *interval-id*

<i>interval-id</i>	XtIntervalId
--------------------	--------------

Return Value

Unspecified.

DWT:REMOVE-WORK-PROC Intrinsic Function

Removes a work procedure.

Format

DWT:REMOVE-WORK-PROC *work-proc-id*

<i>work-proc-id</i>	XtWorkProcId
---------------------	--------------

Return Value

Unspecified.

DWT:RESIZE-WIDGET Intrinsic Function

Resizes a sibling widget of the child making the geometry request.

Format

DWT:RESIZE-WIDGET *widget width height border-width*

<i>widget</i>	DWT:WIDGET
<i>width</i>	INTEGER
<i>height</i>	INTEGER
<i>border-width</i>	INTEGER

Return Value

Unspecified.

DWT:RESIZE-WINDOW Intrinsic Function

DWT:RESIZE-WINDOW Intrinsic Function

Resizes a child widget that already has the new values of its width, height, and border-width fields.

Format

DWT:RESIZE-WINDOW *widget*

widget DWT:WIDGET

Return Value

Unspecified.

DWT:SCALE High-Level Function

Creates a scale widget that allows an application to display a scale for vernier control of a parameter while displaying the current value and range.

Format

DWT:SCALE *parent-widget name x y width height scale-width scale-height title min-value max-value decimal-points value orientation callback drag-callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>width</i>	INTEGER
<i>height</i>	INTEGER
<i>scale-width</i>	INTEGER
<i>scale-height</i>	INTEGER
<i>title</i>	COMPOUND-STRING
<i>min-value</i>	INTEGER
<i>max-value</i>	INTEGER
<i>decimal-points</i>	INTEGER
<i>value</i>	INTEGER
<i>orientation</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>drag-callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SCALE-CREATE Low-Level Function

DWT:SCALE-CREATE Low-Level Function

Creates a scale widget.

Format

DWT:SCALE-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SCALE-GET-SLIDER High-Level Function

Gets the current value of the slider position displayed in the scale.

Format

DWT:SCALE-GET-SLIDER *widget &OPTIONAL value*

<i>widget</i>	DWT:WIDGET
<i>value</i>	INTEGER

Return Value

The value of the current slider position.

DWT:SCALE-SET-SLIDER High-Level Function

Sets or changes the current value of the slider position displayed in the scale.

Format

DWT:SCALE-SET-SLIDER *widget value*

<i>widget</i>	DWT:WIDGET
<i>value</i>	INTEGER

Return Value

Unspecified.

DWT:SCREEN Intrinsic Function

DWT:SCREEN Intrinsic Function

Returns the screen pointer for the specified widget.

Format

DWT:SCREEN *widget*

widget DWT:WIDGET

Return Value

An object of type CLX:SCREEN representing the screen.

DWT:SCROLL-BAR High-Level Function

Creates a scroll bar widget for the application to display and process scroll bar screen operations.

Format

DWT:SCROLL-BAR *parent-widget name x y width height inc page-inc shown int-value min-value max-value orientation callback help-callback unit-inc-callback unit-dec-callback page-inc-callback page-dec-callback to-top-callback to-bottom-callback drag-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>width</i>	INTEGER
<i>height</i>	INTEGER
<i>inc</i>	INTEGER
<i>page-inc</i>	INTEGER
<i>shown</i>	INTEGER
<i>int-value</i>	INTEGER
<i>min-value</i>	INTEGER
<i>max-value</i>	INTEGER
<i>orientation</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE
<i>unit-inc-callback</i>	CALL-BACK-ROUTINE
<i>unit-dec-callback</i>	CALL-BACK-ROUTINE
<i>page-inc-callback</i>	CALL-BACK-ROUTINE
<i>page-dec-callback</i>	CALL-BACK-ROUTINE
<i>to-top-callback</i>	CALL-BACK-ROUTINE
<i>to-bottom-callback</i>	CALL-BACK-ROUTINE
<i>drag-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SCROLL-BAR-CREATE Low-Level Function

DWT:SCROLL-BAR-CREATE Low-Level Function

Creates a scroll bar widget.

Format

DWT:SCROLL-BAR-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SCROLL-BAR-GET-SLIDER High-Level Function

Retrieves the current size and position parameters of the slider in the scroll bar widget.

Format

DWT:SCROLL-BAR-GET-SLIDER *widget &OPTIONAL value shown inc page-inc*

<i>widget</i>	DWT:WIDGET
<i>value</i>	INTEGER
<i>shown</i>	INTEGER
<i>inc</i>	INTEGER
<i>page-inc</i>	INTEGER

Return Value

Four values:

- An integer representing the *value*.
- An integer representing the *shown*.
- An integer representing the *inc*.
- An integer representing the *page-inc*.

DWT:SCROLL-BAR-SET-SLIDER High-Level Function

DWT:SCROLL-BAR-SET-SLIDER High-Level Function

Sets or changes the current size/position parameters of the slider in the scroll bar widget.

Format

DWT:SCROLL-BAR-SET-SLIDER *widget value shown inc page-inc notify*

<i>widget</i>	DWT:WIDGET
<i>value</i>	INTEGER
<i>shown</i>	INTEGER
<i>inc</i>	INTEGER
<i>page-inc</i>	INTEGER
<i>notify</i>	INTEGER

Return Value

Unspecified.

DWT:SCROLL-WINDOW High-Level Function

Creates a scroll window widget for simple applications in the main window widget work area.

Format

DWT:SCROLL-WINDOW *parent-widget name x y width height*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>width</i>	INTEGER
<i>height</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SCROLL-WINDOW-CREATE Low-Level Function

Creates a scroll window widget for simple applications in the main window widget work area.

Format

DWT:SCROLL-WINDOW-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING

DWT:SCROLL-WINDOW-CREATE Low-Level Function

<i>override-arglist</i>	ArgList
<i>override-argcoun</i> t	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SCROLL-WINDOW-SET-AREAS High-Level Function

Sets up or adds the window region, and the horizontal or vertical scroll bar widgets to the scroll window widget.

Format

DWT:SCROLL-WINDOW-SET-AREAS *widget h-scroll v-scroll work-region*

<i>widget</i>	DWT:WIDGET
<i>h-scroll</i>	DWT:WIDGET
<i>v-scroll</i>	DWT:WIDGET
<i>work-region</i>	DWT:WIDGET

Return Value

Unspecified.

DWT:SELECTION High-Level Function

Creates a selection widget.

Format

DWT:SELECTION *parent-widget name x y title value items itemcount visibleitemcount style default-position callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>title</i>	COMPOUND-STRING
<i>value</i>	COMPOUND-STRING
<i>items</i>	COMPOUND-STRING
<i>itemcount</i>	INTEGER
<i>visibleitemcount</i>	INTEGER
<i>style</i>	INTEGER
<i>default-position</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SELECTION-CREATE Low-Level Function

DWT:SELECTION-CREATE Low-Level Function

Creates a selection widget.

Format

DWT:SELECTION-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SEPARATOR High-Level Function

Creates a separator widget for the application to define a border between items in a display.

Format

DWT:SEPARATOR *parent-widget name x y orientation*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>orientation</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SEPARATOR-CREATE Low-Level Function

Creates a separator widget.

Format

DWT:SEPARATOR-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SEPARATOR-GADGET-CREATE Gadget Function

DWT:SEPARATOR-GADGET-CREATE Gadget Function

Creates a separator gadget.

Format

DWT:SEPARATOR-GADGET-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:SET-ERROR-HANDLER Intrinsic Function

Registers an error procedure to be called on a fatal error condition.

Format

DWT:SET-ERROR-HANDLER *handler*

<i>handler</i>	XtErrorHandler
----------------	----------------

Return Value

Unspecified.

DWT:SET-ERROR-MSG-HANDLER Intrinsic Function

Registers a procedure that is called on a fatal error condition.

Format

DWT:SET-ERROR-MSG-HANDLER *handler*

<i>handler</i>	XtErrorMsgHandler
----------------	-------------------

Return Value

Unspecified.

DWT:SET-KEYBOARD-FOCUS Intrinsic Function

DWT:SET-KEYBOARD-FOCUS Intrinsic Function

Redirects keyword input to a child widget of a composite widget.

Format

DWT:SET-KEYBOARD-FOCUS *subtree descended*

<i>subtree</i>	DWT:WIDGET
<i>descended</i>	DWT:WIDGET

Return Value

Unspecified.

DWT:SET-KEY-TRANSLATOR Intrinsic Function

Registers a key translator.

Format

DWT:SET-KEY-TRANSLATOR *display translator*

<i>display</i>	CLX:DISPLAY
<i>translator</i>	XtKeyProc

Return Value

Unspecified.

DWT:SET-MAPPED-WHEN-MANAGED Intrinsic Function

Changes the value of a widget's mapped-when-managed field.

Format

DWT:SET-MAPPED-WHEN-MANAGED *widget mapped-when-managed*

<i>widget</i>	DWT:WIDGET
<i>mapped-when-managed</i>	INTEGER

Return Value

Unspecified.

DWT:SET-SELECTION-TIMEOUT Intrinsic Function

DWT:SET-SELECTION-TIMEOUT Intrinsic Function

Sets the intrinsics selection timeout.

Format

DWT:SET-SELECTION-TIMEOUT *timeout*

timeout INTEGER

Return Value

Unspecified.

DWT:SET-SENSITIVE Intrinsic Function

Sets the sensitivity state of a widget.

Format

DWT:SET-SENSITIVE *widget sensitive*

widget DWT:WIDGET

sensitive INTEGER

Return Value

Unspecified.

DWT:SET-SUBVALUES Intrinsic Function

Sets the current value of a nonwidget resource associated with a widget instance.

Format

DWT:SET-SUBVALUES *base-addr resource-list resource-count arglist argcoun*

base-addr INTEGER or alien structure

resource-list XtResourceList

resource-count INTEGER

arglist ArgList

argcoun INTEGER

Return Value

Unspecified.

DWT:SET-VALUES Intrinsic Function

DWT:SET-VALUES Intrinsic Function

Modifies the current value of a resource associated with a widget instance.

Format

DWT:SET-VALUES *widget arglist argcoun*t

<i>widget</i>	DWT:WIDGET
<i>arglist</i>	ArgList
<i>argcoun</i> t	INTEGER

Return Value

Unspecified.

DWT:SET-WARNING-HANDLER Intrinsic Function

Registers a procedure to be called on nonfatal error conditions.

Format

DWT:SET-WARNING-HANDLER *handler*

<i>handler</i>	XtErrorHandler
----------------	----------------

Return Value

Unspecified.

DWT:SET-WARNING-MSG-HANDLER Intrinsic Function

Registers a procedure that is called on a nonfatal error condition.

Format

DWT:SET-WARNING-MSG-HANDLER *handler*

<i>handler</i>	XtErrorMsgHandler
----------------	-------------------

Return Value

Unspecified.

DWT:S-TEXT High-Level Function

Creates a simple text widget for the application to display a single or multiline text field. The user can enter and edit text in the field.

Format

DWT:S-TEXT *parent-widget name x y cols rows string-value*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>cols</i>	INTEGER
<i>rows</i>	INTEGER
<i>string-value</i>	STRING

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:S-TEXT-CLEAR-SELECTION High-Level Function

Clears the global selection highlighted in the simple text widget.

Format

DWT:S-TEXT-CLEAR-SELECTION *widget time*

<i>widget</i>	DWT:WIDGET
<i>time</i>	Time

Return Value

Unspecified.

DWT:S-TEXT-CREATE Low-Level Function

Creates a simple text widget.

Format

DWT:S-TEXT-CREATE *parent-widget name override-arglist override-argcoun*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcoun</i>	INTEGER

DWT:S-TEXT-CREATE Low-Level Function

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:S-TEXT-GET-EDITABLE High-Level Function

Gets the current permission information concerning whether the text in the simple text widget may be edited by the user.

Format

DWT:S-TEXT-GET-EDITABLE *widget*

widget DWT:WIDGET

Return Value

A boolean value that is true if the user can edit the text in the simple text widget.

DWT:S-TEXT-GET-MAX-LENGTH High-Level Function

Gets the current maximum allowable length of the text string in the simple text widget.

Format

DWT:S-TEXT-GET-MAX-LENGTH *widget*

widget DWT:WIDGET

Return Value

An integer representing the maximum length of the text widget.

DWT:S-TEXT-GET-SELECTION High-Level Function

Gets the global selection, if any, currently highlighted in the simple text widget.

Format

DWT:S-TEXT-GET-SELECTION *widget selection selection-len*

widget DWT:WIDGET

selection STRING

selection-len INTEGER

Return Value

A string containing the currently selected text in the simple text widget.

DWT:S-TEXT-GET-STRING High-Level Function

DWT:S-TEXT-GET-STRING High-Level Function

Gets the text string from the simple text widget.

Format

DWT:S-TEXT-GET-STRING *widget string-value string-len*

<i>widget</i>	DWT:WIDGET
<i>string-value</i>	STRING
<i>string-len</i>	INTEGER

Return Value

An alien structure containing the entire contents of the text widget as an ASCIZ string.

DWT:S-TEXT-REPLACE High-Level Function

Replaces a portion of the current text string in the simple text widget or inserts a new substring in the text.

Format

DWT:S-TEXT-REPLACE *widget from-pos to-pos string-value*

<i>widget</i>	DWT:WIDGET
<i>from-pos</i>	INTEGER
<i>to-pos</i>	INTEGER
<i>string-value</i>	STRING

Return Value

Unspecified.

DWT:S-TEXT-SET-EDITABLE High-Level Function

Sets the permission information that determines whether the text in the widget may be edited by the user.

Format

DWT:S-TEXT-SET-EDITABLE *widget editable*

<i>widget</i>	DWT:WIDGET
<i>editable</i>	INTEGER

Return Value

Unspecified.

DWT:S-TEXT-SET-MAX-LENGTH High-Level Function

DWT:S-TEXT-SET-MAX-LENGTH High-Level Function

Sets the maximum allowable length of the text string in the simple text widget.

Format

DWT:S-TEXT-SET-MAX-LENGTH *widget max-len*

<i>widget</i>	DWT:WIDGET
<i>max-len</i>	INTEGER

Return Value

Unspecified.

DWT:S-TEXT-SET-SELECTION High-Level Function

Makes specified text in the simple text widget the current global selection and highlights it in the simple text widget.

Format

DWT:S-TEXT-SET-SELECTION *widget first last time*

<i>widget</i>	DWT:WIDGET
<i>first</i>	INTEGER
<i>last</i>	INTEGER
<i>time</i>	Time

Return Value

Unspecified.

DWT:S-TEXT-SET-STRING High-Level Function

Sets the text string in the simple text widget.

Format

DWT:S-TEXT-SET-STRING *widget string-value*

<i>widget</i>	DWT:WIDGET
<i>string-value</i>	STRING

Return Value

Unspecified.

DWT:STRING Compound String Function

See DWT:XTSTRING compound string function.

DWT:SUPERCLASS Intrinsic Function

Identifies the widget's superclass.

Format

DWT:SUPERCLASS *widget*

widget DWT:WIDGET

Return Value

A WidgetClass alien structure representing a pointer to the widget's superclass class structure.

DWT:TOGGLE-BUTTON High-Level Function

Creates a toggle button widget for the application to display screen settable switches for the user.

Format

DWT:TOGGLE-BUTTON *parent-widget name x y label value callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>x y</i>	INTEGER
<i>label</i>	COMPOUND-STRING
<i>value</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:TOGGLE-BUTTON-CREATE Low-Level Function

DWT:TOGGLE-BUTTON-CREATE Low-Level Function

Creates a toggle button widget.

Format

DWT:TOGGLE-BUTTON-CREATE *parent-widget name override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:TOGGLE-BUTTON-GADGET-CREATE Gadget Function

Creates a toggle button gadget.

Format

DWT:TOGGLE-BUTTON-GADGET-CREATE *parent-widget name
override-arglist override-argcount*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcount</i>	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:TOGGLE-BUTTON-GET-STATE High-Level Function

Gets the current state of the toggle button.

Format

DWT:TOGGLE-BUTTON-GET-STATE *widget*

<i>widget</i>	INTEGER
---------------	---------

Return Value

An INTEGER representing the value attribute of the toggle button.

DWT:TOGGLE-BUTTON-SET-STATE High-Level Function

DWT:TOGGLE-BUTTON-SET-STATE High-Level Function

Sets or changes the current state of the toggle button.

Format

DWT:TOGGLE-BUTTON-SET-STATE *widget value notify*

<i>widget</i>	DWT:WIDGET
<i>value</i>	INTEGER
<i>notify</i>	INTEGER

Return Value

Unspecified.

DWT:TOOLKIT-INITIALIZE Intrinsic Function

Initializes the XUI Toolkit internals.

Format

DWT:TOOLKIT-INITIALIZE

Return Value

Unspecified.

DWT:TRANSLATE-COORDS Intrinsic Function

Translates X and Y widget coordinates to root coordinates.

Format

DWT:TRANSLATE-COORDS *widget x y*

<i>widget</i>	DWT:WIDGET
<i>x y</i>	INTEGER

Return Value

Two values:

- An INTEGER representing the root-relative X coordinate.
- An INTEGER representing the root-relative Y coordinate.

DWT:TRANSLATE-KEYCODE Intrinsic Function

DWT:TRANSLATE-KEYCODE Intrinsic Function

Invokes the currently registered KeyCode to the KeySym translator.

Format

DWT:TRANSLATE-KEYCODE *display keycode modifiers modifiers-return keysym-return*

<i>display</i>	CLX:DISPLAY
<i>keycode</i>	KeyCode
<i>modifiers</i>	Modifiers
<i>modifiers-return</i>	Modifiers
<i>keysym-return</i>	KeySym

Return Value

Unspecified.

DWT:UNDO-COPY-TO-CLIPBOARD Cut-and-Paste Function

Deletes the last item placed on the clipboard.

Format

DWT:UNDO-COPY-TO-CLIPBOARD *display windowwidget*

<i>display</i>	CLX:DISPLAY
<i>windowwidget</i>	DWT:WIDGET

Return Value

One of two possible constants whose integers represent the status as follows:

DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-LOCKED	4

DWT:UNINSTALL-TRANSLATIONS Intrinsic Function

Removes all existing translations.

Format

DWT:UNINSTALL-TRANSLATIONS *widget*

<i>widget</i>	DWT:WIDGET
---------------	------------

Return Value

Unspecified.

DWT:UNMANAGE-CHILD Intrinsic Function

DWT:UNMANAGE-CHILD Intrinsic Function

Removes a single child widget from its parent's set of managed children.

Format

DWT:UNMANAGE-CHILD *widget*

widget DWT:WIDGET

Return Value

Unspecified.

DWT:UNMANAGE-CHILDREN Intrinsic Function

Removes a list of children from the parent widget's managed list.

Format

DWT:UNMANAGE-CHILDREN *widget-list num-children*

widget-list WidgetList

num-children INTEGER

Return Value

Unspecified.

DWT:UNREALIZE-WIDGET Intrinsic Function

Destroys the windows associated with a widget and its descendants.

Format

DWT:UNREALIZE-WIDGET *widget*

widget DWT:WIDGET

Return Value

Unspecified.

DWT:VMS-CLEAR-STRING Convenience Function

DWT:VMS-CLEAR-STRING Convenience Function

Frees a string in an argument list.

Format

DWT:VMS-CLEAR-STRING *arglist argcount*

<i>arglist</i>	ArgList
<i>argcount</i>	INTEGER

Return Value

Unspecified.

DWT:VMS-FREE-ARGNAMES Convenience Function

Frees memory allocated for argument names.

Format

DWT:VMS-FREE-ARGNAMES *arglist argcount*

<i>arglist</i>	ArgList
<i>argcount</i>	INTEGER

Return Value

Unspecified.

DWT:VMS-SET-ARG Convenience Function

Places an argument in the argument list.

Format

DWT:VMS-SET-ARG *arg arglist argnumber argname*

<i>arg</i>	ALIEN-STRUCTURE
<i>arglist</i>	ArgList
<i>argnumber</i>	INTEGER
<i>argname</i>	STRING

Return Value

Unspecified.

DWT:VMS-SET-CALLBACK-ARG Convenience Function

DWT:VMS-SET-CALLBACK-ARG Convenience Function

Places a callback in the argument list.

Format

DWT:VMS-SET-CALLBACK-ARG *callback-arg arglist argnum argname*

<i>callback-arg</i>	CALL-BACK-ROUTINE
<i>arglist</i>	ArgList
<i>argnum</i>	INTEGER
<i>argname</i>	STRING

Return Value

Unspecified.

DWT:VMS-SET-DESC-ARG Convenience Function

Places a descriptor in the argument list.

Format

DWT:VMS-SET-DESC-ARG *arg arglist argnum argname*

<i>arg</i>	STRING
<i>arglist</i>	ArgList
<i>argnum</i>	INTEGER
<i>argname</i>	STRING

Return Value

Unspecified.

DWT:WARNING Intrinsic Function

See DWT:XTWARNING intrinsic function.

DWT:WARNING-MSG Intrinsic Function

Calls the installed high-level warning handler.

Format

DWT:WARNING-MSG *resource-name resource-type resource-class default-string
params num-params*

DWT:WARNING-MSG Intrinsic Function

<i>resource-name</i>	STRING
<i>resource-type</i>	STRING
<i>resource-class</i>	STRING
<i>default-string</i>	STRING
<i>params</i>	STRING
<i>num-params</i>	INTEGER

Return Value

Unspecified.

DWT:WIDGET-TO-APPLICATION-CONTEXT Intrinsic Function

Returns the application context for a given widget.

Format

DWT:WIDGET-TO-APPLICATION-CONTEXT *widget*

widget DWT:WIDGET

Return Value

An XtAppContext integer representing the application context.

DWT:WINDOW Intrinsic Function

Returns the window of the specified widget.

Format

DWT:WINDOW *widget*

widget DWT:WIDGET

Return Value

An object of type CLX:WINDOW representing the widget's window.

DWT:WINDOW-CREATE Low-Level Function

Creates a window widget.

Format

DWT:WINDOW-CREATE *parent-widget* *name* *override-arglist* *override-argc*

parent-widget DWT:WIDGET
name STRING

DWT:WINDOW-CREATE Low-Level Function

<i>override-arglist</i>	ArgList
<i>override-argcoun</i> t	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:WINDOW-TO-WIDGET Intrinsic Function

Translates a window and display pointer into a widget instance.

Format

DWT:WINDOW-TO-WIDGET *display window*

<i>display</i>	CLX:DISPLAY
<i>window</i>	CLX:WINDOW

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:WORK-BOX High-Level Function

Creates a work box widget for the application to display work in progress messages.

Format

DWT:WORK-BOX *parent-widget name default-position x y style label cancel-label use-percent-scale callback help-callback*

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>default-position</i>	INTEGER
<i>x y</i>	INTEGER
<i>style</i>	INTEGER
<i>label</i>	COMPOUND-STRING
<i>cancel-label</i>	COMPOUND-STRING
<i>use-percent-scale</i>	INTEGER
<i>callback</i>	CALL-BACK-ROUTINE
<i>help-callback</i>	CALL-BACK-ROUTINE

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:WORK-BOX-CREATE Low-Level Function

DWT:WORK-BOX-CREATE Low-Level Function

Creates a work box widget.

Format

DWT:WORK-BOX-CREATE *parent-widget name override-arglist override-argcoun*t

<i>parent-widget</i>	DWT:WIDGET
<i>name</i>	STRING
<i>override-arglist</i>	ArgList
<i>override-argcoun</i> t	INTEGER

Return Value

An object of type DWT:WIDGET representing the new widget.

DWT:XERROR Intrinsic Function

Calls the installed fatal error procedure and passes the message specified.

Format

DWT:XERROR *message*

<i>message</i>	STRING
----------------	--------

Return Value

Unspecified.

DWT:XTSTRING Compound String Function

Creates a compound string.

Format

DWT:XTSTRING *text charset dir-r-to-l*

<i>text</i>	STRING
<i>charset</i>	INTEGER
<i>dir-r-to-l</i>	INTEGER

Return Value

An object of type COMPOUND-STRING representing the resulting compound string. A null pointer is returned if the text is null.

DWT:XTWARNING Intrinsic Function

DWT:XTWARNING Intrinsic Function

Calls the installed nonfatal error procedure.

Format

DWT:XTWARNING *message*

message STRING

Return Value

Unspecified.

புதுக்கால போக்குவரத்து

வினாக்கள் மற்றும் விடைகள்

பொது வினாக்கள் மற்றும் விடைகள்

காலாட்சி நிலைமை

உயர் போக்குவரத்து

நிதான நிலைமை

பொது வினாக்கள்

நிதான வினாக்கள்

நிதான விடைகள்

நிதான விடைகள்

நிதான விடைகள்

நிதான விடைகள்

நிதான விடைகள்

Part III

Guide to Programming CLX

2005

1000 ~~1000~~ ~~1000~~

Chapter 4

Programming Overview of CLX

VAX LISP provides an implementation of the Common LISP X (CLX) interface to the X Window System™ protocol. This chapter provides an overview of CLX, a description of CLX data types, and an introductory CLX program. The program is annotated with references to relevant programming descriptions in other chapters of this guide.

4.1 Overview of CLX

CLX is a package of LISP routines that give LISP programmers access to the capabilities of the X Window System (X) without having to call out to external routines or define non-LISP data structures. X was developed by Project Athena and the Laboratory for Computer Science at the Massachusetts Institute of Technology with funding and participation by Digital Equipment Corporation and International Business Machines, Inc. VAX LISP Version 3.0 supports X Version 11, Release 3 (X11R3). CLX runs on all VAX hardware except the VAX 11/725 and the MicroVAX I.

The program that controls devices such as screens, keyboards, and mice is called the "server." X servers run on computers with either monochrome or color bitmap screens (workstations). The server distributes user input to and accepts output requests from "client" (application) programs. The server returns information to clients by means of either replies or events. Replies are returned synchronously but events are returned asynchronously.

CLX processes some client requests, such as queries about the characteristics of windows, within the CLX library. It sends other client requests, such as those pertaining to displaying graphics on a screen or receiving device input, to the server. The server may or may not reside on the same system as the client and CLX. In the latter case, CLX communicates with the server via DECnet or TCP/IP.

CLX, like Xlib, enables clients to communicate with the X server to create and manage operations with windows, graphics, cursors, fonts and text, pixmaps and offscreen images, and events. The Xlib documentation explains the basic principles of the X Window System, such as the window hierarchy, the coordinate system, and so on. This manual assumes that you are familiar with these concepts and documents primarily the differences between CLX and Xlib. The Xlib manuals are:

- VMS DECwindows Guide to Xlib Programming: MIT C Binding

TM X Window System is a trademark of the Massachusetts Institute of Technology.

- *VMS DECwindows Xlib Routines Reference Manual*
- *X Window System: C Library and Protocol Reference*

All VAX LISP objects implementing CLX are in package CLX: and are exported. This package is part of the LISP executable image. It is installed when VAX LISP is installed. Some sample programs are included in the VAX LISP example directory, which can be referred to by the logical name LISP\$EXAMPLES.

4.2 CLX Data Types

The CLX specification defines numerous data types. These are listed in Table 4-1 in alphabetical order.

Table 4-1: CLX Data Types

Type Specifier	Representation
CLX:ALIST	LIST An association list, with elements of the form: <i>key-type-and-name datum-type-and-name</i>
CLX:ANGLE	A number between (* -2 pi) and (* 2 pi), inclusive. Note that this angle representation is different from the one actually transmitted in the protocol.
CLX:ARC-SEQ	(CLX:REPEAT-SEQ (FIXNUM X) (FIXNUM Y) (FIXNUM WIDTH) (FIXNUM HEIGHT) (CLX:ANGLE ANGLE1) (CLX:ANGLE ANGLE2))
CLX:ARRAY-INDEX	An integer between 0 and ARRAY-DIMENSION-LIMIT, inclusive.
CLX:BIT-GRAVITY	(MEMBER :FORGET :STATIC :CENTER :NORTH :NORTH-EAST :EAST :SOUTH-EAST :SOUTH :SOUTH-WEST :WEST :NORTH-WEST)
CLX:BITMAP	(ARRAY BIT (* *))
CLX:BITMAP-FORMAT	LISP structure.
CLX:BOOLEAN	(OR NULL (NOT NULL))
CLX:CARD8	FIXNUM
CLX:CARD16	FIXNUM
CLX:CARD29	INTEGER
CLX:COLOR	VAX LISP alien structure.
CLX:COLORMAP	LISP structure.
CLX:DEVICE-EVENT-MASK	(OR CLX:MASK32 (LIST CLX:DEVICE-EVENT-MASK-CLASS))
CLX:DEVICE-EVENT-MASK-CLASS	(MEMBER :KEY-PRESS :KEY-RELEASE :BUTTON-PRESS :BUTTON-RELEASE :POINTER-MOTION :BUTTON-1-MOTION :BUTTON-2-MOTION :BUTTON-3-MOTION :BUTTON-4-MOTION :BUTTON-5-MOTION :BUTTON-MOTION)
CLX:DISPLAY	VAX LISP alien structure.
CLX:DRAWABLE	(OR CLX:WINDOW CLX:PIXMAP)
CLX:DRAW-DIRECTION	(MEMBER :LEFT-TO-RIGHT :RIGHT-TO-LEFT)

(continued on next page)

Table 4–1 (Cont.): CLX Data Types

Type Specifier	Representation
CLX:EVENT-KEY	(MEMBER :KEY-PRESS :KEY-RELEASE :BUTTON-PRESS :BUTTON-RELEASE :MOTION-NOTIFY :ENTER-NOTIFY :LEAVE-NOTIFY :FOCUS-IN :FOCUS-OUT :KEYMAP-NOTIFY :EXPOSURE :GRAPHICS-EXPOSURE :NO-EXPOSURE :VISIBILITY-NOTIFY :CREATE-NOTIFY :DESTROY-NOTIFY :UNMAP-NOTIFY :MAP-NOTIFY :MAP-REQUEST :REARENT-NOTIFY :CONFIGURE-NOTIFY :GRAVITY-NOTIFY :RESIZE-REQUEST :CONFIGURE-REQUEST :CIRCULATE-NOTIFY :CIRCULATE-REQUEST :PROPERTY-NOTIFY :SELECTION-CLEAR :SELECTION-REQUEST :SELECTION-NOTIFY :COLORMAP-NOTIFY :CLIENT-MESSAGE)
CLX:EVENT-MASK	(OR CLX:MASK32 (LIST CLX:EVENT-MASK-CLASS))
CLX:EVENT-MASK-CLASS	(MEMBER :KEY-PRESS :KEY-RELEASE :OWNER-GRAB-BUTTON :BUTTON-PRESS :BUTTON-RELEASE :ENTER-WINDOW :LEAVE-WINDOW :POINTER-MOTION :POINTER-MOTION-HINT :BUTTON-1-MOTION :BUTTON-2-MOTION :BUTTON-3-MOTION :BUTTON-4-MOTION :BUTTON-5-MOTION :BUTTON-MOTION :EXPOSURE :VISIBILITY-CHANGE :STRUCTURE-NOTIFY :RESIZE-REDIRECT :SUBSTRUCTURE-NOTIFY :SUBSTRUCTURE-REDIRECT :FOCUS-CHANGE :PROPERTY-CHANGE :COLORMAP-CHANGE :KEYMAP-STATE)
CLX:FONT	LISP structure.
CLX:FONTABLE	(OR CLX:STRINGABLE CLX:FONT)
CLX:FONT-PROPS	LIST
	The list contains alternating keywords and integers.
CLX:GCONTEXT-KEY	(MEMBER :FUNCTION :PLANE-MASK :FOREGROUND :BACKGROUND :LINE-WIDTH :LINE-STYLE :CAP-STYLE :JOIN-STYLE :FILL-STYLE :FILL-RULE :ARC-MODE :TILE :STIPPLE :TS-X :TS-Y :FONT :SUBWINDOW-MODE :EXPOSURES :CLIP-X :CLIP-Y :CLIP-MASK :DASH-OFFSET :DASHES)
CLX:GRAB-STATUS	(MEMBER :SUCCESS :ALREADY-GRABBED :FROZEN :INVALID-TIME :NOT-VIEWABLE)
CLX:IMAGE	VAX LISP alien structure.
CLX:IMAGE-DEPTH	(INTEGER 0 32)
CLX:INDEX-SIZE	(MEMBER :DEFAULT 8 16)
CLX:KEYSYM	INTEGER
CLX:LOGICAL-OP	(MEMBER :GX-CLEAR :GX-AND :GX-AND-REVERSE :GX-COPY :GX-AND-INVERTED :GX-NOOP :GX-XOR :GX-OR :GX-NOR :GX-EQUIV :GX-INVERT :GX-OR-REVERSE :GX-COPY-INVERTED :GX-OR-INVERTED :GX-NAND :GX-SET)
	Or a BOOLE constant (see <i>Common LISP: The Language</i>).
CLX:MASK16	FIXNUM

(continued on next page)

Table 4-1 (Cont.): CLX Data Types

Type Specifier	Representation
CLX:MASK32	INTEGER
CLX:MODIFIER-KEY	(MEMBER :SHIFT :LOCK :CONTROL :MOD-1 :MOD-2 :MOD-3 :MOD-4 :MOD-5)
CLX:MODIFIER-MASK	(OR (MEMBER :ANY) CLX:MASK16 (LIST CLX:MODIFIER-KEY))
CLX:PIXARRAY	(ARRAY CLX:PIXEL (* *))
CLX:PIXEL	(UNSIGNED-BYTE 32)
CLX:PIXMAP	LISP structure.
CLX:PIXMAP-FORMAT	VAX LISP alien structure.
CLX:POINTER-EVENT-MASK	(OR CLX:MASK32 (LIST CLX:POINTER-EVENT-MASK-CLASS))
CLX:POINTER-EVENT-MASK-CLASS	(MEMBER :BUTTON-PRESS :BUTTON-RELEASE :ENTER-WINDOW :LEAVE-WINDOW :POINTER-MOTION :POINTER-MOTION-HINT :BUTTON-1-MOTION :BUTTON-2-MOTION :BUTTON-3-MOTION :BUTTON-4-MOTION :BUTTON-5-MOTION :BUTTON-MOTION :KEYMAP-STATE)
CLX:POINT-SEQ	(CLX:REPEAT-SEQ (CLX:CARD16 X) (CLX:CARD16 Y))
CLX:RECT-SEQ	(CLX:REPEAT-SEQ (CLX:CARD16 X) (CLX:CARD16 Y) (CLX:CARD16 WIDTH) (CLX:CARD16 HEIGHT))
CLX:REPEAT-SEQ	SEQUENCE A sequence containing zero or more elements of the form (<i>type name</i>).
CLX:RESOURCE-ID	INTEGER
CLX:RGB-VAL	(FLOAT 0.0 1.0)
CLX:SCREEN	VAX LISP alien structure.
CLX:SEG-SEQ	(CLX:REPEAT-SEQ (FIXNUM X1) (FIXNUM Y1) (FIXNUM X2) (FIXNUM Y2))
CLX:STATE-MASK-KEY	(OR CLX:MODIFIER-KEY (MEMBER :BUTTON-1 :BUTTON-2 :BUTTON-3 :BUTTON-4 :BUTTON-5))
CLX:STRINGABLE	(OR STRING SYMBOL)
CLX:TIMESTAMP	(OR NULL INTEGER) NIL stands for the current time.
CLX:VISUAL	CLX:CARD29
CLX:VISUAL-INFO	VAX LISP alien structure.
CLX:WINDOW	LISP structure.
CLX:WIN-GRAVITY	(MEMBER :FORGET :STATIC :CENTER :NORTH :NORTH-EAST :EAST :SOUTH-EAST :SOUTH :SOUTH-WEST :WEST :NORTH-WEST)
CLX:XATOM	(OR STRING SYMBOL)

CLX structures (such as CLX:DISPLAY, CLX:FONT, CLX:WINDOW, and so on) are described in Chapters 5 through 13.

4.3 A Sample CLX Program

The sample program in Example 4-1 shows how to open a display, create windows, display text, draw graphics, and handle events.

Example 4-1: Sample CLX Program

```
;;; This is a CLX transcription of the sample Xlib program in the "VMS DECwindows
;;; Guide to Xlib Programming: MIT C Binding." Some utility functions that are
;;; not strictly part of the example precede the transcription.

;;; The following functions avoid repeatedly closing and reopening the display.
;;; DISPLAY-OPEN could be enhanced to check that *DISPLAY* actually contains a
;;; display and also close the old display and open a new one if a different host
;;; is specified.

(defvar *display* nil)
(defvar font-name "-ADOLE-NEW CENTURY SCHOOLBOOK-MEDIUM-R-NORMAL--*-140-*-*-*")
(defvar window-name "Sample CLX Program")

(defun display-open (&optional (host ""))
  (or *display*
    ① (setq *display* (clx:open-display host)) ) )

(defun display-close ()
  (unless (null *display*)
    (setq *display* (clx:close-display *display*)) ) )

;; This helper function is part of the Xlib example
(defun define-color (screen visual color)
  (case (clx:visual-info-class visual)
    (:pseudo-color :direct-color)
    ⑥ (clx:alloc-color
        (clx:screen-default-colormap screen)
        (svref '#("DarkSlateBlue" "LightGray" "Firebrick") color) ) )
  (otherwise
    (case color
      ((0 2) (clx:screen-black-pixel screen))
      (1 (clx:screen-white-pixel screen)) ) ) )

;; Transcription of Xlib Programming Guide example

(defun xlib-example (&optional (host ""))
  "XLIB-EXAMPLE &OPTIONAL (host "")"
A CLX transcription of the sample Xlib program in the VMS DECwindows Guide to
Xlib Programming: MIT C Binding. If a display has already been opened by
DISPLAY-OPEN, it will be used. The display will be left open upon exit."
```

(continued on next page)

Example 4-1 (Cont.): Sample CLX Program

```
(let ((display (display-open host))
      window-1
      window-2
      gc )
  (let* ((screen (first (clx:display-roots display)))
         (depth (clx:screen-root-depth screen))
         (visual (clx:make-visual-info
                  :data (clx:screen-root-visual screen) )))
    (window-1w 400)
    (window-1h 300)
    (window-1x (floor (- (clx:screen-width screen) window-1w) 2))
    (window-1y (floor (- (clx:screen-height screen) window-1h) 2))
    (window-2w 300)
    (window-2h 150)
    (window-2x 50)
    (window-2y 75)
    (font (clx:open-font display font-name ))
    (event-mask (clx:make-event-mask :exposure :button-press)) )
  ③ (setq window-1 (clx:create-window
                      :parent (clx:screen-root screen)
                      :x window-1x :y window-1y
                      :width window-1w :height window-1h
                      :depth depth :border-width 0
                      :class :input-output :visual visual
                      :background (define-color screen visual 0)
                      :event-mask event-mask ))
  (setq window-2 (clx:create-window
                  :parent window-1
                  :x window-2x :y window-2y
                  :width window-2w :height window-2h
                  :depth depth :border-width 4
                  :class :input-output :visual visual
                  :background (define-color screen visual 1)
                  :event-mask event-mask ))
  ④ (setq gc (clx:create-gcontext :drawable window-2
                                    :foreground (define-color screen visual 2)
                                    :background (define-color screen visual 1)
                                    :font font ))
  ⑤
  ⑦ (setf (clx:wm-normal-hints window-1)
          (clx:make-wm-size-hints :x 362 :y 282 :width 400 :height 300) )
  (setf (clx:wm-name window-1) window-name)
  ⑧ (clx:map-window window-1)
  (clx:map-window window-2)

  ;;The event loop -- the core of an Xlib program
  ⑨ (clx:event-case (display)
      (:button-press (window)
        (if (eq window window-1)
            (progn
              (clx:draw-image-glyphs window-2 gc 75 75 "Click HERE to exit")
              nil )
            t ) )
      (:exposure (window)
        (if (eq window window-2)
            (clx:draw-image-glyphs window-2 gc 75 75 "Click here to exit") )
        nil) ) )
```

(continued on next page)

Example 4-1 (Cont.): Sample CLX Program

```
10  (clx:destroy-window window-1)
 2   (clx:display-force-output display) ) )
```

- ➊ The CLX:OPEN-DISPLAY function establishes a connection between the client (the LISP program) and the X server, and returns a CLX:DISPLAY object. See Chapter 5 for more information on displays and screens.
- ➋ CLX buffers client requests and sends them to the server asynchronously. The CLX:DISPLAY-FORCE-OUTPUT function flushes the buffer, forcing the requested output to be displayed.
- ➌ The keyword arguments to the CLX:CREATE-WINDOW function specify the characteristics of the CLX:WINDOW object returned by the function. CLX:CREATE-WINDOW does *not* cause the window actually to appear on the display. See Chapter 6 for more information on windows.
- ➍ The CLX:CREATE-GCONTEXT function returns a CLX:GCONTEXT object with the specified graphics characteristics. See Chapter 7 for details on graphics contexts.
- ➎ The :FONT slot of the CLX:GCONTEXT object determines which font the program uses to write text in WINDOW-2. See Chapter 11 for more information on drawing text. For a list of font names, see the *VMS DECwindows Guide to Xlib Programming: MIT C Binding*.
- ➏ The CLX:ALLOC-COLOR function uses the predefined DECwindows colors named DarkSlateBlue, LightGray, and Firebrick. For more information on colors, see Chapter 8. For a list of predefined color names, see the *VMS DECwindows Guide to Xlib Programming: MIT C Binding*.
- ➐ These functions provide hints to the window manager (that is, DECwindows) as to what size and title the client would like its window to have. See Chapter 6 for more information on window management functions.
- ➑ The CLX:MAP-WINDOW function places a MapRequest event in the server queue; it does not actually make the window visible on the screen. See the following steps and Chapter 6 for more information.
- ➒ The CLX:EVENT-CASE macro loops until events occur (that is, events of the types requested by the client in the :EVENT-CASE slot of a window). The CLX:MAP-WINDOW functions place :EXPOSURE events in the queue, so that clause is executed first. Chapter 12 has more information on events.
- ➓ The CLX:CLOSE-DISPLAY function unmaps the window(s) and ends the connection between the client and the server. See Chapter 5 for details.

C

C

C

C

C

Managing the Client-Server Connection

Before your client program can use a display, you must establish a connection to the X server driving that display. Once you have established this connection, you can use the CLX macros and functions discussed in this chapter to get information about the display. This chapter explains how to perform the following operations:

- Open (connect to) the display.
- Get information about the display and screen.
- Send requests to the server.
- Close (disconnect from) the display.

The chapter concludes with a general discussion of what happens when the connection to the X server is closed.

5.1 Opening the Display

The first call a client makes is to open a display. After opening a display, the client can get information from and send requests to the server. To increase the efficiency of the client-server connection, CLX buffers client requests.

The CLX:OPEN-DISPLAY function establishes a connection from the client to the X server driving the specified display, and returns a CLX:DISPLAY object. The format of the CLX:OPEN-DISPLAY function is:

`CLX:OPEN-DISPLAY host &KEY :DISPLAY display :PROTOCOL system`

host A string that specifies the name of the host machine.
display An integer that specifies the number of the display. The default is 0.
system A keyword that specifies the type of network, either :DECNET or TCP. The default is :DECNET.

The default value of zero for the *display* argument corresponds to the default screen on the specified *host*. If the call to CLX:OPEN-DISPLAY is successful, the function returns a CLX:DISPLAY object; if not, it returns NIL. The function may fail if the host is unknown or unreachable, or if you do not have access to the host.

The following example shows how to establish a connection to the server on a machine named SMITH and obtain some information about the display and screen.

```

Lisp> (setf display (clx:open-display "SMITH"))
#<XDisplay 10485760>
Lisp> (setf screen (clx:display-default-screen display))
#<Screen 22120368>
Lisp> (clx:display-vendor-name display)
"DECWINDOWS DigitalEquipmentCorp."
Lisp> (clx:screen-width-in-millimeter screen)
346

```

The functions that return information about CLX:DISPLAY and CLX:SCREEN objects are discussed in Section 5.2.

5.2 Getting Information About the Display and Screen

After opening a display, clients can get information about the display itself, the screens the server is driving on that display, and the format of images created on those screens. The information functions are listed in Tables 5-1 and 5-2.

Information functions are useful for supplying arguments to other CLX routines. See Part IV for the syntax of information functions. For examples and descriptions of the use of information functions, see the examples throughout Part III.

Table 5-1: Information Functions for DISPLAY Objects

Function	Use with SETF?	Return Value
CLX:DISPLAY-BITMAP-FORMAT	No	The default CLX:BITMAP-FORMAT object for the specified display.
CLX:DISPLAY-DEFAULT-SCREEN	No	The default CLX:SCREEN object for the display.
CLX:DISPLAY-ERROR-HANDLER	Yes	TBS.
CLX:DISPLAY-IMAGE-LSB-FIRST-P	No	T if the byte order for images is least significant bit first or NIL if it is most significant bit first.
CLX:DISPLAY-MAX-KEYCODE	No	An integer that indicates the largest keycode value transmitted by the X server.
CLX:DISPLAY-MAX-REQUEST-LENGTH	No	An integer that indicates the maximum length of a request accepted by the X server, in 4-byte units.
CLX:DISPLAY-MIN-KEYCODE	No	An integer that indicates the smallest keycode value transmitted by the X server.
CLX:DISPLAY-MOTION-BUFFER-SIZE	No	An integer that indicates the approximate size of the X server's buffer containing the recent history of pointer motion.
CLX:DISPLAY-PIXMAP-FORMATS	No	A list of CLX:PIXMAP-FORMAT values that correspond to the depth values of the default screen.
CLX:DISPLAY-PROTOCOL-MAJOR-VERSION	No	An integer that indicates the version number of the X protocol.
CLX:DISPLAY-PROTOCOL-MINOR-VERSION	No	An integer that indicates the release number of the X protocol.
CLX:DISPLAY-RELEASE-NUMBER	No	An integer that indicates the release number of the X server.

(continued on next page)

Table 5–1 (Cont.): Information Functions for DISPLAY Objects

Function	Use with SETF?	Return Value
CLX:DISPLAY-ROOTS	No	A list of CLX:SCREEN objects that represent every screen attached to the display.
CLX:DISPLAY-VENDOR-NAME	No	A string that contains the name of the implementor of the X server.

Table 5–2: Information Functions for CLX:SCREEN Objects

Function	Use with SETF?	Return Value
CLX:SCREEN-BACKING-STORES	No	A keyword that specifies when the server saves the contents of obscured windows: :WHEN-MAPPED, :NOT-USEFUL, or :ALWAYS.
CLX:SCREEN-BLACK-PIXEL	No	The CLX:PIXEL value that produces black on this screen.
CLX:SCREEN-DEFAULT-COLORMAP	No	The default CLX:COLORMAP of the specified screen.
CLX:SCREEN-DEPTHES	No	The depths supported by the display hardware, in an association list whose elements have the form: <i>(depth visuals)</i> where <i>depth</i> is a CLX:IMAGE-DEPTH value and <i>visuals</i> is a list of CLX:VISUAL-INFO.
CLX:SCREEN-EVENT-MASK-AT-OPEN	No	A CLX:EVENT-MASK value that contains the event types requested in the screen's root window.
CLX:SCREEN-HEIGHT	No	An integer that represents the height of the screen in pixels.
CLX:SCREEN-HEIGHT-IN-MILLIMETERS	No	An integer that represents the height of the screen in millimeters.
CLX:SCREEN-MAX-INSTALLED-MAPS	No	An integer that represents the maximum number of colormaps that can be installed.
CLX:SCREEN-MIN-INSTALLED-MAPS	No	An integer that represents the number of colormaps that can be guaranteed to be installed, regardless of the number of entries allocated in each map.
CLX:SCREEN-ROOT	No	The root window of the specified screen.
CLX:SCREEN-ROOT-DEPTH	No	A CLX:IMAGE-DEPTH value that represents the number of planes of the specified screen.
CLX:SCREEN-ROOT-VISUAL	No	An integer pointer to the default visual type of the screen.
CLX:SCREEN-SAVE-UNDERS-P	No	T if the server saves the contents of windows that the client windows obscure; otherwise NIL.
CLX:SCREEN-WHITE-PIXEL	No	The CLX:PIXEL value that produces white on this screen.
CLX:SCREEN-WIDTH	No	An integer that represents the width of the screen in pixels.

(continued on next page)

Table 5–2 (Cont.): Information Functions for CLX:SCREEN Objects

Function	Use with SETF?	Return Value
CLX:SCREEN-WIDTH-IN-MILLIMETERS	No	An integer that represents the width of the screen in millimeters.

5.3 Sending Requests to the Server

CLX normally buffers client requests instead of sending individual requests to the server as they are specified. This increases the efficiency of client-to-server communication. The functions and macro listed in Table 5–3 allow you to control how requests are output from the buffer.

Table 5–3: Output Buffer Routines

Routine	Description
CLX:DISPLAY-FORCE-OUTPUT	Flushes the buffer, that is, forces all requested output to be displayed.
CLX:DISPLAY-FINISH-OUTPUT	Flushes the buffer and waits until the server has received and processed all events, including errors.
CLX:DISPLAY-AFTER-FUNCTION	Specifies the function the client calls after processing any protocol request.
CLX:WITH-DISPLAY	Causes the server to process requests synchronously.

The CLX:DISPLAY-AFTER-FUNCTION function can be used with SETF. While debugging a program, you may wish to set this function to CLX:DISPLAY-FINISH-OUTPUT so that error messages will appear immediately after the request that caused them. If you want to program interactively, set the function to CLX:DISPLAY-FORCE-OUTPUT so that output will be displayed immediately after you request it. For example:

```
(setf (clx:display-after-function display) #'clx:display-force-output)
```

5.4 Closing the Display

CLX automatically destroys windows and resources related to a process when the process exits the server. This means that clients usually do not have to close their connection with a server explicitly. However, clients can explicitly close the connection by using the CLX:CLOSE-DISPLAY function. This function destroys all windows associated with the display and all resources the client has allocated. Its format is:

CLX:CLOSE-DISPLAY *display*

where *display* is a CLX:DISPLAY object that specifies the display to be closed.

References to resources (such as windows) associated with a display that has been closed cause errors.

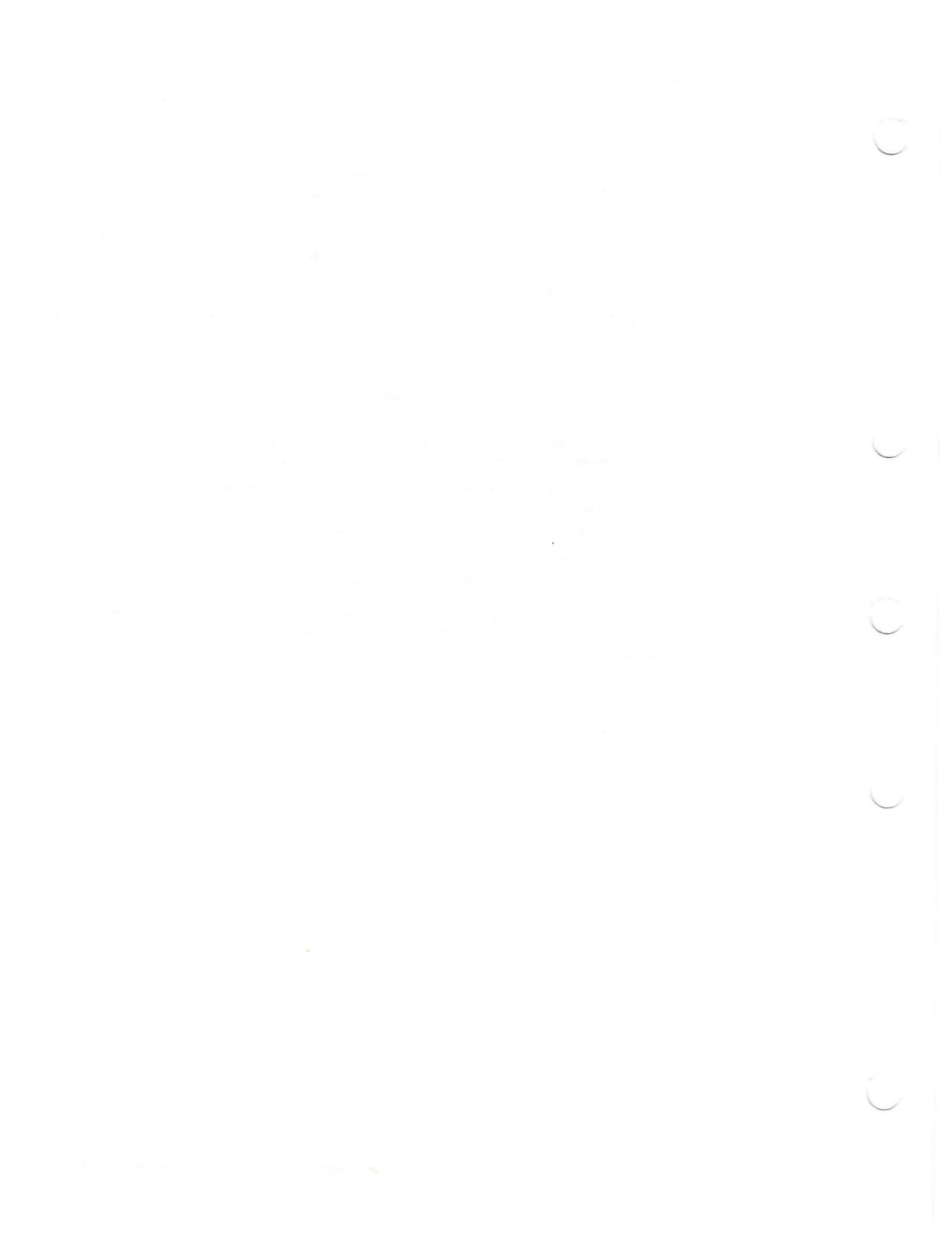
When the X server's connection to a client is closed, either by an explicit call to CLX:CLOSE-DISPLAY or by a process that exits, the X server does the following:

- Discards all input events selected by the client (see CLX:SET-SELECTION-OWNER in Chapter 12).
- Releases all active and passive grabs made by the client (see CLX:UNGRAB-KEYBOARD, CLX:UNGRAB-POINTER, and CLX:UNGRAB-SERVER in Chapter 12).
- Determines whether to retain or destroy client resources after the display is closed (see CLX:SET-CLOSE-DOWN-MODE in Chapter 13).

If the server is to destroy all client resources, it does so as follows:

- Examines each window in the client's saveset to determine if it is a subwindow, or child, of a window created by the client. (The saveset is a list of other clients' windows; windows in this list are called saveset windows.) If a window is a subwindow, the X server reparents the saveset window to its closest ancestor so that it is no longer a child of a window created by the client.
- Maps the saveset window if it is unmapped. The X server does this even if the saveset window is not a subwindow of a window created by the client.
- Destroys all windows created by the client, after examining each one in the client saveset.
- Frees each nonwindow resource (font, pixmap, cursor, colormap, and GContext) created by the client.
- Frees all colors and colormap entries allocated by the client.

Additional processing occurs when you log out of the process in which you invoked CLX functions. See the *VMS DECwindows Guide to Xlib Programming: MIT C Binding* for details.



Chapter 6

Working with Windows

CLX window routines allow client programs to create multiple windows and define window size, location, and appearance on one or more screens. A window may be input-only (in which case it is never visible on the screen) or input-output. A window is always clipped by its parent.

Conflicts between clients about displaying windows are handled by a window manager, which has final control over the size and placement of windows and, in some cases, characteristics such as title bars and borders. The window manager also keeps clients informed about what it is doing with their windows. For example, the window manager may tell a client that one of its windows has been resized so that the client can reformat information displayed in the window. The default window manager is DECwindows.

This chapter describes the following topics:

- Creating and destroying windows
- Mapping and unmapping windows
- Associating properties with windows
- Communicating with the window manager
- Exchanging properties between clients
- Stacking windows
- Changing window attributes

For more information about X window types, hierarchy, and visibility, see the *VMS DECwindows Guide to Xlib Programming: MIT C Binding*.

6.1 Creating Windows

After opening a display, you can create windows. Note that creating a window does not make it visible on the screen. To be visible, a window must meet the following conditions: it must be input-output; it must be mapped; its ancestors must be mapped; and it must not be completely underneath another window. Mapping windows is explained in Section 6.3.

The `CLX:CREATE-WINDOW` function allows you to specify all the characteristics of a window, or to copy some attributes from the parent window. Its format is:

```
CLX:CREATE-WINDOW &KEY :PARENT :X :Y :WIDTH :HEIGHT :DEPTH :BORDER-WIDTH  
:CLASS :VISUAL :BACKGROUND :BORDER :GRAVITY  
:BIT-GRAVITY :BACKING-STORE :BACKING-PLANES  
:BACKING-PIXEL :SAVE-UNDER :EVENT-MASK
```

:DO-NOT-PROPAGATE-MASK :OVERWRITE-REDIRECT
 :COLORMAP :CURSOR

The values of the keyword arguments specify the attributes of the CLX:WINDOW object returned by the CLX:CREATE-WINDOW function, as described in Table 6-1. Unless otherwise noted, the keywords have no default values.

Table 6-1: CLX:CREATE-WINDOW Keywords

:PARENT	A CLX:WINDOW object that specifies the parent of the window being created.
:X and :Y	Integers that specify the X and Y coordinates of the top-left corner of the window, measured in pixels.
:WIDTH and :HEIGHT	Integers that specify the width and height of the window, measured in pixels, but not including the border.
:DEPTH	An integer that specifies the number of bits per pixel. The default value is 0.
:BORDER-WIDTH	An integer that specifies the width of the window's border, measured in pixels. The default value is 0.
:CLASS	:COPY, :INPUT-OUTPUT, or :INPUT-ONLY, specifying whether the window can display output. The default value is :COPY, which means the same class as the parent window.
:VISUAL	Either a CLX:VISUAL value or :COPY. The default value is :COPY, specifying the same visual information as the parent window.
:BACKGROUND	A CLX:PIXEL value, a CLX:PIXMAPP value, :NONE, :PARENT-RELATIVE, or NIL, specifying the background color.
:BORDER	A CLX:PIXEL value, a CLX:PIXMAPP value, or :COPY, specifying the border color.
:GRAVITY	A CLX:WIN-GRAVITY value that specifies how the window will be positioned after its parent is resized.
:BIT-GRAVITY	A CLX:BIT-GRAVITY value that specifies which region of the window will be retained after it is resized.
:BACKING-STORE	:NOT-USEFUL, :WHEN-MAPPED, :ALWAYS, or NIL, specifying when the server will maintain the contents of the window.
:BACKING-PLANES	A CLX:PIXEL value or NIL, specifying which bit planes of the window hold dynamic data that must be preserved in :BACKING-STORE.
:BACKING-PIXEL	A CLX:PIXEL value or NIL, specifying the value to use in planes not covered by :BACKING-PLANES.
:SAVE-UNDER	:ON or :OFF, specifying whether the server will save the contents of windows obscured by the created window.
:EVENT-MASK	A CLX:EVENT-MASK value or NIL, specifying for which events in this window the client requests notification from the server.
:DO-NOT-PROPAGATE-MASK	A CLX:DEVICE-EVENT-MASK value or NIL, specifying which events should not be propagated to ancestor windows.

(continued on next page)

Table 6-1 (Cont.): CLX:CREATE-WINDOW Keywords

: OVERRIDE-REDIRECT	:ON or :OFF, specifying whether calls to map and configure the window should override a request from another client to redirect those calls.
: COLORMAP	An object of type CLX:COLORMAP, :COPY, or NIL, specifying the colormap that best reflects the true colors of the window.
: CURSOR	An object of type CLX:CURSOR, :NONE, or NIL, specifying the cursor shape to be used when the pointer is in the window.

Information about the display is obtained from the parent window. (See Table 5-1 for a complete list of display information functions.)

Example 6-1 shows the creation, on a machine named SMITH, of a window whose parent is the root; top-left corner is the top-left corner of the screen; width and height are 200 pixels; and border is 1 pixel wide.

Example 6-1: Creating a Window

```
Lisp> (setf *display* (clx:open-display "smith"))
#<XDisplay 13043164>
Lisp> (setf *screen* (clx:display-default-screen *display*))
#<Screen 13302544>
Lisp> (setf *root* (clx:screen-root *screen*))
#<Window 524395>
Lisp> (setf *window* (clx:create-window :parent *root* :x 0 :y 0
                                         :width 200 :height 200
                                         :border-width 1))
#<Window 12582913>
Lisp> (clx:map-window *window*)
Lisp> (clx:display-force-output *display*)
```

The final call to CLX:DISPLAY-FORCE-OUTPUT is necessary to make the window visible.

Because no value was supplied for the :BACKGROUND argument, the window is transparent. The window's title, "No name," is supplied by the DECwindows Session Manager. The DECwindows Session Manager reparents each client window.

6.2 Destroying Windows

Destroying a window frees all storage allocated for that window object. If the window is mapped, the server notifies clients using the window that it has been destroyed. The server also notifies clients of exposure events on windows formerly obscured by destroyed windows. The CLX:DESTROY-WINDOW function destroys a specified window and all its subwindows; the CLX:DESTROY-SUBWINDOWS function destroys all the subwindows of a parent window.

The format of the CLX:DESTROY-WINDOW function is:

CLX:DESTROY-WINDOW *window*

where *window* is the CLX:WINDOW object that is destroyed.

The format of the CLX:DESTROY-SUBWINDOWS function is:

CLX:DESTROY-SUBWINDOWS *window*

where *window* is the CLX:WINDOW object whose subwindows are destroyed.

Normally, all windows belonging to a client are destroyed when the client exits. If your client uses windows created by another client, you can prevent them from being destroyed by adding them to your saveset. The saveset is a list, which the X server maintains for each client, of windows created by other clients which should be remapped rather than destroyed when those clients exit.

The CLX:ADD-TO-SAVE-SET function adds the specified window to your saveset. Its format is:

CLX:ADD-TO-SAVE-SET *window*

where *window* is a CLX:WINDOW object.

The CLX:REMOVE-FROM-SAVE-SET function removes a window from your saveset. Its format is:

CLX:REMOVE-FROM-SAVE-SET *window*

where *window* is a CLX:WINDOW object.

The *window* argument passed to the CLX:ADD-TO-SAVE-SET and CLX:REMOVE-FROM-SAVE-SET functions must have been created by some other client or an error occurs.

6.3 Mapping and Unmapping Windows

Mapping a window, if its ancestors are also mapped, allows it to be viewed on the screen. The window is also visible unless it is totally obscured by other windows. CLX provides the CLX:MAP-WINDOW function for mapping a single window, and the CLX:MAP-SUBWINDOWS function for mapping multiple windows.

The format of CLX:MAP-WINDOW is:

CLX:MAP-WINDOW *window*

where *window* is the CLX:WINDOW object that is mapped.

The format of CLX:MAP-SUBWINDOWS is:

CLX:MAP-SUBWINDOWS *window*

where *window* is the CLX:WINDOW object whose subwindows are mapped.

Unmapping a window removes it from the screen, but leaves the window object itself intact. The CLX:UNMAP-WINDOW function removes a single window; the CLX:UNMAP-SUBWINDOWS function removes all the subwindows of a parent window. See Section 6.2 for information on deallocating storage for window objects.

The format of CLX:UNMAP-WINDOW is:

CLX:UNMAP-WINDOW *window*

where *window* is the CLX:WINDOW object that is unmapped.

The format of CLX:UNMAP-SUBWINDOWS is:

CLX:UNMAP-SUBWINDOWS *window*

where *window* is the CLX:WINDOW object whose subwindows are unmapped.

6.4 Associating Properties with Windows

CLX enables clients to associate additional data with a window. This data is considered a property of the window. Although a property must be data of only one type, it can be stored in 8-bit, 16-bit, and 32-bit formats. The indicators for a property list in CLX are called atoms. Atoms may be passed as strings or symbols and are always returned as keywords. For a list of predefined atoms, see the VMS DECwindows Guide to Xlib Programming: MIT C Binding.

The CLX:CHANGE-PROPERTY function allows you to set a new value for a window property. Its format is:

CLX:CHANGE-PROPERTY *window* *property* *data* *type* *format*
&KEY :MODE :START :END :TRANSFORM

<i>window</i>	The CLX:WINDOW object whose properties are changed.
<i>property</i>	A CLX:XATOM value that specifies the property to change.
<i>data</i>	A sequence that specifies the new property data.
<i>type</i>	A CLX:XATOM value that defines the type of property to be changed.
<i>format</i>	The data format size, either 8, 16, or 32.
:MODE	:REPLACE, :PREPEND, or :APPEND, specifying the manner in which the window's property list is to be changed. The default value is :REPLACE.
:START	An integer that defines the starting position within the window's property list. The default value is 0.
:END	An integer that specifies the last position with the window's property list, or NIL.
:TRANSFORM	A function that is applied to each extracted element.

The values :START and :END affect sub-sequences extracted from *data*.

The CLX:GET-PROPERTY function returns information about a window property. Its format is:

CLX:GET-PROPERTY *window* *property*
&KEY :TYPE :START :END :DELETE-P :RESULT-TYPE :TRANSFORM

<i>window</i>	The CLX:WINDOW object whose property is being queried.
<i>property</i>	A CLX:XATOM value that specifies the property being queried.
:TYPE	A CLX:XATOM value that specifies the type of the property, or NIL.
:START	An integer that defines the starting position within the window's property list. The default value is 0.
:END	An integer that specifies the last position with the window's property list, or NIL.
:DELETE-P	A CLX:BOOLEAN value that determines whether the specified property will be deleted.
:RESULT-TYPE	The type specifier of the value returned by the function. The default type is 'LIST.

:TRANSFORM A function that is applied to each returned integer.

The CLX:GET-PROPERTY function returns a sequence of four values: the property data, its type, its format, and the number of bytes remaining in the property if a partial read operation was performed.

The CLX:LIST-PROPERTIES function returns all the properties defined for a specified window. Its format is:

CLX:LIST-PROPERTIES *window* &**KEY :RESULT-TYPE**

window A CLX:WINDOW object.

:RESULT-TYPE The type specifier of the value returned by the function. The default type is 'LIST'.

The function returns a sequence of keywords naming the properties of the *window* argument. Atoms are always returned as keywords.

The CLX:ROTATE-PROPERTIES function rotates the properties of a specified window and generates a :PROPERTY-NOTIFY event in that window. The format of the CLX:ROTATE-PROPERTIES function is:

CLX:ROTATE-PROPERTIES *window properties* &**OPTIONAL delta**

window The CLX:WINDOW object whose properties are rotated.

properties A sequence of CLX:XATOM values. Each value specifies a property to be rotated.

delta An integer that specifies the direction and amount of rotation. The default is 1.

NOTE

A positive value for *delta* rotates the properties to the left, and a negative value rotates right. This is the opposite of the X protocol.

The CLX:DELETE-PROPERTY function removes a specified property from the property list of a window. Its format is:

CLX:DELETE-PROPERTY *window property*

window The CLX:WINDOW whose properties are deleted.

property A CLX:XATOM that specifies the property to be deleted.

Given an integer resource-id, the CLX:ATOM-NAME function returns an atom name as a keyword. Its format is:

CLX:ATOM-NAME *display atom-id*

display A CLX:DISPLAY object.

atom-id An integer.

Given a keyword, the CLX:FIND-ATOM and CLX:INTERN-ATOM functions return the integer resource-id of that atom. The format of CLX:FIND-ATOM is:

CLX:FIND-ATOM *display name*

display A CLX:DISPLAY object.

name A CLX:XATOM value.

The format of CLX:INTERN-ATOM is:

CLX:INTERN-ATOM *display name*

display A CLX:DISPLAY object.

name A CLX:XATOM value.

Both functions return an integer corresponding to the atom.

6.4.1 Communicating with the Window Manager

CLX provides two structures, CLX:WM-HINTS and CLX:WM-SIZE-HINTS, to enable clients to communicate hints to the window manager about the following window properties:

- Pixmaps used to define icons
- Window and icon names
- Position and size of windows
- Initial state of windows
- Input that windows accept
- Commands used to start the application
- Names used to retrieve application resources

All accessor functions for these two structures are valid SETF places so that you can set individual properties of a window. The functions that return information about window manager properties are described in Tables 6-2 and 6-3. Unless otherwise noted, a null value indicates that the property has not been defined for the window.

Table 6-2: Information Functions for CLX:WM-HINTS Objects

Function Name	Use with SETF?	Return Value
CLX:WM-HINTS-INPUT	Yes	:ON if the client relies on the window manager to get keyboard input; :OFF if it does not.
CLX:WM-HINTS-INITIAL-STATE	Yes	A keyword that defines how the window should appear in its initial configuration. Possible values are: :DONT-CARE Client is not interested in the initial state :NORMAL Initial state used most often :ZOOM Window starts zoomed :ICONIC Window starts as an icon :INACTIVE Window is seldom used
CLX:WM-HINTS-ICON-PIXMAP	Yes	The CLX:PIXMAP value used to create the window icon.
CLX:WM-HINTS-ICON-WINDOW	Yes	The CLX:WINDOW object to be used as an icon.
CLX:WM-HINTS-ICON-X	Yes	An integer that specifies the initial X coordinate of the icon.
CLX:WM-HINTS-ICON-Y	Yes	An integer that specifies the initial Y coordinate of the icon.

(continued on next page)

Table 6–2 (Cont.): Information Functions for CLX:WM-HINTS Objects

Function Name	Use with SETF?	Return Value
CLX:WM-HINTS-ICON-MASK	Yes	A CLX:PIXMAP value that specifies which pixels of the icon pixmap are used to create the icon.
CLX:WM-HINTS-WINDOW-GROUP	Yes	A CLX:RESOURCE-ID that identifies a group of other windows to which the specified window belongs.
CLX:WM-HINTS-FLAGS	Yes	An integer that specifies which elements of the data structure are defined.

The DECwindows window manager overrides most calls to resize and relocate windows. To prevent this, create the window with :OVERWRITE-REDIRECT set to :ON. Most applications, however, can use the CLX:WM-SIZE-HINTS structure to work with the window manager.

Table 6–3: Information Functions for CLX:WM-SIZE-HINTS Objects

Function Name	Use with SETF?	Return Value
CLX:WM-SIZE-HINTS-USER-SPECIFIED-POSITION-P	Yes	T if the user wants the window moved to the specified X and Y position; otherwise NIL. The default value is T.
CLX:WM-SIZE-HINTS-USER-SPECIFIED-SIZE-P	Yes	T if the user wants the window resized; otherwise NIL. The default value is T.
CLX:WM-SIZE-HINTS-X	Yes	An integer that specifies the X coordinate of the window position.
CLX:WM-SIZE-HINTS-Y	Yes	An integer that specifies the Y coordinate of the window position.
CLX:WM-SIZE-HINTS-WIDTH	Yes	An integer that specifies the width of the window.
CLX:WM-SIZE-HINTS-HEIGHT	Yes	An integer that specifies the height of the window.
CLX:WM-SIZE-HINTS-MIN-WIDTH	Yes	An integer that specifies the smallest useful width of the window.
CLX:WM-SIZE-HINTS-MIN-HEIGHT	Yes	An integer that specifies the smallest useful height of the window.
CLX:WM-SIZE-HINTS-MAX-WIDTH	Yes	An integer that specifies the largest useful width of the window.
CLX:WM-SIZE-HINTS-MAX-HEIGHT	Yes	An integer that specifies the largest useful height of the window.
CLX:WM-SIZE-HINTS-WIDTH-INC	Yes	An integer that specifies the increments by which the width of the window can be resized.
CLX:WM-SIZE-HINTS-HEIGHT-INC	Yes	An integer that specifies the increments by which the height of the window can be resized.
CLX:WM-SIZE-HINTS-MIN-ASPECT	Yes	A number that specifies the smallest aspect ratio of the window.
CLX:WM-SIZE-HINTS-MAX-ASPECT	Yes	A number that specifies the largest aspect ratio of the window.

Table 6–4 lists additional window manager property functions defined in CLX. Not all the functions can be used with SETF.

Table 6–4: Additional Window Manager Property Functions

Name	Use with SETF?	Description
CLX:GET-WM-CLASS	No	Returns the class of a specified window.
CLX:ICON-SIZES	Yes	Returns the recommended sizes for a window's icon.
CLX:SET-WM-CLASS	No	Sets the class of a specified window.
CLX:WM-HINTS	Yes	Returns the values for the window manager hints.
CLX:WM-ICON-NAME	Yes	Returns the name of the icon for a specified window.
CLX:WM-NAME	Yes	Returns the name of a window.
CLX:WM-NORMAL-HINTS	Yes	Returns recommended values for the size and location of a window in its most frequent initial state.
CLX:WM-ZOOM-HINTS	Yes	Returns the recommended size and location for a window in the zoomed state.

6.4.2 Exchanging Properties Between Clients

Properties that are global to the server are called selections and may be exchanged between clients. An example is text cut from one window and pasted into another. The text cut in window A is a property owned by client A. Ownership of the property transfers to client B, who then pastes the text into window B.

Clients can determine the owner of a selection by calling the CLX:SELECTION-OWNER function. Its format is:

CLX:SELECTION-OWNER *display selection*

display A CLX:DISPLAY object.

selection The CLX:XATOM value that identifies the selection.

The CLX:SELECTION-OWNER function returns the CLX:WINDOW object that currently owns the specified selection, or NIL if no window owns the selection. Clients establish ownership of a selection by using the SETF form of CLX:SELECTION-OWNER.

The CLX:CONVERT-SELECTION function asks the owner of a selection to convert it to a particular data type. Its format is:

CLX:CONVERT-SELECTION *selection type requestor &OPTIONAL property time*

selection The CLX:XATOM value of the requested selection.

type A CLX:XATOM value.

requestor A CLX:WINDOW object.

property A CLX:XATOM value, or NIL.

time A CLX:TIMESTAMP value.

Clients request selections and notify other clients of them by using events. For information about using events to request and convert selections, and to notify clients of them, see Chapter 12. For style guidelines about making selections, see the *XUI Style Guide*.

6.5 Stacking Windows

CLX provides functions that allow you to perform the following operations:

- Move a window to the top or bottom of the stacking order.
- Find the parent and children of a window.
- Change the parent of a window.

The CLX:CIRCULATE-WINDOW-UP function raises to the top of the stacking order the lowest mapped window (if any) that is obscured by a sibling. Its format is:

CLX:CIRCULATE-WINDOW-UP *window*

where *window* is the CLX:WINDOW object that is placed on the top of the stacking order.

The CLX:CIRCULATE-WINDOW-DOWN function lowers to the bottom of the stacking order the highest mapped window (if any) that obscures a sibling. Its format is:

CLX:CIRCULATE-WINDOW-DOWN *window*

where *window* is the CLX:WINDOW object that is shuffled to the bottom of the stacking order.

With both circulate functions, exposure events are generated in formerly obscured windows.

You can find a window's position in the hierarchy by calling the CLX:QUERY-TREE function. Its format is:

CLX:QUERY-TREE *window* &KEY :RESULT-TYPE

window A CLX:WINDOW object.

:RESULT-TYPE The type specifier of the first value returned by the function. The default type is 'LIST.'

The CLX:QUERY-TREE function returns three values: a sequence of CLX:WINDOW objects, listing the children of the window; the parent window; and the root window.

The CLX:REARENT-WINDOW function changes the parent of a window. Its format is:

CLX:REARENT-WINDOW *window* *parent* *x* *y*

window The CLX:WINDOW object whose parent is changed.

parent The CLX:WINDOW object that is the new parent of *window*.

x and *y* Integers that specify the X and Y coordinates of the top left corner of the window relative to the new parent.

6.6 Changing Window Attributes

Many of the accessor functions for CLX:WINDOW objects may be used with SETF so that you can change a window's attributes after you have created it. The functions that return information about windows are listed in Table 6-5, with an explanation of their return value and whether they can be used with SETF.

Table 6–5: Information Functions for CLX:WINDOW Objects

Function Name	Use with SETF?	Return Value
CLX:WINDOW-ALL-EVENT-MASKS	No	CLX:MASK32.
CLX:WINDOW-BACKING-PIXEL	Yes	A CLX:PIXEL value or NIL, specifying the value to use in planes not covered by :BACKING-PLANES.
CLX:WINDOW-BACKING-PLANES	Yes	A CLX:PIXEL value or NIL, specifying which bit planes hold data that must be preserved in :BACKING-STORE.
CLX:WINDOW-BACKING-STORE	Yes	:NOT-USEFUL, :WHEN-MAPPED, or :ALWAYS.
CLX:WINDOW-BIT-GRAVITY	Yes	A CLX:WIN-GRAVITY value that defines how the contents of the window should be moved when it is resized.
CLX:WINDOW-CLASS	No	The class of the window, either :INPUT-OUTPUT or :INPUT-ONLY.
CLX:WINDOW-COLORMAP	Yes	The CLX:COLORMAP that best reflects the colors of the window, or :COPY if the colormap is copied from the parent.
CLX:WINDOW-COLORMAP-INSTALLED-P	No	A CLX:BOOLEAN value that indicates whether a colormap is installed.
CLX:WINDOW-DISPLAY	No	The CLX:DISPLAY object on which the window appears.
CLX:WINDOW-DO-NOT-PROPAGATE-MASK	Yes	A CLX:MASK32 value that specifies which events should not be propagated to ancestor windows.
CLX:WINDOW-EQUAL	No	A CLX:BOOLEAN value that indicates whether two windows are the same window.
CLX:WINDOW-EVENT-MASK	Yes	A CLX:MASK32 value that defines which types of events occurring in the window should be reported to the client.
CLX:WINDOW-GRAVITY	No	A CLX:WIN-GRAVITY value that defines how the server should reposition the window when its parent is resized.
CLX:WINDOW-ID	No	An integer pointer to the CLX:WINDOW object.
CLX:WINDOW-MAP-STATE	No	The visibility status of the window, one of :UNMAPPED, :UNVIEWABLE, or :VIEWABLE.
CLX:WINDOW-OVERRIDE-REDIRECT	Yes	:ON or :OFF, specifying whether requests to map and configure the window should override a request from another client to redirect those calls.
CLX:WINDOW-P	No	A CLX:BOOLEAN value that indicates whether its argument is a CLX:WINDOW object.
CLX:WINDOW-SAVE-UNDER	Yes	:ON or :OFF, specifying whether the server will save the contents of windows obscured by the created window.
CLX:WINDOW-VISUAL	No	INTEGER.

Table 6–6 lists additional functions that may be used with SETF but that may not be called.

Table 6–6: SETF Functions for CLX:WINDOW Objects

Function Name	Type of Value
CLX:WINDOW-BACKGROUND	Specifies the background color; may be NIL or :NONE, :PARENT-RELATIVE, an integer, or a CLX:PIXMAP value.
CLX:WINDOW-BORDER	An integer, NIL, :COPY, or a CLX:PIXMAP value that defines the window border.
CLX:WINDOW-CURSOR	A CLX:CURSOR object that specifies the cursor shape to be used when the pointer is in the window, or :NONE.
CLX:WINDOW-PRIORITY	:ABOVE, :BELOW, :TOP-IF, :BOTTOM-IF, or :OPPOSITE.

Example 6–2 shows how to change the background color of a window. The variables *WINDOW*, *SCREEN*, and *DISPLAY* were assigned in Example 6–1.

Example 6–2: Changing Window Attributes

```
Lisp> (setf (clx:window-background *window*)
             (clx:screen-black-pixel *screen*))
252
Lisp> (clx:clear-area *window*)
Lisp> (clx:display-force-output *display*)
```

Some components of the CLX:WINDOW structure have no accessor function. These attributes can be identified, and in some cases set, with the functions listed in Table 6–7. (A drawable can be either a window or a pixmap.)

Table 6–7: Information Functions for CLX:DRAWABLE Objects

Function Name	Use with SETF?	Return Value
CLX:DRAWABLE-BORDER-WIDTH	Yes	An integer that represents the width of the drawable's border in pixels.
CLX:DRAWABLE-DEPTH	No	An integer that represents the number of planes in the drawable.
CLX:DRAWABLE-HEIGHT	Yes	An integer that represents the height of the drawable in pixels.
CLX:DRAWABLE-ROOT	No	A CLX:DRAWABLE object that represents the root of the drawable.
CLX:DRAWABLE-WIDTH	Yes	An integer that represents the width of the drawable in pixels.
CLX:DRAWABLE-X	Yes	An integer that represents the X coordinate of the top left corner of the drawable, measured in pixels from the origin of its parent.
CLX:DRAWABLE-Y	Yes	An integer that represents the Y coordinate of the top left corner of the drawable, measured in pixels from the origin of its parent.

The CLX:WITH-STATE macro coerces multiple window queries and changes into as few protocol requests as possible, thereby improving performance. Its format is:

CLX:WITH-STATE *drawable* {*body*}*

drawable The CLX:WINDOW or CLX:PIXMAP object whose attributes are being accessed or set.

body LISP forms to be executed.

The body is not surrounded by a CLX:WITH-DISPLAY macro. For more information, see the description in Part IV.

1937-2-27

— 100 —

— 101 —

— 102 —

— 103 —

— 104 —

— 105 —

— 106 —

— 107 —

— 108 —

— 109 —

— 110 —

— 111 —

— 112 —

— 113 —

— 114 —

— 115 —

Chapter 7

Defining Graphics Characteristics

In CLX, graphics characteristics, such as line width and style, are stored in CLX:GCONTEXT structures. Clients specify a GContext to be used when drawing graphics objects. This chapter describes how to define graphics characteristics prior to drawing objects. The following procedures are covered:

- Creating a graphics context
- Querying and setting GContext components
- Copying and freeing GContexts
- Using GContexts efficiently

Chapter 9 describes how to draw graphics objects. Chapter 11 describes how to display text.

7.1 Creating a Graphics Context

CLX provides default values for all graphics characteristics. Clients create a GContext by calling the CLX:CREATE-GCONTEXT function. For all GContext components, a value of NIL causes the default GContext value to be used.

The format of the CLX:CREATE-GCONTEXT function is:

```
(CLX:CREATE-GCONTEXT &KEY :DRAWABLE :FUNCTION :PLANE-MASK :FOREGROUND  
:BACKGROUND :LINE-WIDTH :LINE-STYLE :CAP-STYLE  
:JOIN-STYLE :FILL-STYLE :FILL-RULE :ARC-MODE :TILE  
:STIPPLE :TS-X :TS-Y :FONT :SUBWINDOW-MODE  
:EXPOSURES :CLIP-X :CLIP-Y :CLIP-MASK  
:CLIP-ORDERING :DASH-OFFSET :DASHES :CACHE-P)
```

where the values of the keyword arguments specify the graphics characteristics of a window. Table 7-1 explains the keyword arguments to CLX:CREATE-GCONTEXT.

Table 7-1: CLX:CREATE-GCONTEXT Keywords

Keyword	Value
:DRAWABLE	A CLX:DRAWABLE object. This argument is required.
:FUNCTION	A CLX:LOGICAL-OP value (see Table 7-2) that defines how the server computes pixel values when the client updates a section of the screen. The default operation is :GX-COPY.
:PLANE-MASK	An integer that defines the planes on which the server performs the bitwise computation of pixels defined by the value of :FUNCTION. The default plane mask is all 1s.
:FOREGROUND and :BACKGROUND	CLX:PIXEL values that specify the foreground and background colors. The default values are 0 and 1, respectively.
:LINE-WIDTH	An integer that defines the width of a line, in pixels. The default value is 0.
:LINE-STYLE	A keyword that specifies which sections of a line the server draws. The possible values are :SOLID (the default), :DASH, and :DOUBLE-DASH.
:CAP-STYLE	A keyword that specifies how the server draws the endpoints of a line. The possible values are :BUTT (the default), :NOT-LAST, :ROUND, and :PROJECTING.
:JOIN-STYLE	A keyword that specifies how the servers draw corners for wide lines. The possible values are :MITER (the default), :ROUND, and :BEVEL.
:FILL-STYLE	A keyword that specifies the contents of the source for line, text, and fill operations. The possible values are :SOLID (the default), :TILED, :OPAQUE-STIPPLED, and :STIPPLED.
:FILL-RULE	A keyword that specifies how the server computes which points are inside when filling a polygon. The possible values are :EVEN-ODD (the default) and :WINDING.
:ARC-MODE	A keyword that specifies how the server connects the endpoints of an arc. The possible values are :PIE-SLICE (the default) or :CHORD.
:TILE and :STIPPLE	The CLX:PIXMAP objects used for tile and stipple operations, respectively. The default pixmaps are of unspecified size, filled with the foreground pixel for tiling but 1s for stippling.
:TS-X and :TS-Y	Integers that define the X and Y coordinates of the origin for tile and stipple operations. The default values are both 0.
:FONT	A CLX:FONTABLE that defines the font used in text operations. Using a string for the :FONT component will cause an implicit call to CLX:OPEN-FONT.
:SUBWINDOW-MODE	A keyword that specifies whether inferior windows clip superior windows. The possible values are :CLIP-BY-CHILDREN (the default) and :INCLUDE-INFERIORS.
:EXPOSURES	A keyword that specifies whether the server informs the client when the contents of a window are lost. The possible values are :ON (the default) and :OFF.
:CLIP-X and :CLIP-Y	Integers that define the X and Y coordinates of the origin for clip operations. The default values are both 0.

(continued on next page)

Table 7-1 (Cont.): CLX:CREATE-GCONTEXT Keywords

Keyword	Value
:CLIP-MASK	A CLX:PIXMAP or CLX:RECT-SEQ value that the server uses to restrict write operations to the drawable. The default is no clip mask.
:CLIP-ORDERING	A hint to the server that improves performance of clip operations. The possible values are :UNSORTED, :Y-SORTED, :YX-SORTED, and :YX-BANDED.
:DASH-OFFSET	An integer that defines the pixel within the dash length sequence (specified by the :DASHES keyword) at which the server starts drawing a dashed line. The default is 0.
:DASHES	An integer or a sequence of integers that defines the length, in pixels, of each section of a dashed line. The default is the list (4 4).
:CACHE-P	A CLX:BOOLEAN value that specifies whether the graphics context is to be cached locally by CLX. The default is T.

Table 7-2 lists the operations that may be specified for the :FUNCTION component of a CLX:GCONTEXT structure.

Table 7-2: CLX:LOGICAL-OP Values

Keyword	Description
:GX-CLEAR	0
:GX-AND	<i>src</i> AND <i>dst</i>
:GX-AND-REVERSE	<i>src</i> AND NOT <i>dst</i>
:GX-COPY	<i>src</i>
:GX-AND-INVERTED	(NOT <i>src</i>) AND <i>dst</i>
:GX-NOOP	<i>dst</i>
:GX-XOR	<i>src</i> XOR <i>dst</i>
:GX-OR	<i>src</i> OR <i>dst</i>
:GX-NOR	(NOT <i>src</i>) AND NOT <i>dst</i>
:GX-EQUIV	(NOT <i>src</i>) XOR <i>dst</i>
:GX-INVERT	NOT <i>dst</i>
:GX-OR-REVERSE	<i>src</i> OR NOT <i>dst</i>
:GX-COPY-INVERTED	NOT <i>src</i>
:GX-OR-INVERTED	(NOT <i>src</i>) OR <i>dst</i>
:GX-NAND	(NOT <i>src</i>) OR NOT <i>dst</i>
:GX-SET	1

The screen the client is updating is the destination (*dst*). The graphics object that the client is drawing on the screen is the source (*src*). The keyword value specifies how the server computes new bits from the existing destination bits and the source bits. The default (and most common) logical function is specified by :GX-COPY, which uses only the source bits to update the screen.

7.2 Querying and Setting GContext Components

Objects of type CLX:GCONTEXT are returned by the CLX:CREATE-GCONTEXT function (see Section 7.1) and are passed to the drawing functions (see Chapter 9).

Table 7-3 lists the functions that return information about CLX:GCONTEXT objects, with descriptions of their return values and whether they can be used with SETF.

Table 7-3: Information Functions for CLX:GCONTEXT Objects

Function Name	Use with SETF?	Return Value
CLX:GCONTEXT-ARC-MODE	Yes	The method used to join the endpoints of an arc, either :CHORD or :PIE-SLICE.
CLX:GCONTEXT-BACKGROUND	Yes	A CLX:PIXEL value that specifies the background color.
CLX:GCONTEXT-CACHE-P	Yes	A CLX:BOOLEAN value that indicates whether the structure is stored locally by CLX.
CLX:GCONTEXT-CAP-STYLE	Yes	The method used to draw the endpoints of lines. The possible values are :BUTT, :NOT-LAST, :ROUND, and :PROJECTING.
CLX:GCONTEXT-CLIP-MASK	Yes	The mask used to restrict write operations, either :NONE or a CLX:PIXMAP or CLX:RECT-SEQ value.
CLX:GCONTEXT-CLIP-ORDERING	Yes	:UNSORTED, :Y-SORTED, :YX-SORTED, or :YX-BANDED.
CLX:GCONTEXT-CLIP-X	Yes	The integer X coordinate of the origin for the clip mask.
CLX:GCONTEXT-CLIP-Y	Yes	The integer Y coordinate of the origin for the clip mask.
CLX:GCONTEXT-DASHES	Yes	An integer or list of integers. Each integer specifies the length of a dash or a gap in a line drawn with :LINE-STYLE set to :DASH or :DOUBLE-DASH.
CLX:GCONTEXT-DASH-OFFSET	Yes	An integer index into the dashes list.
CLX:GCONTEXT-DISPLAY	No	The CLX:DISPLAY object on which the GContext was created.
CLX:GCONTEXT-EQUAL	No	A CLX:BOOLEAN value that indicates whether the predicate's two arguments are the same GContext.
CLX:GCONTEXT-EXPOSURES	Yes	:ON when the server processes exposure events for the window, otherwise :OFF.
CLX:GCONTEXT-FILL-RULE	Yes	The method used to determine which pixels are "inside" a shape being filled, either :EVEN-ODD or :WINDING.
CLX:GCONTEXT-FILL-STYLE	Yes	The method used to fill a shape. The possible values are :SOLID, :TILED, :OPAQUE-STIPPLED, and :STIPPLED.
CLX:GCONTEXT-FONT	Yes	A CLX:FONTABLE value that specifies the font to be used when displaying text.
CLX:GCONTEXT-FOREGROUND	Yes	A CLX:PIXEL value that specifies the foreground color.
CLX:GCONTEXT-FUNCTION	Yes	A CLX:LOGICAL-OP value (see Table 7-2) that specifies the method used to combine bits.
CLX:GCONTEXT-ID	No	The integer resource-id of the GContext.
CLX:GCONTEXT-JOIN-STYLE	Yes	The method used to join lines and line segments. The possible values are :MITER, :ROUND, and :BEVEL.

(continued on next page)

Table 7-3 (Cont.): Information Functions for CLX:GCONTEXT Objects

Function Name	Use with SETF?	Return Value
CLX:GCONTEXT-LINE-STYLE	Yes	The method used to draw lines. The possible values are :SOLID, :DASH, and :DOUBLE-DASH.
CLX:GCONTEXT-LINE-WIDTH	Yes	An integer that specifies the width of a line in pixels.
CLX:GCONTEXT-P	No	A CLX:BOOLEAN value that indicates whether the argument is of type CLX:GCONTEXT.
CLX:GCONTEXT-PLANE-MASK	Yes	An integer that specifies the planes on which graphics objects are drawn.
CLX:GCONTEXT-STIPPLE	Yes	The CLX:PIXMAP value used for stippling operations.
CLX:GCONTEXT-SUBWINDOW-MODE	Yes	A keyword that indicates whether children clip their ancestors, either :CLIP-BY-CHILDREN or :INCLUDE-INFERIORS.
CLX:GCONTEXT-TILE	Yes	The CLX:PIXMAP value used for tiling operations.
CLX:GCONTEXT-TS-X	Yes	An integer that specifies the X coordinate of the origin for tile and stipple operations.
CLX:GCONTEXT-TS-Y	Yes	An integer that specifies the Y coordinate of the origin for tile and stipple operations.

GContext components can also be set temporarily by using the CLX:WITH-GCONTEXT macro. See Section 7.4 for information on CLX:WITH-GCONTEXT.

7.3 Copying and Freeing GContexts

In addition to creating a graphics context, clients can copy defined characteristics from one CLX:GCONTEXT structure into another. The CLX:COPY-GCONTEXT function copies all components of the source GContext into the destination GContext. Its format is:

(CLX:COPY-GCONTEXT *source destination*)

source The CLX:GCONTEXT object to copy from.

destination The CLX:GCONTEXT object to copy into.

The CLX:COPY-GCONTEXT-COMPONENTS function copies only specified components into the destination GContext. Its format is:

(CLX:COPY-GCONTEXT-COMPONENTS *source destination &REST keys*)

source The CLX:GCONTEXT object to copy from.

destination The CLX:GCONTEXT object to copy into.

keys Keywords that specify which components are copied.

The keyword arguments correspond to those listed for the CLX:CREATE-GCONTEXT function in Section 7.1, Table 7-1, except for :DRAWABLE.

When a graphics context is no longer needed, you should deallocate it by calling the CLX:FREE-GCONTEXT function in the following format:

(CLX:FREE-GCONTEXT *gcontext*)

where *gcontext* is the CLX:GCONTEXT object whose memory is to be deallocated.

7.4 Using GContexts Efficiently

Changing GContext components frequently can seriously degrade performance, as the server must revalidate a GContext whenever a client redefines it. Grouping server requests that identify the same window and GContext will improve performance. When graphics characteristics need to be changed frequently, creating a new GContext is more efficient than redefining an existing one. Clients using the DECwindows server should create no more than 50 GContexts.

When the :CACHE-P component is true, which is the default, CLX stores the GContext locally to the client so that server requests are batched. Component changes (by means of SETF and WITH-GCONTEXT) are always deferred regardless of the cache mode, and are sent over the protocol only when required by a local operation or by an explicit call to CLX:FORCE-GCONTEXT-CHANGES.

The format of the CLX:FORCE-GCONTEXT-CHANGES function is:

(CLX:FORCE-GCONTEXT-CHANGES *gcontext*)

where *gcontext* is the CLX:GCONTEXT object whose components are changed.

The CLX:WITH-GCONTEXT macro changes GContext components within the dynamic scope of the body.

(CLX:WITH-GCONTEXT *gcontext*
 &KEY :FUNCTION :PLANE-MASK :BACKGROUND :LINE-WIDTH
 :LINE-STYLE :CAP-STYLE :JOIN-STYLE :FILL-STYLE :FILL-RULE :ARC-MODE
 :TILE :STIPPLE :TS-X :TS-Y :FONT :SUBWINDOW-MODE :EXPOSURES
 :CLIP-X :CLIP-Y :CLIP-MASK :CLIP-ORDERING :DASHES :DASH-OFFSET
 &BODY *forms*)

gcontext The CLX:GCONTEXT object whose components are changed.

:FUNCTION ... The keyword arguments specify the components of *gcontext* that are changed.

forms LISP forms to be executed while the changes to the graphics context are in effect.

The body is not surrounded by a CLX:WITH-DISPLAY. If :CACHE-P is NIL or some component states are unknown, CLX:WITH-GCONTEXT implements save/restore by creating a temporary GContext and using CLX:COPY-GCONTEXT-COMPONENTS to copy components to it and from it.

Chapter 8

Using Color

Color is an attribute of both windows and graphics objects. Depending on display hardware, clients can define color as black or white, as shades of gray, or as a spectrum of hues. For a detailed description of color definitions, including VAXstations and the visual types they support, see the *VMS DECwindows Guide to Xlib Programming: MIT C Binding*.

This chapter introduces managing color with CLX and describes how to share and allocate color resources; the following topics are covered:

- Pixels and colormaps
- Matching color requirements to screen types
- Sharing color resources
- Allocating colors for exclusive use
- Storing color values
- Freeing color resources
- Querying colormap entries

8.1 Pixels and Colormaps

The color of a window or graphics object depends on the values of the pixels that constitute it. The number of bits associated with each pixel determines the number of possible pixel values. Monochrome screens have one bit per pixel, which gives two possible pixel values, black and white. On a monochrome screen, all bits that define an image reside on a single "plane," an allocation of memory in which there is a one-to-one correspondence between bits and pixels. The number of planes is called the depth of the screen. The greater the depth of the screen, the more colors it can display at one time.

Color and intensity (gray-scale) screens have more than one plane, so more than one bit defines the value of a pixel. Each of these bits resides on a different plane in memory. For example, a four-plane intensity display can produce up to sixteen levels of brightness. A four-plane color display can produce as many as sixteen colors. The `CLX:SCREEN-DEPTH$` function returns a list of all the depths supported by the display hardware.

CLX uses a table, called a colormap, to define the color of each pixel. A colormap contains a collection of color cells, each of which defines a pixel value in terms of its red, green, and blue (RGB) components. In CLX, RGB components are floating-point numbers between zero and one, inclusive, not integers between zero (off) and 65535 (brightest). Each pixel value is an index into a colormap.

Because most VAXstation workstations have a global hardware colormap that applies to the entire display, clients should use the same colormap whenever possible. If a client allocates and installs its own colormap, other clients on the display appear in the wrong colors.

CLX provides a default colormap to which all clients have access. This is returned by the CLX:SCREEN-DEFAULT-COLORMAP function. Clients can either allocate color cells for exclusive use or allocate colors for shared use from the default colormap. Sharing colors conserves space in the default colormap.

In cases where the client cannot use the default colormap and must use a new colormap, CLX creates virtual colormaps that are swapped in and out of the hardware colormap. Color values of a virtual colormap are loaded or unloaded into the hardware lookup table when a client calls the CLX:INSTALL-COLORMAP or CLX:UNINSTALL-COLORMAP function.

The format of the CLX:INSTALL-COLORMAP function is:

CLX:INSTALL-COLORMAP *colormap*

where *colormap* is the CLX:COLORMAP object that is installed.

The format of the CLX:UNINSTALL-COLORMAP function is:

CLX:UNINSTALL-COLORMAP *colormap*

where *colormap* is the CLX:COLORMAP object that is swapped out.

CLX also provides the CLX:INSTALLED-COLORMAPS function, which returns the colormaps currently installed. Its format is:

CLX:INSTALLED-COLORMAPS *window* &KEY :RESULT-TYPE

window The CLX:WINDOW object whose colormaps are listed.

:RESULT-TYPE The LISP type of the return value. The default is 'LIST.

The default colormap on a specified screen is returned by the CLX:SCREEN-DEFAULT-COLORMAP function.

8.2 Matching Color Requirements to Screen Types

Each screen has certain visual characteristics, such as color or monochrome capability, that are collectively called the visual information of the screen. CLX stores visual information in the CLX:VISUAL-INFO structure. Table 8-1 lists the accessor functions for CLX:VISUAL-INFO objects. None of these functions can be used with SETF.

Table 8-1: Information Functions for CLX:VISUAL-INFO Objects

Function Name	Return Value
CLX:VISUAL-INFO-ID	An integer resource-id.
CLX:VISUAL-INFO-CLASS	The visual type of the screen: :STATIC-GRAY, :STATIC-COLOR, :TRUE-COLOR, :GRAY-SCALE, :PSEUDO-COLOR, or :DIRECT-COLOR.
CLX:VISUAL-INFO-RED-MASK	A CLX:PIXEL value.

(continued on next page)

Table 8-1 (Cont.): Information Functions for CLX:VISUAL-INFO Objects

Function Name	Return Value
CLX:VISUAL-INFO-GREEN-MASK	A CLX:PIXEL value.
CLX:VISUAL-INFO-BLUE-MASK	A CLX:PIXEL value.
CLX:VISUAL-INFO-BITS-PER-RGB	An integer specifying the number of bits in each red, green, or blue value of a color cell.
CLX:VISUAL-INFO-COLORMAP-ENTRIES	An integer specifying the number of entries in the colormap.

The constructor function for CLX:VISUAL-INFO objects is CLX:MAKE-VISUAL-INFO. This function takes a single argument, which is an integer resource-id (a CLX:VISUAL value). The CLX:VISUAL value associated with a specified window is returned by the CLX:WINDOW-VISUAL function; with a specified screen by the CLX:SCREEN-ROOT-VISUAL function. For example:

```
Lisp> *window*
#<Window 12582913>
Lisp> (setf *visual* (clx:window-visual *window*))
14041000
Lisp> (setf *info* (clx:make-visual-info *visual*))
#<Visual-Info 524389>
Lisp> (clx:visual-info-colormap-entries *info*)
256
```

8.3 Sharing Color Resources

CLX provides the following ways to share color resources:

- Using named colors
- Specifying exact color values

8.3.1 Using Named Colors

X11 servers provide named colors that clients can share. (For a list of VMS DECwindows color names, see the *VMS DECwindows Guide to Xlib Programming: MIT C Binding*. Such names, however, are not portable to other servers.) To use a named color, call the CLX:ALLOC-COLOR function with a CLX:STRINGABLE value for the *color* argument. The CLX:ALLOC-COLOR function determines whether the colormap defines a value for the specified color. If the color exists, the server returns the index to the colormap (that is, a pixel value). If the color does not exist, the server returns an error.

The format of the CLX:ALLOC-COLOR function is:

CLX:ALLOC-COLOR *colormap color*

colormap The CLX:COLORMAP object that is to be accessed.
color Either a CLX:COLOR object or a string or symbol value that names the color to be allocated.

The function returns three values: a pixel, the screen color, and the exact color. The screen color contains the RGB values supported by the display hardware. The exact color contains the RGB values specified by the named color.

The files RADAR.LSP and REBOUND.LSP are examples of using CLX:ALLOC-COLOR with named colors. These programs are in the directory referenced by the logical name LISP\$EXAMPLES.

8.3.2 Specifying Exact Color Values

To specify exact color values, use the following method:

1. Assign values to a CLX:COLOR object.
2. Call the CLX:ALLOC-COLOR function, specifying the color created in Step 1. The CLX:ALLOC-COLOR function returns the pixel value of the color and the RGB values of the closest color supported by the hardware.

Table 8-2 lists the functions that return information about CLX:COLOR objects.

Table 8-2: Information Functions for CLX:COLOR Objects

Function Name	Use with SETF?	Return Value
CLX:COLOR-BLUE	Yes	A number between 0 and 1, inclusive, indicating the percentage of blue in the color.
CLX:COLOR-GREEN	Yes	A number between 0 and 1, inclusive, indicating the percentage of green in the color.
CLX:COLOR-RED	Yes	A number between 0 and 1, inclusive, indicating the percentage of red in the color.
CLX:COLOR-RGB	No	Three CLX:RGB-VAL values for red, green, and blue.

The constructor function for CLX:COLOR objects is CLX:MAKE-COLOR. The format of this function is:

CLX:MAKE-COLOR &KEY :RED GREEN BLUE
&ALLOW-OTHER-KEYS

For example:

```
Lisp> (setf sea-green (clx:make-color :red 0 :green 1 :blue .8))  
#<Color red: 0.00 green: 1.00 blue: 0.80>  
Lisp> (clx:alloc-color *colormap* sea-green)  
243 ;  
#<Color red: 0.00 green: 1.00 blue: 0.80>  
Lisp> (setf (clx:gcontext-foreground *gcontext*) 243)  
243
```

8.4 Allocating Colors for Exclusive Use

When a client wants to change color values that are already in a colormap, it must allocate them for its exclusive use. CLX provides two methods for allocating colors for the exclusive use of a client. First, the client can allocate cells and store color values in the default colormap. Second, if the default colormap does not contain enough storage, the client can create its own colormap in which to store color values.

This section describes how to specify a colormap, how to allocate cells for exclusive use, and how to store values in the color cells.

8.4.1 Specifying a Colormap

To create your own colormap, call the CLX:CREATE-COLORMAP function. Its format is:

CLX:CREATE-COLORMAP *visual window &OPTIONAL alloc-p*

- visual* A CLX:VISUAL value, that is, an integer pointing to the CLX:VISUAL-INFO data supported by the screen hardware.
- window* A CLX:WINDOW object, used to specify the screen.
- alloc-p* A CLX:BOOLEAN value that specifies whether the client allocates all colormap entries for its exclusive use or creates a colormap with no defined entries.

If the visual type is :STATIC-GRAY, :STATIC-COLOR, or :TRUE-COLOR, the client must allocate no entries. Specifying NIL for the *alloc-p* argument is useful when two or more clients are to share the newly created colormap. The function returns the specified CLX:COLORMAP object.

Table 8-3 lists the functions that are available for CLX:COLORMAP objects. None of these functions can be used with SETF.

Table 8-3: Information Functions for CLX:COLORMAP Objects

Function Name	Return Value
CLX:COLORMAP-DISPLAY	CLX:DISPLAY
CLX:COLORMAP-EQUAL	CLX:BOOLEAN
CLX:COLORMAP-ID	INTEGER
CLX:COLORMAP-P	CLX:BOOLEAN

8.4.2 Allocating Color Cells

After specifying a colormap, allocate color cells in it. The CLX:ALLOC-COLOR-CELLS function allocates cells for a pseudocolor or a gray-scale device. The CLX:ALLOC-COLOR-PLANES function simulates a direct color device.

The format of the CLX:ALLOC-COLOR-CELLS function is:

CLX:ALLOC-COLOR-CELLS *colormap colors*
&KEY :PLANES :CONTIGUOUS-P :RESULT-TYPE

- colormap* The CLX:COLORMAP object in which cells are allocated.
- colors* An integer that specifies the number of pixels to be returned.
- :PLANES An integer that specifies the number of plane masks to be returned. The default value is 0.
- :CONTIGUOUS-P A CLX:BOOLEAN value that specifies whether the colormap entries must be contiguous.
- :RESULT-TYPE The LISP type of the return values. The default is 'LIST.

The function returns a sequence of CLX:PIXEL values and a sequence of CLX:MASK16 values. For detailed information on plane masks, see the *X Window System: C Library and Protocol Reference*.

The format of the CLX:ALLOC-COLOR-PLANES function is:

CLX:ALLOC-COLOR-PLANES *colormap colors*
&KEY :REDS :GREENS :BLUES :CONTIGUOUS-P
:RESULT-TYPE

<code>colormap</code>	The CLX:COLORMAP object that is to be accessed.
<code>colors</code>	An integer that specifies the number of pixel values returned.
<code>:REDS</code>	An integer that specifies the number of red planes. The default value is 0.
<code>:GREENS</code>	An integer that specifies the number of green planes. The default value is 0.
<code>:BLUES</code>	An integer that specifies the number of blue planes. The default value is 0.
<code>:CONTIGUOUS-P</code>	A CLX:BOOLEAN value that specifies whether the colormap entries are contiguous.
<code>:RESULT-TYPE</code>	The type specifier of the first return value. The default is ' <code>LIST</code> '.

The function returns four values: a sequence of CLX:PIXEL values and three CLX:MASK16 values for the red, green, and blue planes.

When allocating colors from a shared colormap, the client may exhaust the resources of the colormap. In this case, CLX provides the CLX:COPY-COLORMAP-AND-FREE function for copying existing colormap entries into a new client-created colormap. The function returns a colormap of the same visual type and for the same screen as the previously shared colormap. The previously shared colormap can be either the default colormap or a client-created colormap.

The format of the CLX:COPY-COLORMAP-AND-FREE function is:

`CLX:COPY-COLORMAP-AND-FREE colormap`

where `colormap` is the CLX:COLORMAP object that is to be copied.

The function returns the newly copied CLX:COLORMAP object.

CLX:COPY-COLORMAP-AND-FREE copies all allocated cells from the previously shared colormap to the new colormap, keeping color values intact. The new colormap is created with the same value for the `alloc-p` argument as the previously shared colormap. This value determines which entries are moved to the new colormap and which entries in the shared colormap are freed:

- `T` All entries are copied from the previously shared colormap and are then freed to create writable map entries.
- `NIL` The entries moved are all pixels and planes that have been allocated with the following routines and that have not been freed since they were allocated: CLX:ALLOC-COLOR, CLX:ALLOC-COLOR-CELLS, CLX:ALLOC-COLOR-PLANES.

8.5 Storing Color Values

After allocating color entries in the colormap, you can store RGB values in specified colormap cells with the following method:

1. Assign color values to a CLX:COLOR structure.
2. Call the CLX:STORE-COLOR function to store one color or the CLX:STORE-COLORS function to store multiple colors.

To store named colors, use a CLX:STRINGABLE value in place of a CLX:COLOR object. CLX looks up the color name.

The format of the CLX:STORE-COLOR function is:

`CLX:STORE-COLOR colormap pixel spec &KEY :RED-P :GREEN-P :BLUE-P`

<i>colormap</i>	The CLX:COLORMAP object whose color cell will be stored in memory.
<i>pixel</i>	A CLX:PIXEL value that specifies where to store the color.
<i>spec</i>	Either a CLX:COLOR object or a CLX:STRINGABLE value that names the color to be stored.
:RED-P	A CLX:BOOLEAN value that indicates whether the red value should be stored. The default is T.
:GREEN-P	A CLX:BOOLEAN value that indicates whether the green value should be stored. The default is T.
:BLUE-P	A CLX:BOOLEAN value that indicates whether the blue value should be stored. The default is T.

The format of the CLX:STORE-COLORS function is:

CLX:STORE-COLORS *colormap specs &KEY :RED-P :GREEN-P :BLUE-P*

<i>colormap</i>	The CLX:COLORMAP object whose color cells will be stored in memory.
<i>specs</i>	A CLX:REPEAT-SEQ value with elements of the form: <i>(pixel color)</i>
	where <i>pixel</i> is a CLX:PIXEL value and <i>color</i> is either a CLX:COLOR object or a CLX:STRINGABLE value that names the color to be stored.
:RED-P	A CLX:BOOLEAN value that indicates whether the red value should be stored. The default is T.
:GREEN-P	A CLX:BOOLEAN value that indicates whether the green value should be stored. The default is T.
:BLUE-P	A CLX:BOOLEAN value that indicates whether the blue value should be stored. The default is T.

8.6 Freeing Color Resources

To free storage allocated for client colors, call the CLX:FREE-COLORS function. CLX:FREE-COLORS releases all storage allocated by the following color functions: CLX:ALLOC-COLOR, CLX:ALLOC-COLOR-CELLS, and CLX:ALLOC-COLOR-PLANES. The format of the CLX:FREE-COLORS function is:

CLX:FREE-COLORS *colormap pixels &OPTIONAL :PLANE-MASK*

<i>colormap</i>	The CLX:COLORMAP object whose color cells are to be erased from memory.
<i>pixels</i>	A sequence of CLX:PIXEL values, specifying the cells to erase.
:PLANE-MASK	A CLX:MASK16 value that specifies which planes are erased. The default value is 0, meaning all planes.

To delete the association between a colormap ID and a colormap, call the CLX:FREE-COLORMAP function. CLX:FREE-COLORMAP has no effect on the screen's default colormap. If the colormap is installed, CLX:FREE-COLORMAP removes it. The format of the CLX:FREE-COLORMAP function is:

CLX:FREE-COLORMAP *colormap*

where *colormap* is the CLX:COLORMAP object that is to be erased from memory.

8.7 Querying Colormap Entries

CLX provides functions to return the RGB values of both colormap entries and named colors.

To query the RGB values of specified entries in a colormap, use the CLX:QUERY-COLORS function. Its format is:

CLX:QUERY-COLORS *colormap pixels* &KEY :RESULT-TYPE

colormap The CLX:COLORMAP object whose colors will be returned.

pixels A sequence of CLX:PIXEL values specifying the entries being queried.

:RESULT-TYPE The LISP type of the return value. The default is 'LIST.

CLX:QUERY-COLORS returns a sequence of the CLX:COLOR objects defined in the specified colormap.

To look up the RGB values associated with a named color, use the CLX:LOOKUP-COLOR function. This function uses the specified colormap to find out the value with respect to a particular screen. The format of the CLX:LOOKUP-COLOR function is:

CLX:LOOKUP-COLOR *colormap name*

colormap The CLX:COLORMAP object of interest.

name A CLX:STRINGABLE value that names the color.

The function returns both the screen values (what the hardware supports) and the true values of the named color. For example, assuming that the variable *COLORMAP* is bound to the default colormap of a screen:

```
Lisp> (clx::lookup-color *colormap* "turquoise")
#<Color red: 0.68 green: 0.91 blue: 0.91> ;
#<Color red: 0.68 green: 0.92 blue: 0.92>
```

Chapter 9

Graphics Functions

CLX provides routines for drawing graphics into windows and pixmaps. This chapter describes how to create and manage graphics drawn into windows, and covers the following topics:

- Drawing points
- Drawing lines
- Drawing and filling rectangles
- Drawing and filling arcs
- Clearing and copying areas
- Creating cursors

All the drawing functions in this chapter take a drawable and a GContext as their first two arguments. For better performance, group calls with the same, unmodified GContext.

Chapter 10 has additional information on drawing graphics into pixmaps.

9.1 Drawing Points

CLX provides the CLX:DRAW-POINT function for drawing a single point and the CLX:DRAW-POINTS function for drawing multiple points. CLX batches drawing requests to the server; however, when you are drawing multiple points, performance is better if you use a single call to DRAW-POINTS rather than many calls to DRAW-POINT.

The format of the CLX:DRAW-POINT function is:

`CLX:DRAW-POINT drawable gcontext x y`

drawable The CLX:DRAWABLE object in which the point is displayed.

gcontext The CLX:GCONTEXT object that specifies the graphics characteristics of the point (such as color).

x and *y* Integers that specify the X and Y coordinates of the point.

The X and Y coordinates are relative to the origin of the drawable.

The CLX:DRAW-POINTS function can draw multiple points in a single request. The coordinates of the points can be stored in a CLX:POINT-SEQ value before the function is called. The format of the CLX:DRAW-POINTS function is:

`CLX:DRAW-POINTS drawable gcontext points &OPTIONAL relative-p`

- drawable* The CLX:DRAWABLE object in which the points are displayed.
- gcontext* The CLX:GCONTEXT object that specifies the graphics characteristics of the points (such as color).
- points* A value of type CLX:POINT-SEQ that specifies the coordinates of the points.
- relative-p* An optional argument whose CLX:BOOLEAN value specifies whether the coordinates of each point are relative to the previous point.

The coordinates of the first point in the sequence are relative to the origin of the drawable. If *relative-p* is false, the coordinates of all other points are also relative to the origin. If *relative-p* is true, they are relative to the point that precedes them in the sequence.

Example 9-1 uses the CLX:DRAW-POINTS function to draw a circle of points around each click of MB1. This is translated from Example 6-1 in the VMS DECwindows Guide to Xlib Programming: MIT C Binding.

Example 9-1: Drawing Points

```
;;;This code creates the window for the CLX:DRAW-POINTS example
(defvar *window* nil "The window to show examples in.")
(defvar *gcontext* nil "The gcontext for the window to show examples in.")

(defun create-example-window (host)
  (let* ((display (clx:open-display host))
         (screen (clx:display-default-screen display))
         (root-window (clx:screen-root screen))
         (white-pixel (clx:screen-white-pixel screen)))
    (setq *window*
          (clx:create-window :parent root-window
                             :x 100 :y 100 :width 600 :height 600
                             :background white-pixel
                             :event-mask '(:exposure :button-press)))
    (setq *gcontext*
          (clx:create-gcontext :drawable *window*)))
  (clx:map-window *window*)
  (clx:display-force-output (clx:window-display *window*)))

;; Help function for point-example
(defun write-point-messages (window gcontext)
  (clx:draw-glyphs window gcontext 150 25 "To create points, press MB1")
  (clx:draw-glyphs window gcontext 150 50
    "Each press creates a new circle of points")
  (clx:draw-glyphs window gcontext 150 75 "To exit, press MB2"))
```

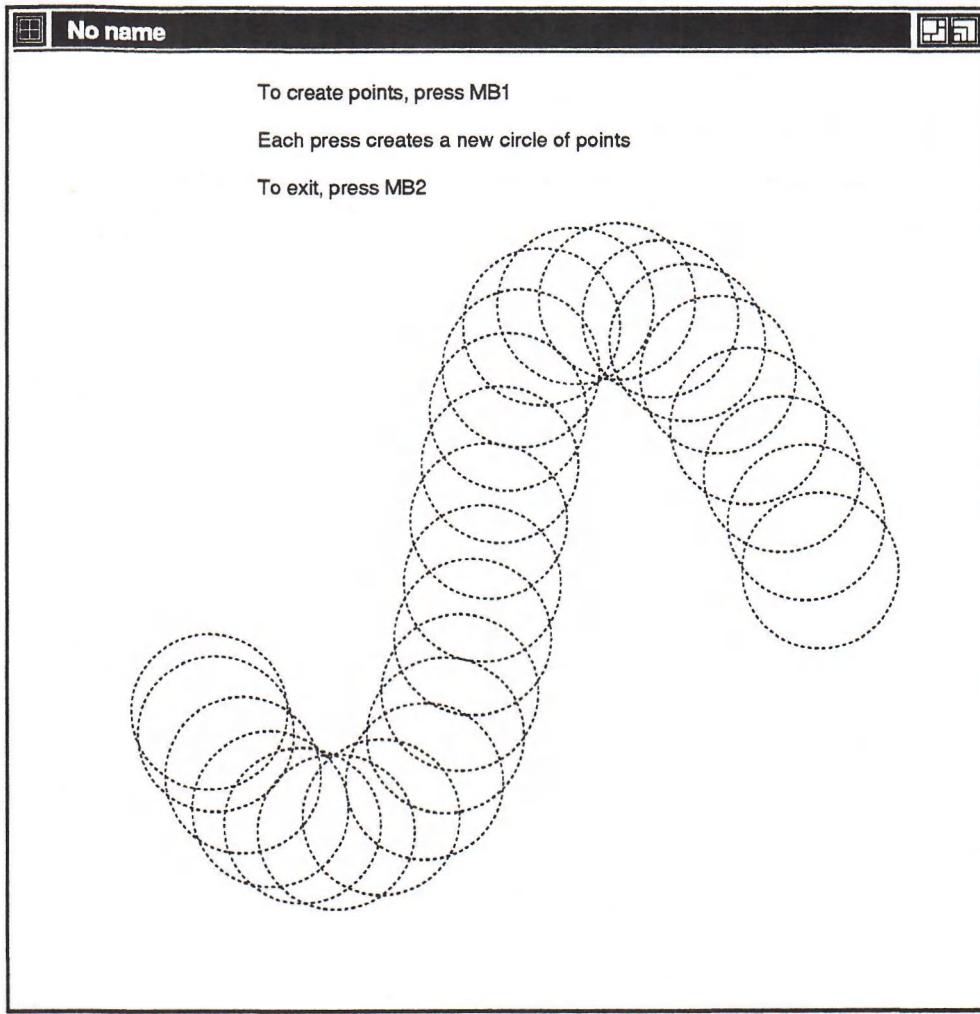
(continued on next page)

Example 9-1 (Cont.): Drawing Points

```
(defun point-example (win gcontext)
  ;; Window should be width 600, height 600, and event mask set to
  ;; button-press and exposures. This may be done as follows:
  ;; (setf (window-event-mask <window>)
  ;;       (make-event-mask :button-press :exposure))
  (write-point-messages win gcontext)
  (let ((point-vector (make-array 200 :element-type 'fixnum)))
    (clx:event-case ((clx:window-display win) :discard-p t)
      ;; On exposure events, rewrite the text
      (:exposure () (write-point-messages win gcontext) nil)
      (:button-press (x y code)
        (if (= code 1) ; that's MB1
            (progn
              ;; Fill in the array points
              (dotimes (index 100)
                (setf (aref point-vector (* 2 index))
                      (+ x (truncate (* 50 (cos index))))))
              (setf (aref point-vector (1+ (* 2 index)))
                      (+ y (truncate (* 50 (sin index))))))
              ;; and draw them
              (clx:draw-points win gcontext point-vector) nil)
            ;; Otherwise it's MB2 or MB3, so exit event-case
            t)))))
```

Figure 9-1 shows the output from Example 9-1. To run this example, first invoke CREATE-EXAMPLE-WINDOW and then pass *WINDOW* and *GCONTEXT* to POINT-EXAMPLE.

Figure 9–1: Drawing Points



MLO-003391

9.2 Drawing Lines

CLX provides three functions for drawing lines and line segments. The CLX:DRAW-LINE function draws a straight line between two specified endpoints. The CLX:DRAW-LINES function draws a series of straight lines connecting each point in a sequence (a CLX:POINT-SEQ value). Similarly, the CLX:DRAW-SEGMENTS function draws straight lines connecting each *pair* of points in a sequence (a CLX:SEG-SEQ value). That is, CLX:DRAW-LINES treats consecutive lines as having a common endpoint, but CLX:DRAW-SEGMENTS does not.

The format of the CLX:DRAW-LINE function is:

CLX:DRAW-LINE *drawable gcontext x1 y1 x2 y2 &OPTIONAL relative-p*

drawable The CLX:DRAWABLE object in which the line is drawn.

<i>gcontext</i>	The CLX:GCONTEXT object that specifies the graphics characteristics of the line (line style, line width, and so on).
<i>x1</i> and <i>y1</i>	Integers that specify the X and Y coordinates of the first endpoint of the line.
<i>x2</i> and <i>y2</i>	Integers that specify the X and Y coordinates of the second endpoint of the line.
<i>relative-p</i>	A CLX:BOOLEAN value that specifies whether the coordinates of the second endpoint are relative to the first endpoint.

If *relative-p* is false, the coordinates of both endpoints are relative to the origin of *drawable*. This is the default.

The format of the CLX:DRAW-LINES function is:

CLX:DRAW-LINES *drawable gcontext points &KEY :RELATIVE-P :FILL-P :SHAPE*

<i>drawable</i>	The CLX:DRAWABLE object in which the lines are drawn.
<i>gcontext</i>	The CLX:GCONTEXT object that specifies the graphics characteristics of the lines (line style, line width, and so on).
<i>points</i>	A CLX:POINT-SEQ value that contains the X and Y coordinates of the lines, in the form: <i>(x1 y1 x2 y2 ...)</i>
:RELATIVE-P	A CLX:BOOLEAN value that specifies whether the coordinates of the second and following points are relative to the point that precedes them in the point sequence.
:FILL-P	A CLX:BOOLEAN value that specifies whether the region enclosed by the lines is filled.
:SHAPE	A keyword used by the server to optimize fill operations. Possible values are explained below: :COMPLEX The region enclosed by the lines can intersect itself. :NON-CONVEX The region does not intersect itself. :CONVEX The region is wholly convex. The default value is :COMPLEX.

If :FILL-P is true and the :SHAPE is known and correctly specified, performance may be improved.

The format of the CLX:DRAW-SEGMENTS function is:

CLX:DRAW-SEGMENTS *drawable gcontext segments*

<i>drawable</i>	The CLX:DRAWABLE in which the line segments are drawn.
<i>gcontext</i>	The CLX:GCONTEXT that specifies the graphics characteristics of the line segments (line style, line width, and so on).
<i>segments</i>	A CLX:SEG-SEQ value that contains the X and Y coordinates of the endpoints of the line segments.

With CLX:DRAW-SEGMENTS, line segments are not connected unless you specify a common endpoint in the CLX:SEG-SEQ value. Example 9-2 illustrates the difference between CLX:DRAW-LINES and CLX:DRAW-SEGMENTS.

Example 9–2: Drawing Lines and Line Segments

```
(defun lines-segments-demo (host)
  (let* ((d (clx:open-display host))
         (s (clx:display-default-screen d))
         (w1 (clx:create-window :parent (clx:screen-root s)
                                :x 0 :y 0 :width 200 :height 100
                                :background (clx:screen-white-pixel s)))
         (w2 (clx:create-window :parent (clx:screen-root s)
                                :x 210 :y 0 :width 200 :height 100
                                :background (clx:screen-white-pixel s)))
         (gc (clx:create-gcontext :drawable w1
                                :foreground (clx:screen-black-pixel s)
                                :background (clx:screen-white-pixel s)))
         (points ' ( 10 10 190 10    10 50 190 50    10 90 190 90)))
    (clx:map-window w1)
    (clx:map-window w2)

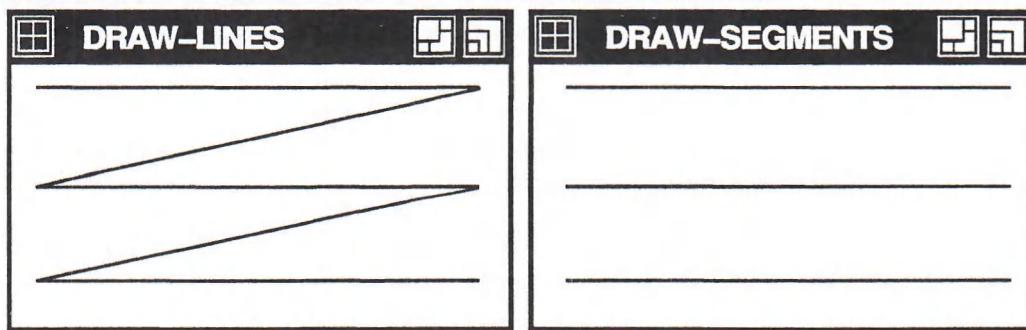
    ;; Make windows appear before drawing in them
    (clx:display-force-output d)

    ;; Give windows names
    (setf (clx:wm-name w1) "DRAW-LINES")
    (setf (clx:wm-name w2) "DRAW-SEGMENTS")

    ;; Draw same points as lines and as segments
    (clx:draw-lines w1 gc points)
    (clx:draw-segments w2 gc points)
    (clx:display-force-output d)))
```

Figure 9–2 shows the output of this example.

Figure 9–2: Drawing Lines and Line Segments



MLO-003392

9.3 Drawing and Filling Rectangles

Rectangles can either be drawn as outlines or can be filled. A single call to the DRAW-RECTANGLES function is more efficient than multiple calls to the DRAW-RECTANGLE function.

The format of the CLX:DRAW-RECTANGLE function is:

CLX:DRAW-RECTANGLE *drawable gcontext x y width height &OPTIONAL fill-p*

<i>drawable</i>	The CLX:DRAWABLE object in which the rectangle is drawn.
<i>gcontext</i>	The CLX:GCONTEXT object that specifies the graphics characteristics of the rectangle (line width, color, and so on).
<i>x</i> and <i>y</i>	Integers that specify the X and Y coordinates of the top-left corner of the rectangle.
<i>width</i> and <i>height</i>	Integers that specify the width and height, in pixels, of the rectangle.
<i>fill-p</i>	A CLX:BOOLEAN value that specifies whether the rectangle is filled.

If *fill-p* is true, the rectangle is filled with the pixmap specified by the :FILL-STYLE component of *gcontext*.

The CLX:DRAW-RECTANGLES function has a required argument of type CLX:RECT-SEQ that must contain the X and Y coordinates, width, and height of each rectangle to be drawn. The format of this function is:

CLX:DRAW-RECTANGLES *drawable gcontext rectangles* &OPTIONAL *fill-p*

<i>drawable</i>	The CLX:DRAWABLE object in which the rectangles are drawn.
<i>gcontext</i>	The CLX:GCONTEXT object that specifies the graphics characteristics of the rectangles (line width, color, and so on).
<i>rectangles</i>	A CLX:RECT-SEQ value that specifies the origin and size of the rectangles.
<i>fill-p</i>	A CLX:BOOLEAN value that specifies whether the rectangles are filled.

9.4 Drawing and Filling Arcs

To draw an arc with CLX, you must specify a rectangle that defines the boundaries of the arc as well as two angles that indicate the starting point and extent of the arc. That is, an arc is specified as a portion of an ellipse inscribed within a rectangle. For example, to draw a circle, you would specify a square as the bounding rectangle, any angle as the start, and 2π radians as the extent. CLX provides the CLX:DRAW-ARC function for drawing a single arc and the CLX:DRAW-ARCS function for drawing multiple arcs. A single call to CLX:DRAW-ARCS is more efficient than multiple calls to CLX:DRAW-ARC.

The format of the CLX:DRAW-ARC function is:

CLX:DRAW-ARC *drawable gcontext x y width height angle1 angle2*
&OPTIONAL *fill-p degrees-p*

<i>drawable</i>	The CLX:DRAWABLE object in which the arc is drawn.
<i>gcontext</i>	The CLX:GCONTEXT object that specifies the graphics characteristics of the arc (arc-mode, color, and so on).
<i>x</i> and <i>y</i>	Integers that specify the X and Y coordinates of the top left corner of the arc's bounding rectangle.
<i>width</i> and <i>height</i>	Integers that specify the dimensions, in pixels, of the arc's bounding rectangle.
<i>angle1</i>	A CLX:ANGLE value that specifies the start of the arc.
<i>angle2</i>	A CLX:ANGLE value that specifies the extent of the arc.
<i>fill-p</i>	A CLX:BOOLEAN value that specifies whether the arc is filled.
<i>degrees-p</i>	A CLX:BOOLEAN value that specifies whether the angles are measured in degrees.

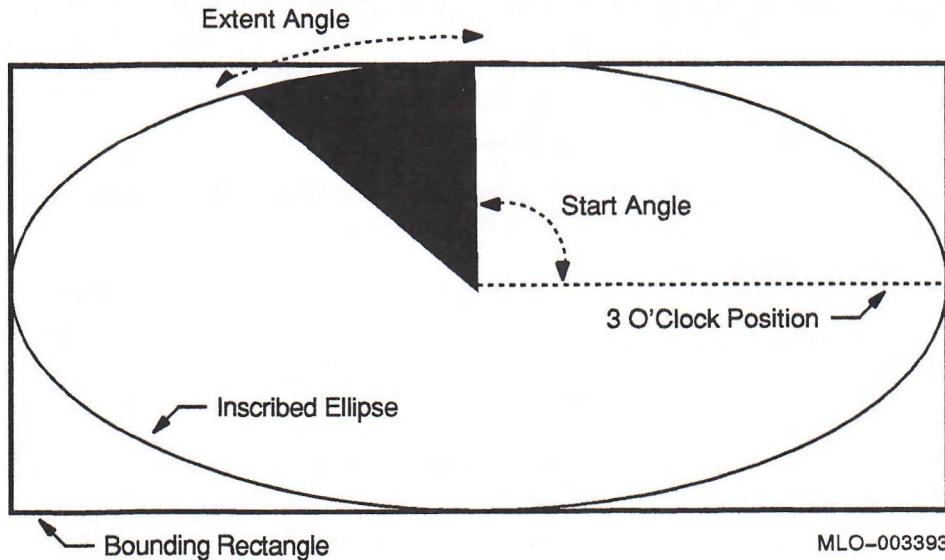
The angles are measured in radians unless *degrees-p* is true. Positive values indicate counterclockwise motion; negative values indicate clockwise motion. The start of the arc is relative to the 3 o'clock position from the center of the rectangle. The extent of the arc is relative to the start of the arc. Example 9-3 illustrates the relationships among the rectangle and the angles that specify an arc.

Example 9-3: Drawing an Arc

```
(defun arcs-demo (host)
  (let* ((display (clx:open-display host))
         (screen (clx:display-default-screen display))
         (window (clx:create-window :parent (clx:screen-root screen)
                                      :x 0 :y 0 :width 400 :height 200))
         (gc (clx:create-gcontext :drawable window
                                   :arc-mode :pie-slice)))
    ;; Make window visible
    (clx:map-window window)
    (clx:display-force-output display)
    ;; Draw bounding rectangle (for demonstration only)
    (clx:draw-rectangle window gc 10 10 380 180)
    ;; Draw complete ellipse (for demonstration only)
    (clx:draw-arc window gc 10 10 380 180 0 360 nil t)
    ;; Draw arc from 12:00 to 11:00, filled
    (clx:draw-arc window gc 10 10 380 180 90 30 t t)
    (clx:display-force-output display)))
```

Figure 9-3 shows the output from Example 9-3, with labels added.

Figure 9-3: Drawing an Arc



If *fill-p* is true, the endpoints of the arc are joined according to the :ARC-MODE component of *gc*, and the resulting shape is filled with the pixmap specified by the :BACKGROUND component of *gc*.

The format of the CLX:DRAW-ARCS function is:

CLX:DRAW-ARCS *drawable gcontext arcs &OPTIONAL fill-p degrees-p*

<i>drawable</i>	The CLX:DRAWABLE object in which the arcs are drawn.
<i>gcontext</i>	The CLX:GCONTEXT object that specifies the graphics characteristics of the arcs (arc-mode, color, and so on).
<i>arcs</i>	A CLX:ARC-SEQ value that specifies the origins and dimensions of the arcs' bounding rectangles.
<i>fill-p</i>	A CLX:BOOLEAN value that specifies whether the arcs are filled.
<i>degrees-p</i>	A CLX:BOOLEAN value that specifies whether the angles are measured in degrees.

9.5 Clearing and Copying Areas

A rectangular area of a window can be cleared (filled with the background color) with the CLX:CLEAR-AREA function or copied to another window or pixmap with the CLX:COPY-AREA function. Areas of pixmaps can also be copied, but because pixmaps do not have defined backgrounds, you must use the CLX:DRAW-RECTANGLE function to clear an area of a pixmap. A single bit plane of a window or pixmap can be copied with the CLX:COPY-PLANE function.

The format of the CLX:CLEAR-AREA function is:

CLX:CLEAR-AREA *window &KEY :X :Y :WIDTH :HEIGHT :EXPOSURES-P*

<i>window</i>	The CLX:WINDOW object that contains the area to clear.
<i>:x</i> and <i>:y</i>	Integers that specify the X and Y coordinates of the top left corner of the area. The default value for both keywords is 0, which specifies the origin of the window.
<i>:WIDTH</i> and <i>:HEIGHT</i>	Integers that specify the dimensions, in pixels, of the area.
<i>:EXPOSURES-P</i>	A CLX:BOOLEAN value that specifies whether exposure events are generated in <i>window</i> .

Null values for *:WIDTH* and *:HEIGHT* default to the current width of the window minus *:x* and the current height of the window minus *:y*, respectively. In other words, using the default values for *:x*, *:y*, *:WIDTH*, and *:HEIGHT* results in clearing the entire window. Passing in a zero *:WIDTH* or *:HEIGHT* is a null operation.

The format of the CLX:COPY-AREA function is:

CLX:COPY-AREA *src gcontext src-x src-y width height dst dst-x dst-y*

<i>src</i>	The CLX:DRAWABLE object that contains the area to copy.
<i>gcontext</i>	The CLX:GCONTEXT object of the source area.
<i>src-x</i> and <i>src-y</i>	Integers that specify the X and Y coordinates of the origin of the source area.
<i>width</i> and <i>height</i>	Integers that specify the dimensions, in pixels, of the source area.
<i>dst</i>	The CLX:DRAWABLE object in which the copy is displayed.

dst-x and *dst-y* Integers that specify the X and Y coordinates of the origin where the copied area is displayed.

Using the GContext of the source area ensures that the copy looks the same as the source.

The format of the CLX:COPY-PLANE function is:

CLX:COPY-PLANE *src gcontext plane src-x src-y width height dst dst-x dst-y*

src The CLX:DRAWABLE object that contains the area to modify.

gcontext The CLX:GCONTEXT object of the source area.

plane An integer that specifies which bit plane of the source area is copied.

src-x and *src-y* Integers that specify the X and Y coordinates of the origin of the source area.

width and *height* Integers that specify the dimensions, in pixels, of the source area.

dst The CLX:DRAWABLE object in which the modified area is displayed.

dst-x and *dst-y* Integers that specify the X and Y coordinates of the origin where the modified area is displayed.

The source and destination drawables must have the same root but may have different depths.

9.6 Creating Cursors

You can create your own pointer cursor from an image with the CLX:CREATE-CURSOR function, or a character of a font with the CLX:CREATE-GLYPH-CURSOR function. You can also recolor a cursor, query the hardware for best cursor size, and free the memory allocated to a cursor.

The format of the CLX:CREATE-CURSOR function is:

CLX:CREATE-CURSOR &KEY :SOURCE :MASK :X :Y :FOREGROUND :BACKGROUND

:SOURCE A CLX:PIXMAP object that contains the cursor shape.

:MASK A CLX:PIXMAP object that modifies the cursor shape. If no mask is provided, all bits of the source are used.

:X and :Y Integers that specify the X and Y coordinates of the cursor's hot spot.

:foreground and :background Two CLX:COLOR objects that specify the cursor's foreground and background colors.

The function returns a CLX:CURSOR object with the specified characteristics.

The format of the CLX:CREATE-GLYPH-CURSOR function is:

CLX:CREATE-GLYPH-CURSOR &KEY :SOURCE-FONT :SOURCE-CHAR :MASK-FONT
:MASK-CHAR :foreground :background

:SOURCE-FONT The CLX:FONT object that includes the glyph used to create the cursor.

:SOURCE-CHAR An integer that identifies the glyph in the source font to be used as the source for the cursor shape.

:MASK-FONT A CLX:FONT object that contains masks that control how the cursor is displayed on the screen. The default value is NIL.

:MASK-CHAR	An integer that identifies the glyph in the mask font to be used as the mask for the cursor shape. The default value is NIL.
:BACKGROUND	Two CLX:COLOR objects that specify the foreground and background colors of the cursor.

The :SOURCE-FONT, :SOURCE-CHAR, :FOREGROUND, and :BACKGROUND arguments are required. If no mask is provided, all bits of the source are used. If :MASK-CHAR is specified but :MASK-FONT is not, the :SOURCE-FONT is used. The CLX:CREATE-GLYPH-CURSOR function returns a CLX:CURSOR with the specified characteristics.

Example 9-4 creates a cursor by calling the CLX:CREATE-GLYPH-CURSOR function with the specified arguments. If none of the arguments are supplied, it creates the wait cursor in the DECW\$CURSOR font with a black or red foreground (depending on the depth of the display) and a white background. This example is derived from CURSORS.LSP in the directory referenced by the logical name LISP\$EXAMPLES.

Example 9-4: Creating a Cursor

```
(defun create-cursor (display &key
                           cursor-font
                           (cursor-font-name "decw$cursor")
                           (source-char clx:+decw$c_wait_cursor)
                           (background ;white
                           (clx:make-color :red 1 :green 1 :blue 1))
                           foreground ;set below if unsupplied
                           )
  (cond ((clx:font-p cursor-font)) ; fine, do nothing
        ((null cursor-font)
         (unless (simple-string-p cursor-font-name)
           (error "CURSOR-FONT-NAME must be of type ~
                  SIMPLE-STRING, received ~s" cursor-font-name)))
        (let ((font-name-list
              (clx:list-font-names display cursor-font-name
                                   :max-fs 1)))
          (if font-name-list
              (setf cursor-font (clx:open-font display (car font-name-list)))
              (error "No font matches ~s" cursor-font-name)))
        (t (error "CURSOR-FONT must be of type CLX:FONT, received ~s"
                  cursor-font)))
        (unless (integerp source-char)
          (error "SOURCE-CHAR must be of type INTEGER, received ~s"
                 source-char))
        (unless (clx:color-p background)
          (error "BACKGROUND must be of type CLX:COLOR, received ~s"
                 background)))
        ;; here we'll use black or red, depending on the depth of
        ;; the display.
        (cond ((clx:color-p foreground)
               ((null foreground)
                (setf foreground
                      ;; red or black, depending on display depth
                      (if
                        (< 1 (clx:screen-root-depth
                                  (clx:display-default-screen display)))
                        (clx:make-color :red 1 :green 0 :blue 0) ; red
                        (clx:make-color :red 0 :green 0 :blue 0))) ; black
               (t (error "FOREGROUND must be of type CLX:COLOR, received ~s"
                         foreground))))
```

(continued on next page)

Example 9-4 (Cont.): Creating a Cursor

```
(clx:create-glyph-cursor :source-font cursor-font
                           :mask-font   cursor-font
                           :source-char source-char
                           :mask-char   (1+ source-char)
                           :foreground  foreground
                           :background  background))
```

Use the CLX:RECOLOR-CURSOR function to change the foreground and background colors of a cursor. Its format is:

CLX:RECOLOR-CURSOR *cursor foreground background*

cursor The CLX:CURSOR object whose colors change.

foreground and *background* Two CLX:COLOR objects that specify the new colors.

The CLX:QUERY-BEST-CURSOR function queries the display hardware for the most efficient cursor size. Its format is:

CLX:QUERY-BEST-CURSOR *width height display*

width and *height* Integers that specify the desired dimensions of a cursor.

display The CLX:DISPLAY device on which the cursor will be used.

The function returns two integer values: the width and height, supported by the display hardware, that most closely match the desired dimensions.

To free the memory allocated to a cursor, call the CLX:FREE-CURSOR function. Its format is:

CLX:FREE-CURSOR *cursor*

where *cursor* is a CLX:CURSOR object originally returned by CLX:CREATE-CURSOR or CLX:CREATE-GLYPH-CURSOR.

Chapter 10

Using Pixmaps and Images

CLX enables clients to create and work with both on-screen graphics, such as lines and cursors, and off-screen graphics, such as pixmaps and images. This chapter describes how to work with off-screen graphics, and includes the following topics:

- Creating and freeing pixmaps
- Creating and managing bitmap files
- Working with images

Chapter 7 and Chapter 9 explain how to work with on-screen graphics objects.

10.1 Creating and Freeing Pixmaps

A pixmap is an area of memory into which clients can either draw objects or temporarily save part of a screen. Pixmaps are useful for graphics that are displayed repeatedly, such as cursors and icons; for creating tiling patterns; and for saving portions of windows that have been obscured. In addition, drawing complicated graphics sequences into pixmaps and then copying the pixmaps to a window may be faster than drawing the sequences directly into a window.

The `CLX:CREATE-PIXMAP` function returns a `CLX:PIXMAP` object with the specified width, height, and depth. It is an error to specify a width or height of zero, or a depth that is not supported by the drawable's root window. The format of the `CLX:CREATE-PIXMAP` function is:

`CLX:CREATE-PIXMAP &KEY :WIDTH :HEIGHT :DEPTH :DRAWABLE`

<code>:WIDTH</code> and <code>:HEIGHT</code>	Integers that specify the dimensions of the pixmap.
<code>:DEPTH</code>	An integer that specifies the depth of the pixmap.
<code>:DRAWABLE</code>	A <code>CLX:DRAWABLE</code> object that specifies the screen on which the pixmap is created.

The functions that return information about pixmaps are listed in Table 10-1. None of these functions is a valid `SETF` place.

Table 10–1: Information Functions for Pixmaps

Function	Return Value
CLX:PIXMAP-DISPLAY	The CLX:DISPLAY object on which the pixmap was created.
CLX:PIXMAP-EQUAL	A CLX:BOOLEAN value that indicates whether the two arguments are the same pixmap.
CLX:PIXMAP-ID	The integer resource-id of the pixmap.
CLX:PIXMAP-P	A CLX:BOOLEAN value that indicates whether the argument is of type CLX:PIXMAP.

Objects of type CLX:PIXMAP-FORMAT show what formats the display hardware supports, and are returned by the CLX:DISPLAY-PIXMAP-FORMATS function. The functions that return information about CLX:PIXMAP-FORMAT objects are described in Table 10–2. All of these functions can be used with SETF.

Table 10–2: Information Functions for Pixmap Formats

Function	Return Value
CLX:PIXMAP-FORMAT-BITS-PER-PIXEL	The number of bits used to hold each pixel, a member of (1 4 8 16 24 32).
CLX:PIXMAP-FORMAT-DEPTH	A CLX:IMAGE-DEPTH value that specifies the depth of the display.
CLX:PIXMAP-FORMAT-PAD	The number of bits to a multiple of which each scanline is padded, a member of (8 16 32).

When your program no longer needs a pixmap, use the CLX:FREE-PIXMAP function to free the memory allocated to it. The format of this function is:

CLX:FREE-PIXMAP *Pixmap*

where *Pixmap* is the CLX:PIXMAP object to be freed in server memory. CLX:FREE-PIXMAP first deletes the association between the pixmap and its resource-id and then frees the pixmap storage.

10.2 Creating and Managing Bitmap Files

A bitmap is a pixmap of depth 1, or an array of bits. CLX enables clients to create files of bitmap data and then use those files to create either bitmaps or pixmaps. To create a bitmap data file, use the CLX:WRITE-BITMAP-FILE function. Its format is:

**CLX:WRITE-BITMAP-FILE *pathname image*
&OPTIONAL *name*
&KEY :PIXMAP-P :WIDTH :HEIGHT :X-HOT :Y-HOT :DRAWABLE**

pathname A pathname that specifies the file to write.
image The CLX:IMAGE or CLX:PIXMAP object to write.
name A CLX:STRINGABLE value that names the bitmap and is used for variable prefixes. The default is "IMAGE-".
:PIXMAP-P A CLX:BOOLEAN value that specifies whether the *image* argument is a pixmap or an image.

<code>:WIDTH</code> and <code>:HEIGHT</code>	Two integers that specify the dimensions of the image or pixmap.
<code>:X-HOT</code> and <code>:Y-HOT</code>	Two integers that specify the hot spot of a cursor. The defaults are 0.
<code>:DRAWABLE</code>	A CLX:DRAWABLE object that provides information on the screen and display.

The CLX:WRITE-BITMAP-FILE function creates a C include file in standard X11 format.

To create an image or a pixmap from an X11 bitmap file, use the CLX:READ-BITMAP-FILE function. Its format is:

`CLX:READ-BITMAP-FILE pathname &KEY :PIXMAP-P :DRAWABLE`

<code>pathname</code>	A pathname or string that specifies the file to read.
<code>:PIXMAP-P</code>	A CLX:BOOLEAN value that specifies whether the function returns a pixmap or an image.
<code>:DRAWABLE</code>	A CLX:DRAWABLE object that identifies the screen on which to create the pixmap. This argument is required only if <code>:PIXMAP-P</code> is true.

If `:PIXMAP-P` is true, the function returns three values: the CLX:PIXMAP object and two integers that specify the X and Y coordinates of a cursor's hot spot. If `:PIXMAP-P` is false, the function returns the CLX:IMAGE object only.

Objects of type CLX:BITMAP-FORMAT store the format in which images are transmitted and received by the server, and are returned by the CLX:DISPLAY-BITMAP-FORMAT function. Bitmaps are represented in scanline order. The functions that return information about objects of type CLX:BITMAP-FORMAT are described in Table 10-3. None of these functions is a valid SETF place.

Table 10-3: Information Functions for Bitmap-Formats

Function	Return Value
<code>CLX:BITMAP-FORMAT-UNIT</code>	The number of bits in a unit of the scanline, a member of (8 16 32).
<code>CLX:BITMAP-FORMAT-PAD</code>	The number of bits to a multiple of which each scanline is padded, a member of (8 16 32).
<code>CLX:BITMAP-FORMAT-LSB-FIRST-P</code>	A CLX:BOOLEAN value that indicates whether the least significant bit or the most significant bit is the leftmost bit in each unit.

10.3 Working with Images

Off-screen images may be simple patterns stored directly in an array or more complex forms drawn into a pixmap. Instances of the CLX:IMAGE structure include a pointer to image data that may be stored in an array or pixmap. Clients can perform operations on images which are not possible with pixmaps. The following is a list of direct manipulations that can be performed on images:

- Destroying an image
- Getting a pixel from an image
- Storing a pixel in an image
- Extracting part of an image
- Adding a constant to the image

CLX lets clients store image data as a bitmap (:BITMAP format), as a stack of bitmaps (:XY-PIXMAP format), or as a list of pixel values (:Z-PIXMAP format). To create a CLX:IMAGE object, use the CLX:CREATE-IMAGE function. Its format is:

```
CLX:CREATE-IMAGE &KEY :WIDTH :HEIGHT :DEPTH :DATA :RED-MASK :GREEN-MASK  
:BLUE-MASK :BITS-PER-PIXEL :FORMAT :SCANLINE-PAD  
:BYTES-PER-LINE :BYTE-LSB-FIRST-P :BIT-LSB-FIRST-P
```

:WIDTH and :HEIGHT	Two CLX:CARD16 values that specify the dimensions of the image. These arguments are required.
:DEPTH	A CLX:CARD8 value that specifies the depth of the image. The default is 1.
:DATA	A sequence in which to store the data. Its type depends on the value of the :FORMAT argument: :FORMAT :BITMAP (ARRAY CLX:CARD8 (*)) :FORMAT :XY-PIXMAP (LIST CLX:BITMAP) :FORMAT :Z-PIXMAP CLX:PIXARRAY
:RED-MASK :GREEN-MASK and :BLUE-MASK	Three CLX:PIXEL values that specify the red, green, and blue values for pixels in :Z-PIXMAP format images.
:BITS-PER-PIXEL	The number of bits used to store a pixel, a member of (1 4 8 16 24 32).
:FORMAT	The type of image returned, one of :BITMAP, :XY-PIXMAP, or :Z-PIXMAP.
:SCANLINE-PAD	An integer that specifies the number of bits each scanline is padded to a multiple of, a member of (8 16 32).
:BYTES-PER-LINE	An integer that specifies the number of bytes in a scanline. The default is calculated from the :WIDTH and :SCANLINE-PAD arguments.
:BYTE-LSB-FIRST-P	A CLX:BOOLEAN value that specifies whether the least significant byte is first.
:BIT-LSB-FIRST-P	A CLX:BOOLEAN value that specifies whether the least significant bit is first.

The functions that return information on CLX:IMAGE objects are listed in Table 10-4.

Table 10-4: Information Functions for CLX:IMAGE Objects

Function	Use with SETF?	Return Value
CLX:IMAGE-BIT-LSB-FIRST-P	Yes	A CLX:BOOLEAN value that specifies whether the least significant bit is first.
CLX:IMAGE-BLUE-MASK	Yes	A CLX:PIXEL value that specifies the blue value in a pixel, in :Z-PIXMAP format.

(continued on next page)

Table 10-4 (Cont.): Information Functions for CLX:IMAGE Objects

Function	Use with SETF?	Return Value
CLX: IMAGE-BYTE-LSB-FIRST-P	Yes	A CLX:BOOLEAN value that specifies whether the least significant byte is first.
CLX: IMAGE-BYTES-PER-LINE	Yes	An integer that specifies the number of bytes in a scanline.
CLX: IMAGE-FORMAT	Yes	A keyword that specifies the format of the image data: :BITMAP, :XY-PIXMAP, or :Z-PIXMAP.
CLX: IMAGE-GREEN-MASK	Yes	A CLX:PIXEL value that specifies the green value in a pixel, in :Z-PIXMAP format.
CLX: IMAGE-DEPTH	No	A CLX:CARD8 value that specifies the depth of the image. The default value is 1, which means a bitmap.
CLX: IMAGE-HEIGHT	No	A CLX:CARD16 value that specifies the height of the image. The default value is 0.
CLX: IMAGE-RED-MASK	Yes	A CLX:PIXEL value that specifies the red value in a pixel, in :Z-PIXMAP format.
CLX: IMAGE-SCANLINE-PAD	Yes	Each scanline in the image is a multiple of this value, which may be 8, 16, or 32.
CLX: IMAGE-WIDTH	No	A CLX:CARD16 value that specifies the width of the image. The default value is 0.

The functions listed in Table 10-4 access images in all three formats. Tables 10-5 and 10-6 describe the additional accessors for images in :XY-PIXMAP and :Z-PIXMAP formats. None of the following functions is a valid SETF place:

Table 10-5: Information Functions for :XY-PIXMAP Images

Function	Return Value
CLX: IMAGE-XY-BITMAP-LIST	A list of the CLX:BITMAP values that contain the image data.

Table 10-6: Information Functions for :Z-PIXMAP Images

Function	Return Value
CLX: IMAGE-Z-BITS-PER-PIXEL	The number of bits used to store each pixel, a member of (1 4 8 16 24 32).
CLX: IMAGE-Z-PIXARRAY	The CLX:PIXARRY value that contains the image data.

10.3.1 Getting Images from the Server

Images can be brought into LISP memory with the CLX:GET-IMAGE and CLX:GET-RAW-IMAGE functions. The CLX:GET-IMAGE function returns a CLX:IMAGE object; the CLX:GET-RAW-IMAGE function returns a sequence of bytes.

The format of the CLX:GET-IMAGE function is:

```
CLX:GET-IMAGE drawable &KEY :X :Y :WIDTH :HEIGHT :PLANE-MASK :FORMAT
```

<i>drawable</i>	The CLX:DRAWABLE object that contains the image to be stored in memory.
:X and :Y	Integer X and Y coordinates of the origin of the image in the drawable. These arguments are required.
:WIDTH and :HEIGHT	Integers that specify the dimensions of the image. These arguments are required.
:PLANE-MASK	A CLX:PIXEL value that specifies the planes to be retrieved. The default is #xFFFFFFF, which means all planes.
:FORMAT	The data format stored in the server, either :XY-PIXMAP or :Z-PIXMAP. The default is :Z-PIXMAP.

The format of the CLX:GET-RAW-IMAGE function is:

```
CLX:GET-RAW-IMAGE drawable &KEY :DATA :START :X :Y :WIDTH :HEIGHT :PLANE-MASK  
:FORMAT :RESULT-TYPE
```

<i>drawable</i>	The CLX:DRAWABLE that contains the image to be stored in memory.
:DATA	A sequence of 8-bit quantities.
:START	A CLX:ARRAY-INDEX value specifying the starting position in the data. The default value is 0.
:X and :Y	Integers that specify the X and Y coordinates of the origin of the image in the drawable.
:WIDTH and :HEIGHT	Integers that specify the dimensions of the image.
:PLANE-MASK	A CLX:PIXEL value that specifies the planes to be retrieved. The default value is #xFFFFFFF.
:FORMAT	A keyword that specifies the type of image format, either :XY-PIXMAP or Z-PIXMAP.
:RESULT-TYPE	The LISP type of the return value. The default is '(VECTOR (UNSIGNED-BYTE 8)).

The function returns a sequence of 8-bit quantities, in transmission format. If :DATA is given, it is modified in place (and returned); otherwise, a new sequence is created and returned, with a size computed from the other arguments and the returned depth.

10.3.2 Displaying Images

Images can be displayed with the CLX:PUT-IMAGE and CLX:PUT-RAW-IMAGE functions. The CLX:PUT-IMAGE function outputs a CLX:IMAGE object; the CLX:PUT-RAW-IMAGE function outputs a sequence of bytes.

The format of the CLX:PUT-IMAGE function is:

```
CLX:PUT-IMAGE drawable gcontext image  
&KEY :SRC-X :SRC-Y :X :Y :WIDTH :HEIGHT :BITMAP-P
```

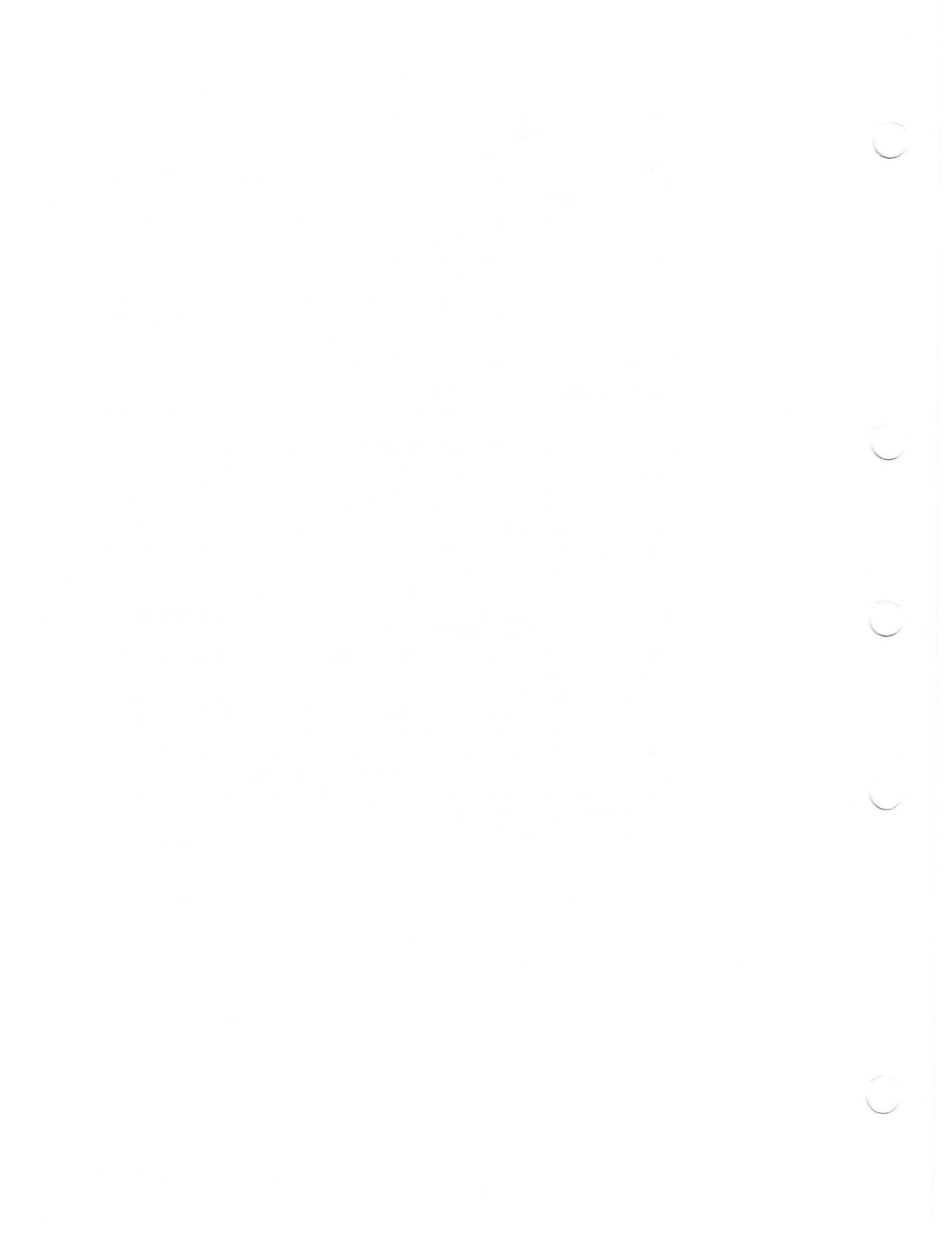
<i>drawable</i>	The CLX:DRAWABLE object in which the image is displayed.
<i>gcontext</i>	The CLX:GCONTEXT object that specifies the graphics characteristics of the image.
<i>image</i>	The CLX: IMAGE object to be displayed.
:SRC-X and :SRC-Y	Integers that specify the X and Y coordinates of the image. The defaults are 0.
:X and :Y	Integers that specify the X and Y coordinates of the image. These arguments are required.
:WIDTH and :HEIGHT	Integers that specify the dimensions of the image.
:BITMAP-P	A CLX:BOOLEAN value that indicates whether the image is a bitmap when depth is 1. Setting this argument to T causes <i>gcontext</i> to supply the foreground and background pixels.

The format of the CLX:PUT-RAW-IMAGE function is:

```
CLX:PUT-RAW-IMAGE drawable gcontext data
                  &KEY :START :DEPTH :X :Y :WIDTH :HEIGHT :LEFT-PAD :FORMAT
```

<i>drawable</i>	The CLX:DRAWABLE object where the image is to be displayed.
<i>gcontext</i>	The CLX:GCONTEXT object that specifies the graphics characteristics used to display the image: function, plane-mask, subwindow-mode, clip-x, clip-y, clip-mask, foreground, and background.
<i>data</i>	A sequence of 8-bit quantities that contain the image.
:START	A CLX:ARRAY-INDEX value that specifies the starting point within <i>data</i> . The default value is 0.
:DEPTH	An integer that specifies the depth of the drawable.
:X and :Y	Two integers that specify the X and Y coordinates of the rectangle where the image is drawn in the drawable.
:WIDTH and :HEIGHT	Two integers that specify the dimensions of the rectangle where the image is drawn in the drawable.
:LEFT-PAD	An integer that specifies the number of bits to the left of each scanline which are ignored by the server. The default value is 0; this value is required when the format is :Z-PIXMAP.
:FORMAT	The type of image format, either :XY-PIXMAP or :Z-PIXMAP.

The *data* argument must be a sequence of 8-bit quantities already in the appropriate format for transmission. The :START argument is the starting index in *data*; the end is computed from the other arguments. If :DEPTH is 1 and :FORMAT is :XY-PIXMAP, *data* is a bitmap.



Chapter 11

Writing Text

This chapter describes how to write text with CLX, and includes the following topics:

- Characters and fonts
- Specifying fonts
- Computing text size
- Drawing text on the screen

VMS DECwindows provides a font compiler that enables programmers to convert ASCII files into binary form. For information on the font compiler, see the VMS *DECwindows Guide to Xlib Programming: MIT C Binding*.

11.1 Characters and Fonts

A character is the smallest unit of text the server can display. A font is a group of characters that have the same style and size. CLX refers to characters by using indexes into fonts. The accessor functions for characters are listed in Table 11-1. None of these functions is a valid SETF place.

Table 11-1: Information Functions for Characters

Function Name	Return Value
CHAR-ASCENT	An integer that specifies the distance from the baseline to the top of the bounding box.
CHAR-ATTRIBUTES	A positive integer whose 1 bits specify the attributes of the character defined in the bitmap distribution format (BDF) file. A character is not guaranteed to have any attributes.
CHAR-DESCENT	An integer that specifies the distance from the baseline to the bottom of the bounding box.
CHAR-LEFT-BEARING	An integer that specifies the distance from the origin to the left edge of the bounding box.
CHAR-RIGHT-BEARING	An integer that specifies the distance from the origin to the right edge of the bounding box.
CHAR-WIDTH	An integer that specifies the distance from the origin of the current character to the origin of the next character.

Each character accessor function takes two arguments: the first is a CLX:FONT object, the second an index of type CARD16. For 8-bit fonts, the index is usually the ASCII value of the character so you can pass a CHARACTER as the second argument. For example:

```

Lisp> *font*
#<Font 9x15>
Lisp> (clx:char-ascent *font* #\f)
12
Lisp> (clx:char-descent *font* #\f)
3

```

The character accessor functions return NIL if the index is out of bounds for the font.

The functions that return information about CLX:FONT objects are described in Table 11-2. None of these functions is a valid SETF place.

Table 11-2: Information Functions for CLX:FONT Objects

Function Name	Return Value
FONT-ALL-CHARS-EXIST-P	A CLX:BOOLEAN value that indicates whether all characters in the font have nonzero bounding boxes.
FONT-ASCENT	An integer that specifies the distance from the baseline to the top of the bounding box.
FONT-DEFAULT-CHAR	The integer index to the character used when an undefined or nonexistent character is printed.
FONT-DESCENT	An integer that specifies the distance from the baseline to the bottom of the bounding box.
FONT-DIRECTION	A CLX:DRAW-DIRECTION value that gives a hint about the direction in which the font is painted. The possible values are :LEFT-TO-RIGHT and :RIGHT-TO-LEFT.
FONT-MAX-CHAR	The integer index to the last character in the font.
FONT-MAX-BYTE1	An integer that specifies the last row of the font.
FONT-MAX-BYTE2	An integer that specifies the last byte in a 16-bit font. In 8-bit fonts, this is the same as FONT-MAX-CHAR.
FONT-MIN-CHAR	The integer index to the first character in the font.
FONT-MIN-BYTE1	An integer that specifies the first row of the font.
FONT-MIN-BYTE2	An integer that specifies the first byte in a 16-bit font. In 8-bit fonts, this is the same as FONT-MIN-CHAR.
FONT-NAME	A string that names the font
FONT-PROPERTIES	A CLX:FONT-PROPS value that contains a list of additional font properties, as alternating keywords and integers.
FONT-PROPERTY	This function requires two arguments: the first is a CLX:FONT object, the second a keyword naming a font property. The return value is an integer that specifies the value of that property.
MAX-CHAR-ASCENT	The largest ascent of any character in the font.
MAX-CHAR-ATTRIBUTES	The largest number of attributes of any character in the font.
MAX-CHAR-DESCENT	The largest descent of any character in the font.
MAX-CHAR-LEFT-BEARING	The largest left-bearing of any character in the font.
MAX-CHAR-RIGHT-BEARING	The largest right-bearing of any character in the font.

(continued on next page)

Table 11–2 (Cont.): Information Functions for CLX:FONT Objects

Function Name	Return Value
MAX-CHAR-WIDTH	The largest width of any character in the font.
MIN-CHAR-ASCENT	The smallest ascent of any character in the font.
MIN-CHAR-ATTRIBUTES	The smallest number of attributes owned by any character in the font.
MIN-CHAR-DESCENT	The smallest descent of any character in the font.
MIN-CHAR-LEFT-BEARING	The smallest left-bearing of any character in the font.
MIN-CHAR-RIGHT-BEARING	The smallest right-bearing of any character in the font.
MIN-CHAR-WIDTH	The smallest width of any character in the font.

Clients can compute the bounding box of the font from the maximum and minimum character metrics. The upper-left coordinate of the bounding box whose origin is at (x, y) is defined as follows:

```
(+ x (CLX:MIN-CHAR-LEFT-BEARING font)) , (- y (CLX:MAX-CHAR-ASCENT font))
```

The width of the font bounding box is defined as follows:

```
(+ (CLX:MIN-CHAR-LEFT-BEARING font) (CLX:MAX-CHAR-RIGHT-BEARING font))
```

The height of the font bounding box is defined as follows:

```
(+ (CLX:MAX-CHAR-ASCENT font) (CLX:MAX-CHAR-DESCENT font))
```

The following additional functions are defined for CLX:FONT objects:

CLX:FONT-DISPLAY	Returns the CLX:DISPLAY object on which the font is open.
CLX:FONT-ID	Returns the integer resource-id of the font.
CLX:FONT-EQUAL	Predicate returns true if arguments are identical.
CLX:FONT-P	Predicate returns true if argument is a CLX:FONT object.

11.2 Specifying Fonts

The font used when drawing text is specified by the :FONT slot of the GContext passed to the draw function. You can provide a font value when you create the GContext, or you can first open a font and then set the slot to that value. In either case, you need to know the name of the font.

The CLX:LIST-FONT-NAMES function returns a list of font names that match a given pattern and the number of matches found. Its format is:

```
CLX:LIST-FONT-NAMES display pattern &KEY :MAX-FONTS :RESULT-TYPE
```

display	A CLX:DISPLAY object.
pattern	A string that specifies the search pattern to be matched by the font names that are returned.
:MAX-FONTS	An integer that specifies the maximum number of font names returned. The default value is 1024.
:RESULT-TYPE	The LISP type of the return value. The default is 'LIST.

See the VMS DECwindows Guide to Xlib Programming: MIT C Binding for a complete list of DECwindows font names.

The CLX:LIST-FONTS function returns “pseudo” fonts that contain basic font metrics and properties but no per-character metrics and no resource-ids. These pseudo fonts are automatically converted to real fonts dynamically as needed, by issuing an OpenFont protocol request. The format of the CLX:LIST-FONTS function is:

CLX:LIST-FONTS *display pattern* &KEY :MAX-FONTS :RESULT-TYPE

<i>display</i>	A CLX:DISPLAY object.
<i>pattern</i>	A string that specifies the search pattern to be matched by the names of fonts that are returned.
:MAX-FONTS	An integer that specifies the maximum number of fonts returned. The default value is 1024.
:RESULT-TYPE	The LISP type of the return value. The default is 'LIST.

Once you have a font name, use the CLX:OPEN-FONT function to load it into LISP memory. Font objects may be cached and reference counted locally within the display object. This function may not execute a CLX:WITH-DISPLAY if the font is cached. The format of the CLX:OPEN-FONT function is:

CLX:OPEN-FONT *display name*

<i>display</i>	A CLX:DISPLAY object.
<i>name</i>	A string that specifies the font.

Example 11-1 shows how to use CLX:LIST-FONT-NAMES to search for a font whose name matches a wildcard string. The function OPEN-WILDCARD-FONT opens and returns the first font that matches the specified string. If no font name matches the wildcard, the function returns NIL.

Example 11-1: Specifying Fonts

```
Lisp> (defun open-wildcard-font (display wildcard-string)
  (let ((font-list (clx:list-font-names display wildcard-string
                                         :max-fon
                                         t 1)))
    (when font-list
      (clx:open-font display (car font-list)))))

OPEN-WILDCARD-FONT
Lisp> (open-wildcard-font *d* "*Lubalin*24*")
#<Font -Adobe-ITC Lubalin Graph-R-Normal--24-240-75-75-P-139-ISO8859-1>
```

This example assumes that *D* is a CLX:DISPLAY object previously returned by the CLX:OPEN-DISPLAY function.

The CLX:DISCARD-FONT-INFO function discards any state that can be reobtained with a QueryFont request. This is simply a performance hint for memory-limited systems. The format of this function is:

CLX:DISCARD-FONT-INFO *font*

where *font* is the CLX:FONT object whose state is to be discarded.

The CLX:CLOSE-FONT function may not generate a protocol request if the font is reference counted locally or if it is a pseudo font.

CLX:CLOSE-FONT *font*

where *font* is the CLX:FONT object that is no longer needed.

The CLX:FONT-PATH function returns a list (by default) of strings or pathnames that show where the server looks for font files. You can change the search path with SETF. The format of the CLX:FONT-PATH function is:

CLX:FONT-PATH *display* &KEY :RESULT-TYPE

display A CLX:DISPLAY object.
:RESULT-TYPE The LISP type of the return value.

11.3 Computing Text Size

CLX provides functions that calculate the width and extent of 8-bit strings. To draw 2-byte strings, you must provide a translate function that returns indexes into the 2-byte fonts.

The CLX:TEXT-WIDTH function computes the sum of the width of each character in a specified string. Its format is:

CLX:TEXT-WIDTH *font sequence* &KEY :START :END :TRANSLATE

font Either a CLX:FONT or a CLX:GCONTEXT object that specifies the font used.
sequence A sequence of characters (that is, a string) to measure.
:START and Two integers that specify the first and last characters in the sequence.
:END The default value for :START is 0; for :END the length of the string.
:TRANSLATE A function that translates *sequence* into font indexes. The default function is CLX:TRANSLATE-DEFAULT.

The CLX:TEXT-EXTENTS function computes the bounding box of a specified string. Its format is:

CLX:TEXT-EXTENTS *fonts sequence* &KEY :START :END :TRANSLATE

fonts Either a CLX:FONT or a CLX:GCONTEXT object that specifies the font used.
sequence A sequence of characters (that is, a string) to measure.
:START and Integers that specify the first and last characters in *sequence*. The default for :START is 0; for :END the end of the sequence.
:END The default for :END is the end of the sequence.
:TRANSLATE A function that translates *sequence* into font indexes. The default function is CLX:TRANSLATE-DEFAULT.

The function returns the width, ascent, descent, left-bearing, right-bearing, font-ascent, font-descent, and direction of the specified string, and an array index value. The index indicates the position within the string where the translation failed; if the entire string was translated, the last return value is NIL. The :TRANSLATE function will always be called with a 16-bit destination buffer.

You need the CLX:TRANSLATE-DEFAULT function only if you are using two-byte fonts. Its format is:

CLX:TRANSLATE-DEFAULT *src src-start src-end font dst dst-start*

src A sequence of characters to measure or draw.
src-start and Two CLX:ARRAY-INDEX values that specify the first and last positions in
src-end the source.
font The CLX:FONT object to use.

<i>dst</i>	A vector that holds the font indexes.
<i>dst-start</i>	A CLX:ARRAY-INDEX value that specifies the starting position within the destination.

The *dst* is guaranteed to have room for (- src-end src-start) integer elements, starting at *dst-start*; whether *dst* holds 8-bit or 16-bit elements depends on context. If known, *font* is the current font. The function should translate as many elements of *src* as possible into indexes in the current font, and store them into *dst*. The first return value should be the *src* index of the first untranslated element. If no further elements need to be translated, the second return value should be NIL. If a horizontal motion is required before further translation, the second return value should be the delta in X coordinate. If known, the pixel width of the translated text can be returned as the third value; this can allow for appending of subsequent output to the same protocol request if no overall width has been specified at the higher level.

11.4 Drawing Text on the Screen

In the text-drawing functions below, if width is specified, it is assumed to be the total pixel width of whatever string of glyphs is actually drawn. Specifying width will allow for appending the output of subsequent calls to the same protocol request, provided GContext has not been modified in the interim. If width is not specified, appending of subsequent output may not occur (unless the translate function returns the width). Specifying width is simply a hint, for performance.

The functions in this section expect the text to be drawn to be a LISP string. If you provide your own :TRANSLATE function it should output a string or a vector of characters, which is effectively the same thing.

The CLX:DRAW-GLYPH and CLX:DRAW-GLYPHS functions draw only pixels set to on in the glyph, that is, the foreground but not the background. The CLX:DRAW-GLYPH function paints a single character; the CLX:DRAW-GLYPHS function can handle strings. The format of the CLX:DRAW-GLYPH function is:

```
CLX:DRAW-GLYPH drawable gcontext x y elt
                  &KEY :TRANSLATE :WIDTH :SIZE
```

<i>drawable</i>	The CLX:DRAWABLE object where the glyph will be displayed.
<i>gcontext</i>	The CLX:GCONTEXT object that determines the graphics characteristics of the glyph. It contains the font.
<i>x</i> and <i>y</i>	Integers that specify the X and Y coordinates of the origin of the glyph.
<i>elt</i>	An element (such as a character) to draw.
:TRANSLATE	A function that translates <i>elt</i> into a font index. The default function is CLX:TRANSLATE-DEFAULT.
:WIDTH	An integer that specifies the width of the glyph in pixels.
:SIZE	A CLX:INDEX-SIZE value that specifies the size of the font. The default value is :DEFAULT, which means an 8-bit font.

The first return value is true if *elt* is output, or NIL if the :TRANSLATE function refuses to output it (for example, if there is no font specified in *gcontext*). The second result is the width of the glyph displayed, measured in pixels, if known, or NIL if unknown.

The format of the CLX:DRAW-GLYPHS function is:

```
CLX:DRAW-GLYPHS drawable gcontext x y sequence
                  &KEY :START :END :TRANSLATE :WIDTH :SIZE
```

<i>drawable</i>	The CLX:DRAWABLE object where the glyphs will be displayed.
<i>gcontext</i>	The CLX:GCONTEXT object that determines the graphics characteristics of the text, such as the font.
<i>x</i> and <i>y</i>	Integers that specify the X and Y coordinates of the origin of the first character displayed.
<i>sequence</i>	A sequence of characters (that is, a string) to draw.
:START and :END	Two CLX:ARRAY-INDEX values that specify the first and last positions within the sequence. The default value for :START is 0. If :END is null, the length of the sequence is used.
:TRANSLATE	A function that translates <i>sequence</i> into font indexes. The default function is CLX:TRANSLATE-DEFAULT.
:WIDTH	An integer that specifies the width of the glyphs, in pixels.
:SIZE	A CLX:INDEX-SIZE that specifies the size of the font. The default value is :DEFAULT, which means an 8-bit font.

The function returns two values: the first is the new start, if the end was not reached; the second is the overall width if known.

The CLX:DRAW-IMAGE-GLYPH and CLX:DRAW-IMAGE-GLYPHS functions display both the foreground and background of the character matrix. This is useful when writing over previously displayed text. The format of the CLX:DRAW-IMAGE-GLYPH function is:

```
CLX:DRAW-IMAGE-GLYPH drawable gcontext x y elt
                      &KEY :TRANSLATE :WIDTH :SIZE
```

<i>drawable</i>	The CLX:DRAWABLE object where the glyph will be displayed.
<i>gcontext</i>	The CLX:GCONTEXT object that determines the graphics characteristics of the glyph. It contains the font.
<i>x</i> and <i>y</i>	Integers that specify the X and Y coordinates of the origin of the glyph.
<i>elt</i>	An item (that is, a character) to draw.
:TRANSLATE	A function that translates <i>elt</i> into a font index. The default function is CLX:TRANSLATE-DEFAULT.
:WIDTH	An integer that specifies the width of the glyph, in pixels.
:SIZE	A CLX:INDEX-SIZE value that specifies the size of the font. The default value is :DEFAULT, which means an 8-bit font.

The function returns two values. The first is T if *elt* is output, NIL if the TRANSLATE function refuses to output it. The second result is the overall width, if known. The :TRANSLATE function may specify a new font.

The format of the CLX:DRAW-IMAGE-GLYPHS function is:

```
CLX:DRAW-IMAGE-GLYPHS drawable gcontext x y sequence
                      &KEY :START :END :WIDTH :TRANSLATE :SIZE
```

<i>drawable</i>	The CLX:DRAWABLE object where the glyphs will be displayed.
<i>gcontext</i>	The CLX:GCONTEXT object that determines the graphics characteristics of the glyphs, such as the font.
<i>x</i> and <i>y</i>	Integers that specify the X and Y coordinates of the origin of the first character displayed.
<i>sequence</i>	A sequence of items (that is, a string) to draw.
:START and :END	Two CLX:ARRAY-INDEX values that specify the first and last positions within the sequence. The default value for :START is 0. If :END is null, the length of the sequence is used.

:WIDTH	An integer that specifies the width of the glyphs, in pixels.
:TRANSLATE	A function that translates <i>sequence</i> into font indexes. The default function is CLX:TRANSLATE-DEFAULT.
:SIZE	A CLX:INDEX-SIZE value that specifies the size of the font. The default value is :DEFAULT, meaning an 8-bit font.

The first return value is the new start if :END was not reached. The second result is overall width if known.

Example 11-2 shows the difference between CLX:DRAW-GLYPHS and CLX:DRAW-IMAGE-GLYPHS.

Example 11-2: Drawing Text

```
(defun clx-text-example (&optional (machine (machine-instance)))
  (let* ((d (clx:open-display machine))
         (s (first (clx:display-roots d)))
         (r (clx:screen-root s))
         (black-pixel (clx:screen-black-pixel s))
         (white-pixel (clx:screen-white-pixel s))
         (w (clx:create-window
              :parent r :x 10 :y 100
              :width 300 :height 100
              :background white-pixel
              :event-mask '(:button-press)))
         (font (clx:open-font
                 d
                 (first (clx:list-font-names d
                                              "*times*bold*24*")))))
    (gc (clx::create-gcontext :drawable w
                               :foreground black-pixel
                               :background white-pixel
                               :font font)))
  ;; Display the window
  (setf (clx:wm-name w) "Drawing Text")
  (clx:map-window w)
  (clx:display-finish-output d)

  ;; Make some distracting background
  (clx:draw-lines w gc '(0 100 100 0 100 100 200 0 200 100 300 0))

  ;; Draw some text
  (clx:draw-glyphs w gc 10 40 "These are glyphs.")
  (clx:draw-image-glyphs w gc 10 80 "These are image-glyphs.")

  (clx:display-force-output d)

  (clx:event-case
    (d :force-output-p t)
    (:button-press ()
      (clx:destroy-window w)
      t))

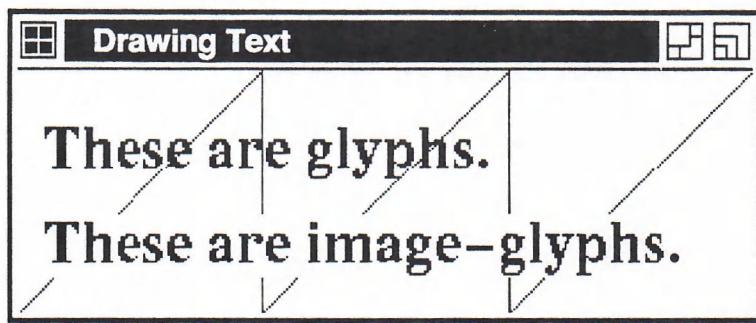
  (clx:display-force-output d)))
```

Figure 11-1 shows the output of this example on a system where the first font that matches "*times*bold*24*" is:

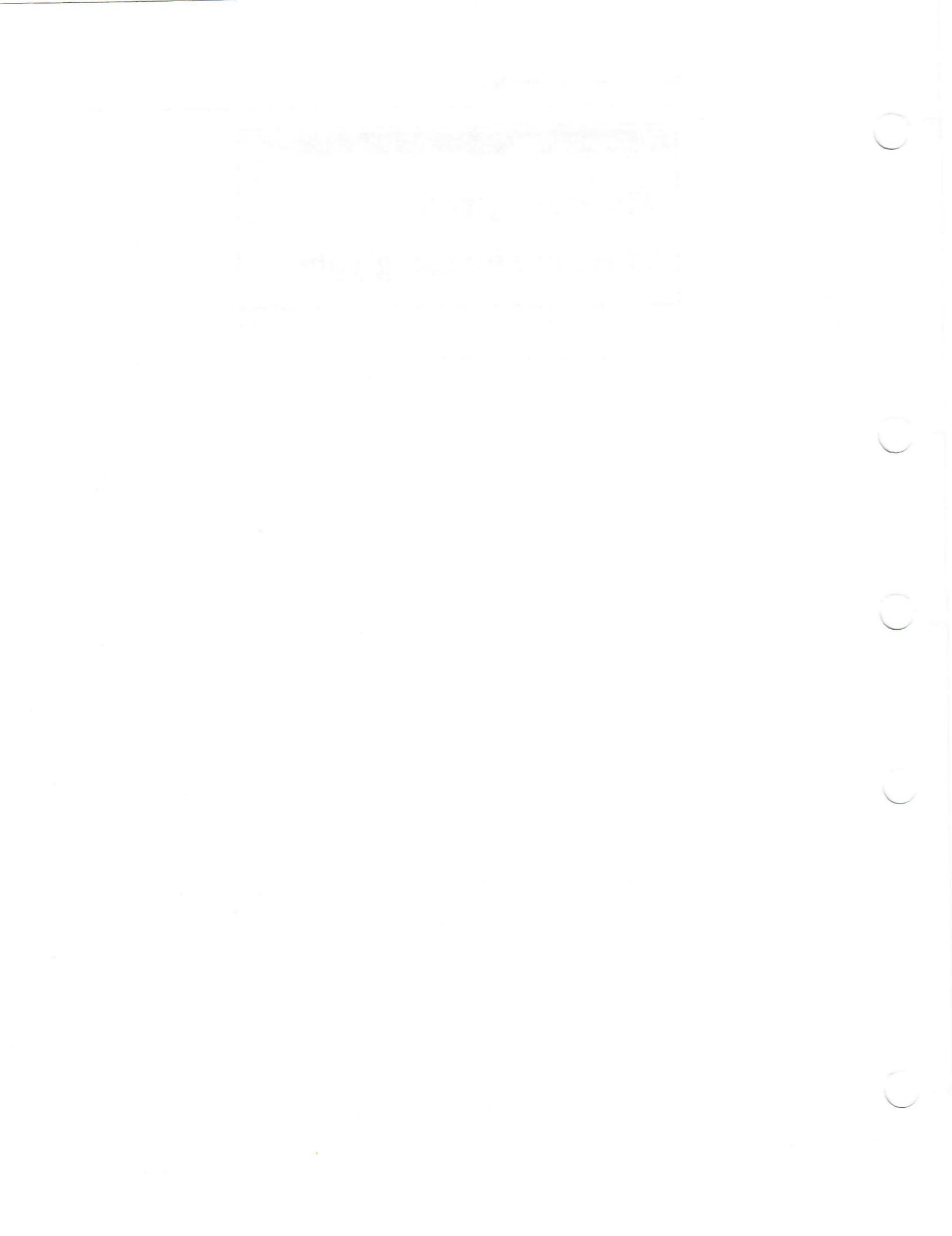
"-Adobe-Times-Bold-R-Normal--24-240-75-75-P-132-ISO8859-1"

Depending on which fonts are loaded on your system and the order in which they were loaded, Example 11-2 may not run on your system, or may not produce the same results.

Figure 11–1: Drawing Text



MLO-003394



Chapter 12

Event Functions

An event is a report from the X server. Events indicate either a change in the state of a device (such as a mouse button being pressed) or the execution of a routine called by a client (such as a window being mapped). When an event occurs, CLX notifies interested clients by placing information about the event in a queue. The visible window that contains the pointer cursor when an event occurs is called the source window. CLX searches the window hierarchy upward from the source window until one of the following applies:

- CLX finds a window in which one or more clients have selected that event type. This window is called the event window. After CLX locates an event window, it sends information about the event to the appropriate clients.
- CLX finds a window whose :DO-NOT-PROPAGATE-MASK component has been set by a client to include that event type. Setting this component specifies that CLX should not notify clients about events occurring in the window and its children.
- CLX reaches the top of the window hierarchy without finding an event window.

The CLX event routines allow client programs to select the types of events they want to be informed of and to manage event processing. This chapter explains how to perform the following operations:

- Select the event types to be reported to your window.
- Process events reported by CLX.
- Control which clients are notified of device events.
- Send an event to a window.

The last section in this chapter provides an alphabetical reference to each event type.

12.1 Selecting Events

CLX can report 33 types of events related to keyboards, mice, windowing, and graphics operations. Table 12-1 lists the event types, grouped by category. Usually, CLX sends information about an event only to clients that have specified an interest in that event type. Clients use one of the following methods to select the event types in which they are interested:

- Specifying an event mask when creating a window with CLX:CREATE-WINDOW.
- Changing a window's event mask with SETF or CLX:WITH-STATE.

- Creating or changing the graphics exposure mask of a graphics context.

Five events, however, are useful to almost all clients and are reported regardless of event masks. CLX reports client messages, mapping notifications, selection clearings, selection notifications, and selection requests to all clients.

Table 12–1: Event Keys

Client Communication Events	Keyboard Events	Window Crossing Events
:CLIENT-MESSAGE	:KEY-PRESS	:ENTER-NOTIFY
:PROPERTY-NOTIFY	:KEY-RELEASE	:LEAVE-NOTIFY
:SELECTION-CLEAR	Keymap State Events	Window State Events
:SELECTION-NOTIFY	:KEYMAP-NOTIFY	:CIRCULATE-NOTIFY
:SELECTION-REQUEST	Pointer Events	:CONFIGURE-NOTIFY
Colormap State Events	:BUTTON-PRESS	:CREATE-NOTIFY
:COLORMAP-NOTIFY	:BUTTON-RELEASE	:DESTROY-NOTIFY
Exposure Events	:MOTION-NOTIFY	:GRAVITY-NOTIFY
:EXPOSURE	Structure Control Events	:MAP-NOTIFY
:GRAPHICS-EXPOSURE	:CIRCULATE-REQUEST	:REARENT-NOTIFY
:NO-EXPOSURE	:CONFIGURE-REQUEST	:UNMAP-NOTIFY
Input Focus Events	:MAP-REQUEST	:VISIBILITY-NOTIFY
:FOCUS-IN	:RESIZE-REQUEST	
:FOCUS-OUT		

See Section 12.5 for details on what information is reported for each event type.

Sections 12.1.1 through 12.1.3 demonstrate how to create, specify, and change an event mask. For more information about specifying the :EXPOSURE component of a GContext, see Chapter 7.

12.1.1 Constructing Event Masks

Event masks are used to specify which event types are reported to a client. Table 12–2 lists the event mask keywords provided by CLX with the corresponding event types. An event mask of 0 specifies that no events are requested.

Table 12–2: Event Masks

Keyword	Events Reported (Event Type)
:BUTTON-MOTION	At least one button on the pointing device is pressed while the pointer moves (:MOTION-NOTIFY).
:BUTTON-1-MOTION	Button 1 of the pointing device is pressed while the pointer moves (:MOTION-NOTIFY).
:BUTTON-2-MOTION	Button 2 of the pointing device is pressed while the pointer moves (:MOTION-NOTIFY).
:BUTTON-3-MOTION	Button 3 of the pointing device is pressed while the pointer moves (:MOTION-NOTIFY).
:BUTTON-4-MOTION	Button 4 of the pointing device is pressed while the pointer moves (:MOTION-NOTIFY).

(continued on next page)

Table 12-2 (Cont.): Event Masks

Keyword	Events Reported (Event Type)
:BUTTON-5-MOTION	Button 5 of the pointing device is pressed while the pointer moves (:MOTION-NOTIFY).
:BUTTON-PRESS	A button on the pointing device is pressed (:BUTTON-PRESS).
:BUTTON-RELEASE	A button on the pointing device is released (:BUTTON-RELEASE).
:COLORMAP-CHANGE	A client installs, changes, or removes a colormap (:COLORMAP-NOTIFY).
:ENTER-WINDOW	The pointer enters a window (:ENTER-NOTIFY).
:EXPOSURE	A window becomes visible, a graphics region cannot be computed, a graphics request exposes a region, or all sources are available and a no expose event is generated (:EXPOSURE, :GRAPHICS-EXPOSURE, :NO-EXPOSURE).
:FOCUS-CHANGE	The keyboard focus changes (:FOCUS-IN, :FOCUS-OUT).
:KEYMAP-STATE	The keymap changes (:KEYMAP-NOTIFY).
:KEY-PRESS	A key on the keyboard is pressed (:KEY-PRESS).
:KEY-RELEASE	A key on the keyboard is released (:KEY-RELEASE).
:LEAVE-WINDOW	The pointer leaves a window (:LEAVE-WINDOW).
:OWNER-GRAB-BUTTON	Automatic grabs should activate with :OWNER-P set to T (no event type).
:POINTER-MOTION	The pointer moves (:MOTION-NOTIFY).
:POINTER-MOTION-HINT	CLX is free to report only one pointer motion event (:MOTION-NOTIFY) until one of the following occurs: <ul style="list-style-type: none">• Either the key or button state changes.• The pointer leaves the window.• The client calls CLX:QUERY-POINTER or CLX:MOTION-EVENTS.
:PROPERTY-CHANGE	A client changes a property (:PROPERTY-NOTIFY).
:RESIZE-REDIRECT	Another client changes the size of a window (:RESIZE-REQUEST).
:STRUCTURE-NOTIFY	One of the following operations occurs on a window: <ul style="list-style-type: none">• Circulate (:CIRCULATE-NOTIFY)• Configure (:CONFIGURE-NOTIFY)• Destroy (:DESTROY-NOTIFY)• Map (:MAP-NOTIFY)• Move (:GRAVITY-NOTIFY)• Reparent (:REPARENT-NOTIFY)• Unmap (:UNMAP-NOTIFY)
:SUBSTRUCTURE-NOTIFY	One of the following operations occurs on a child of a window: <ul style="list-style-type: none">• Circulate (:CIRCULATE-NOTIFY)• Configure (:CONFIGURE-NOTIFY)• Destroy (:DESTROY-NOTIFY)• Map (:MAP-NOTIFY)• Move (:GRAVITY-NOTIFY)• Reparent (:REPARENT-NOTIFY)• Unmap (:UNMAP-NOTIFY)

(continued on next page)

Table 12–2 (Cont.): Event Masks

Keyword	Events Reported (Event Type)
:SUBSTRUCTURE-REDIRECT	Another client performs one of the following operations on a window: <ul style="list-style-type: none">• Circulate (:CIRCULATE-REQUEST)• Configure (:CONFIGURE-REQUEST)• Map (:MAP-REQUEST)
:VISIBILITY-CHANGE	The visibility of a window changes (:VISIBILITY-NOTIFY).

Event masks are encoded as CLX:MASK32 values by the CLX:MAKE-EVENT-MASK function and are deciphered by the CLX:MAKE-EVENT-KEYS function. These functions are defined only for core events (events that are listed in Table 12–1). The format of the CLX:MAKE-EVENT-MASK function is:

CLX:MAKE-EVENT-MASK &REST *keys*

where *keys* is a list of event mask keywords shown in Table 12–2. The function returns a CLX:MASK32 value that may be used to specify the event mask of a window (see Section 12.1.2) or passed to the CLX:GRAB-POINTER, CLX:GRAB-BUTTON, and CLX:SEND-EVENT functions (see Sections 12.3 and 12.4).

The format of the CLX:MAKE-EVENT-KEYS function is:

CLX:MAKE-EVENT-KEYS *event-mask*

where *event-mask* is a CLX:MASK32 value originally returned by the CLX:MAKE-EVENT-MASK function. The function returns a list of the event keys coded in the event mask. For example:

```
Lisp> (clx:make-event-mask :button-press :exposure)
32772
Lisp> (clx:make-event-keys 32772)
(:BUTTON-PRESS :EXPOSURE)
```

12.1.2 Specifying an Event Mask

To specify which event types you are interested in when creating a window, provide a CLX:MASK32 value or a list of event mask keywords as the value of the :EVENT-MASK keyword of the CLX:CREATE-WINDOW function. For example, the following fragment from Example 4–1 specifies that the client program is interested in exposure and button press events:

```
...
(let* (...)
  (window (clx:create-window :parent (clx:screen-root screen)
    ...
    :event-mask '(:exposure :button-press)))
  ...)
```

12.1.3 Changing an Event Mask

To change the event selection of a window, use a SETF form, within a CLX:WITH-STATE macro if you prefer. The example below adds key press events to the event mask of the window created in Section 12.1.1:

```
(setf (clx:window-event-mask window) '(:exposure :button-press :key-press))
```

The next example makes the same change to the event mask but uses the CLX:WITH-STATE macro:

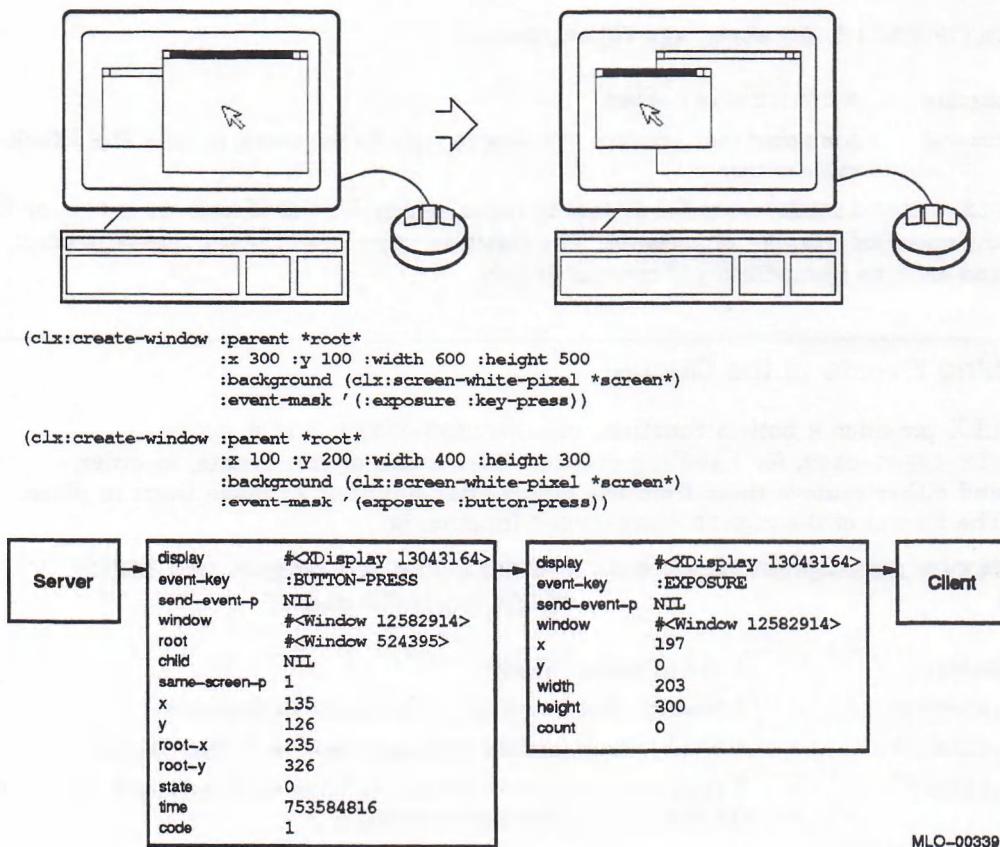
```
(clx:with-state (window)
  (setf (window-event-mask window) '(:exposure :button-press :key-press))
  ...)
```

Using the CLX:WITH-STATE macro improves performance when several window accessors are used.

12.2 Processing Events

Clients have one event queue for each display connection, not one for each window. Events in different windows on the same display are interleaved in the event queue. Figure 12-1 shows two CLX-created windows, a fragment of code, and a representation of an event queue. When a user moves the pointer cursor from the larger window to the smaller one and clicks a button, the events generated include :MOTION-NOTIFY and :LEAVE-NOTIFY in the larger window and :ENTER-NOTIFY, :KEYMAP-NOTIFY, :BUTTON-PRESS, :EXPOSURE, :BUTTON-RELEASE, and :FOCUS-IN in the smaller window. However, the windows' event masks filter out all but the :BUTTON-PRESS and :EXPOSURE events in the smaller window.

Figure 12-1: The Event Queue



The routines in this section allow you to perform the following operations:

- Lock the event queue.
- Find the length of the queue.

- Handle events in the queue.
- Add an event to the queue.
- Remove an event from the queue.

12.2.1 Locking the Event Queue

The CLX:WITH-EVENT-QUEUE macro locks the event queue so that your client has exclusive access. Its format is:

CLX:WITH-EVENT-QUEUE *display* &BODY *forms*

display A CLX:DISPLAY object.

forms One or more LISP forms to be executed while the event queue is locked.

A call to the CLX:WITH-EVENT-QUEUE macro is implicit in these two functions and macro: CLX:EVENT-LISTEN, CLX:PROCESS-EVENT, and CLX:EVENT-CASE.

12.2.2 Finding the Length of the Event Queue

The CLX:EVENT-LISTEN function returns the number of events queued for a display. Its format is:

CLX:EVENT-LISTEN *display* &OPTIONAL *timeout*

display A CLX:DISPLAY object.

timeout A number that specifies how long to wait for the event, or NIL. The default value is zero.

CLX:EVENT-LISTEN waits for events to occur, either forever if *timeout* is NIL, or for the specified number of seconds. The function returns NIL if the queue is empty and returns immediately if *timeout* is zero.

12.2.3 Handling Events in the Queue

CLX provides both a function, CLX:PROCESS-EVENT, and a macro, CLX:EVENT-CASE, for handling events. Clients can obtain events, in order, and either remove them from the queue after handling or leave them in place. The format of the CLX:PROCESS-EVENT function is:

CLX:PROCESS-EVENT *display* &KEY :HANDLER :TIMEOUT :PEEK-P :DISCARD-P
:FORCE-OUTPUT-P :EVENT

display A CLX:DISPLAY object.

:HANDLER A function that is applied to the events in the queue.

:TIMEOUT A number that specifies how many seconds to wait for an event.

:PEEK-P Has a CLX:BOOLEAN value that specifies whether events are left on the queue after they are processed.

:DISCARD-P Has a CLX:BOOLEAN value that specifies whether events for which the :HANDLER function returns NIL are removed from the queue.

:FORCE-OUTPUT-P Has a CLX:BOOLEAN value that specifies whether the CLX:DISPLAY-FORCE-OUTPUT function is called before this function begins execution. The default value is T.

:EVENT A DWT:EVENT value, or NIL. If a value is provided, the CLX:PROCESS-EVENT function handles that event rather than querying the event queue.

If :FORCE-OUTPUT-P is true, CLX:DISPLAY-FORCE-OUTPUT is invoked before any events are processed. The :HANDLER function is invoked on each event in the queue until it returns non-NIL. The object that the handler returns is then returned by the CLX:PROCESS-EVENT function. The values of :PEEK-P and :DISCARD-P determine what happens to events: If :PEEK-P is true, the event that returned non-NIL remains on the queue; if :DISCARD-P is true, all events for which the handler returns NIL are removed from the queue. Otherwise, they are left in place. CLX:PROCESS-EVENT hangs until a non-NIL value is returned for an event or until the number of seconds specified by :TIMEOUT have elapsed. The function returns NIL on timeout.

The arguments to the :HANDLER function are described further in Section 12.5. Example 12-1 shows how to use the CLX:PROCESS-EVENT function to print the contents of an event queue.

Example 12-1: Using the CLX:PROCESS-EVENT Function

```
(defun print-queue (display)
  (clx:process-event display :handler #'print-event
                      :discard-p nil
                      :timeout 0))

(defun print-event (&key event-key)
  (print event-key)
  nil)
```

The :HANDLER function, PRINT-EVENT, never returns true, so :TIMEOUT must be set to zero. CLX:PROCESS-EVENT is guaranteed to look through the entire queue once.

If CLX:PROCESS-EVENT is invoked recursively, the nested invocation begins with the event after the one currently being processed.

The format of the CLX:EVENT-CASE macro is:

CLX:EVENT-CASE *display*
 &KEY :TIMEOUT :PEEK-P :DISCARD-P :FORCE-OUTPUT-P
 &BODY *clauses*

<i>display</i>	A CLX:DISPLAY object.
:TIMEOUT	A number that specifies how many seconds to wait until CLX:EVENT-CASE returns; or NIL, meaning until one of the <i>clauses</i> returns non-NIL.
:PEEK-P	Has a CLX:BOOLEAN value that specifies whether events are left on the queue.
:DISCARD-P	Has a CLX:BOOLEAN value that specifies whether events for which the handler returns NIL are removed from the queue.
:FORCE-OUTPUT-P	Has a CLX:BOOLEAN value that specifies whether the CLX:DISPLAY-FORCE-OUTPUT function is called before this function starts reading events from the queue. The default value is true.
<i>clauses</i>	Specify the event keys and the LISP forms to be executed when the events occur.

The *clauses* have the following syntax:

(*event-key* | {*event-key*}*) (*{arg}** | {*{arg var}*}*) *forms*)

event-key An event key shown in Table 12-1. The names may be typed as variables or as keywords.

arg An argument appropriate to the event key, as listed in Section 12.5.

var A variable you want to be bound to the event key argument.

forms One or more LISP forms to be executed for the event.

The following fragment illustrates a call to CLX:EVENT-CASE with a single *clause*:

```
(clx:event-case (*my-display*)
  ;; | event-key | ({arg}*) | forms |
  (:button-press (window x y) (if (member window *my-window-list*)
    (progn (move-cursor x y) t) nil)))
```

This example uses three components of a :BUTTON-PRESS event: the window and the X and Y coordinates of the pointer cursor. If the window is a member of the set named by *MY-WINDOW-LIST*, the user-defined function MOVE-CURSOR is passed the X and Y coordinates. The clause returns NIL so that event processing continues.

The last *clause* in a CLX:EVENT-CASE may contain T or OTHERWISE in place of an event key. If you do not provide such a clause, the implicit clause (otherwise nil) is used.

Example 12-2 is a portion of the file BLACKBOARD.LSP, which is in the LISP\$EXAMPLES directory.

Example 12-2: Using the CLX:EVENT-CASE Macro

```
;Now we go into the event-case, which sits at the event queue
;;and processes button events. When a keyboard event occurs, it
;;exits the loop.
(clx:event-case
  (*blackboard-display* :discard-p t)
  (:exposure (window)
    ; This is a simple way to handle the exposure -- just
    ; redraw our entire line-list.
    (when (eq window *blackboard-window*)
      (clx:draw-segments *blackboard-window*
        *blackboard-gcontext*
        *blackboard-line-list*)))
  (:button-press (x y code)
    (case code ;this tells which button was pushed
      (1 (clx:draw-point *blackboard-window*
        *blackboard-gcontext*
        (setq old-x last-x last-x x)
        (setq old-y last-y last-y y))
        (setq last-button 1))
      (2 (draw-and-record-line last-x last-y
        (setq old-x last-x last-x x)
        (setq old-y last-y last-y y))
        (setq last-button 2))
      (3 (if (= last-button 3)
        (clx:clear-area *blackboard-window*)
        (progn
          (undraw-and-unrecord-line
            old-x old-y last-x last-y)
          (setq last-x old-x last-y old-y))))
```

(continued on next page)

Example 12-2 (Cont.): Using the CLX:EVENT-CASE Macro

```
(setq last-button 3))  
      nil);this tells event-case not to exit  
(:key-press () t);pressing a key returns t, thus exiting event-case  
))
```

12.2.4 Adding an Event to the Queue

The format of the CLX:QUEUE-EVENT function is:

CLX:QUEUE-EVENT *display event-key*
&REST *args*
&KEY :APPEND-P
&ALLOW-OTHER-KEYS

display A CLX:DISPLAY object.
event-key A CLX:EVENT-KEY value.
args Keyword-value pairs that specify information appropriate to the *event-key* (see Section 12.5).
:APPEND-P Has a CLX:BOOLEAN value that determines whether the event is put at the tail of the queue (if true) or head.

Additional keyword arguments depend on *event-key* and are as specified in Section 12.5.

12.2.5 Removing an Event from the Queue

The CLX:DISCARD-CURRENT-EVENT function deletes the event at the head of the queue. Its format is:

CLX:DISCARD-CURRENT-EVENT *display*

where *display* is a CLX:DISPLAY object. The CLX:DISCARD-CURRENT-EVENT function returns T if it succeeded; NIL if the queue was empty. For example:

```
Lisp> (clx:event-listen *display*)  
27  
Lisp> (clx:discard-current-event *display*)  
T  
Lisp> (clx:event-listen *display*)  
26
```

12.3 Controlling Events

A client may “grab” exclusive use of the keyboard, keyboard keys, pointer, pointer buttons, or server. The keyboard or pointer can be explicitly grabbed by calling the CLX:GRAB-KEYBOARD or CLX:GRAB-POINTER function: this is called an active grab. A particular key or button can be passively grabbed in a particular window: the grab activates when the key or button is actually pressed. When a device is actively grabbed, keyboard or pointer events are reported to the client that established the grab rather than the client that owns the event window. If the grab places the keyboard or pointer in asynchronous mode, further keyboard or pointer events continue to be processed. If the keyboard or pointer is in synchronous mode, no further events are processed until the grabbing client

allows them (see Section 12.3.4). The keyboard or pointer is “frozen” during a synchronous grab. Actual keyboard or pointer changes are not lost while the device is frozen: they are simply queued for later processing. The event that triggered the grab can also be replayed.

When the server is grabbed, no requests from other clients are processed. Grabbing the server is highly discouraged.

Several of the functions in this section take a pointer event, state, or modifier mask argument. Tables 12–3 through 12–5 list the valid keywords for these arguments.

Table 12–3: Pointer Event Masks

:BUTTON-MOTION	:BUTTON-5-MOTION	:KEYMAP-STATE
:BUTTON-1-MOTION	:BUTTON-PRESS	:LEAVE-WINDOW
:BUTTON-2-MOTION	:BUTTON-RELEASE	:POINTER-MOTION
:BUTTON-3-MOTION	:ENTER-WINDOW	:POINTER-MOTION-HINT
:BUTTON-4-MOTION		

Table 12–4: State Masks

:SHIFT	:MOD-1	:BUTTON-1
:LOCK	:MOD-2	:BUTTON-2
:CONTROL	:MOD-3	:BUTTON-3
	:MOD-4	:BUTTON-4
	:MOD-5	:BUTTON-5

Table 12–5: Modifier Masks

:SHIFT	:CONTROL	:MOD-2	:MOD-4
:LOCK	:MOD-1	:MOD-3	:MOD-5

Like event masks, state masks are encoded as CLX:MASK16 values with the CLX:MAKE-STATE-MASK function and are deciphered with the CLX:MAKE-STATE-KEYS function. The format of the CLX:MAKE-STATE-MASK function is:

CLX:MAKE-STATE-MASK &REST keys

where *keys* are CLX:STATE-MASK-KEYS values. The function returns a CLX:MASK16 value that may be passed to the CLX:MAKE-STATE-KEYS and CLX:GRAB-BUTTON (:MODIFIER argument) functions.

The format of the CLX:MAKE-STATE-KEYS function is:

CLX:MAKE-STATE-KEYS state-mask

where *state-mask* is a CLX:MASK16 value that encodes a state mask. The function returns a list of CLX:STATE-MASK-KEY values. For example:

```
Lisp> (clx:make-event-mask :button-1 :shift :lock)
259
Lisp> (clx:make-event-keys 259)
(:SHIFT :LOCK :BUTTON-1)
```

12.3.1 Grabbing the Pointer

The functions in this section allow you to control input from the pointer, which is usually a mouse. The pointer may be actively grabbed by calling the CLX:GRAB-POINTER function or passively grabbed by calling the CLX:GRAB-BUTTON function. Passive grabs are convenient for implementing reliable pop-up menus. For example, you can guarantee that the pop-up is mapped before the pointer button is released by grabbing the button and requesting synchronous behavior. The button press will trigger the grab and freeze further event reporting until you can map the pop-up window. When you then allow further event processing, the button release will be correctly processed relative to the pop-up window.

The format of the CLX:GRAB-POINTER function is:

```
CLX:GRAB-POINTER window event-mask
                  &KEY :OWNER-P :SYNC-POINTER-P :SYNC-KEYBOARD-P
                  :CONFINE-TO :CURSOR :TIME
```

<i>window</i>	The CLX:WINDOW within which the client grabs the pointer.
<i>event-mask</i>	The CLX:POINTER-EVENT-MASK value that specifies for which events the client is notified.
:OWNER-P	Has a CLX:BOOLEAN value that specifies whether all pointer events are reported to the client. When false, pointer events are reported only when they occur in <i>window</i> and are selected by its event mask.
:SYNC-POINTER-P	Has a CLX:BOOLEAN value that specifies whether pointer events are processed synchronously.
:SYNC-KEYBOARD-P	Has a CLX:BOOLEAN value that specifies whether keyboard events are processed synchronously.
:CONFINE-TO	Either has a CLX:WINDOW value that specifies the window to which the pointer cursor is confined, or is null, meaning that the cursor can move to any window.
:CURSOR	Either has a CLX:CURSOR value that specifies the cursor's shape during the grab, or is null, meaning that the cursor does not change during the grab.
:TIME	Has a CLX:TIMESTAMP value that specifies the time at which events are to be released.

The function returns a keyword that indicates the grab status of the pointer cursor. The possible values are :GRAB-SUCCESS, :ALREADY-GRABBED, :GRAB-FROZEN, :GRAB-INVALID-TIME, and :GRAB-NOT-VIEWABLE.

The format of the CLX:GRAB-BUTTON function is:

```
CLX:GRAB-BUTTON window button event-mask
                  &KEY :MODIFIERS :OWNER-P :SYNC-POINTER-P
                  :SYNC-KEYBOARD-P :CONFINE-TO :CURSOR
```

<i>window</i>	The CLX:WINDOW object within which the client grabs the pointer button.
<i>button</i>	An integer that specifies which button is grabbed, or the keyword :ANY.
<i>event-mask</i>	The CLX:POINTER-EVENT-MASK value that specifies for which events the client is notified.
:MODIFIERS	Has a CLX:MODIFIER-MASK value that specifies which keyboard modifiers are grabbed. The default value is zero.

:OWNER-P	Has a CLX:BOOLEAN value that specifies whether all pointer events are reported to the client. When false, pointer events are reported only when they occur in <i>window</i> and are selected by its event mask.
:SYNC-POINTER-P	Has a CLX:BOOLEAN value that specifies whether pointer events are processed synchronously.
:SYNC-KEYBOARD-P	Has a CLX:BOOLEAN value that specifies whether keyboard events are processed synchronously.
:CONFINE-TO	Either has a CLX:WINDOW value that specifies the window to which the pointer cursor is confined, or is null, meaning that the cursor can move to any window.
:CURSOR	Either has a CLX:CURSOR value that specifies the cursor that is displayed during the grab, or is null, meaning that the shape does not change.

The CLX:UNGRAB-POINTER and CLX:UNGRAB-BUTTON functions release pointer and button grabs, respectively. The format of the CLX:UNGRAB-POINTER function is:

CLX:UNGRAB-POINTER *display &KEY :TIME*

- display* The CLX:DISPLAY whose pointer cursor is to be released.
:TIME Has a CLX:TIMESTAMP value that specifies when pointer events are released.

The format of the CLX:UNGRAB-BUTTON function is:

CLX:UNGRAB-BUTTON *window button &KEY :MODIFIERS*

- window* The CLX:WINDOW object within which the client releases the pointer button.
button An integer that specifies which button is released, or the keyword :ANY.
:MODIFIERS Has a CLX:MODIFIER-MASK value that specifies which keyboard modifiers are released. The default value is zero.

Use the CLX:CHANGE-ACTIVE-POINTER-GRAB function to change the event mask or cursor of an ongoing pointer grab. Its format is:

CLX:CHANGE-ACTIVE-POINTER-GRAB *display event-mask
&OPTIONAL cursor time*

- display* The CLX:DISPLAY whose active pointer grab is changed.
event-mask The CLX:POINTER-EVENT-MASK that specifies for which events the client is notified.
cursor NIL or the CLX:CURSOR that specifies the shape of the cursor that is displayed. If no value is provided, the cursor shape does not change.
time A CLX:TIMESTAMP that specifies when the grab is changed.

The specified *time* must be no earlier than the time at which the grab was established, and no later than the current server time.

12.3.2 Grabbing the Keyboard

The keyboard may be actively grabbed by calling the CLX:GRAB-KEYBOARD function; individual keys or combinations of keys may be passively grabbed by calling the CLX:GRAB-KEY function. The format of the CLX:GRAB-KEYBOARD function is:

CLX:GRAB-KEYBOARD *window
&KEY :OWNER-P :SYNC-POINTER-P :SYNC-KEYBOARD-P :TIME*

<i>window</i>	The CLX:WINDOW within which the keyboard is grabbed.
:OWNER-P	Has a CLX:BOOLEAN value that specifies whether all keyboard events are reported to the client. When false, keyboard events are reported only when they occur in <i>window</i> and are selected by its event mask.
:SYNC-POINTER-P	Has a CLX:BOOLEAN value that specifies whether pointer events are processed synchronously.
:SYNC-KEYBOARD-P	Has a CLX:BOOLEAN value that specifies whether keyboard events are processed synchronously.
:TIME	Has a CLX:TIMESTAMP that specifies when the grab is established.

The function returns a keyword that indicates the status of the grab:

:GRAB-SUCCESS, :ALREADY-GRABBED, :GRAB-FROZEN, :GRAB-INVALID-TIME, or
:GRAB-NOT-VIEWABLE.

The format of the CLX:GRAB-KEY function is:

CLX:GRAB-KEY *window key*
&KEY :MODIFIERS :OWNER-P :SYNC-POINTER-P :SYNC-KEYBOARD-P

<i>window</i>	The CLX:WINDOW object in which the key is grabbed.
<i>key</i>	An INTEGER or :ANY and specifies the key that is grabbed.
:MODIFIERS	Has a CLX:MODIFIER-MASK value that specifies which keyboard modifiers are grabbed. The default value is zero.
:OWNER-P	Has a CLX:BOOLEAN value that specifies whether all keyboard events are reported to the client. When false, keyboard events are reported only when they occur in <i>window</i> and only if they are selected by <i>event-mask</i> .
:SYNC-POINTER-P	Has a CLX:BOOLEAN value that specifies whether pointer events are processed synchronously.
:SYNC-KEYBOARD-P	Has a CLX:BOOLEAN value that specifies whether keyboard events are processed synchronously.

The CLX:UNGRAB-KEYBOARD and CLX:UNGRAB-KEY functions release active and passive keyboard grabs, respectively. The format of the CLX:UNGRAB-KEYBOARD function is:

CLX:UNGRAB-KEYBOARD *display &KEY :TIME*

<i>display</i>	The CLX:DISPLAY whose keyboard is released.
:TIME	Has a CLX:TIMESTAMP that specifies when keyboard events are released.

The format of the CLX:UNGRAB-KEY function is:

CLX:UNGRAB-KEY *window key &KEY :MODIFIERS*

<i>window</i>	The CLX:WINDOW in which the keyboard is released.
<i>key</i>	An INTEGER or :ANY and specifies the key that is grabbed.
:MODIFIERS	Has a CLX:MODIFIER-MASK value that specifies which keyboard modifiers are released. The default value is zero.

12.3.3 Grabbing the Server

The CLX:GRAB-SERVER function allows a client program to take exclusive possession of the server for a specified display. No requests are processed, including requests to close connections, while the server is grabbed. A client automatically ungrabs the server when it closes its connection to that server. Clients should not grab the server any more than is absolutely necessary.

The format of the CLX:GRAB-SERVER function is:

CLX:GRAB-SERVER *display*

where *display* is a CLX:DISPLAY object.

The format of the CLX:WITH-SERVER-GRABBED macro is:

CLX:WITH-SERVER-GRABBED *display* &BODY *body*

display A CLX:DISPLAY object.

body LISP forms to be executed while the server is grabbed.

The body is not surrounded by a CLX:WITH-DISPLAY macro.

The CLX:UNGRAB-SERVER function releases an active server grab. Its format is:

CLX:UNGRAB-SERVER *display*

where *display* is a CLX:DISPLAY object.

12.3.4 Allowing Events

The CLX:ALLOW-EVENTS function releases events that were queued because a device was grabbed.

The format of the CLX:ALLOW-EVENTS function is:

CLX:ALLOW-EVENTS *display* *mode* &OPTIONAL *time*

display The CLX:DISPLAY object whose events are released.

mode A keyword that specifies which events are released.

time The CLX:TIMESTAMP value that specifies when events are released.

The possible values for the *mode* argument are explained below:

:ASYNC-POINTER	Allows pointer event processing to continue normally after a pointer has been stopped.
:SYNC-POINTER	Allows pointer event processing to continue normally after a pointer has been frozen or grabbed until the next BUTTON-PRESS or :BUTTON-RELEASE event is reported to the client.
:REPLAY-POINTER	If the pointer is actively grabbed by the client, and is frozen, the pointer grab is released and the event is completely reprocessed.
:ASYNC-KEYBOARD	Allows keyboard event processing to continue normally after a keyboard has been frozen.
:SYNC-KEYBOARD	If the keyboard is frozen and actively grabbed by the client, keyboard event processing continues as usual until the next KEY-PRESS or :KEY-RELEASE event is reported to the client.

:REPLAY-KEYBOARD	If the keyboard is actively grabbed by the client, and is frozen as the result of an event having been sent to the client either from the activation of CLX:GRAB-KEY or from a previous CLX:ALLOW-EVENTS with mode :SYNC-KEYBOARD (but not from a CLX:GRAB-KEYBOARD), the keyboard grab is released and the event is completely reprocessed.
:ASYNC-BOTH	If the pointer and the keyboard are frozen by the client, event processing (for both devices) continues normally.
:SYNC-BOTH	If the pointer and keyboard are frozen by the client, event processing (for both devices) continues normally until the next BUTTON-PRESS, BUTTON-RELEASE, :KEY-PRESS, or KEY-RELEASE event is reported to the client for a grabbed device, at which time the devices again appear to freeze.

12.4 Sending Events

Use the CLX:SEND-EVENT function to send an event to a specified window. The function identifies the destination window, determines which client should receive the event, and ignores any active grabs. The contents of the event are specified by additional keyword arguments, and are not checked or altered by the X server except that the :SEND-EVENT component is set to T.

The format of the CLX:SEND-EVENT function is:

```
CLX:SEND-EVENT window event-key event-mask
  &REST args
  &KEY :PROPAGATE-P :DISPLAY
  &ALLOW-OTHER-KEYS
```

<i>window</i>	A CLX:WINDOW object or a keyword specifying a special window. The possible keywords are :POINTER-WINDOW and INPUT-FOCUS.
<i>event-key</i>	A CLX:EVENT-KEY or NIL; specifies what kinds of events to generate.
<i>event-mask</i>	A CLX:EVENT-MASK; if nonzero, sends the event to the client that created the window.
<i>args</i>	Keyword-value pairs that contain information about the event being sent.
:PROPAGATE-P	Has a CLX:BOOLEAN value that specifies whether to propagate the event to other clients.
:DISPLAY	A CLX:DISPLAY object or NIL.

The function returns T if the X server successfully generates the event, NIL if the conversion to the wire protocol failed.

The valid &REST arguments vary according to the *event-key* argument. The tables in Section 12.5 list the keywords. If an event component has synonyms, supply a value for only one of them. The fragment below illustrates the syntax of the additional keyword arguments:

```
(send-event *application-window* :button-press 0
  :state 0
  :code 1
  :child *application-popup*
  :propagate-p t)
```

This example generates an MB1 press with no modifiers in the *APPLICATION-POPUP* window. For a complete list of components of a :BUTTON-PRESS event, see Table 12-6.

The `:DISPLAY` argument is required only if the *window* is `:POINTER-WINDOW` or `:INPUT-FOCUS`.

12.5 Event Keys

The tables show the components of each core event reported by CLX. These components determine the arguments that may be passed to event-handling routines such as `CLX:PROCESS-EVENT` and `CLX:EVENT-CASE`.

Table 12-6: :BUTTON-PRESS Event-Key

Argument	Type	Meaning
<i>display</i>	<code>CLX:DISPLAY</code>	The display on which the event occurred. This argument is available only to <code>CLX:PROCESS-EVENT</code> handler functions.
<i>event-key</i>	<code>CLX:EVENT-KEY</code>	The current event.
<i>send-event-p</i>	<code>CLX:BOOLEAN</code>	True if the event came from a <code>CLX:SEND-EVENT</code> request.
<i>[window event-window]</i>	<code>CLX:WINDOW</code>	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>root</i>	<code>CLX:WINDOW</code>	The root window in which the event occurred.
<i>child</i>	(OR NULL <code>CLX:WINDOW</code>)	The source window in which the event occurred.
<i>same-screen-p</i>	<code>CLX:BOOLEAN</code>	Indicates whether the event window is on the same screen as the root window.
<i>x</i>	INTEGER	The X coordinate of the pointer in the event window.
<i>y</i>	INTEGER	The Y coordinate of the pointer in the event window.
<i>root-x</i>	INTEGER	The X coordinate of the pointer relative to the root window.
<i>root-y</i>	INTEGER	The Y coordinate of the pointer relative to the root window.
<i>state</i>	<code>CLX:MASK16</code>	The state of buttons and modifier keys just prior to the event.
<i>time</i>	<code>CLX:TIMESTAMP</code>	The time in milliseconds at which the event occurred.
<i>code</i>	INTEGER	The number of the button that was pressed.

Table 12-7: :BUTTON-RELEASE Event-Key

Argument	Type	Meaning
<i>display</i>	<code>CLX:DISPLAY</code>	The display on which the event occurred. This argument is available only to <code>CLX:PROCESS-EVENT</code> handler functions.
<i>event-key</i>	<code>CLX:EVENT-KEY</code>	The current event.
<i>send-event-p</i>	<code>CLX:BOOLEAN</code>	True if the event came from a <code>CLX:SEND-EVENT</code> request.
<i>[window event-window]</i>	<code>CLX:WINDOW</code>	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>root</i>	<code>CLX:WINDOW</code>	The root window in which the event occurred.
<i>child</i>	(OR NULL <code>CLX:WINDOW</code>)	The source window in which the event occurred.
<i>same-screen-p</i>	<code>CLX:BOOLEAN</code>	Indicates whether the event window is on the same screen as the root window.
<i>x</i>	INTEGER	The X coordinate of the pointer in the event window.

(continued on next page)

Table 12-7 (Cont.): :BUTTON-RELEASE Event-Key

Argument	Type	Meaning
<i>y</i>	INTEGER	The Y coordinate of the pointer in the event window.
<i>root-x</i>	INTEGER	The X coordinate of the pointer relative to the root window.
<i>root-y</i>	INTEGER	The Y coordinate of the pointer relative to the root window.
<i>state</i>	CLX:MASK16	The state of buttons and modifier keys just prior to the event.
<i>time</i>	CLX:TIMESTAMP	The time in milliseconds at which the event occurred.
<i>code</i>	INTEGER	The number of the button that was released.

Table 12-8: :CIRCULATE-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
<i>event-window</i>	CLX:WINDOW	The event window.
<i>window</i>	CLX:WINDOW	The window that has been circulated.
<i>place</i>	(MEMBER :TOP :BOTTOM)	The position of the window on the stack after it has been circulated.

Table 12-9: :CIRCULATE-REQUEST Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>parent</i> <i>event-window</i>]	CLX:WINDOW	The parent of the window to be circulated (<i>parent</i> and <i>event-window</i> are synonyms).
<i>window</i>	CLX:WINDOW	The window that is to be circulated.
<i>place</i>	(MEMBER :TOP :BOTTOM)	The requested position of the window on the stack.

Table 12-10: :CLIENT-MESSAGE Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.

(continued on next page)

Table 12–10 (Cont.): :CLIENT-MESSAGE Event-Key

Argument	Type	Meaning
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The window to which the message is sent (<i>window</i> and <i>event-window</i> are synonyms).
<i>format</i>	(MEMBER 8 16 32)	Indicates whether the data is in units of 8, 16, or 32 bits.
<i>data</i>	(SEQUENCE INTEGER)	The message data.

Table 12–11: :COLORMAP-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The window whose colormap has changed (<i>window</i> and <i>event-window</i> are synonyms).
<i>colormap</i>	(OR NIL CLX:COLORMAP)	NIL if the colormap was changed as a result of a call to CLX:FREE-COLORMAP; otherwise, the new colormap.
<i>new-p</i>	CLX:BOOLEAN	True if the colormap has changed; false if the colormap has been installed or removed.
<i>installed-p</i>	CLX:BOOLEAN	True if the colormap is installed, otherwise false.

Table 12–12: :CONFIGURE-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
<i>event-window</i>	CLX:WINDOW	The event window.
<i>window</i>	CLX:WINDOW	The window that has been configured.
<i>x</i>	INTEGER	The X coordinate of the top-left corner of the region that is exposed (relative to the origin of the drawable).
<i>y</i>	INTEGER	The Y coordinate of the top-left corner of the region that is exposed (relative to the origin of the drawable).
<i>width</i>	INTEGER	The width of the exposed region.
<i>height</i>	INTEGER	The height of the exposed region.
<i>border-width</i>	INTEGER	The width of the border, in pixels.
<i>above-sibling</i>	(OR NIL CLX:WINDOW)	The sibling window above which <i>window</i> is stacked. If this is NIL, CLX places <i>window</i> on the bottom of the stack.
<i>override-redirect-p</i>	CLX:BOOLEAN	Specifies whether the server ignores requests to reconfigure <i>window</i> .

Table 12–13: :CONFIGURE-REQUEST Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>parent</i> <i>event-window</i>]	CLX:WINDOW	The parent of the window to be reconfigured (<i>parent</i> and <i>event-window</i> are synonyms).
<i>window</i>	CLX:WINDOW	The window that is to be reconfigured.
<i>x</i>	INTEGER	The requested X coordinate of the top-left corner of the region that is exposed (relative to the origin of the drawable).
<i>y</i>	INTEGER	The requested Y coordinate of the top-left corner of the region that is exposed (relative to the origin of the drawable).
<i>width</i>	INTEGER	The requested width of the exposed region.
<i>height</i>	INTEGER	The requested height of the exposed region.
<i>border-width</i>	INTEGER	The requested width of the border, in pixels.
<i>stack-mode</i>	(MEMBER :ABOVE :BELOW :TOP-IF :BOTTOM-IF :OPPOSITE)	The requested position of the window in the stack. See the description of the CLX:WINDOW-PRIORITY function in Part IV for details.
<i>above-sibling</i>	(OR NIL CLX:WINDOW)	The sibling window above which <i>window</i> is stacked. If NIL, <i>window</i> is placed at the bottom of the stack.
<i>value-mask</i>	CLX:MASK16	A bit mask that describes which of the arguments were specified in the ConfigureWindow request.

Table 12–14: :CREATE-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
<i>window</i>	CLX:WINDOW	The window just created.
[<i>parent</i> <i>event-window</i>]	CLX:WINDOW	The parent of the created window (<i>parent</i> and <i>event-window</i> are synonyms).
<i>x</i>	INTEGER	The X coordinate of the origin of <i>window</i> .
<i>y</i>	INTEGER	The Y coordinate of the origin of <i>window</i> .
<i>width</i>	INTEGER	The width of <i>window</i> .
<i>height</i>	INTEGER	The height of <i>window</i> .
<i>border-width</i>	INTEGER	The width of the border, in pixels.
<i>override-redirect-p</i>	CLX:BOOLEAN	Specifies whether the server ignores requests to create <i>window</i> .

Table 12-15: :DESTROY-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
<i>event-window</i>	CLX:WINDOW	The event window.
<i>window</i>	CLX:WINDOW	The window that has been destroyed.

Table 12-16: :ENTER-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>root</i>	CLX:WINDOW	The root of the source window.
<i>child</i>	(OR NULL CLX:WINDOW)	The source window.
<i>same-screen-p</i>	CLX:BOOLEAN	Specifies whether the event window and the root window are on the same screen.
<i>x</i>	INTEGER	The X coordinate of the pointer in the event window.
<i>y</i>	INTEGER	The Y coordinate of the pointer in the event window.
<i>root-x</i>	INTEGER	The X coordinate of the pointer relative to the root window.
<i>root-y</i>	INTEGER	The Y coordinate of the pointer relative to the root window.
<i>state</i>	CLX:MASK16	The state of buttons and keys just prior to the event.
<i>time</i>	CLX:TIMESTAMP	The time in milliseconds at which the event occurred.
<i>mode</i>	(MEMBER :NORMAL :GRAB :UNGRAB)	Indicates whether the event is normal or pseudomotion.
<i>kind</i>	(MEMBER :ANCESTOR :VIRTUAL :INFERIOR :NON-LINEAR :NONLINEAR-VIRTUAL)	Indicates which windows CLX notifies of the window entry event.
<i>focus-p</i>	CLX:BOOLEAN	If true, specifies that the event window is the focus window. If false, one of the children of the event window is the focus window.

Table 12-17: :EXPOSURE Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>x</i>	INTEGER	The X coordinate of the top-left corner of the region that is exposed (relative to the origin of the drawable).
<i>y</i>	INTEGER	The Y coordinate of the top-left corner of the region that is exposed (relative to the origin of the drawable).
<i>width</i>	INTEGER	The width of the exposed region.
<i>height</i>	INTEGER	The height of the exposed region.
<i>count</i>	INTEGER	The number of exposure events that are to follow.

Table 12-18: :FOCUS-IN Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>mode</i>	(MEMBER :NORMAL :WHILE-GRABBED :GRAB :UNGRAB)	Specifies whether the event is the result of normal keyboard input; keyboard input after a client has grabbed the keyboard; keyboard input at the time the client activates a keyboard grab; or keyboard input at the time the client deactivates a keyboard grab.
<i>kind</i>	(MEMBER :ANCESTOR :VIRTUAL :INFERIOR :NONLINEAR :NONLINEAR-VIRTUAL : POINTER : POINTER-ROOT :NONE)	Indicates which windows and pointers CLX notifies of the input focus change.

Table 12-19: :FOCUS-OUT Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.

(continued on next page)

Table 12–19 (Cont.): :FOCUS-OUT Event-Key

Argument	Type	Meaning
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>mode</i>	(MEMBER :NORMAL : WHILE-GRABBED : GRAB :UNGRAB)	Specifies whether the event is the result of normal keyboard input; keyboard input after a client has grabbed the keyboard; keyboard input at the time the client activates a keyboard grab; or keyboard input at the time the client deactivates a keyboard grab.
<i>kind</i>	(MEMBER :ANCESTOR :VIRTUAL : INFERIOR : NONLINEAR : NONLINEAR-VIRTUAL : POINTER : POINTER-ROOT :NONE)	Indicates which windows and pointers CLX notifies of the input focus change.

Table 12–20: :GRAPHICS-EXPOSURE Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>drawable</i> <i>event-window</i>]	CLX:DRAWABLE	The drawable reporting the event (<i>drawable</i> and <i>event-window</i> are synonyms).
<i>x</i>	INTEGER	The X coordinate of the top-left corner of the exposed region (relative to the origin of the drawable).
<i>y</i>	INTEGER	The Y coordinate of the top-left corner of the exposed region (relative to the origin of the drawable).
<i>width</i>	INTEGER	The width of the exposed region.
<i>height</i>	INTEGER	The height of the exposed region.
<i>count</i>	INTEGER	The number of exposure events that are to follow.
<i>major</i>	INTEGER	Indicates whether the graphics request was a CopyArea (62) or a CopyPlane (63).
<i>minor</i>	INTEGER	The value zero. Reserved for use by extensions.

Table 12–21: :GRAVITY-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.

(continued on next page)

Table 12–21 (Cont.): :GRAVITY-NOTIFY Event-Key

Argument	Type	Meaning
<i>event-window</i>	CLX:WINDOW	The event window.
<i>window</i>	CLX:WINDOW	The window that has been moved.
<i>x</i>	INTEGER	The X coordinate of the new origin, relative to the parent window.
<i>y</i>	INTEGER	The Y coordinate of the new origin, relative to the parent window.

Table 12–22: :KEYMAP-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>keymap</i>	(BIT-VECTOR 256)	The bit vector of the keyboard. Each 1 bit indicates that the corresponding key is currently pressed.

Table 12–23: :KEY-PRESS Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>root</i>	CLX:WINDOW	The root window of the source window.
<i>child</i>	(OR NULL CLX:WINDOW)	The source window.
<i>same-screen-p</i>	CLX:BOOLEAN	Indicates whether the event window is on the same screen as the root window.
<i>x</i>	INTEGER	The X coordinate of the pointer in the event window.
<i>y</i>	INTEGER	The Y coordinate of the pointer in the event window.
<i>root-x</i>	INTEGER	The X coordinate of the pointer relative to the root window.
<i>root-y</i>	INTEGER	The Y coordinate of the pointer relative to the root window.
<i>state</i>	CLX:MASK16	The state of buttons and modifier keys just prior to the event.
<i>time</i>	CLX:TIMESTAMP	The time in milliseconds at which the event occurred.
<i>code</i>	INTEGER	The keycode of the key that was pressed.

Table 12-24: :KEY-RELEASE Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> [<i>event-window</i>]]	CLX:WINDOW	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>root</i>	CLX:WINDOW	The root window of the window in which the event occurred.
<i>child</i>	(OR NULL CLX:WINDOW)	The source window.
<i>same-screen-p</i>	CLX:BOOLEAN	Indicates whether the event window is on the same screen as the root window.
<i>x</i>	INTEGER	The X coordinate of the pointer in the event window.
<i>y</i>	INTEGER	The Y coordinate of the pointer in the event window.
<i>root-x</i>	INTEGER	The X coordinate of the pointer relative to the root window.
<i>root-y</i>	INTEGER	The Y coordinate of the pointer relative to the root window.
<i>state</i>	CLX:MASK16	The state of buttons and modifier keys just prior to the event.
<i>time</i>	CLX:TIMESTAMP	The time in milliseconds at which the event occurred.
<i>code</i>	INTEGER	The keycode of the key that was released.

Table 12-25: :LEAVE-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> [<i>event-window</i>]]	CLX:WINDOW	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>root</i>	CLX:WINDOW	The root of the source window.
<i>child</i>	(OR NULL CLX:WINDOW)	The source window.
<i>same-screen-p</i>	CLX:BOOLEAN	Specifies whether the event window and the root window are on the same screen.
<i>x</i>	INTEGER	The X coordinate of the pointer in the event window.
<i>y</i>	INTEGER	The Y coordinate of the pointer in the event window.
<i>root-x</i>	INTEGER	The X coordinate of the pointer relative to the root window.
<i>root-y</i>	INTEGER	The Y coordinate of the pointer relative to the root window.

(continued on next page)

Table 12-25 (Cont.): :LEAVE-NOTIFY Event-Key

Argument	Type	Meaning
<i>state</i>	INTEGER	The state of buttons and keys just prior to the event.
<i>time</i>	INTEGER	The time in milliseconds at which the event occurred.
<i>mode</i>	(MEMBER :NORMAL :GRAB :UNGRAB)	Indicates whether the event is normal or pseudomotion.
<i>kind</i>	(MEMBER :ANCESTOR :VIRTUAL :INFERIOR :NON-LINEAR :NONLINEAR-VIRTUAL)	Indicates which windows CLX notifies of the window exit event.
<i>focus-p</i>	CLX:BOOLEAN	If true, specifies that the event window is the focus window. If false, one of the children of the event window is the focus window.

Table 12-26: :MAPPING-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
<i>request</i>	(MEMBER :MODIFIER :KEYBOARD :POINTERT)	The type of mapping change being reported.
<i>start</i>	INTEGER	The first number of the range of altered keys, set only if <i>request</i> is :KEYBOARD.
<i>count</i>	INTEGER	The last number of the range of altered keys, set only if <i>request</i> is :KEYBOARD.

Table 12-27: :MAP-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
<i>event-window</i>	CLX:WINDOW	The event window.
<i>window</i>	CLX:WINDOW	The window that has been mapped.
<i>override-redirect-p</i>	CLX:BOOLEAN	Specifies whether the server ignores requests to map the window.

Table 12–28: :MAP-REQUEST Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>parent</i> <i>event-window</i>]	CLX:WINDOW	The parent of the window to be mapped (<i>parent</i> and <i>event-window</i> are synonyms).
<i>window</i>	CLX:WINDOW	The window that is to be mapped.

Table 12–29: :MOTION-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The event window (<i>window</i> and <i>event-window</i> are synonyms).
<i>root</i>	CLX:WINDOW	The root of the source window.
<i>child</i>	(OR NULL CLX:WINDOW)	The source window.
<i>same-screen-p</i>	CLX:BOOLEAN	Indicates whether the event window is on the same screen as the root window.
<i>x</i>	INTEGER	The X coordinate of the pointer in the event window.
<i>y</i>	INTEGER	The Y coordinate of the pointer in the event window.
<i>root-x</i>	INTEGER	The X coordinate of the pointer relative to the root window.
<i>root-y</i>	INTEGER	The Y coordinate of the pointer relative to the root window.
<i>state</i>	CLX:MASK16	The state of the mouse buttons just prior to the event.
<i>time</i>	CLX:TIMESTAMP	The time in milliseconds at which the event occurred.
<i>hint-p</i>	CLX:BOOLEAN	Indicates whether other motion events that CLX has not yet reported have occurred.

Table 12–30: :NO-EXPOSURE Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>drawable</i> <i>event-window</i>]	CLX:DRAWABLE	The window or pixmap reporting the event (<i>drawable</i> and <i>event-window</i> are synonyms).
<i>major</i>	INTEGER	Indicates whether the graphics request was a CopyArea (62) or a CopyPlane (63).
<i>minor</i>	INTEGER	The value zero. Reserved for use by extensions.

Table 12-31: :PROPERTY-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The window whose property was changed (<i>window</i> and <i>event-window</i> are synonyms).
<i>atom</i>	KEYWORD	Identifies the property that was changed. For more information on properties and atoms, see Section 6.4.
<i>state</i>	(MEMBER :NEW-VALUE :DELETED)	Indicates whether the property is being changed (or rotated) or is being deleted.
<i>time</i>	CLX:TIMESTAMP	Server time that the property changed.

Table 12-32: :REPARENT-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
<i>event-window</i>	CLX:WINDOW	The event window.
<i>window</i>	CLX:WINDOW	The window that is to be reparented.
<i>parent</i>	CLX:WINDOW	The new parent of <i>window</i> .
<i>x</i>	INTEGER	The X coordinate of the origin of the new parent window.
<i>y</i>	INTEGER	The Y coordinate of the origin of the new parent window.
<i>override-redirect-p</i>	CLX:BOOLEAN	Specifies whether the server ignores requests to reparent <i>window</i> .

Table 12-33: :RESIZE-REQUEST Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The window that is to be resized (<i>window</i> and <i>event-window</i> are synonyms).
<i>width</i>	INTEGER	The new width of the window.
<i>height</i>	INTEGER	The new height of the window.

Table 12–34: :SELECTION-CLEAR Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The window losing ownership of the selection (<i>window</i> and <i>event-window</i> are synonyms).
<i>selection</i>	KEYWORD	The selection atom.
<i>time</i>	CLX:TIMESTAMP	Last time change recorded for the selection.

Table 12–35: :SELECTION-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The window that owns the selection (<i>window</i> and <i>event-window</i> are synonyms).
<i>selection</i>	KEYWORD	The selection atom.
<i>target</i>	KEYWORD	The target type atom.
<i>property</i>	(OR NULL KEYWORD)	The atom that specifies a property.
<i>time</i>	(OR CLX:TIMESTAMP KEYWORD)	A timestamp.

Table 12–36: :SELECTION-REQUEST Event-Key

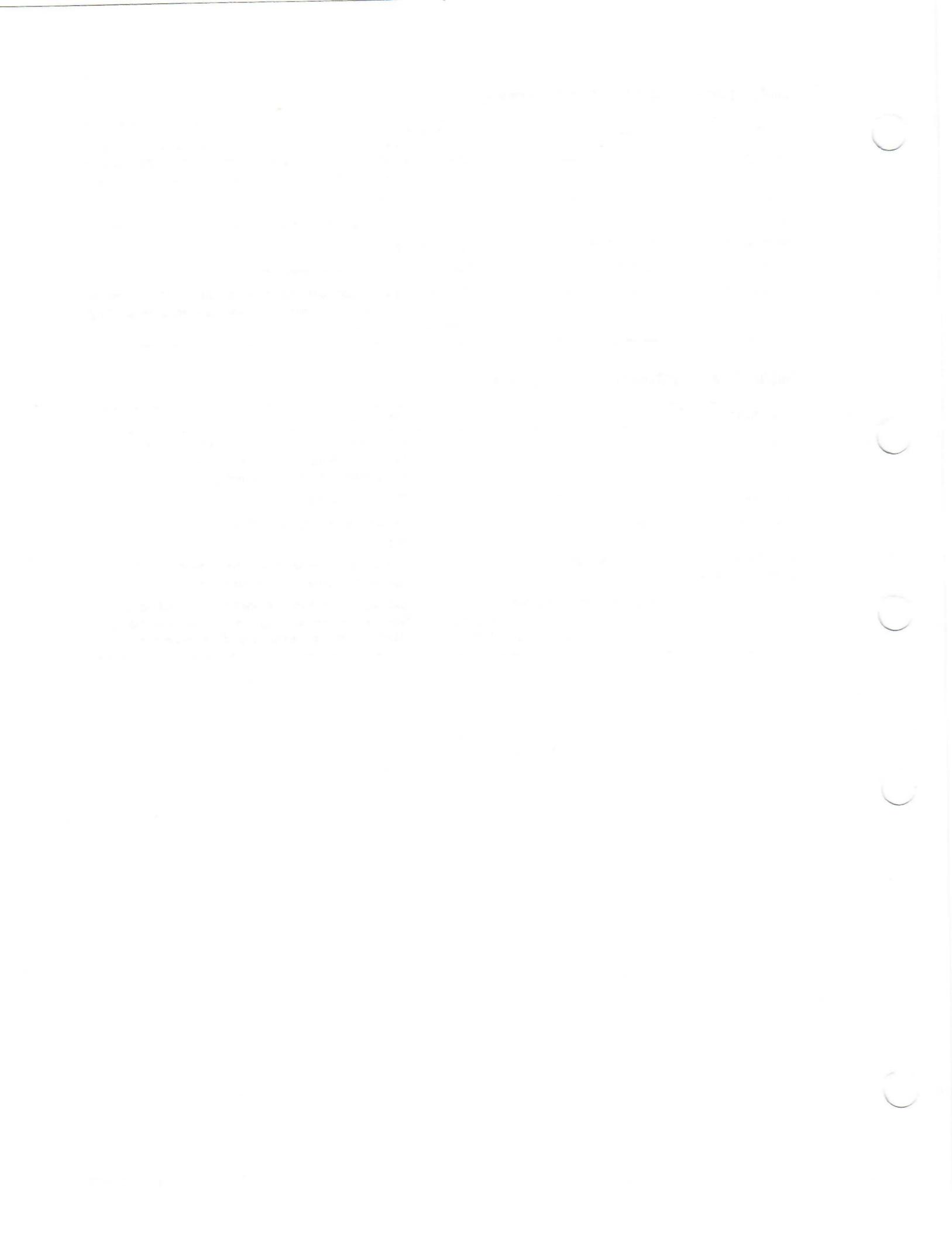
Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The window that owns the selection (<i>window</i> and <i>event-window</i> are synonyms).
<i>requestor</i>	CLX:WINDOW	The window that requests the selection.
<i>selection</i>	KEYWORD	The selection atom.
<i>target</i>	KEYWORD	The target type atom.
<i>property</i>	(OR NULL KEYWORD)	The atom that specifies a property.
<i>time</i>	INTEGER	A timestamp, expressed in milliseconds, or the keyword :CURRENT-TIME from the ConvertSelection request.

Table 12–37: :UNMAP-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
<i>event-window</i>	CLX:WINDOW	The event window.
<i>window</i>	CLX:WINDOW	The window that was unmapped.
<i>configure-p</i>	CLX:BOOLEAN	If true, indicates that the unmap event occurred as a result of resizing the parent of <i>window</i> when the window gravity of <i>window</i> is :UNMAP-GRAVITY.

Table 12–38: :VISIBILITY-NOTIFY Event-Key

Argument	Type	Meaning
<i>display</i>	CLX:DISPLAY	The display on which the event occurred. This argument is available only to CLX:PROCESS-EVENT handler functions.
<i>event-key</i>	CLX:EVENT-KEY	The current event.
<i>send-event-p</i>	CLX:BOOLEAN	True if the event came from a CLX:SEND-EVENT request.
[<i>window</i> <i>event-window</i>]	CLX:WINDOW	The window whose visibility changed (<i>window</i> and <i>event-window</i> are synonyms).
<i>state</i>	(MEMBER :UNOBSCURED :PARTIALLY-OBSCURED :FULLY-OBSCURED)	Indicates whether the window has changed from being obscured to being visible, or from being visible to being partially or fully obscured.



Chapter 13

Window and Session Management

A window or session manager program completes the following types of tasks:

- Reparenting windows
- Customizing the keyboard and pointer
- Using the screen saver
- Controlling network access
- Closing the connection
- Finding extensions

NOTE

Most clients are not responsible for window or session management and do not need to use these routines. A client could use these routines if there were no formal window or session manager program. Window or session management routines must be used with great care, as they can affect the operation of other applications.

13.1 Reparenting Windows

You can change a window's parent to another window on the same screen with the CLX:REARENT-WINDOW function. Its format is:

`CLX:REARENT-WINDOW window parent x y`

window The CLX:WINDOW object whose parent is changed.
parent The CLX:WINDOW object that is the new parent.
x and *y* Integers that specify the X and Y coordinates of the origin of *window* in *parent*.

If the specified *window* is mapped, CLX:REARENT-WINDOW automatically unmaps it; removes it from its current position in the window hierarchy; inserts it as the child of the specified *parent*; and maps it again. The *window* is placed on top of the stacking order with respect to its new siblings.

13.2 Customizing the Keyboard and the Pointer

CLX provides functions that you can use to change the keyboard controls; set or obtain the list of auto-repeat keys; ring the bell; set or obtain the pointer button or keyboard mappings; and obtain a bit vector for the keyboard. This section discusses the user-preference options of bell and key click loudness, pointer behavior, and so on.

13.2.1 Ringing the Bell

The CLX:BELL function rings the keyboard bell. Its format is:

CLX:BELL *display* &OPTIONAL *percent-from-normal*

display A CLX:DISPLAY object.

percent-from-normal An integer from -100 (off) to 100 (loudest), inclusive. The default value is zero.

The optional *percent-from-normal* argument specifies a percentage increase (positive values) or decrease (negative values) from the volume set in a LISP program with the CLX:CHANGE-KEYBOARD-CONTROL function or, if the DECwindows Session Manager is present, with the Keyboard... item of the Customize menu. The pitch and duration of the bell can be modified only with the CLX:CHANGE-KEYBOARD-CONTROL function.

13.2.2 Keyboard and Pointer Mappings

A "key code" represents a physical (or logical) key on the keyboard. Key codes are between 8 and 255, inclusive. The mapping between keys and key codes cannot be changed. A "keysym" is an encoding of the symbol on the cap of a key. A list of keysyms is associated with each key code. The first keysym in each list is the one for no modifiers. See the *X Window System: C Library and Protocol Reference* for information on keysym encoding. This section explains how to control the bindings of keysyms to keys and modifiers.

The CLX:KEYBOARD-MAPPING function returns the valid key codes for a specified display. Its format is:

CLX:KEYBOARD-MAPPING *display*
&KEY :FIRST-KEYCODE :START :END :DATA

display A CLX:DISPLAY object.

:FIRST-KEYCODE An integer that specifies which keycode to start at, or NIL. The default value is the :MIN-KEYCODE component of *display*.

:START A CLX:ARRAY-INDEX value that specifies where (in the return array) to put the first keycode, or NIL. The default value is :FIRST-KEYCODE.

:END A CLX:ARRAY-INDEX value or NIL. The difference between :END and :START is the number of keycodes to return. The default :END is one more than the :MAX-KEYCODE of *display*.

:DATA An array in which to return the key codes, or NIL.

The function returns an array of keysyms. If :DATA is specified, the results are put there.

To change the mapping of keysyms to key codes, use the CLX:CHANGE-KEYBOARD-MAPPING function. Its format is:

CLX:CHANGE-KEYBOARD-MAPPING *display keysyms*
&KEY :START :END :FIRST-KEYCODE

display A CLX:DISPLAY object.

keysyms An array.

<code>:START</code> and <code>:END</code>	Two CLX:ARRAY-INDEX values that specify the subrange of keyyms.
<code>:FIRST-KEYCODE</code>	An integer that specifies the first keycode to store at. The default value is that of <code>:START</code> .

X permits at most eight modifier keys. The CLX:MODIFIER-MAPPING function returns the key codes currently being used as modifiers. Its format is:

`CLX:MODIFIER-MAPPING display`

where *display* is a CLX:DISPLAY object. The function returns multiple values, each of which is a list of integers representing the key codes that have a modifier bound to them:

```
shift
lock
control
mod1
mod2
mod3
mod4
mod5
```

Use the CLX:SET-MODIFIER-MAPPING function to change the key codes of the keys (if any) that are to be used as modifiers. The format of the CLX:SET-MODIFIER-MAPPING function is:

`CLX:SET-MODIFIER-MAPPING display &KEY :SHIFT :LOCK :CONTROL
:MOD1 :MOD2 :MOD3 :MOD4 :MOD5`

`display` A CLX:DISPLAY object.
`:SHIFT` Sequences of integers that specify the key codes of the modifier keys.
`:LOCK`
`:CONTROL`
and
`:MODn`

The CLX:SET-MODIFIER-MAPPING function returns a keyword indicating the status of the change: `:SUCCESS`, `:BUSY`, meaning that a modifier whose key codes are being changed is logically down, or `:FAILED`, meaning that the specified key codes are not valid for the modifier.

The CLX:POINTER-MAPPING function returns a mapping list that defines which buttons are enabled for the pointer cursor on the specified display. The format of the CLX:POINTER-MAPPING function is:

`CLX:POINTER-MAPPING display &KEY :RESULT-TYPE`

`display` A CLX:DISPLAY object.
`:RESULT-TYPE` The LISP type of the return value. The default value is 'LIST.
The function returns a sequence of INTEGER.

13.2.3 Keycode Mapping

The functions in this section provide translations among characters, key codes, and keyyms.

The CLX:CHARACTER->KEYSYMS function returns a list of the keysyms that match a specified character. If a display is provided, translations specific to that display are used; if not, global translations are used. The format of the CLX:CHARACTER->KEYSYMS function is:

CLX:CHARACTER->KEYSYMS *character* &OPTIONAL *display*

character Any LISP object.

display A CLX:DISPLAY object.

The CLX:KEYCODE->KEYSYM function returns the keysym to which a specified keycode is bound. Its format is:

CLX:KEYCODE->KEYSYM *display keycode keysym-index*

display A CLX:DISPLAY object.

keycode An integer.

keysym-index An integer or NIL.

The CLX:KEYSYM->CHARACTER function finds the character or string associated with a keysym. Its format is:

CLX:KEYSYM->CHARACTER *display keysym* &OPTIONAL *state*

display A CLX:DISPLAY object.

keysym A CLX:KEYSYM value.

state An integer or NIL. This argument is ignored by VAX LISP.

The CLX:KEYSYM->CHARACTER function returns multiple values if *keysym* is bound to more than one character. The first return value is the character to which *keysym* is bound, or the first character of the string; the second is the string, if any; and the third is the length of the string.

The CLX:KEYSYM->KEYCODES function returns the keycodes corresponding to the specified CLX:KEYSYM, or NIL if there is none. Its format is:

CLX:KEYSYM->KEYCODES *display keysym*

display A CLX:DISPLAY object.

keysym A CLX:KEYSYM.

The CLX:KEYCODE->CHARACTER function returns the character corresponding to the specified keycode, or NIL if there isn't one. Its format is:

CLX:KEYCODE->CHARACTER *display keycode state*
&KEY :KEYSYM-INDEX :KEYSYM-INDEX-FUNCTION

display A CLX:DISPLAY object.

keycode An integer.

state A CLX:MASK16 value that specifies the state of modifier keys (shift, control, and so on).

:KEYSYM-INDEX An integer or NIL. The default value is the result of the :KEYSYM-INDEX-FUNCTION function.

<code>:KEYSYM-INDEX-FUNCTION</code>	NIL or a function that takes the following arguments: <i>(char0 state caps-lock-p keysyms-per-keycode)</i>
<code>char0</code>	The character associated with the first :KEYSYM-INDEX value.
<code>state</code>	A CLX:MASK16 value that specifies the state of modifier keys (shift, control, and so on).
<code>caps-lock-p</code>	True when the keysym associated with the lock modifier is for caps-lock.
<code>keysyms-per-keycode</code>	The number of keysyms per keycode.

The default is #'CLX:KEYSYM-INDEX-FUNCTION.

The CLX:KEYCODE->CHARACTER function returns multiple values if *keycode* is bound to more than one character. The first return value is the character to which *keycode* is bound, or the first character of the string; the second is the string, if any; and the third is the length of the string.

13.2.4 Keyboard and Pointer Controls

CLX allows client programs to query and to change several keyboard and pointer settings. The CLX:KEYBOARD-CONTROL function returns the current keyboard settings. Its format is:

`CLX:KEYBOARD-CONTROL display`

where *display* is a CLX:DISPLAY object. The keyboard settings are returned as multiple values:

<code>key-click-percent</code>	The volume of key clicks between 0 (off) and 100 (loudest).
<code>bell-percent</code>	The volume of a bell ring between 0 (off) and 100 (loudest).
<code>bell-pitch</code>	The pitch (specified in hertz) of the bell.
<code>bell-duration</code>	The duration of a bell ring, in milliseconds.
<code>led-mask</code>	Each 1 bit indicates an LED that is on. The least significant bit corresponds to the first LED.
<code>global-auto-repeat</code>	:ON if auto-repeat is enabled for the keyboard as a whole; :OFF if it is not.
<code>auto-repeats</code>	A bit vector where each 1 bit indicates that auto-repeat is enabled for the corresponding key.

The CLX:CHANGE-KEYBOARD-CONTROL allows you to change the keyboard settings. Its format is:

`CLX:CHANGE-KEYBOARD-CONTROL display
&KEY :KEY-CLICK-PERCENT :BELL-PERCENT :BELL-PITCH :BELL-DURATION
:LED :LED-MODE :KEY :AUTO-REPEAT-MODE`

<code>display</code>	A CLX:DISPLAY object.
<code>:KEY-CLICK-PERCENT</code>	An integer that represents the volume of a key click from 0 (off) to 100 (loudest), :DEFAULT, or NIL.
<code>:BELL-PERCENT</code>	An integer that specifies the volume of a bell ring from 0 (off) to 100 (loudest), :DEFAULT, or NIL.

:BELL-PITCH	An integer that specifies the pitch of the bell in hertz, :DEFAULT, or NIL.
:BELL-DURATION	An integer that specifies the duration of a bell ring in milliseconds, :DEFAULT, or NIL.
:LED	An integer whose 1 bits specify which LEDs (counting from zero) are affected by the :LED-MODE argument, or NIL.
:LED-MODE	The keyword :ON or :OFF that specifies the state of LEDs on the keyboard, or NIL.
:KEY	An integer whose 1 bits specify that the corresponding key has auto-repeat mode enabled, or NIL.
:AUTO-REPEAT-MODE	The keyword :ON, :OFF or :DEFAULT that specifies the global auto-repeat mode for the keyboard, or NIL.

If both :LED-MODE and :LED are specified, the state of the appropriate LEDs is changed. If only :LED-MODE is specified, the state of all LEDs is changed.

Similarly, if both :AUTO-REPEAT-MODE and :KEY are specified, the auto-repeat mode of that key is changed. If only :AUTO-REPEAT-MODE is specified, the global auto-repeat mode for the entire keyboard is changed, without affecting the individual key settings. It is an error to specify a :KEY without an :AUTO-REPEAT-MODE.

When the global auto-repeat mode is :ON, keys obey their individual auto-repeat modes; when the global mode is :OFF, no keys auto-repeat. An auto-repeating key generates alternating :KEY-PRESS and :KEY-RELEASE events. When a key is used as a modifier, it does not auto-repeat regardless of its auto-repeat setting.

13.2.5 Setting Pointer Controls

The way the pointer cursor moves in response to movement of the pointing device is affected by two pointer controls: acceleration and threshold. Acceleration, expressed as a ratio, is a multiplier for cursor movement. For example, an acceleration of 3/1 means that the pointer moves three times as fast as the default. Acceleration takes effect only if the pointer moves more than threshold pixels at a time, and applies only to the distance moved beyond the threshold. CLX provides functions that allow you to find the current pointer controls and to set their values.

The CLX:POINTER-CONTROL function returns the pointer movement values for acceleration and the threshold at which acceleration should be applied. Its format is:

CLX:POINTER-CONTROL *display*

where *display* is a CLX:DISPLAY object. The function returns two numeric values:

acceleration The ratio of the current pointer speed to the default speed.

threshold The distance, in pixels, that the pointer cursor must move before *acceleration* is applied.

The CLX:CHANGE-POINTER-CONTROL function lets you define how the pointer cursor moves. If you supply a numeric value for :ACCELERATION, it must be positive. CLX will rationalize the value if necessary, and the ratio may be rounded arbitrarily by the X server.

The format of the CLX:CHANGE-POINTER-CONTROL function is:

CLX:CHANGE-POINTER-CONTROL *display* &KEY :ACCELERATION :THRESHOLD

<i>display</i>	A CLX:DISPLAY object.
:ACCELERATION	A number, :DEFAULT, or NIL.
:THRESHOLD	An INTEGER, :DEFAULT, or NIL.

13.3 Using the Screen Saver

Users can set the timeout interval for, and enable or disable, the screen saver feature interactively in the DECwindows Session Manager with the Window... item of the Customize menu. CLX functions allow client programs to override the screen saver settings.

13.3.1 Querying the Screen Saver

The CLX:SCREEN-SAVER function returns the current screen saver settings for the specified display. The format of this function is:

CLX:SCREEN-SAVER *display*

where *display* is a CLX:DISPLAY object. The values returned are:

<i>timeout</i>	The time, in seconds, that must elapse with no input from the keyboard or pointer before the screen saver turns on. A value of zero means that the screen saver is disabled.						
<i>interval</i>	The time, in seconds, between invocations of the screen saver.						
<i>blanking</i>	One of three keywords specifying how the screen is cleared: <table> <tr> <td>:YES</td> <td>Blank the screen. This can be returned only if the display hardware supports video blanking.</td> </tr> <tr> <td>:NO</td> <td>Do not blank the screen. If exposures are allowed (see below), or if the screen can be regenerated without sending exposure events to clients, the screen is tiled with the root window background tile. If exposures are not allowed or the exposure events are sent to clients, the screen does not change.</td> </tr> <tr> <td>:DEFAULT</td> <td>The default blanking method is used.</td> </tr> </table>	:YES	Blank the screen. This can be returned only if the display hardware supports video blanking.	:NO	Do not blank the screen. If exposures are allowed (see below), or if the screen can be regenerated without sending exposure events to clients, the screen is tiled with the root window background tile. If exposures are not allowed or the exposure events are sent to clients, the screen does not change.	:DEFAULT	The default blanking method is used.
:YES	Blank the screen. This can be returned only if the display hardware supports video blanking.						
:NO	Do not blank the screen. If exposures are allowed (see below), or if the screen can be regenerated without sending exposure events to clients, the screen is tiled with the root window background tile. If exposures are not allowed or the exposure events are sent to clients, the screen does not change.						
:DEFAULT	The default blanking method is used.						
<i>exposures</i>	One of three keywords specifying whether exposure events are generated: <table> <tr> <td>:YES</td> <td>Exposures are allowed. This is the default.</td> </tr> <tr> <td>:NO</td> <td>Exposures are not allowed.</td> </tr> <tr> <td>:DEFAULT</td> <td>The default value is used.</td> </tr> </table>	:YES	Exposures are allowed. This is the default.	:NO	Exposures are not allowed.	:DEFAULT	The default value is used.
:YES	Exposures are allowed. This is the default.						
:NO	Exposures are not allowed.						
:DEFAULT	The default value is used.						

13.3.2 Setting the Screen Saver

The CLX:SET-SCREEN-SAVER function allows client programs to set the controls for the screen saver. The format of this function is:

CLX:SET-SCREEN-SAVER *display timeout interval blanking exposures*

<i>display</i>	A CLX:DISPLAY object.
<i>timeout</i>	An integer that specifies the time that must elapse without input from the keyboard or pointer before the screen saver turns on, or the keyword :DEFAULT. A value of zero means that the screen saver is disabled. :DEFAULT means that the default setting is restored.
<i>interval</i>	An integer or the keyword :DEFAULT that specifies the time between invocations of the screen saver.

blanking : YES, :NO, or :DEFAULT.

exposures : YES, :NO, or :DEFAULT.

The values of *timeout* and *interval* are specified in seconds. The keyword values for the *exposures* argument are:

- :YES Blank the screen. This can be used only if the display hardware supports video blanking.
- :NO Do not blank the screen. If exposures are allowed (see below), or if the screen can be regenerated without sending exposure events to clients, the screen is tiled with the root window background tile. If exposures are not allowed or the exposure events are sent to clients, the screen does not change.
- :DEFAULT The default blanking method is used.

The keyword values for the *blanking* argument are:

- :YES Exposures are allowed.
- :NO Exposures are not allowed.
- :DEFAULT The default value is used.

The CLX:RESET-SCREEN-SAVER function resets the timeout clock to zero, as if input had just been received. The format of this function is:

CLX:RESET-SCREEN-SAVER *display*

where *display* is a CLX:DISPLAY object.

13.3.3 Enabling the Screen Saver

The CLX:ACTIVATE-SCREEN-SAVER function enables the screen saver, even if it is currently disabled. The format of this function is:

CLX:ACTIVATE-SCREEN-SAVER *display*

where *display* is a CLX:DISPLAY object.

13.4 Controlling Network Access

For each display, CLX maintains an "access control list" of the machines that can connect across the network to the X server controlling that display. CLX provides four routines for controlling network access:

- CLX:ADD-ACCESS-HOST
- CLX:REMOVE-ACCESS-HOST
- CLX:ACCESS-HOSTS
- CLX:ACCESS-CONTROL

The first two allow client programs to specify which machines have access to a particular display. The last two return information about network access to a display.

Users can manipulate the access control list in the DECwindows Session Manager by using the Security... item of the Customize menu. Such changes may also be saved from one session to the next. See the VMS DECwindows User's Guide for details.

13.4.1 Adding and Removing Hosts

The CLX:ADD-ACCESS-HOST function dynamically adds one host to the list of hosts that can connect to the server controlling a display. This function does not execute successfully unless the client issuing the command resides on the same host as the server.

The format of the CLX:ADD-ACCESS-HOST function is:

CLX:ADD-ACCESS-HOST *display host*

display A CLX:DISPLAY object.

host A STRING that names the machine to be added to the access list.

The REMOVE-ACCESS-HOST function removes a host from the access control list of the display. Its format is:

CLX:REMOVE-ACCESS-HOST *display host*

display A CLX:DISPLAY object.

host A STRING that names the machine to be removed from the access list.

NOTE

The server does not retain changes to the access control list made with CLX:ADD-ACCESS-HOST or CLX:REMOVE-ACCESS-HOST when its host is rebooted.

13.4.2 Getting Information About Hosts

The access control list of a display may be enabled or disabled. When the access control list is enabled, only hosts on the list can connect to that display.

The CLX:ACCESS-HOSTS function returns the access control list of a display and indicates whether the access list is enabled. The format of the CLX:ACCESS-HOSTS function is:

CLX:ACCESS-HOSTS *display* &KEY :RESULT-TYPE

display A CLX:DISPLAY object.

:RESULT-TYPE The LISP type of the first return value. The default is 'LIST.

The function returns two values:

hosts A sequence of strings naming the hosts on the access control list of *display*.

enabled-p A CLX:BOOLEAN value that is true if the access list is enabled for *display*.

The CLX:ACCESS-CONTROL function also returns a CLX:BOOLEAN value that is true if the access control list is enabled for *display*, false if it is disabled. The format of the CLX:ACCESS-CONTROL function is:

CLX:ACCESS-CONTROL *display*

where *display* is a CLX:DISPLAY object. The CLX:ACCESS-CONTROL function may be used with SETF if your client program is on the same machine as the display.

13.5 Closing the Connection

The functions described in this section allow client programs to control the processing that takes place when a connection to the X server is closed. Specifically, you can:

- Determine what happens to a client's resources when the client disconnects.
- Disconnect clients associated with particular resources.
- Save windows from being destroyed when their client disconnects.

13.5.1 Deallocating Resources

The CLX:CLOSE-DOWN-MODE function indicates what happens to a client's resources when the client disconnects from the server. The default close-down mode is :DESTROY, which frees all client resources. The :RETAIN-TEMPORARY and :RETAIN-PERMANENT modes keep the client's resources after it disconnects until a client invokes CLX:KILL-TEMPORARY-CLIENTS or CLX:KILL-CLIENTS, respectively.

The format of the CLX:CLOSE-DOWN-MODE function is:

CLX:CLOSE-DOWN-MODE *display*

where *display* is a CLX:DISPLAY object. The function returns the current close-down mode of the specified display, either :DESTROY, :RETAIN-PERMANENT or :RETAIN-TEMPORARY. This value is cached locally in the display object so that invoking CLX:CLOSE-DOWN-MODE generates no server request.

The CLX:CLOSE-DOWN-MODE function may be used with SETF:

(SETF (CLX:CLOSE-DOWN-MODE *display*) *mode*)

13.5.2 Disconnecting Other Clients

CLX provides the CLX:KILL-CLIENT function for disconnecting the client associated with a particular resource, and the CLX:KILL-TEMPORARY-CLIENT function for destroying the resources associated with clients that disconnected in :RETAIN-TEMPORARY close-down mode. (See Section 13.5.1 for more details on close-down mode.)

The format of the CLX:KILL-CLIENT function is:

CLX:KILL-CLIENT *display resource-id*

display A CLX:DISPLAY object.

resource-id A CLX:RESOURCE-ID value that identifies the resource (window, colormap, and so on) associated with the client to be disconnected.

The format of the CLX:KILL-TEMPORARY-CLIENTS function is:

CLX:KILL-TEMPORARY-CLIENTS *display*

where *display* is a CLX:DISPLAY object.

13.5.3 Saving Windows

The saveset is a list of other clients' windows that, if they are children of your client's windows, should not be destroyed when your client disconnects from the server and that should be remapped if your window is unmapped. When a connection is closed, the server reparents saveset windows to an ancestor not created by the disconnecting client and maps them if they are unmapped.

The CLX:ADD-TO-SAVE-SET function adds a window to your client's saveset. Its format is:

CLX:ADD-TO-SAVE-SET *window*

where *window* is a CLX:WINDOW object created by another client.

The server automatically removes windows from the saveset when they are destroyed. You use the CLX:REMOVE-FROM-SAVE-SET function to remove an existing window from your client's saveset. Its format is:

CLX:REMOVE-FROM-SAVE-SET *window*

where *window* is a CLX:WINDOW object created by some other client.

13.6 Finding Extensions

CLX provides two functions for finding extensions included in a particular implementation. The CLX:QUERY-EXTENSION function returns information on a specific extension, and the CLX:LIST-EXTENSIONS function returns the extensions supported by the server.

The format of the CLX:QUERY-EXTENSION function is:

CLX:QUERY-EXTENSION *display name*

display A CLX:DISPLAY object.

name A CLX:STRINGABLE that names an extension.

If the extension specified by *name* is supported on *display*, CLX:QUERY-EXTENSION returns three values:

<i>major-opcode</i>	The major opcode assigned to the extension by the server, or NIL if it has none.
<i>first-event</i>	The base event type code assigned to the extension, or NIL if the extension generates no additional event types.
<i>first-error</i>	The base error code assigned to the extension, or NIL if the extension generates no additional error codes.

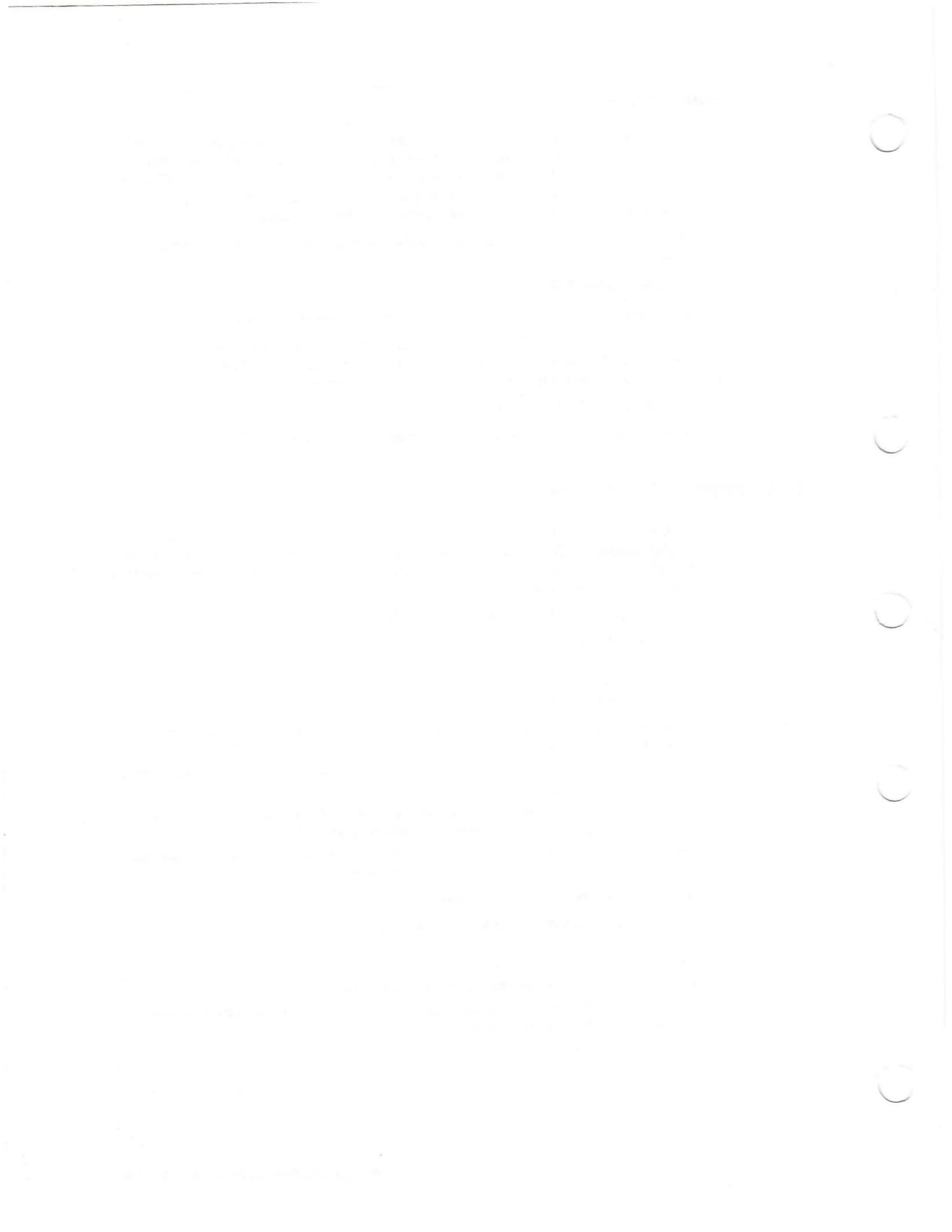
The format of the CLX:LIST-EXTENSIONS function is:

CLX:LIST-EXTENSIONS *display* &KEY :RESULT-TYPE

display A CLX:DISPLAY object.

:RESULT-TYPE The LISP type of the returned value.

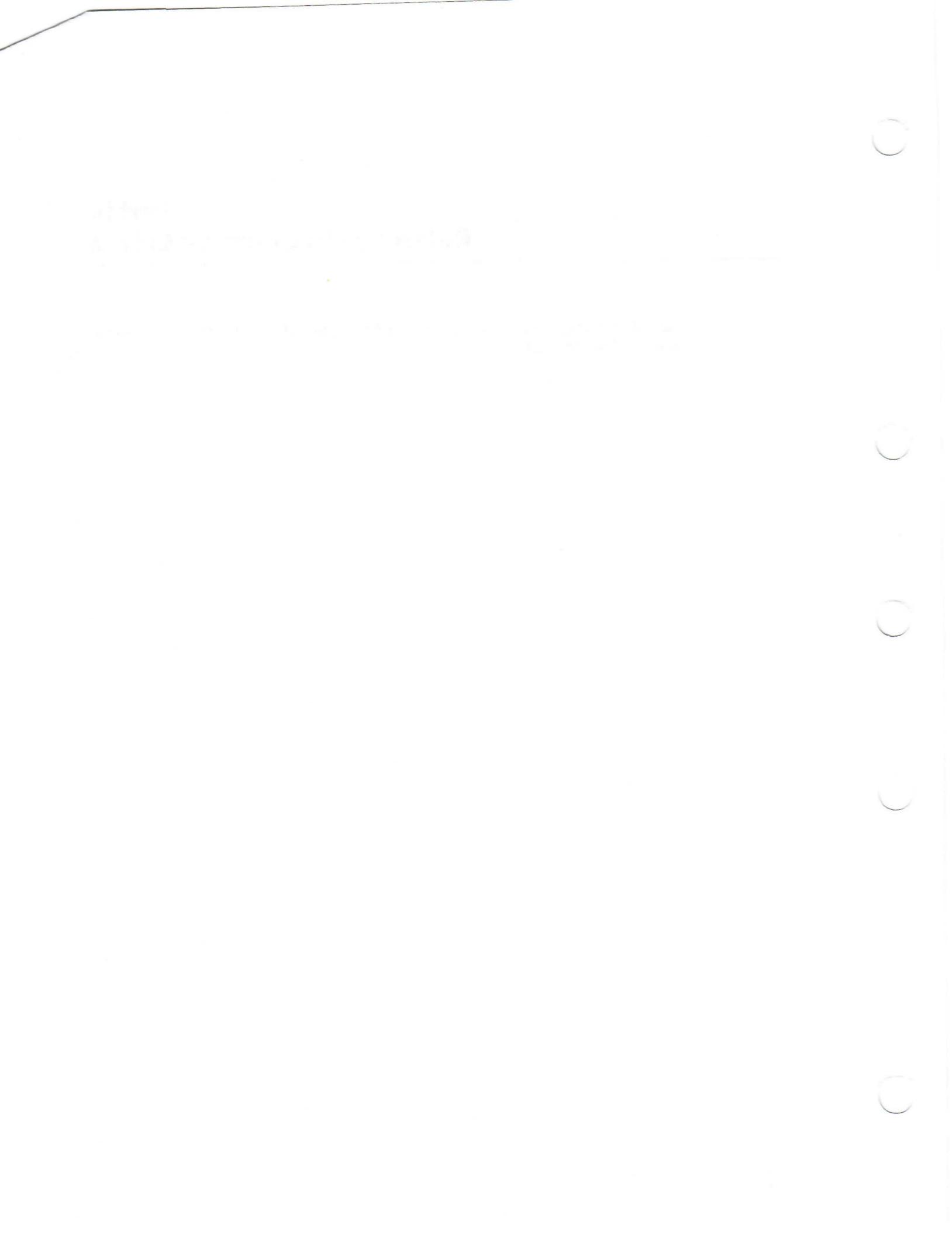
The function returns a sequence of strings naming all the extensions supported by the server that controls *display*.



Part IV

Reference to Common LISP X

Part IV contains, in alphabetical order, a description of each function and macro exported from the CLX: package.



CLX:ACCESS-CONTROL Function

CLX:ACCESS-CONTROL Function

Returns `T` if the access control list is enabled on the specified display, `NIL` if it is not. This function can be used with `SETF` if your client program is on the same machine as the display.

Format

CLX:ACCESS-CONTROL *display*

Argument

display

A CLX:DISPLAY object.

Return Value

A CLX:BOOLEAN value.

CLX:ACCESS-HOSTS Function

Returns the access control list of the specified display, and indicates whether the list is enabled. In VAX LISP, hosts are strings.

Format

CLX:ACCESS-HOSTS *display* &KEY :RESULT-TYPE

Arguments

display

A CLX:DISPLAY object.

:RESULT-TYPE

The LISP type of the first return value. The default value is 'LIST.

Return Values

Two values:

- A sequence (by default, a list) of hosts.
- A CLX:BOOLEAN value.

CLX:ACTIVATE-SCREEN-SAVER Function

CLX:ACTIVATE-SCREEN-SAVER Function

Enables the screen saver feature.

Format

CLX:ACTIVATE-SCREEN-SAVER *display*

Argument

display

A CLX:DISPLAY object.

Return Value

Unspecified.

CLX:ADD-ACCESS-HOST Function

Adds the specified host to the access control list of the specified display.

Format

CLX:ADD-ACCESS-HOST *display host*

Arguments

display

A CLX:DISPLAY object.

host

A STRING that names a machine.

Return Value

Unspecified.

CLX:ADD-TO-SAVE-SET Function

Adds the specified window to your client's saveset so that it is remapped rather than destroyed when its client exits. The window must have been created by some other client or an error occurs.

Format

CLX:ADD-TO-SAVE-SET *window*

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:ALIST Type Specifier

An association list of key names and data types. This type is an abstraction that is used for describing the structure of other CLX types.

Representation

key-type-and-name datum-type-and-name

CLX:ALLOC-COLOR Function

Allocates a named or exact color for shared use.

To use a named color, pass a CLX:STRINGABLE value for the *color* argument. To specify an exact color, create a CLX:COLOR structure and pass it and its associated colormap to the CLX:ALLOC-COLOR function.

Format

CLX:ALLOC-COLOR *colormap color*

CLX:ALLOC-COLOR Function

Arguments

colormap

A CLX:COLORMAP object.

color

A CLX:STRINGABLE value or CLX:COLOR object.

Return Values

Three values:

- The CLX:PIXEL value of the allocated color.
- A CLX:COLOR structure that contains the RGB values supported by the display (the "screen color").
- A CLX:COLOR structure that contains the RGB values specified by the DECwindows named color (the "exact color").

CLX:ALLOC-COLOR-CELLS Function

Allocates cells in a specified colormap for exclusive use by a client program. Use this function on a pseudocolor or a gray-scale device. Use the CLX:ALLOC-COLOR-PLANES function on a direct color device.

Format

CLX:ALLOC-COLOR-CELLS *colormap colors &KEY :PLANES :CONTIGUOUS-P :RESULT-TYPE*

Arguments

colormap

The CLX:COLORMAP in which cells are allocated.

colors

The INTEGER number of pixels allocated.

:PLANES

The INTEGER number of planes returned. The default is 0.

:CONTIGUOUS-P

A CLX:BOOLEAN value: T specifies that the allocated color cells must be contiguous entries in the colormap.

:RESULT-TYPE

The LISP type of the return values. The default value is 'LIST.

CLX:ALLOC-COLOR-CELLS Function

Return Values

Two values:

- A sequence of CLX:PIXEL values.
- A sequence of CLX:MASK16 values representing plane masks.

CLX:ALLOC-COLOR-PLANES Function

Allocates color cells for exclusive use by a client program. Use this function to simulate a direct color device. Use the CLX:ALLOC-COLOR-CELLS function on a pseudocolor or gray-scale device.

Format

CLX:ALLOC-COLOR-PLANES *colormap colors &KEY :REDS :GREENS :BLUES :CONTIGUOUS-P :RESULT-TYPE*

Arguments

colormap

The CLX:COLORMAP in which cells are allocated.

colors

The INTEGER number of pixels returned.

:REDS :GREENS :BLUES

Three INTEGER values specifying the number of bits set in the returned plane masks. The default values are 0.

:CONTIGUOUS-P

A CLX:BOOLEAN value: T specifies that the allocated color cells must be contiguous entries in the colormap.

:RESULT-TYPE

The LISP type of the first return value. The default value is 'LIST.

Return Values

Four values:

- A sequence of CLX:PIXEL values.
- A CLX:MASK16 value representing the plane-mask for red values.
- A CLX:MASK16 value representing the plane-mask for green values.
- A CLX:MASK16 value representing the plane-mask for blue values.

CLX:ALLOW-EVENTS Function

CLX:ALLOW-EVENTS Function

Allows processing of events that were not reported because a device was grabbed.

Format

CLX:ALLOW-EVENTS *display mode &OPTIONAL time*

Arguments

display

A CLX:DISPLAY object.

mode

A keyword that specifies which events are to be released. The possible values are:

:ASYNC-POINTER	Allows pointer event processing to continue normally after a pointer has been stopped.
:SYNC-POINTER	Allows pointer event processing to continue normally after a pointer has been frozen or grabbed until the next :BUTTON-PRESS or :BUTTON-RELEASE event is reported to the client.
:REPLY-POINTER	If the pointer is actively grabbed by the client and is frozen, the pointer grab is released and that event is completely reprocessed.
:ASYNC-KEYBOARD	Allows keyboard event processing to continue normally after a keyboard has been frozen.
:SYNC-KEYBOARD	If the keyboard is frozen and actively grabbed by the client, keyboard event processing continues as usual until the next :KEY-PRESS or :KEY-RELEASE event is reported to the client.
:REPLAY-KEYBOARD	If the keyboard is actively grabbed by the client, and is frozen as the result of an event having been sent to the client either from the activation of a passive grab established by a call to CLX:GRAB-KEY or from a previous CLX:ALLOW-EVENTS with mode :SYNC-KEYBOARD (but not from a CLX:GRAB-KEYBOARD), the keyboard grab is released and that event is completely reprocessed.
:ASYNC-BOTH	If the pointer and the keyboard are frozen by the client, event processing (for both devices) continues normally.
:SYNC-BOTH	If the pointer and keyboard are frozen by the client, event processing (for both devices) continues normally until the next :BUTTON-PRESS, :BUTTON-RELEASE, :KEY-PRESS, or :KEY-RELEASE event is reported to the client for a grabbed device, at which time the devices again appear to freeze.

time

The CLX:TIMESTAMP when events are released.

CLX:ALLOW-EVENTS Function

Return Value

Unspecified.

CLX:ANGLE Type Specifier

Note that the representation of a CLX:ANGLE differs from the one actually transmitted in the protocol.

Representation

(NUMBER , (* -2 PI) , (* 2 PI))

CLX:ARC-SEQ Type Specifier

An arc sequence is a CLX:REPEAT-SEQUENCE of FIXNUM's (rectangles) and CLX:ANGLE's. Objects of type CLX:ARC-SEQ may be passed to the CLX:DRAW-ARCS function.

Representation

(CLX::REPEAT-SEQ (FIXNUM *x*) (FIXNUM *y*)
(FIXNUM *width*) (FIXNUM *height*)
(CLX:ANGLE *angle1*) (CLX:ANGLE *angle2*))

CLX:ARRAY-INDEX Type Specifier

This type is an abstraction that places upper and lower bounds on integers used as array indexes in CLX.

Representation

(INTEGER 0, ARRAY-DIMENSION-LIMIT)

CLX:ATOM-NAME Function

Given an integer resource-id, this function returns the name of the corresponding atom on a specified display.

Format

CLX:ATOM-NAME *display atom-id*

CLX:ATOM-NAME Function

Arguments

display

A CLX:DISPLAY object.

atom-id

An INTEGER value.

Return Value

A keyword.

CLX:BELL Function

Sounds the keyboard bell. The volume may be specified.

Format

CLX:BELL *display* &OPTIONAL *percent-from-normal*

Arguments

display

A CLX:DISPLAY object.

percent-from-normal

A number between -100 and 100, inclusive. The default is 0.

Return Value

Unspecified.

CLX:BIT-GRAVITY Type Specifier

The bit gravity of a window specifies where its contents could be relocated when it is resized. Objects of type CLX:BIT-GRAVITY are returned by the CLX:WINDOW-BIT-GRAVITY function, and can be used as a SETF value for that accessor or for the :BIT-GRAVITY component of the CLX:CREATE-WINDOW function.

Representation

(MEMBER :FORGET :STATIC :CENTER :NORTH :NORTH-EAST :EAST :SOUTH-EAST
:SOUTH :SOUTH-WEST :WEST :NORTH-WEST)

CLX:BITMAP Type Specifier

Bitmaps are pixmaps of depth 1, that is, arrays of bits.

Representation

(ARRAY BIT (* *))

CLX:BITMAP-FORMAT Structure

Objects of type CLX:BITMAP-FORMAT store the format in which images are transmitted and received by the server. The CLX:DISPLAY-BITMAP-FORMAT function returns the bitmap-format used by the specified display.

Constructor Function

CLX:MAKE-BITMAP-FORMAT

Accessor Functions

None of the following functions is a valid SETF place:

Name	Type Definition
CLX:BITMAP-FORMAT-LSB-FIRST-P	CLX:BOOLEAN
CLX:BITMAP-FORMAT-PAD	(MEMBER 8 16 32)
CLX:BITMAP-FORMAT-UNIT	(MEMBER 8 16 32)

Deallocator Function

Bitmap-formats are subject to garbage collection.

CLX:BITMAP-FORMAT-LSB-FIRST-P Function

Predicate indicates whether the least significant bit or the most significant bit is the leftmost bit in each unit of a bitmap.

Format

CLX:BITMAP-FORMAT-LSB-FIRST-P *bitmap-format*

Argument

bitmap-format

A CLX:BITMAP-FORMAT object.

CLX:BITMAP-FORMAT-LSB-FIRST-P Function

Return Value

A CLX:BOOLEAN value.

CLX:BITMAP-FORMAT-P Function

Predicate returns true if its argument is a bitmap-format.

Format

CLX:BITMAP-FORMAT-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:BITMAP-FORMAT-PAD Function

Returns the number of bits to a multiple of which each scanline is padded.

Format

CLX:BITMAP-FORMAT-PAD *bitmap-format*

Argument

bitmap-format

A CLX:BITMAP-FORMAT object.

Return Value

A member of (8 16 32).

CLX:BITMAP-FORMAT-UNIT Function

Returns the number of bits in a unit of a bitmap.

Format

CLX:BITMAP-FORMAT-UNIT *bitmap-format*

Argument

bitmap-format

A CLX:BITMAP-FORMAT object.

Return Value

A member of (8 16 32).

CLX:BOOLEAN Type Specifier

Represents values that are either true or false.

Representation

(OR NULL (NOT NULL))

CLX:CARD8 Type Specifier

An abstract type used for masks and other integer values less than 256.

Representation

FIXNUM

CLX:CARD16 Type Specifier

An abstract type used for masks and other integer values less than 65536.

Representation

FIXNUM

CLX:CARD29 Type Specifier

CLX:CARD29 Type Specifier

An abstract type used for masks and resource-ids.

Representation

INTEGER

CLX:CHANGE-ACTIVE-POINTER-GRAB Function

Modifies the active pointer grab on the specified display.

Format

CLX:CHANGE-ACTIVE-POINTER-GRAB *display event-mask*
 &OPTIONAL *cursor time*

Arguments

display

A CLX:DISPLAY object.

event-mask

The CLX:POINTER-EVENT-MASK to be changed.

cursor

The CLX:CURSOR to be displayed during the grab.

time

The CLX:TIMESTAMP when the grab is changed.

Return Value

Unspecified.

CLX:CHANGE-KEYBOARD-CONTROL Function

Modifies the keyboard settings on the specified display.

Format

CLX:CHANGE-KEYBOARD-CONTROL *display*
 &KEY **:KEY-CCLICK-PERCENT :BELL-PERCENT :BELL-PITCH**
 :BELL-DURATION :LED :LED-MODE :KEY :AUTO-REPEAT-MODE

CLX:CHANGE-KEYBOARD-CONTROL Function

Arguments

display

A CLX:DISPLAY object.

:KEY-CCLICK-PERCENT

An integer that represents the volume of a key click from 0 (off) to 100 (loudest).

:BELL-PERCENT

An integer that specifies the volume of a bell ring from 0 (off) to 100 (loudest), :DEFAULT, or NIL.

:BELL-PITCH

An integer that specifies the pitch of the bell in hertz, :DEFAULT, or NIL.

:BELL-DURATION

An integer that specifies the duration of a bell ring in milliseconds, :DEFAULT, or NIL.

:LED

An integer whose 1 bits specify that the corresponding LEDs (counting from 0) are affected by the :LED-MODE argument, or NIL.

:LED-MODE

Keyword :ON or :OFF that specifies the state of LEDs on the keyboard, or NIL.

:KEY

An integer that specifies that the corresponding key has auto-repeat mode enabled, or NIL.

:AUTO-REPEAT-MODE

Keyword :ON, :OFF, or :DEFAULT that specifies the global auto-repeat mode for the keyboard, or NIL.

Return Value

Unspecified.

CLX:CHANGE-KEYBOARD-MAPPING Function

Changes the mapping of keysyms to key codes.

Format

CLX:CHANGE-KEYBOARD-MAPPING *display keysyms*
 &KEY :START :END :FIRST-KEYCODE

CLX:CHANGE-KEYBOARD-MAPPING Function

Arguments

display

A CLX:DISPLAY object.

keysyms

An array.

:START :END

Two CLX:ARRAY-INDEX values that specify a subrange of keysyms. The default value for :START is 0.

:FIRST-KEYCODE

The first keycode to store. The default value is the value of the :START keyword.

Return Value

Unspecified.

CLX:CHANGE-POINTER-CONTROL Function

Defines how fast the pointer cursor moves compared to the default speed (:ACCELERATION) and how far it must move before acceleration is applied (:THRESHOLD).

If you supply a numeric value for :ACCELERATION, it must be positive. CLX rationalizes the value if necessary, and the ratio may be rounded arbitrarily by the X server.

Format

CLX:CHANGE-POINTER-CONTROL *display*
&KEY :ACCELERATION :THRESHOLD

Arguments

display

A CLX:DISPLAY object.

:ACCELERATION

A number, :DEFAULT, or NIL.

:THRESHOLD

An INTEGER, :DEFAULT, or NIL.

CLX:CHANGE-POINTER-CONTROL Function

Return Value

Unspecified.

CLX:CHANGE-PROPERTY Function

Alters the property list of the specified window. :START and :END affect sub-sequences extracted from *data*. :TRANSFORM is applied to each extracted element.

Format

CLX:CHANGE-PROPERTY *window* *property* *data* *type* *format*
 &KEY **:MODE** **:START** **:END** **:TRANSFORM**

Arguments

window

A CLX:WINDOW object.

property

The CLX:XATOM value that names the property to change.

data

A SEQUENCE containing the new property data.

type

A CLX:XATOM value.

format

The size of the data, either 8, 16, or 32.

:MODE

One of :REPLACE, :PREPEND, or :APPEND. The default value is :REPLACE.

:START

An integer. The default value is 0.

:END

An integer or NIL.

:TRANSFORM

A function that takes an integer argument, or NIL.

Return Value

Unspecified.

CLX:CHARACTER->KEYSYMS Function

CLX:CHARACTER->KEYSYMS Function

Given a character, returns a list of all matching keysyms. If a display is provided, translations specific to that display are used; if not, global translations are used.

Format

CLX:CHARACTER->KEYSYMS *character* &**OPTIONAL** *display*

Arguments

character

Any LISP object.

display

A CLX:DISPLAY object.

Return Value

A list.

CLX:CHAR-ASCENT Function

Returns the vertical distance from the baseline to the top of a character.

Format

CLX:CHAR-ASCENT *font index*

Arguments

font

A CLX:FONT object.

index

The character number.

Return Value

A FIXNUM or NIL if *index* is out of bounds for *font*.

CLX:CHAR-ATTRIBUTES Function

CLX:CHAR-ATTRIBUTES Function

Returns the attributes of a character.

Format

CLX:CHAR-ATTRIBUTES *font index*

Arguments

font

A CLX:FONT object.

index

The character number.

Return Value

A FIXNUM or NIL if *index* is out of bounds for *font*.

CLX:CHAR-DESCENT Function

Returns the vertical distance from the baseline to the bottom of a character.

Format

CLX:CHAR-DESCENT *font index*

Arguments

font

A CLX:FONT object.

index

The character number.

Return Value

A FIXNUM or NIL if *index* is out of bounds for *font*.

CLX:CHAR-LEFT-BEARING Function

CLX:CHAR-LEFT-BEARING Function

Returns the horizontal distance from the origin to the left edge of a character's bounding box.

Format

CLX:CHAR-LEFT-BEARING *font index*

Arguments

font

A CLX:FONT object.

index

The character number.

Return Value

A FIXNUM or NIL if *index* is out of bounds for *font*.

CLX:CHAR-RIGHT-BEARING Function

Returns the horizontal distance from the origin to the right edge of a character's bounding box.

Format

CLX:CHAR-RIGHT-BEARING *font index*

Arguments

font

A CLX:FONT object.

index

The character number.

Return Value

A FIXNUM or NIL if *index* is out of bounds for *font*.

CLX:CHAR-WIDTH Function

Returns the horizontal distance from the origin of the specified character to the origin of the next.

Format

CLX:CHAR-WIDTH *font index*

Arguments

font

A CLX:FONT object.

Index

The character number.

Return Value

A FIXNUM or NIL if *index* is out of bounds for *font*.

CLX:CIRCULATE-WINDOW-DOWN Function

Shuffles the specified window to the bottom of the stacking order so that none of its siblings is obscured by it.

Format

CLX:CIRCULATE-WINDOW-DOWN *window*

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:CIRCULATE-WINDOW-UP Function

CLX:CIRCULATE-WINDOW-UP Function

Brings the specified window up to the top of the stacking order so that it is not obscured by any of its siblings.

Format

CLX:CIRCULATE-WINDOW-UP *window*

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:CLEAR-AREA Function

Clears (fills with the background color or pixmap) a rectangular area of a window. Passing in a zero width or height is a no-op. A null width or height defaults to the width or height of the window.

Format

CLX:CLEAR-AREA *window* &**KEY :X :Y :WIDTH :HEIGHT :EXPOSURES-P**

Arguments

window

The CLX:WINDOW object that contains the area to be cleared.

:X :Y

Two integers that specify the X and Y coordinates of the top left corner of the area to be cleared. The default value for both keywords is 0, meaning the origin of the window.

:WIDTH :HEIGHT

Two integers that specify the width and height, in pixels, of the area to be cleared.

:EXPOSURES-P

A CLX:BOOLEAN value that specifies whether the server sends exposure events for the visible portions of *window* when they are cleared.

Return Value

Unspecified.

CLX:CLOSE-DISPLAY Function

Closes the connection between your client program and the X server that controls the specified display. CLX:CLOSE-DISPLAY destroys all client windows and frees all client resources.

Format

CLX:CLOSE-DISPLAY *display*

Argument

display

The object of type CLX:DISPLAY whose connection is to be closed.

Return Value

Unspecified.

CLX:CLOSE-DOWN-MODE Function

Specifies what happens to a client's resources when the client disconnects from the server. The default close-down mode is :DESTROY, which frees all client resources. The :RETAIN-TEMPORARY and :RETAIN-PERMANENT modes keep the client's resources after it disconnects, until another client invokes CLX:KILL-TEMPORARY-CLIENTS or CLX:KILL-CLIENTS, respectively.

This function can be used with SETF.

Format

CLX:CLOSE-DOWN-MODE *display*

Argument

display

A CLX:DISPLAY object.

CLX:CLOSE-DOWN-MODE Function

Return Value

:DESTROY, :RETAIN-PERMANENT, or :RETAIN-TEMPORARY, indicating the close-down mode of the specified display.

CLX:CLOSE-FONT Function

Unloads a font from LISP memory. It is an error to access a closed font.

Format

CLX:CLOSE-FONT *font*

Argument

font

A CLX:FONT object.

Return Value

Unspecified.

CLX:COLOR Structure

Holds the representation of a color. Note that CLX red, green, and blue values range from 0 to 1, indicating percentages, not from 0 to 65535 as in Xlib.

Constructor Function

CLX:MAKE-COLOR

Accessor Functions

Unless otherwise indicated, the following functions can be used with SETF:

Name	Type Definition
CLX:COLOR-BLUE	(NUMBER 0 1)
CLX:COLOR-GREEN	(NUMBER 0 1)
CLX:COLOR-RED	(NUMBER 0 1)
CLX:COLOR-RGB	Three CLX:RGB-VAL values. This function is not a valid SETF place.

Deallocator Function

Instances of type CLX:COLOR are subject to garbage collection.

CLX:COLOR-BLUE Function

Returns the blue value of the specified color.

Format

CLX:COLOR-BLUE *color*

Argument

color

A CLX:COLOR object.

Return Value

A CLX:RGB-VAL value.

CLX:COLOR-GREEN Function

Returns the green value of the specified color.

Format

CLX:COLOR-GREEN *color*

Argument

color

A CLX:COLOR object.

Return Value

A CLX:RGB-VAL value.

CLX:COLORMAP Structure

Colormaps are tables of color cells, each of which defines the color of a pixel in terms of its red, green, and blue components.

Constructor Function

CLX:MAKE-COLORMAP

CLX:COLORMAP Structure

Accessor Functions

Unless otherwise indicated, the following functions are not valid SETF places:

Name	Type Definition
CLX:COLORMAP-DISPLAY	CLX:DISPLAY
CLX:COLORMAP-ID	INTEGER

Predicates

CLX:COLORMAP-EQUAL, CLX:COLORMAP-P

Deallocator Function

CLX:FREE-COLORMAP

CLX:COLORMAP-DISPLAY Function

Returns the display on which the specified colormap was created.

Format

CLX:COLORMAP-DISPLAY *colormap*

Argument

colormap

A CLX:COLORMAP object.

Return Value

A CLX:DISPLAY object.

CLX:COLORMAP-EQUAL Function

Returns T if its two arguments are the same CLX:COLORMAP object.

Format

CLX:COLORMAP-EQUAL *object-1 object-2*

Arguments

object-1 object-2

Any two LISP objects.

CLX:COLORMAP-EQUAL Function

Return Value

A CLX:BOOLEAN value.

CLX:COLORMAP-ID Function

Returns the integer resource-id associated with the specified colormap.

Format

CLX:COLORMAP-ID *colormap*

Argument

colormap

A CLX:COLORMAP object.

Return Value

A CLX:RESOURCE-ID value.

CLX:COLORMAP-P Function

Returns T if its argument is a CLX:COLORMAP object.

Format

CLX:COLORMAP-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:COLOR-P Function

CLX:COLOR-P Function

Returns true if its argument is a CLX:COLOR object.

Format

CLX:COLOR-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:COLOR-RED Function

Returns the red value of the specified color.

Format

CLX:COLOR-RED *color*

Argument

color

A CLX:COLOR object.

Return Value

A CLX:RGB-VAL value.

CLX:COLOR-RGB Function

Returns the three values of the red, green, and blue components of the specified color.

CLX:COLOR-RGB Function

Format

CLX:COLOR-RGB *color*

Argument

color

A CLX:COLOR object.

Return Values

Three CLX:RGB-VAL values:

red
green
blue

CLX:CONVERT-SELECTION Function

Asks the owner of the specified selection (global property) to convert it to the specified data type.

Format

CLX:CONVERT-SELECTION *selection type requestor* &OPTIONAL *property time*

Arguments

selection type

Two CLX:XATOM values.

requestor

The CLX:WINDOW object that receives selection notification.

property

The CLX:XATOM value of the selection.

time

A CLX:TIMESTAMP value.

Return Value

Unspecified.

CLX:COPY-AREA Function

CLX:COPY-AREA Function

Copies a rectangular area of one drawable to another drawable.

Format

CLX:COPY-AREA *src gcontext src-x src-y width height dst dst-x dst-y*

Arguments

src

The CLX:DRAWABLE object that contains the area to copy.

gcontext

The CLX:GCCONTEXT object that contains the copy function to use (see the description of CLX:LOGICAL-OP values).

src-x src-y

Two integers that specify the X and Y coordinates of the origin of the area.

width height

Two integers that specify the dimensions, in pixels, of the area.

dst

The CLX:DRAWABLE object in which the copy is displayed.

dst-x dst-y

Two integers that specify the X and Y coordinates of the origin where the copied area is displayed.

Return Value

Unspecified.

CLX:COPY-COLORMAP-AND-FREE Function

Creates a new colormap for the same screen as the given colormap. All existing allocations are moved from the given colormap to the new colormap, and freed in the given colormap. The values of other entries in the new colormap are unspecified.

Format

CLX:COPY-COLORMAP-AND-FREE *colormap*

CLX:COPY-COLORMAP-AND-FREE Function

Argument

colormap

A CLX:COLORMAP object.

Return Value

The newly copied CLX:COLORMAP object.

CLX:COPY-GCONTEXT Function

Copies all components of the source GContext to the destination GContext.

Format

CLX:COPY-GCONTEXT *src dst*

Arguments

src dst

Two CLX:GCONTEXT objects.

Return Value

Unspecified.

CLX:COPY-GCONTEXT-COMPONENTS Function

Copies only specified components of the source GContext to the destination GContext.

Format

CLX:COPY-GCONTEXT-COMPONENTS *src dst &REST keys*

Arguments

src dst

Two CLX:GCONTEXT objects.

keys

A list of CLX:GCONTEXT-KEY values.

CLX:COPY-GCONTEXT-COMPONENTS Function

Return Value

Unspecified.

CLX:COPY-IMAGE Function

Copies an image with optional subimaging and format conversion.

Format

CLX:COPY-IMAGE *image &KEY :X :Y :WIDTH :HEIGHT :RESULT-TYPE*

Arguments

image

A CLX:IMAGE object to copy.

:X :Y

The X and Y coordinates of the subimage to copy. The default values are 0.

:WIDTH :HEIGHT

The dimensions of the subimage.

:RESULT-TYPE

A sequence type specifier for the return value. The default is the same as the original image.

Return Value

The newly copied IMAGE object.

CLX:COPY-PLANE Function

Copies a single plane from a specified area of a drawable, and uses it to modify an area of another drawable.

Format

CLX:COPY-PLANE *src gcontext plane src-x src-y width height dst dst-x dst-y*

Arguments

src

The CLX:DRAWABLE object that contains the area to modify.

CLX:COPY-PLANE Function

gcontext

The CLX:GCONTEXT object that contains the copy function to be used (see the description of CLX:LOGICAL-OP values).

plane

An integer that specifies which bit plane of the source area is copied.

src-x src-y

Two integers that specify the X and Y coordinates of the origin of the source area.

width height

Two integers that specify the dimensions, in pixels, of the source area.

dst

The CLX:DRAWABLE object in which the modified area is displayed.

dst-x dst-y

Two integers that specify the X and Y coordinates of the origin where the modified area is displayed.

Return Value

Unspecified.

CLX:CREATE-COLORMAP Function

Creates a CLX:COLORMAP object for the screen that is associated with the *window* argument.

Format

CLX:CREATE-COLORMAP *visual* *window* &OPTIONAL *alloc-p*

Arguments

visual

An INTEGER resource-id.

window

A CLX:WINDOW object.

alloc-p

A CLX:BOOLEAN value that specifies whether the colormap is allocated writable (clients can allocate cells for exclusive use) or shareable.

CLX:CREATE-COLORMAP Function

Return Value

The newly created CLX:COLORMAP object.

CLX:CREATE-CURSOR Function

Creates a CLX:CURSOR object from an image drawn in a pixmap.

Format

**CLX:CREATE-CURSOR &KEY :SOURCE :MASK :X :Y :FOREGROUND
:BACKGROUND**

Arguments

:SOURCE

The CLX:PIXMAP object that contains the cursor image.

:MASK

A CLX:PIXMAP object that specifies which bits of the source pixmap to use (the cursor shape).

:X :Y

Two integers specifying the X and Y coordinates of the cursor's hot spot.

:FOREGROUND :BACKGROUND

Two CLX:COLOR objects that specify the foreground and background of the cursor.

Return Value

The newly created CLX:CURSOR object.

CLX:CREATE-GCONTEXT Function

Returns an instance of the CLX:GCONTEXT structure containing the specified graphics characteristics. For all components, a value of NIL causes the default GContext value to be used.

NOTE

Using a CLX:STRINGABLE value for :FONT causes an implicit call to CLX:OPEN-FONT.

If :CACHE-P is true, then GContext state is cached locally, and changing a GContext component has no effect unless the new value differs from the cached value. Component changes (SETFS and CLX:WITH-GCONTEXT) are always deferred regardless of the cache mode, and are sent over the protocol only when required by a local operation or by an explicit call to CLX:FORCE-GCONTEXT-CHANGES.

CLX:CREATE-GCONTEXT Function

Format

CLX:CREATE-GCONTEXT
&KEY :DRAWABLE :FUNCTION :PLANE-MASK :FOREGROUND
:BACKGROUND :LINE-WIDTH :LINE-STYLE :CAP-STYLE
:JOIN-STYLE :FILL-STYLE :FILL-RULE :ARC-MODE :TILE
:STIPPLE :TS-X :TS-Y :FONT :SUBWINDOW-MODE
:EXPOSURES :CLIP-X :CLIP-Y :CLIP-MASK :CLIP-ORDERING
:DASH-OFFSET :DASHES :CACHE-P

Arguments

:DRAWABLE

A CLX:DRAWABLE value.

:FUNCTION

A CLX:LOGICAL-OP value or NIL.

:PLANE-MASK :BACKGROUND :LINE-WIDTH

Integers or NIL.

:LINE-STYLE

NIL or one of :SOLID, :DASH, or :DOUBLE-DASH.

:CAP-STYLE

NIL or one of :NOT-LAST, :BUTT, :ROUND, or :PROJECTING.

:JOIN-STYLE

NIL or one of :MITER, :ROUND, or :BEVEL.

:FILL-STYLE

NIL or one of :SOLID, :TILED, :OPAQUE-STIPPLED, or :STIPPLED.

:FILL-RULE

NIL or either :EVEN-ODD or :WINDING.

:ARC-MODE

NIL or either :CHORD or :PIE-SLICE.

:TILE :STIPPLE

Two CLX:PIXMAP objects.

:TS-X :TS-Y

Integers.

:FONT

A CLX:FONT object.

:SUBWINDOW-MODE

NIL or either :CLIP-BY-CHILDREN or :INCLUDE-INFERIORS.

:EXPOSURES

NIL or either :ON or :OFF.

CLX:CREATE-GCONTEXT Function

:CLIP-X CLIP-Y

Integers.

:CLIP-MASK

NIL or :NONE, or either a CLX:PIXMAP or a CLX:RECT-SEQ.

:CLIP-ORDERING

NIL or one of :UNSORTED, :Y-SORTED, :YX-SORTED, or :YX-BANDED.

:DASHES

An integer or a sequence of integers.

:DASH-OFFSET

An integer.

:CACHE-P

A CLX:BOOLEAN value. The default is T.

Return Value

The newly created CLX:GCONTEXT object.

CLX:CREATE-GLYPH-CURSOR Function

Creates a CLX:CURSOR object from a character (glyph) in the specified font. The :SOURCE-FONT, :SOURCE-CHAR, :FOREGROUND, and :BACKGROUND arguments are required.

Format

```
CLX:CREATE-GLYPH-CURSOR &KEY :SOURCE-FONT :SOURCE-CHAR  
                      :MASK-FONT :MASK-CHAR  
                      :FOREGROUND :BACKGROUND
```

Arguments

:SOURCE-FONT

The CLX:FONT object that includes the glyph used to create the cursor.

:SOURCE-CHAR

An integer that identifies the glyph in the source font.

:MASK-FONT

A CLX:FONT object that contains the glyph to be used as a mask for the cursor shape. The default value is NIL, which specifies the source font.

:MASK-CHAR

An integer that identifies the glyph used as a mask to determine which pixels of the source character are displayed. The default value is NIL, which specifies that all bits of the source character are used.

CLX:CREATE-GLYPH-CURSOR Function

:FOREGROUND :BACKGROUND

Two CLX-COLOR objects that specify the foreground and background of the cursor to be created.

Return Value

The newly created CLX:CURSOR object.

CLX:CREATE-IMAGE Function

Returns an IMAGE-X, IMAGE-XY, or IMAGE-Z structure, depending on the type of the :DATA parameter.

Format

```
CLX:CREATE-IMAGE &KEY :WIDTH :HEIGHT :DEPTH :DATA :RED-MASK  
:GREEN-MASK :BLUE-MASK :BITS-PER-PIXEL  
:FORMAT :SCANLINE-PAD :BYTES-PER-LINE  
:BYTE-LSB-FIRST-P :BIT-LSB-FIRST-P
```

Arguments

:WIDTH :HEIGHT

Integers that specify the width and height of the image, measured in pixels. These arguments are required.

:DEPTH

An integer that specifies the depth of the image. The default is 1.

:DATA

A sequence in which to store the data. Its type depends on the value of the :FORMAT argument:

```
:FORMAT :BITMAP      (ARRAY CLX:CARD8 (*))  
:FORMAT :XY-PIXMAP   (LIST CLX:BITMAP)  
:FORMAT :Z-PIXMAP    CLX:PIXARRAY
```

:RED-MASK :GREEN-MASK :BLUE-MASK

Three CLX:PIXEL values.

:BITS-PER-PIXEL

(MEMBER 1 4 8 16 24 32)

:FORMAT

(MEMBER :BITMAP :XY-PIXMAP :Z-PIXMAP)

:SCANLINE-PAD

(MEMBER 8 16 32)

CLX:CREATE-IMAGE Function

:BYTES-PER-LINE

Specifies the number of bytes in the client image between the start of one scanline and the start of the next.

:BYTE-LSB-FIRST-P :BIT-LSB-FIRST-P

Two CLX:BOOLEAN values that specify byte and bit ordering.

Return Value

An IMAGE-X, IMAGE-XY, or IMAGE-Z structure, depending on the type of the :DATA parameter.

CLX:CREATE-PIXMAP Function

Creates a CLX:PIXMAP object with the specified characteristics.

Format

CLX:CREATE-PIXMAP &KEY :WIDTH :HEIGHT :DEPTH :DRAWABLE

Arguments

:WIDTH :HEIGHT :DEPTH

Integers that specify the dimensions of the pixmap. These arguments are required.

:DRAWABLE

A CLX:DRAWABLE object that provides display information. This argument is required.

Return Value

The newly created CLX:PIXMAP object.

CLX:CREATE-WINDOW Function

Creates a CLX:WINDOW object with the specified attributes. The appropriate display is obtained from the *parent* argument.

Format

**CLX:CREATE-WINDOW &KEY :PARENT :X :Y :WIDTH :HEIGHT :DEPTH
:BORDER-WIDTH :CLASS :VISUAL
:BACKGROUND :BORDER :GRAVITY
:BIT-GRAVITY :BACKING-STORE**

CLX:CREATE-WINDOW Function

**:BACKING-PLANES :BACKING-PIXEL
:SAVE-UNDER :EVENT-MASK
:DO-NOT-PROPAGATE-MASK
:OVERRIDE-REDIRECT :COLORMAP
:CURSOR**

Arguments

:PARENT

A CLX:WINDOW that specifies the parent of the window returned by this function.

:X :Y

Integers that specify the X and Y coordinates of the top-left corner, measured in pixels from the top-left corner of the parent.

:WIDTH :HEIGHT

Integers that specify the width and height, measured in pixels, not including the border.

:DEPTH

An integer that specifies the number of bits per pixel. The default value is 0.

:BORDER-WIDTH

An integer that specifies the size of the border, measured in pixels. The default value is 0.

:CLASS

One of :COPY, :INPUT-OUTPUT, or :INPUT-ONLY, specifying whether the window can display output. The default value is :COPY, which means the same class as the parent window.

:VISUAL

Either a CLX:VISUAL value or :COPY. The default value is :COPY, specifying the same visual information as the parent window.

:BACKGROUND

A CLX:PIXEL value, a CLX:PIXMAP object, :NONE, :PARENT-RELATIVE, or NIL, specifying the background.

:BORDER

A CLX:PIXEL value, CLX:PIXMAP object, or :COPY, specifying the border.

:GRAVITY

A CLX:WIN-GRAVITY value that specifies how the window is positioned after its parent is resized.

:BIT-GRAVITY

A CLX:BIT-GRAVITY value that specifies which region of the window is retained after it is resized.

:BACKING-STORE

:NOT-USEFUL, :WHEN-MAPPED, :ALWAYS, or NIL, specifying when the server maintains the contents of the window.

CLX:CREATE-WINDOW Function

:BACKING-PLANES

A CLX:PIXEL or NIL, specifying which bit planes of the window hold dynamic data that must be preserved in :BACKING-STORE.

:BACKING-PIXEL

A CLX:PIXEL value or NIL, specifying the value to use in planes not covered by :BACKING-PLANES.

:SAVE-UNDER

Either :ON or :OFF, specifying whether the server saves the contents of windows obscured by the created window.

:EVENT-MASK

A CLX:EVENT-MASK or NIL, specifying which events interest the client in this window.

:DO-NOT-PROPAGATE-MASK

A CLX:DEVICE-EVENT-MASK or NIL, specifying which events should not be propagated to ancestor windows.

: OVERRIDE-REDIRECT

Either :ON or :OFF, specifying whether calls to map and configure the window should override a request from another client to redirect those calls.

:COLORMAP

A CLX:COLORMAP, :COPY, or NIL, specifying the colormap that best reflects the true colors of the window.

:CURSOR

A CLX:CURSOR object, :NONE, or NIL, specifying the cursor shape to be used when the pointer is in the window.

Return Value

The newly created CLX:WINDOW object.

CLX:CURSOR-DISPLAY Function

Returns the CLX:DISPLAY object for which the specified cursor was created.

Format

CLX:CURSOR-DISPLAY *cursor*

Argument

cursor

A CLX:CURSOR object.

CLX:CURSOR-DISPLAY Function

Return Value

A CLX:DISPLAY object.

CLX:CURSOR-EQUAL Function

Returns true if its two arguments are the same CLX:CURSOR object.

Format

CLX:CURSOR-EQUAL *object-1 object-2*

Arguments

object-1 object-2

Any two LISP objects.

Return Value

A CLX:BOOLEAN value.

CLX:CURSOR-ID Function

Returns the integer resource-id of the specified cursor.

Format

CLX:CURSOR-ID *cursor*

Argument

cursor

A CLX:CURSOR object.

Return Value

A CLX:RESOURCE-ID value.

CLX:CURSOR-P Function

CLX:CURSOR-P Function

This predicate returns true if its argument is of type CLX:CURSOR.

Format

CLX:CURSOR-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:DELETE-PROPERTY Function

Removes the specified property from the specified window's property list.

Format

CLX:DELETE-PROPERTY *window property*

Arguments

window

A CLX:WINDOW object.

property

A CLX:XATOM value.

Return Value

Unspecified.

CLX:DESTROY-SUBWINDOWS Function

CLX:DESTROY-SUBWINDOWS Function

Calls CLX:DESTROY-WINDOW on all the children of the specified window, in bottom-to-top stacking order.

Format

CLX:DESTROY-SUBWINDOWS *window*

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:DESTROY-WINDOW Function

If the specified window is mapped, a call to CLX:UNMAP-WINDOW is performed automatically. The window and all its children are then destroyed, in order from the specified window downward. The order among siblings at each level is undefined. A :DESTROY-NOTIFY event is generated for each window.

Normal exposure processing is performed on all windows formerly obscured by the specified window or its children.

If the *window* argument is a root window, this function has no effect.

Format

CLX:DESTROY-WINDOW *window*

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:DEVICE-EVENT-MASK Type Specifier

CLX:DEVICE-EVENT-MASK Type Specifier

Objects of type CLX:DEVICE-EVENT-MASK are returned by the CLX:MAKE-EVENT-MASK function and may be passed to the CLX:CREATE-WINDOW function as the value of the :DO-NOT-PROPAGATE-MASK argument.

Representation

(OR CLX:MASK32 (LIST CLX:DEVICE-EVENT-MASK-CLASS))

CLX:DEVICE-EVENT-MASK-CLASS Type Specifier

Abstract type used to define CLX:DEVICE-EVENT-MASK values.

Representation

(MEMBER :KEY-PRESS :KEY-RELEASE :BUTTON-PRESS :BUTTON-RELEASE
:POINTER-MOTION :BUTTON-1-MOTION :BUTTON-2-MOTION
:BUTTON-3-MOTION :BUTTON-4-MOTION :BUTTON-5-MOTION
:BUTTON-MOTION)

CLX:DISCARD-CURRENT-EVENT Function

Deletes the event at the head of the event queue.

Format

CLX:DISCARD-CURRENT-EVENT *display*

Argument

display

A CLX:DISPLAY object.

Return Value

T if an event was deleted; NIL if the queue was empty.

CLX:DISCARD-FONT-INFO Function

Discards any state that can be re-obtained by functions that generate a QueryFont protocol request, such as CLX:CHAR-ASCENT, CLX:FONT-MAX-WIDTH, and so on. This is simply a performance hint for memory-limited systems.

Format

CLX:DISCARD-FONT-INFO *font*

Argument

font

A CLX:FONT object.

Return Value

Unspecified.

CLX:DISPLAY Structure

This structure contains information about the display itself and the X server controlling the display.

Constructor Function

An instance of the CLX:DISPLAY structure is returned by the CLX:OPEN-DISPLAY function.

Accessor Functions

Unless otherwise indicated, these functions are not valid SETF places.

Name	Type Definition
CLX:DISPLAY-BITMAP-FORMAT	CLX:BITMAP-FORMAT
CLX:DISPLAY-DEFAULT-SCREEN	CLX:SCREEN
CLX:DISPLAY-IMAGE-LSB-FIRST-P	CLX:BOOLEAN
CLX:DISPLAY-MAX-KEYCODE	FIXNUM
CLX:DISPLAY-MAX-REQUEST-LENGTH	FIXNUM
CLX:DISPLAY-MIN-KEYCODE	FIXNUM
CLX:DISPLAY-MOTION-BUFFER-SIZE	FIXNUM
CLX:DISPLAY-PIXMAP-FORMATS	LIST of CLX:PIXMAP-FORMAT
CLX:DISPLAY-PROTOCOL-MAJOR-VERSION	FIXNUM
CLX:DISPLAY-PROTOCOL-MINOR-VERSION	FIXNUM

CLX:DISPLAY Structure

Name	Type Definition
CLX:DISPLAY-RELEASE-NUMBER	FIXNUM
CLX:DISPLAY-ROOTS	LIST of CLX:SCREEN
CLX:DISPLAY-VENDOR-NAME	STRING

Deallocator Function

CLX:CLOSE-DISPLAY

CLX:DISPLAY-AFTER-FUNCTION Function

Specifies a function to be called after every protocol request is generated, even those inside an explicit CLX:WITH-DISPLAY macro. The function is called inside the effective CLX:WITH-DISPLAY for the associated request. Can be set, for example, to #'CLX:DISPLAY-FORCE-OUTPUT or #'CLX:DISPLAY-FINISH-OUTPUT.

The default value is NIL, meaning that no after function is defined. CLX:DISPLAY-AFTER-FUNCTION is never called from inside the after function itself.

Format

CLX:DISPLAY-AFTER-FUNCTION *display*

Argument

display

A CLX:DISPLAY object.

Return Value

NIL, or a function that takes a CLX:DISPLAY object.

CLX:DISPLAY-BITMAP-FORMAT Function

Returns the default bitmap-format on the specified display.

Format

CLX:DISPLAY-BITMAP-FORMAT *display*

CLX:DISPLAY-BITMAP-FORMAT Function

Argument

display

A CLX:DISPLAY object.

Return Value

A CLX:BITMAP-FORMAT object.

CLX:DISPLAY-DEFAULT-SCREEN Function

Returns the default screen on the specified display.

Format

CLX:DISPLAY-DEFAULT-SCREEN *display*

Argument

display

A CLX:DISPLAY object.

Return Value

A CLX:SCREEN object.

CLX:DISPLAY-FINISH-OUTPUT Function

Forces all requested output to be displayed, then causes a round-trip request between server and client to ensure that all possible errors and events have been received. DISPLAY-AFTER-FUNCTION can be set to this function while debugging so that error messages are displayed immediately.

Format

CLX:DISPLAY-FINISH-OUTPUT *display*

Argument

display

A CLX:DISPLAY object.

CLX:DISPLAY-FINISH-OUTPUT Function

Return Value

Unspecified.

CLX:DISPLAY-FORCE-OUTPUT Function

Flushes the output buffer used by CLX to store requests. All requests in the output buffer that have not yet been sent are transmitted to the X server.

Format

CLX:DISPLAY-FORCE-OUTPUT *display*

Argument

display

A CLX:DISPLAY object.

Return Value

Unspecified.

CLX:DISPLAY-IMAGE-LSB-FIRST-P Function

Indicates whether image data is stored with the least significant bit first.

Format

CLX:DISPLAY-IMAGE-LSB-FIRST-P *display*

Argument

display

A CLX:DISPLAY object.

Return Value

T if the byte order for image data is least significant bit first; NIL if it is most significant bit first.

CLX:DISPLAY-MAX-KEYCODE Function

CLX:DISPLAY-MAX-KEYCODE Function

Indicates the largest keycode value transmitted by the X server. For more information, see the *X Window System: C Library and Protocol Reference*.

Format

CLX:DISPLAY-MAX-KEYCODE *display*

Argument

display

A CLX:DISPLAY object.

Return Value

An integer.

CLX:DISPLAY-MAX-REQUEST-LENGTH Function

Returns the maximum length of a request accepted by the X server, in 4-byte units. For more information, see the *X Window System: C Library and Protocol Reference*.

Format

CLX:DISPLAY-MAX-REQUEST-LENGTH *display*

Argument

display

A CLX:DISPLAY object.

Return Value

An integer.

CLX:DISPLAY-MIN-KEYCODE Function

CLX:DISPLAY-MIN-KEYCODE Function

Indicates the smallest keycode value transmitted by the X server. For more information, see the *X Window System: C Library and Protocol Reference*.

Format

CLX:DISPLAY-MIN-KEYCODE *display*

Argument

display

A CLX:DISPLAY object.

Return Value

An integer.

CLX:DISPLAY-MOTION-BUFFER-SIZE Function

The X server may maintain the recent history of pointer motion. This function indicates the approximate size of the pointer motion buffer.

Format

CLX:DISPLAY-MOTION-BUFFER-SIZE *display*

Argument

display

A CLX:DISPLAY object.

Return Value

An integer.

CLX:DISPLAY-P Function

Returns T if its argument is a CLX:DISPLAY object.

Format

CLX:DISPLAY-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:DISPLAY-PIXMAP-FORMATS Function

Lists the pixmap format of each depth supported by the display.

Format

CLX:DISPLAY-PIXMAP-FORMATS *display*

Argument

display

A CLX:DISPLAY object.

Return Value

A list of CLX:PIXMAP-FORMAT objects.

CLX:DISPLAY-PROTOCOL-MAJOR-VERSION Function

Returns the version number of the X protocol.

CLX:DISPLAY-PROTOCOL-MAJOR-VERSION Function

Format

CLX:DISPLAY-PROTOCOL-MAJOR-VERSION *display*

Argument

display

A CLX:DISPLAY object.

Return Value

An integer.

CLX:DISPLAY-PROTOCOL-MINOR-VERSION Function

Returns the release number of the X protocol.

Format

CLX:DISPLAY-PROTOCOL-MINOR-VERSION *display*

Argument

display

A CLX:DISPLAY object.

Return Value

An integer.

CLX:DISPLAY-RELEASE-NUMBER Function

Returns the release number of the X server. This value depends on the implementation.

Format

CLX:DISPLAY-RELEASE-NUMBER *display*

CLX:DISPLAY-RELEASE-NUMBER Function

Argument

display

A CLX:DISPLAY object.

Return Value

An integer.

CLX:DISPLAY-ROOTS Function

Returns a list of CLX:SCREEN objects, one for each screen attached to the specified display.

Format

CLX:DISPLAY-ROOTS *display*

Argument

display

A CLX:DISPLAY object.

Return Value

A list of CLX:SCREEN objects.

CLX:DISPLAY-VENDOR-NAME Function

Returns the name of the implementor of the X server.

Format

CLX:DISPLAY-VENDOR-NAME *display*

Argument

display

A CLX:DISPLAY object.

CLX:DISPLAY-VENDOR-NAME Function

Return Value

A string.

CLX:DRAWABLE Type Specifier

A CLX:DRAWABLE is either a window or a pixmap. Objects of type CLX:DRAWABLE can be passed to any of the CLX:DRAW- and CLX:DRAWABLE- functions.

Representation

(OR CLX:WINDOW CLX:PIXMAP)

CLX:DRAWABLE-BORDER-WIDTH Function

Returns the width of the border of the specified drawable, in pixels. This function can be used with SETF.

Format

CLX:DRAWABLE-BORDER-WIDTH *drawable*

Argument

drawable

A CLX:DRAWABLE object.

Return Value

An integer.

CLX:DRAWABLE-DEPTH Function

Returns the number of bits per pixel in the specified drawable. This function is not a valid SETF place.

Format

CLX:DRAWABLE-DEPTH *drawable*

CLX:DRAWABLE-DEPTH Function

Argument

drawable

A CLX:DRAWABLE object.

Return Value

An integer.

CLX:DRAWABLE-DISPLAY Function

Returns the display on which the specified drawable appears. This function is not a valid SETF place.

Format

CLX:DRAWABLE-DISPLAY *drawable*

Argument

drawable

A CLX:DRAWABLE object.

Return Value

A CLX:DISPLAY object.

CLX:DRAWABLE-EQUAL Function

Returns T if its two arguments are the same CLX:DRAWABLE object.

Format

CLX:DRAWABLE-EQUAL *object-1 object-2*

Arguments

object-1 object-2

Any two LISP objects.

CLX:DRAWABLE-EQUAL Function

Return Value

A CLX:BOOLEAN value.

CLX:DRAWABLE-HEIGHT Function

Returns the height of the specified drawable, in pixels. For windows, this is the inside height, excluding any border. This function can be used with SETF.

Format

CLX:DRAWABLE-HEIGHT *drawable*

Argument

drawable

A CLX:DRAWABLE object.

Return Value

An integer.

CLX:DRAWABLE-ID Function

Returns the integer resource-id of the specified drawable. This function is not a valid SETF place.

Format

CLX:DRAWABLE-ID *drawable*

Argument

drawable

A CLX:DRAWABLE object.

Return Value

A CLX:RESOURCE-ID value.

CLX:DRAWABLE-P Function

Returns T if its argument is a CLX:DRAWABLE object.

Format

CLX:DRAWABLE-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:DRAWABLE-ROOT Function

Returns the root window of the specified drawable. This function is not a valid SETF place.

Format

CLX:DRAWABLE-ROOT *drawable*

Argument

drawable

A CLX:DRAWABLE object.

Return Value

A CLX:WINDOW object.

CLX:DRAWABLE-WIDTH Function

CLX:DRAWABLE-WIDTH Function

Returns the width, in pixels, of the specified drawable. For windows, this is the inside width, excluding any border. This function can be used with SETF.

Format

CLX:DRAWABLE-WIDTH *drawable*

Argument

drawable

A CLX:DRAWABLE object.

Return Value

An integer.

CLX:DRAWABLE-X Function

Returns the X coordinate of the top-left corner of the specified drawable, measured in pixels from the top-left corner of its parent. This function can be used with SETF.

Format

CLX:DRAWABLE-X *drawable*

Argument

drawable

A CLX:DRAWABLE object.

Return Value

An integer.

CLX:DRAWABLE-Y Function

Returns the Y coordinate of the top-left corner of the specified drawable, measured in pixels from the top-left corner of its parent. This function can be used with SETF.

Format

CLX:DRAWABLE-Y *drawable*

Argument

drawable

A CLX:DRAWABLE object.

Return Value

An integer.

CLX:DRAW-ARC Function

Draws the arc defined by a portion of the ellipse inscribed within a specified rectangle. The angle that determines the starting position of the arc is specified relative to the 3 o'clock position of the bounding rectangle; the angle that determines the extent of the arc is specified relative to the starting position of the arc. By default, the angles are measured in radians.

Format

CLX:DRAW-ARC *drawable gcontext x y width height angle1 angle2*
&OPTIONAL *fill-p degrees-p*

Arguments

drawable

The CLX:DRAWABLE object in which the arc is drawn.

gcontext

The CLX:GCONTEXT object that specifies the graphics characteristics (such as arc-mode) of the arc.

x y

Two integers that specify the X and Y coordinates of the top-left corner of the arc's bounding rectangle.

CLX:DRAW-ARC Function

width height

Two integers that specify the dimensions, in pixels, of the arc's bounding rectangle.

angle1 angle2

Two CLX:ANGLE values that specify the starting point and extent of the arc, respectively.

fill-p

A CLX:BOOLEAN value that specifies whether the arc is filled.

degrees-p

A CLX:BOOLEAN value that specifies whether the angles are measured in degrees or radians.

Return Value

Unspecified.

CLX:DRAW-ARCS Function

Draws multiple arcs in the specified drawable.

Format

CLX:DRAW-ARCS *drawable gcontext arcs &OPTIONAL fill-p degrees-p*

Arguments

drawable

The CLX:DRAWABLE object in which the arcs are drawn.

gcontext

The CLX:GCONTEXT object that specifies the graphics characteristics (such as arc-mode) of the arcs.

arcs

A CLX:ARC-SEQ value that specifies the origins and dimensions of the arcs' bounding rectangles.

fill-p

A CLX:BOOLEAN value that specifies whether the arcs are filled.

degrees-p

A CLX:BOOLEAN value that specifies whether the angles are measured in degrees.

Return Value

Unspecified.

CLX:DRAW-DIRECTION Type Specifier

Specifies whether a font should be drawn from left to right (like English) or right to left (like Hebrew). Objects of type CLX:DRAW-DIRECTION are returned by the CLX:FONT-DIRECTION function.

Representation

(MEMBER :LEFT-TO-RIGHT :RIGHT-TO-LEFT)

CLX:DRAW-GLYPH Function

Draws a single character in the specified drawable. Only foreground pixels are drawn; the background is not affected.

Format

CLX:DRAW-GLYPH *drawable gcontext x y elt &KEY :TRANSLATE :WIDTH :SIZE*

Arguments

drawable

The CLX:DRAWABLE in which the glyphs are drawn.

gcontext

The CLX:GCONTEXT that determines the graphics characteristics of the text, such as font and color.

x y

Two integers that specify the X and Y coordinates of the origin of the text.

elt

An element, such as a character, to draw.

:TRANSLATE

A function that translates *elt* into a font index. The default is #'CLX:TRANSLATE-DEFAULT.

:WIDTH

An INTEGER that specifies the width of the text in pixels, or NIL.

CLX:DRAW-GLYPH Function

:SIZE

A CLX:INDEX-SIZE value that specifies the size of the font. The default value is :DEFAULT, which means an 8-bit font.

Return Values

Two values:

- T if *elt* is output but NIL if the :TRANSLATE function refuses to output it.
- An integer that indicates the width of the drawn text if known.

CLX:DRAW-GLYPHS Function

Draws multiple glyphs in the specified drawable. Only foreground pixels are drawn; the background is not affected.

Format

CLX:DRAW-GLYPHS *drawable gcontext x y sequence*
 &KEY :START :END :TRANSLATE :WIDTH :SIZE

Arguments

drawable

The CLX:DRAWABLE in which the glyphs are drawn.

gcontext

The CLX:GCONTEXT that determines the graphics characteristics of the text, such as font and color.

x y

Two integers that specify the X and Y coordinates of the origin of the text.

sequence

A sequence of items to draw; for example, a string.

:START

The CLX:ARRAY-INDEX value that specifies the starting position within *sequence*. The default value is 0.

:END

The CLX:ARRAY-INDEX that specifies the last position within *sequence*. If this is null, the length of *sequence* is used.

:TRANSLATE

A function that translates *sequence* into font indexes. The default is #'CLX:TRANSLATE-DEFAULT.

:WIDTH

An integer or NIL, specifying the width of the glyphs in pixels if known.

CLX:DRAW-GLYPHS Function

:SIZE

A CLX:INDEX-SIZE value that specifies the size of the font. The default is :DEFAULT, meaning an 8-bit font.

Return Values

Two values:

- An ARRAY-INDEX value that indicates the new start if :END was not reached; or NIL if it was.
- An integer that indicates the overall width of the glyphs if known; or NIL if not.

CLX:DRAW-IMAGE-GLYPH Function

Draws a single character in the specified drawable. Both foreground and background pixels are drawn. An initial font change from the :TRANSLATE function is allowed.

Format

CLX:DRAW-IMAGE-GLYPH *drawable gcontext x y elt*
 &KEY :TRANSLATE :WIDTH :SIZE

Arguments

drawable

The CLX:DRAWABLE in which the glyph is drawn.

gcontext

The CLX:GCONTEXT that determines the graphics characteristics of the text, such as font and color.

x y

Two integers that specify the X and Y coordinates of the origin of the text.

elt

An element, such as a character, to draw.

:TRANSLATE

A function that translates *elt* into a font index. The default is #'CLX:TRANSLATE-DEFAULT.

:WIDTH

An integer that specifies the width of the text in pixels, or NIL.

:SIZE

A CLX:INDEX-SIZE value that specifies the size of the font. The default value is :DEFAULT, which means an 8-bit font.

CLX:DRAW-IMAGE-GLYPH Function

Return Values

Two values:

- `T` if `elt` is output but `NIL` if the `:TRANSLATE` function refuses to output it.
- An integer that indicates the overall width of the drawn text if known.

CLX:DRAW-IMAGE-GLYPHS Function

Draws multiple glyphs in the specified drawable. Both foreground and background pixels are drawn. An initial font change is allowed from the `:TRANSLATE` function, but any subsequent font change or horizontal motion causes termination.

Format

CLX:DRAW-IMAGE-GLYPHS *drawable gcontext x y sequence*
 &KEY :START :END :TRANSLATE :WIDTH :SIZE

Arguments

drawable

The CLX:DRAWABLE in which the glyphs are drawn.

gcontext

The CLX:GCCONTEXT that determines the graphics characteristics of the text, such as font and color.

x y

Two integers that specify the X and Y coordinates of the origin of the text.

sequence

A sequence of items to draw; for example, a string.

:START

The CLX:ARRAY-INDEX value that specifies the starting position within *sequence*. The default value is 0.

:END

The CLX:ARRAY-INDEX value that specifies the last position within *sequence*. If this is null, the length of *sequence* is used.

:TRANSLATE

A function that translates *sequence* into font indexes. The default is #'CLX:TRANSLATE-DEFAULT.

:WIDTH

An integer or `NIL`, specifying the width of the glyphs in pixels if known.

CLX:DRAW-IMAGE-GLYPHS Function

:SIZE

A CLX:INDEX-SIZE value that specifies the size of the font. The default is :DEFAULT, meaning an 8-bit font.

Return Values

Two values:

- An ARRAY-INDEX value that indicates the new start if :END was not reached; or NIL if it was.
- An integer that indicates the overall width of the glyphs if known; or NIL if not.

CLX:DRAW-LINE Function

Draws a straight line between two specified points.

Format

CLX:DRAW-LINE *drawable gcontext x1 y1 x2 y2 &OPTIONAL relative-p*

Arguments

drawable

The CLX:DRAWABLE object in which the line is drawn.

gcontext

The CLX:GCONTEXT object that specifies the graphic characteristics of the line, such as line style, line width, and so on.

x1 y1

Two integers that specify the X and Y coordinates of the first endpoint of the line.

x2 y2

Two integers that specify the X and Y coordinates of the second endpoint of the line.

relative-p

A CLX:BOOLEAN value that specifies whether the coordinates of the second endpoint are relative to the first endpoint.

Return Value

Unspecified.

CLX:DRAW-LINES Function

CLX:DRAW-LINES Function

Draws straight lines connecting each point in a sequence or CLX:POINT-SEQ value.

Format

CLX:DRAW-LINES *drawable gcontext points*
 &KEY :RELATIVE-P :FILL-P :SHAPE

Arguments

drawable

The CLX:DRAWABLE object in which the lines are drawn.

gcontext

The CLX:GCONTEXT object that specifies the graphic characteristics of the lines, such as line style, line width, and so on.

points

A sequence or CLX:POINT-SEQ value that contains the X and Y coordinates of the endpoints of the lines.

:RELATIVE-P

A CLX:BOOLEAN value that specifies whether the coordinates of the second and following points are relative to the point that precedes them in the point sequence.

:FILL-P

A CLX:BOOLEAN value that specifies whether the region enclosed by the lines is filled.

:SHAPE

A keyword used by the server to optimize fill operations. Possible values are COMPLEX, :NON-CONVEX, and :CONVEX. The default value is :COMPLEX.

Return Value

Unspecified.

CLX:DRAW-POINT Function

Draws a single point at the specified coordinates. The coordinates are relative to the origin of the drawable.

CLX:DRAW-POINT Function

Format

CLX:DRAW-POINT *drawable gcontext x y*

Arguments

drawable

The CLX:DRAWABLE object in which the point is drawn.

gcontext

The CLX:GCCONTEXT object that specifies the graphics characteristics of the point, such as color.

x y

Two integers that specify the X and Y coordinates of the point.

Return Value

Unspecified.

CLX:DRAW-POINTS Function

Draws multiple points in the specified drawable. The X and Y coordinates of each point may be given in a single CLX:POINT-SEQ value. The coordinates of the first point in the sequence are relative to the origin of the drawable, but the coordinates of the second and following points may be relative to the point that precedes them in the sequence.

Format

CLX:DRAW-POINTS *drawable gcontext points &OPTIONAL relative-p*

Arguments

drawable

The CLX:DRAWABLE object in which the points are drawn.

gcontext

The CLX:GCCONTEXT object that specifies the graphics characteristics of the points (such as color).

points

A CLX:POINT-SEQ value that specifies the coordinates of the points.

relative-p

A CLX:BOOLEAN value that specifies whether the coordinates of each point are relative to the previous point.

CLX:DRAW-POINTS Function

Return Value

Unspecified.

CLX:DRAW-RECTANGLE Function

Draws a solid or outlined rectangle.

Format

CLX:DRAW-RECTANGLE *drawable gcontext x y width height &OPTIONAL fill-p*

Arguments

drawable

The CLX:DRAWABLE object in which the rectangle is drawn.

gcontext

The CLX:GCONTEXT object that specifies the graphics characteristics of the rectangle, such as color, line width, and so on.

x y

Two integers that specify the X and Y coordinates of the top-left corner of the rectangle.

width height

Two integers that specify the dimensions of the rectangle, measured in pixels.

fill-p

A CLX:BOOLEAN value that specifies whether the rectangle is filled.

Return Value

Unspecified.

CLX:DRAW-RECTANGLES Function

Draws multiple solid or outlined rectangles. The origin and dimensions of each rectangle may be stored in a CLX:RECT-SEQ value.

Format

CLX:DRAW-RECTANGLES *drawable gcontext rectangles &OPTIONAL fill-p*

CLX:DRAW-RECTANGLES Function

Arguments

drawable

The CLX:DRAWABLE object in which the rectangles are drawn.

gcontext

The CLX:GCONTEXT object that specifies the graphic characteristics (line width, color, and so on) of the rectangles.

rectangles

A sequence or CLX:RECT-SEQ value that specifies the origins and sizes of the rectangles.

fill-p

A CLX:BOOLEAN value that specifies whether the rectangles are filled.

Return Value

Unspecified.

CLX:DRAW-SEGMENTS Function

Draws multiple line segments connecting each pair of points in a sequence or CLX:SEG-SEQ value.

Format

CLX:DRAW-SEGMENTS *drawable gcontext segments*

Arguments

drawable

The CLX:DRAWABLE object in which the line segments are drawn.

gcontext

The CLX:GCONTEXT object that specifies the graphics characteristics of the line segments, such as color, width, and so on.

segments

A sequence or CLX:SEG-SEQ value that contains the X and Y coordinates of the endpoints of the line segments.

Return Value

Unspecified.

CLX:EVENT-CASE Macro

CLX:EVENT-CASE Macro

Executes the matching *clause* for each event in the queue until a *clause* returns non-NIL. The value returned by that *clause* is then returned by CLX:EVENT-CASE.

Hangs until non-NIL is generated for some event, or for the specified :TIMEOUT. Returns NIL if :TIMEOUT is reached.

Format

```
CLX:EVENT-CASE display &KEY :TIMEOUT :PEEK-P :DISCARD-P  
          :FORCE-OUTPUT-P  
          &BODY clauses
```

Arguments

display

A CLX:DISPLAY object.

:TIMEOUT

The number of seconds to wait before the macro returns.

:PEEK-P

A CLX:BOOLEAN value. If true, the event for which the *clauses* return non-NIL is left on the queue.

:DISCARD-P

A CLX:BOOLEAN value. If true, events for which the *clauses* return NIL are removed from the queue; otherwise they are left in place.

:FORCE-OUTPUT-P

A CLX:BOOLEAN value. If true, the default, the macro invokes CLX:DISPLAY-FORCE-OUTPUT before it starts reading events.

clauses

One or more LISP forms that include the forms to be executed when particular events occur. The *clauses* have the following syntax:

```
(event-key | ({event-key}*) ({arg}* | {(arg var)*) forms )
```

event-key A CLX:EVENT-KEY value. The names may be typed as variables or as keywords. The names are not evaluated, and it is an error for the same *event-key* to appear in more than one *clause*.

arg An argument that is appropriate to the *event-key*, as listed in Section 12.5. As with keyword arguments in a lambda list (*keyword var*) forms are allowed.

var A variable you want to be bound to the *event-key* argument.

forms One or more LISP forms to be executed for the event.

The last *clause* may contain T or OTHERWISE in place of a CLX:EVENT-KEY. If you do not provide such a clause, the implicit clause (otherwise nil) is used.

Return Value

The first non-NIL value returned by one of the *clauses*, or NIL if :TIMEOUT is reached.

CLX:EVENT-KEY Type Specifier

CLX:EVENT-KEY is a data abstraction that specifies the keywords that are usable with the CLX:EVENT-CASE macro and CLX:PROCESS-EVENT function.

Representation

```
(MEMBER :KEY-PRESS :KEY-RELEASE :BUTTON-PRESS :BUTTON-RELEASE  
      :MOTION-NOTIFY :ENTER-NOTIFY :LEAVE-NOTIFY :FOCUS-IN :FOCUS-OUT  
      :KEYMAP-NOTIFY :EXPOSURE :GRAPHICS-EXPOSURE :NO-EXPOSURE  
      :VISIBILITY-NOTIFY :CREATE-NOTIFY :DESTROY-NOTIFY :UNMAP-NOTIFY  
      :MAP-NOTIFY :MAP-REQUEST :REARENT-NOTIFY :CONFIGURE-NOTIFY  
      :GRAVITY-NOTIFY :RESIZE-REQUEST :CONFIGURE-REQUEST  
      :CIRCULATE-NOTIFY :CIRCULATE-REQUEST :PROPERTY-NOTIFY  
      :SELECTION-CLEAR :SELECTION-REQUEST :SELECTION-NOTIFY  
      :COLORMAP-NOTIFY :CLIENT-MESSAGE)
```

CLX:EVENT-LISTEN Function

Returns the number of events queued for the specified display, if any, else NIL. Hangs waiting for events, forever if *timeout* is NIL, else for the specified number of seconds.

Format

CLX:EVENT-LISTEN *display* &OPTIONAL *timeout*

Arguments

display

A CLX:DISPLAY object.

timeout

The number of seconds to wait for events, or NIL if no limit. The default value is 0.

Return Value

The number of events queued locally, or NIL if there are none.

CLX:EVENT-MASK Type Specifier

CLX:EVENT-MASK Type Specifier

Objects of type CLX:EVENT-MASK are returned by the CLX:MAKE-EVENT-MASK function and can be passed to the CLX:SEND-EVENT function.

Representation

(OR CLX:MASK32 (LIST CLX:EVENT-MASK-CLASS))

CLX:EVENT-MASK-CLASS Type Specifier

Objects of type CLX:EVENT-MASK-CLASS are returned by the CLX:MAKE-EVENT-KEYS function and may be passed to the CLX:MAKE-EVENT-MASK function.

Representation

(MEMBER :KEY-PRESS :KEY-RELEASE :OWNER-GRAB-BUTTON :BUTTON-PRESS
:BUTTON-RELEASE :ENTER-WINDOW :LEAVE-WINDOW :POINTER-MOTION
:POINTER-HINT :BUTTON-1-MOTION :BUTTON-2-MOTION :BUTTON-3-MOTION
:BUTTON-4-MOTION :BUTTON-5-MOTION :BUTTON-MOTION :EXPOSURE
:VISIBILITY-CHANGE :STRUCTURE-NOTIFY :RESIZE-REDIRECT
:SUBSTRUCTURE-NOTIFY :SUBSTRUCTURE-REDIRECT :FOCUS-CHANGE
:PROPERTY-CHANGE :COLORMAP-CHANGE :KEYMAP-STATE)

CLX:FIND-ATOM Function

Given the name of a property, returns the atom number.

Format

CLX:FIND-ATOM *display name*

Arguments

display

A CLX:DISPLAY object.

name

A CLX:XATOM value.

Return Value

A FIXNUM that indicates the atom-id.

CLX:FONT Structure

Holds the representation of a font.

Constructor Function

CLX:CREATE-FONT

Accessor Functions

None of the following functions can be used with SETF:

Name	Type Definition
CLX:FONT-ALL-CHARS-EXIST-P	CLX:BOOLEAN
CLX:FONT-ASCENT	INTEGER
CLX:FONT-DEFAULT-CHAR	INTEGER
CLX:FONT-DESCENT	INTEGER
CLX:FONT-DIRECTION	CLX:DRAW-DIRECTION
CLX:FONT-DISPLAY	CLX:DISPLAY
CLX:FONT-EQUAL	CLX:BOOLEAN
CLX:FONT-ID	CLX:RESOURCE-ID
CLX:FONT-MAX-BYTE1	INTEGER
CLX:FONT-MAX-BYTE2	INTEGER
CLX:FONT-MAX-CHAR	INTEGER
CLX:FONT-MIN-BYTE1	INTEGER
CLX:FONT-MIN-BYTE2	INTEGER
CLX:FONT-MIN-CHAR	INTEGER
CLX:FONT-NAME	(OR STRING NULL)
CLX:FONT-P	CLX:BOOLEAN

Deallocator Function

Instances of type CLX:FONT are subject to garbage collection.

CLX:FONTABLE Type Specifier

A data abstraction representing objects that can be coerced into a font specification.

Representation

(OR CLX:STRINGABLE CLX:FONT)

CLX:FONT-ALL-CHARS-EXIST-P Function

CLX:FONT-ALL-CHARS-EXIST-P Function

Returns true if all characters in the specified font have nonzero bounding boxes.

Format

CLX:FONT-ALL-CHARS-EXIST-P *font*

Argument

font

A CLX:FONT object.

Return Value

A CLX:BOOLEAN value.

CLX:FONT-ASCENT Function

Returns the distance from the baseline to the top of the font's bounding box.

Format

CLX:FONT-ASCENT *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:FONT-DEFAULT-CHAR Function

Returns the index to the character used when an undefined or nonexistent character is drawn.

CLX:FONT-DEFAULT-CHAR Function

Format

CLX:FONT-DEFAULT-CHAR *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:FONT-DESCENT Function

Returns the distance from the baseline to the bottom of the font's bounding box.

Format

CLX:FONT-DESCENT *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:FONT-DIRECTION Function

Gives a hint about the direction in which the font is painted.

Format

CLX:FONT-DIRECTION *font*

Argument

font

A CLX:FONT object.

CLX:FONT-DIRECTION Function

Return Value

The CLX:DRAW-DIRECTION of the given font, either :LEFT-TO-RIGHT or :RIGHT-TO-LEFT.

CLX:FONT-DISPLAY Function

Returns the display on which the specified font was opened.

Format

CLX:FONT-DISPLAY *font*

Argument

font

A CLX:FONT object.

Return Value

A DISPLAY object.

CLX:FONT-EQUAL Function

Returns true if its two arguments are the same CLX:FONT object.

Format

CLX:FONT-EQUAL *object-1 object-2*

Arguments

object-1 object-2

Any two LISP objects.

Return Value

A CLX:BOOLEAN value.

CLX:FONT-ID Function

Returns the resource-id of the specified font.

Format

CLX:FONT-ID *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:FONT-MAX-BYTE1 Function

Returns the last row of the font.

Format

CLX:FONT-MAX-BYTE1 *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:FONT-MAX-BYTE2 Function

For 16-bit fonts, returns the last byte of the font. For 8-bit fonts, this is the same as CLX:FONT-MAX-CHAR.

CLX:FONT-MAX-BYTE2 Function

Format

CLX:FONT-MAX-BYTE2 *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:FONT-MAX-CHAR Function

Returns the index to the last character of the font.

Format

CLX:FONT-MAX-CHAR *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:FONT-MIN-BYTE1 Function

Returns the first row of the font.

Format

CLX:FONT-MIN-BYTE1 *font*

Argument

font

A CLX:FONT object.

CLX:FONT-MIN-BYTE1 Function

Return Value

An integer.

CLX:FONT-MIN-BYTE2 Function

For 16-bit fonts, returns the first byte in the font. For 8-bit fonts, this is the same as CLX:FONT-MIN-CHAR.

Format

CLX:FONT-MIN-BYTE2 *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:FONT-MIN-CHAR Function

Returns the index of the first character in the font.

Format

CLX:FONT-MIN-CHAR *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:FONT-NAME Function

CLX:FONT-NAME Function

Returns a string that names the font. Returns NIL for a pseudo font returned by CLX:GCONTEXT-FONT.

Format

CLX:FONT-NAME *font*

Argument

font

A CLX:FONT object.

Return Value

A STRING or NIL.

CLX:FONT-P Function

Returns true if its argument is a CLX:FONT object.

Format

CLX:FONT-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:FONT-PATH Function

Returns the search path used by the server to find font files.

Format

CLX:FONT-PATH *display* &KEY :RESULT-TYPE

Arguments

display

A CLX:DISPLAY object.

:RESULT-TYPE

The LISP type of the return value. The default is 'LIST.

Return Value

A sequence of strings or pathnames.

CLX:FONT-PROPERTIES Function

Returns any additional properties defined for the specified font.

Format

CLX:FONT-PROPERTIES *font*

Argument

font

A CLX:FONT object.

Return Value

A CLX:FONT-PROPS value.

CLX:FONT-PROPERTY Function

CLX:FONT-PROPERTY Function

Given a font and the name of a property, returns the value of that property.

Format

CLX:FONT-PROPERTY *font name*

Arguments

font

A CLX:FONT object.

name

A keyword that names a property.

Return Value

An integer, or NIL if the property is not defined.

CLX:FONT-PROPS Type Specifier

A list containing alternating keywords and integers. Objects of type CLX:FONT-PROPS are returned by the CLX:FONT-PROPERTIES function.

Representation

LIST

CLX:FORCE-GCONTEXT-CHANGES Function

Forces any batched changes to a GContext to be sent to the X server.

Format

CLX:FORCE-GCONTEXT-CHANGES *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

CLX:FORCE-GCONTEXT-CHANGES Function

Return Value

Unspecified.

CLX:FREE-COLORMAP Function

Deletes the association between the specified colormap and its resource-id. This function has no effect on the default colormap of the screen.

Format

CLX:FREE-COLORMAP *colormap*

Argument

colormap

A CLX:COLORMAP object.

Return Value

Unspecified.

CLX:FREE-COLORS Function

Releases all storage allocated by the following functions: CLX:ALLOC-COLOR, CLX:ALLOC-COLOR-CELLS, and CLX:ALLOC-COLOR-PLANES.

Format

CLX:FREE-COLORS *colormap pixels &OPTIONAL plane-mask*

Arguments

colormap

A CLX:COLORMAP object.

pixels

A sequence of CLX:PIXEL values to erase.

plane-mask

A CLX:MASK value that specifies which planes to erase. The default is 0, meaning all planes.

CLX:FREE-COLORS Function

Return Value

Unspecified.

CLX:FREE-CURSOR Function

Frees the memory allocated for the specified cursor.

Format

CLX:FREE-CURSOR *cursor*

Argument

cursor

A CLX:CURSOR object.

Return Value

Unspecified.

CLX:FREE-GCONTEXT Function

Frees the memory allocated to a GContext.

Format

CLX:FREE-GCONTEXT *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

Unspecified.

CLX:FREE-PIXMAP Function

Deallocates the memory used by a pixmap.

Format

CLX:FREE-PIXMAP *pixmap*

Argument

pixmap

A CLX:PIXMAP object.

Return Value

Unspecified.

CLX:GCONTEXT Structure

Contains a set of graphics characteristics to be used when drawing graphics objects.

Constructor Function

An instance of the CLX:GCONTEXT structure is returned by the CLX:CREATE-GCONTEXT function.

Accessor Functions

Unless otherwise indicated, the following functions can be used with SETF:

Name	Type Definition
CLX:GCONTEXT-ARC-MODE	(OR NULL (MEMBER :CHORD :PIE-SLICE))
CLX:GCONTEXT-BACKGROUND	CLX:PIXEL
CLX:GCONTEXT-CACHE-P	CLX:BOOLEAN
	This function is not a valid SETF place.
CLX:GCONTEXT-CAP-STYLE	(OR NULL (MEMBER :NOT-LAST :BUTT :ROUND :PROJECTING))
CLX:GCONTEXT-CLIP-MASK	(OR NULL (MEMBER :NONE) CLX:PIXMAP CLX:RECT-SEQ)
CLX:GCONTEXT-CLIP-ORDERING	(OR NULL (MEMBER :UNSORTED :Y-SORTED :YX-SORTED :YX-BANDED))
CLX:GCONTEXT-CLIP-X	INTEGER
CLX:GCONTEXT-CLIP-Y	INTEGER

CLX:GCONTEXT Structure

Name	Type Definition
CLX:GCONTEXT-DASH-OFFSET	(OR NULL INTEGER)
CLX:GCONTEXT-DASHES	(OR NULL (OR INTEGER (SEQUENCE INTEGER)))
CLX:GCONTEXT-DISPLAY	CLX:DISPLAY
	This function is not a valid SETF place.
CLX:GCONTEXT-EQUAL	CLX:BOOLEAN
	This function is not a valid SETF place.
CLX:GCONTEXT-EXPOSURES	(OR NULL (MEMBER :ON :OFF))
CLX:GCONTEXT-FILL-RULE	(OR NULL (MEMBER :EVEN-ODD :WINDING))
CLX:GCONTEXT-FILL-STYLE	(OR NULL (MEMBER :SOLID :TILED :OPAQUE-STIPPLED :STIPPLED))
	(OR NULL FONTABLE)
CLX:GCONTEXT-FONT	CLX:PIXEL
CLX:GCONTEXT-FOREGROUND	CLX:LOGICAL-OP
CLX:GCONTEXT-FUNCTION	INTEGER
CLX:GCONTEXT-ID	This function is not a valid SETF place.
	(OR NULL (MEMBER :MITER :ROUND :BEVEL))
CLX:GCONTEXT-JOIN-STYLE	(MEMBER :SOLID :DASH :DOUBLE-DASH)
CLX:GCONTEXT-LINE-STYLE	INTEGER
CLX:GCONTEXT-LINE-WIDTH	CLX:BOOLEAN
CLX:GCONTEXT-P	This function is not a valid SETF place.
	INTEGER
CLX:GCONTEXT-PLANE-MASK	(OR NULL CLX:PIXMAP)
CLX:GCONTEXT-STIPPLE	(OR NULL (MEMBER :CLIP-BY-CHILDREN :INCLUDE-INFERIORS))
CLX:GCONTEXT-SUBWINDOW-MODE	(OR NULL CLX:PIXMAP)
CLX:GCONTEXT-TILE	INTEGER
CLX:GCONTEXT-TS-X	INTEGER
CLX:GCONTEXT-TS-Y	

Deallocator Function

CLX:FREE-GCONTEXT

CLX:GCONTEXT-ARC-MODE Function

Returns a keyword indicating how the server connects the endpoints of an arc drawn with the specified graphics context.

Format

CLX:GCONTEXT-ARC-MODE *gcontext*

CLX:GCONTEXT-ARC-MODE Function

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

The arc mode of the given GCONTEXT, either :CHORD or :PIE-SLICE.

CLX:GCONTEXT-BACKGROUND Function

Returns the background color of the specified graphics context.

Format

CLX:GCONTEXT-BACKGROUND *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

A CLX:PIXEL value.

CLX:GCONTEXT-CACHE-P Function

Returns the caching mode of the specified graphics context.

Format

CLX:GCONTEXT-CACHE-P *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

CLX:GCONTEXT-CACHE-P Function

Return Value

`T` if the GContext is stored locally by CLX; otherwise `NIL`.

CLX:GCONTEXT-CAP-STYLE Function

Returns a keyword indicating how the server draws the endpoints of wide lines in the specified graphics context.

Format

CLX:GCONTEXT-CAP-STYLE *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

`:BUTT`, `:NOT-LAST`, `:ROUND`, or `:PROJECTING`; or `NIL` if the cap style is not defined.

CLX:GCONTEXT-CLIP-MASK Function

Returns the mask used to restrict write operations to a drawable.

Format

CLX:GCONTEXT-CLIP-MASK *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

A CLX:PIXMAP or CLX:RECT-SEQ value containing the clip mask, or `:NONE`.

CLX:GCONTEXT-CLIP-X Function

Returns the X coordinate of the origin for clipping operations in the specified graphics context.

Format

CLX:GCONTEXT-CLIP-X *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

An integer.

CLX:GCONTEXT-CLIP-Y Function

Returns the Y coordinate of the origin for clipping operations in the specified graphics context.

Format

CLX:GCONTEXT-CLIP-Y *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

An integer.

CLX:GCONTEXT-DASHES Function

CLX:GCONTEXT-DASHES Function

Returns the length, in pixels, of each section of a dashed line.

Format

CLX:GCONTEXT-DASHES *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

An integer, or a list of integers.

CLX:GCONTEXT-DASH-OFFSET Function

Returns the position within the dashes list (returned by CLX:GCONTEXT-DASHES) at which the server begins drawing a dashed line.

Format

CLX:GCONTEXT-DASH-OFFSET *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

An integer.

CLX:GCONTEXT-DISPLAY Function

Returns the CLX:DISPLAY object on which the specified graphics context was created.

Format

CLX:GCONTEXT-DISPLAY *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

A CLX:DISPLAY object.

CLX:GCONTEXT-EQUAL Function

Returns true if its two arguments are the same CLX:GCONTEXT object.

Format

CLX:GCONTEXT-EQUAL *object-1 object-2*

Arguments

object-1 object-2

Any two LISP objects.

Return Value

A CLX:BOOLEAN value.

CLX:GCONTEXT-EXPOSURES Function

CLX:GCONTEXT-EXPOSURES Function

Returns a keyword indicating whether the server informs the client when a CLX:COPY-AREA or COPY-PLANE function is called.

Format

CLX:GCONTEXT-EXPOSURES *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

:ON, :OFF, or NIL if exposures is not defined.

CLX:GCONTEXT-FILL-RULE Function

Returns the method used by the server to determine whether a point is "inside" a shape to be filled.

Format

CLX:GCONTEXT-FILL-RULE *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

:EVEN-ODD, :WINDING, or NIL if the fill rule is not defined.

CLX:GCONTEXT-FILL-STYLE Function

CLX:GCONTEXT-FILL-STYLE Function

Returns the method used by the server when filling lines, text, and shapes.

Format

CLX:GCONTEXT-FILL-STYLE *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

:SOLID, :TILED, :OPAQUE-STIPPLED, or :STIPPLED; or NIL if the fill style is not defined.

CLX:GCONTEXT-FONT Function

Returns the font stored in the specified GContext if it is known. If it is not known and :METRICS-P is false, NIL is returned. If it is not known and :METRICS-P is true, then a pseudo font is returned. Full metric and property information can be obtained from a pseudo font, but the pseudo font does not have a name or a resource-id; attempts to use a pseudo font where a resource-id is required result in an invalid font error.

Format

CLX:GCONTEXT-FONT *gcontext* &OPTIONAL :METRICS-P

Arguments

gcontext

A CLX:GCONTEXT object.

:METRICS-P

A CLX:BOOLEAN value.

Return Value

A CLX:FONTABLE value or NIL if the font is not defined.

CLX:GCONTEXT-FOREGROUND Function

CLX:GCONTEXT-FOREGROUND Function

Returns the foreground color of the specified graphics context.

Format

CLX:GCONTEXT-FOREGROUND *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

A CLX:PIXEL value.

CLX:GCONTEXT-FUNCTION Function

Returns the logical function used by the server to combine bits from a graphics object to be displayed with bits from the existing drawable.

Format

CLX:GCONTEXT-FUNCTION *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

A CLX:LOGICAL-OP value.

CLX:GCONTEXT-ID Function

Returns the resource-id of the specified graphics context.

Format

CLX:GCONTEXT-ID *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

An integer.

CLX:GCONTEXT-JOIN-STYLE Function

Returns the method used by the server when drawing connections between line segments in the specified graphics context.

Format

CLX:GCONTEXT-JOIN-STYLE *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

:MITER, :ROUND, or :BEVEL; or NIL if the join style is not defined.

CLX:GCONTEXT-KEY Type Specifier

CLX:GCONTEXT-KEY Type Specifier

Objects of type CLX:GCONTEXT-KEY may be passed to the CLX:COPY-GCONTEXT-COMPONENTS function. The values correspond to those components of the CLX:GCONTEXT structure that are valid SETF places.

Representation

```
(MEMBER :FUNCTION :PLANE-MASK :FOREGROUND :BACKGROUND :LINE-WIDTH  
      :LINE-STYLE :CAP-STYLE :JOIN-STYLE :FILL-STYLE :FILL-RULE :ARC-MODE  
      :TILE :STIPPLE :TS-X :TS-Y :FONT :SUBWINDOW-MODE :EXPOSURES :CLIP-X  
      :CLIP-Y :CLIP-MASK :DASH-OFFSET :DASHES)
```

CLX:GCONTEXT-LINE-STYLE Function

Returns the method used by the server to draw lines in the specified graphics context.

Format

CLX:GCONTEXT-LINE-STYLE *gcontext*

Argument

gcontext
A CLX:GCONTEXT object.

Return Value

:SOLID, :DASH, or :DOUBLE-DASH; or NIL if the line style is not defined.

CLX:GCONTEXT-LINE-WIDTH Function

Returns the width, in pixels, of lines drawn in the specified graphics context.

Format

CLX:GCONTEXT-LINE-WIDTH *gcontext*

Argument

gcontext
A CLX:GCONTEXT object.

CLX:GCONTEXT-LINE-WIDTH Function

Return Value

An integer.

CLX:GCONTEXT-P Function

Returns T if its argument is a CLX:GCONTEXT object.

Format

CLX:GCONTEXT-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:GCONTEXT-PLANE-MASK Function

Specifies the planes to which graphics operations are limited when drawing with the specified graphics context.

Format

CLX:GCONTEXT-PLANE-MASK *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

An integer containing a bit mask.

CLX:GCONTEXT-STIPPLE Function

CLX:GCONTEXT-STIPPLE Function

Returns the CLX:PIXMAP used for stipple operations in the specified graphics context.

Format

CLX:GCONTEXT-STIPPLE *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

A CLX:PIXMAP value.

CLX:GCONTEXT-SUBWINDOW-MODE Function

Returns a keyword indicating whether child windows clip their ancestors.

Format

CLX:GCONTEXT-SUBWINDOW-MODE *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

:CLIP-BY-CHILDREN or :INCLUDE-INFERIORS, or NIL if the subwindow mode is not defined.

CLX:GCONTEXT-TILE Function

Returns the CLX:PIXMAP used for tile operations in the specified graphics context.

Format

CLX:GCONTEXT-TILE *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

A CLX:PIXMAP value.

CLX:GCONTEXT-TS-X Function

Returns the X coordinate of the origin for tile and stipple operations in the specified graphics context.

Format

CLX:GCONTEXT-TS-X *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

An integer.

CLX:GCONTEXT-TS-Y Function

CLX:GCONTEXT-TS-Y Function

Returns the Y coordinate of the origin for tile and stipple operations in the specified graphics context.

Format

CLX:GCONTEXT-TS-Y *gcontext*

Argument

gcontext

A CLX:GCONTEXT object.

Return Value

An integer.

CLX:GET-IMAGE Function

Gets an image from the server.

Format

CLX:GET-IMAGE *drawable* &**KEY :X :Y :WIDTH :HEIGHT :PLANE-MASK
:FORMAT :RESULT-TYPE**

Arguments

drawable

A CLX:DRAWABLE object.

:X :Y

Two integers that specify the X and Y coordinates of the origin of the image. These arguments are required.

:WIDTH :HEIGHT

Two integers that specify the dimensions of the image. These arguments are required.

:PLANE-MASK

An integer that specifies which planes to get. The default value is #xFFFFFFFF, meaning all planes.

:FORMAT

The type of image. The default is :Z-PIXMAP.

CLX:GET-IMAGE Function

:RESULT-TYPE

The type of the return value. The default is the value of the :FORMAT argument.

Return Value

Returns a CLX:IMAGE-X, CLX:IMAGE-XY, or CLX:IMAGE-Z structure, depending on the value of the :RESULT-TYPE argument.

CLX:GET-PROPERTY Function

Returns the specified property and information about it from the property list of the specified window. The :TRANSFORM function is applied to each integer retrieved. NIL is returned for :TYPE when the protocol returns None.

Format

CLX:GET-PROPERTY *window* *property* &KEY :TYPE :START :END :DELETE-P
:RESULT-TYPE :TRANSFORM

Arguments

window

A CLX:WINDOW object.

property

A CLX:XATOM value.

:TYPE

A CLX:XATOM value, or NIL.

:START

An integer. The default value is 0.

:END

An integer.

:DELETE-P

A CLX:BOOLEAN value.

:RESULT-TYPE

The LISP type of the return value. The default value is 'LIST.

:TRANSFORM

A function that takes an integer argument, or NIL.

CLX:GET-PROPERTY Function

Return Values

A sequence of four values:

<i>data</i>	the property itself
<i>type</i>	the type of the property
<i>format</i>	the size of the data format
<i>bytes-after</i>	the number of bytes remaining in the property if a partial read operation was performed

In each case, NIL is returned when the protocol returns None.

CLX:GET-RAW-IMAGE Function

Brings an image into LISP memory.

If :DATA is given, it is modified in place (and returned); otherwise, a new sequence is created and returned, with a size computed from the other arguments and the returned depth. The sequence is filled with 8-bit quantities, in transmission format.

Format

CLX:GET-RAW-IMAGE *drawable* &KEY :DATA :START :X :Y :WIDTH
:HEIGHT :PLANE-MASK :FORMAT
:RESULT-TYPE

Arguments

drawable

The CLX:DRAWABLE that contains the image to be stored in memory.

:DATA

A sequence to store the image in, or NIL.

:START

A CLX:ARRAY-INDEX value that specifies the starting position in :DATA. The default value is 0.

:X :Y

Two integers that specify the X and Y coordinates of the origin of the image.

:WIDTH :HEIGHT

Two integers that specify the dimensions of the image.

:PLANE-MASK

A CLX:PIXEL value that specifies the planes to be retrieved. The default value is #xFFFFFFFF.

:FORMAT

The type of image format, either :XY-PIXMAP or :Z-PIXMAP.

CLX:GET-RAW-IMAGE Function

:RESULT-TYPE

The LISP type of the first return value. The default is '(VECTOR (UNSIGNED-BYTE 8)).

Return Values

Two values:

- A sequence of bytes containing the image data.
- An integer that indicates the depth of the image.

CLX:GET-WM-CLASS Function

Returns the class of a window.

Format

CLX:GET-WM-CLASS *window*

Argument

window

A CLX:WINDOW object.

Return Values

Two values:

- A string containing the window's name, or NIL if none has been defined.
- The window's class, either "INPUT" or "INPUT-OUTPUT"; or NIL if the class was copied from the window's parent when the window was created.

CLX:GLOBAL-POINTER-POSITION Function

Returns the position of the pointer cursor and the root window on the specified display.

Format

CLX:GLOBAL-POINTER-POSITION *display*

CLX:GLOBAL-POSITION Function

Argument

display

A CLX:DISPLAY object.

Return Values

Three values:

- The X coordinate of the pointer cursor.
- The Y coordinate of the pointer cursor.
- The root window.

CLX:GRAB-BUTTON Function

Passively grabs the specified button on the pointer device. The grab is activated when the button is pressed.

Format

```
CLX:GRAB-BUTTON window button event-mask
    &KEY :MODIFIERS OWNER-P :SYNC-POINTER-P
        :SYNC-KEYBOARD-P :CONFINE-TO :CURSOR
```

Arguments

window

A CLX:WINDOW object.

button

An integer or :ANY.

event-mask

A CLX:POINTER-EVENT-MASK value.

:MODIFIERS

A CLX:MODIFIER-MASK value. The default value is 0.

:OWNER-P

A CLX:BOOLEAN value. If true, all pointer events are reported to the client. If false, pointer events are reported only when they occur in *window* and are selected by *event-mask*.

:SYNC-POINTER-P

A CLX:BOOLEAN value. If true, pointer events are processed synchronously.

:SYNC-KEYBOARD-P

A CLX:BOOLEAN value. If true, keyboard events are processed synchronously.

CLX:GRAB-BUTTON Function

:CONFINE-TO

The CLX:WINDOW object to which the cursor is confined for the duration of the grab, or null if the cursor can move to any window.

:CURSOR

The CLX:CURSOR to be displayed for the duration of the grab, or null if the cursor does not change.

Return Value

Unspecified.

CLX:GRAB-KEY Function

Passively grabs the specified key on the keyboard. The grab is activated when the key is pressed.

Format

CLX:GRAB-KEY *window* *key &KEY :MODIFIERS :OWNER-P :SYNC-Pointer-P :SYNC-Keyboard-P*

Arguments

window

A CLX:WINDOW object.

key

An integer or :ANY.

:MODIFIERS

A CLX:MODIFIER-MASK. The default value is 0.

:OWNER-P

A CLX:BOOLEAN value. If true, all keyboard events are reported to the client. If false, only events occurring in *window* and selected by *key* are reported.

:SYNC-Pointer-P

A CLX:BOOLEAN value. If true, pointer events are processed synchronously.

:SYNC-Keyboard-P

A CLX:BOOLEAN value. If true, keyboard events are processed synchronously.

Return Value

Unspecified.

CLX:GRAB-KEYBOARD Function

CLX:GRAB-KEYBOARD Function

Actively grabs the keyboard.

Format

CLX:GRAB-KEYBOARD *window* &**KEY** :OWNER-P :SYNC-POINTER-P
:SYNC-KEYBOARD-P :TIME

Arguments

window

A CLX:WINDOW object.

:OWNER-P

A CLX:BOOLEAN value. If true, all pointer events are reported to the client. If false, keyboard events are reported only when they occur in *window*.

:SYNC-POINTER-P

A CLX:BOOLEAN value. If true, pointer events are processed synchronously.

:SYNC-KEYBOARD-P

A CLX:BOOLEAN value. If true, keyboard events are processed synchronously.

:TIME

The CLX:TIMESTAMP when the grab takes effect.

Return Value

A CLX:GRAB-STATUS value.

CLX:GRAB-POINTER Function

Actively grabs the pointing device.

Format

CLX:GRAB-POINTER *window* *event-mask*
&**KEY** :OWNER-P :SYNC-POINTER-P :SYNC-KEYBOARD-P
:CONFINE-TO :CURSOR :TIME

Arguments

window

A CLX:WINDOW object.

CLX:GRAB-POINTER Function

event-mask

A CLX:POINTER-EVENT-MASK value that specifies which pointer events are reported during the grab.

:OWNER-P

A CLX:BOOLEAN value. If true, all pointer events are reported to the client. When false, pointer events are reported only when they occur in *window*.

:SYNC-POINTER-P

A CLX:BOOLEAN value. If true, pointer events are processed synchronously.

:SYNC-KEYBOARD-P

A CLX:BOOLEAN value. If true, keyboard events are processed synchronously.

:CONFINE-TO

The CLX:WINDOW object to which the pointer cursor is confined for the duration of the grab, or null if the cursor can move to any window.

:CURSOR

The CLX:CURSOR displayed for the duration of the grab, or null if the cursor does not change.

:TIME

The CLX:TIMESTAMP when the grab takes effect.

Return Value

A CLX:GRAB-STATUS value.

CLX:GRAB-SERVER Function

Allows a client program to take exclusive possession of the server for a specified display. No requests are processed while the server is grabbed, not even requests to close connections. You should not grab the server any more than is absolutely necessary.

A server grab is automatically released when the client closes its connection to the server.

Format

CLX:GRAB-SERVER *display*

Argument

display

A CLX:DISPLAY object.

CLX:GRAB-SERVER Function

Return Value

Unspecified.

CLX:GRAB-STATUS Type Specifier

Objects of type CLX:GRAB-STATUS are returned by the CLX:GRAB-KEYBOARD and CLX:GRAB-POINTER functions.

Representation

(MEMBER :SUCCESS :ALREADY-GRABBED :FROZEN :INVALID-TIME :NOT-VIEWABLE)

CLX:ICON-SIZES Function

Uses the CLX:SIZE-HINTS structure to find the recommended size for the icon of a window.

Format

CLX:ICON-SIZES *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:WM-SIZE-HINTS object.

CLX:IMAGE Structure

Images structures allow you to access screen data from LISP, and transfer data between the client and the server.

Constructor Function

CLX:CREATE-IMAGE

Accessor Functions

Unless otherwise indicated, the following functions can be used with SETF:

Name	Type Definition
CLX:IMAGE-BIT-LSB-FIRST-P	CLX:BOOLEAN
CLX:IMAGE-BITS-PER-PIXEL	INTEGER
CLX:IMAGE-BLUE-MASK	CLX:MASK16
CLX:IMAGE-BYTE-LSB-FIRST-P	CLX:BOOLEAN
CLX:IMAGE-BYTES-PER-LINE	CLX:CARD16
CLX:IMAGE-DEPTH	CARD8 The default value is 1. This function is not a valid SETF place.
CLX:IMAGE-FORMAT	(MEMBER :BITMAP :XY-PIXMAP :Z-PIXMAP)
CLX:IMAGE-GREEN-MASK	CLX:MASK16
CLX:IMAGE-HEIGHT	CARD16
CLX:IMAGE-RED-MASK	CLX:MASK16
CLX:IMAGE-SCANLINE-PAD	(MEMBER 8 16 32)
CLX:IMAGE-WIDTH	CARD16

Predicate

CLX:IMAGE-P

Deallocator Function

Objects of type CLX:IMAGE are subject to garbage collection.

CLX:IMAGE-BIT-LSB-FIRST-P Function

Returns true if the data for the specified image is stored with the least significant bit first.

Format

CLX:IMAGE-BIT-LSB-FIRST-P *image*

Argument

Image

A CLX:IMAGE object.

CLX:IMAGE-BIT-LSB-FIRST-P Function

Return Value

A CLX:BOOLEAN value.

CLX:IMAGE-BITS-PER-PIXEL Function

Returns the number of bits used to hold each pixel in the specified image.

Format

CLX:IMAGE-BITS-PER-PIXEL *image*

Argument

image

A CLX:IMAGE object.

Return Value

A member of (1 4 8 16 24 32).

CLX:IMAGE-BLUE-MASK Function

Returns the blue value of a pixel in an image whose format is :Z-PIXMAP.

Format

CLX:IMAGE-BLUE-MASK *image*

Argument

image

A CLX:IMAGE object.

Return Value

A CLX:PIXEL value.

CLX:IMAGE-BYTE-LSB-FIRST-P Function

Returns true if the data for the specified image is stored with the least significant byte first.

Format

CLX:IMAGE-BYTE-LSB-FIRST-P *image*

Argument

image

A CLX:IMAGE object.

Return Value

A CLX:BOOLEAN value.

CLX:IMAGE-BYTES-PER-LINE Function

Returns the number of bytes in each scanline of the specified image.

Format

CLX:IMAGE-BYTES-PER-LINE *image*

Argument

image

A CLX:IMAGE object.

Return Value

A CLX:CARD16 value.

CLX:IMAGE-DEPTH Function

CLX:IMAGE-DEPTH Function

Returns the depth of the specified image.

Format

CLX:IMAGE-DEPTH *image*

Argument

image

A CLX:IMAGE object.

Return Value

A CLX:CARD8 value.

CLX:IMAGE-DEPTH Type Specifier

An abstract type used for the depth of an image, pixmap-format, or visual-info.

Representation

(INTEGER 0 32)

CLX:IMAGE-FORMAT Function

Returns the format in which image data is stored.

Format

CLX:IMAGE-FORMAT *image*

Argument

image

A CLX:IMAGE object.

CLX:IMAGE-FORMAT Function

Return Value

:BITMAP, :XY-PIXMAP, or :Z-PIXMAP.

CLX:IMAGE-GREEN-MASK Function

Returns the green value of a pixel in an image whose format is :Z-PIXMAP.

Format

CLX:IMAGE-GREEN-MASK *image*

Argument

image

A CLX:IMAGE object.

Return Value

A CLX:PIXEL value.

CLX:IMAGE-HEIGHT Function

Returns the height of the specified image.

Format

CLX:IMAGE-HEIGHT *image*

Argument

image

A CLX:IMAGE object.

Return Value

A CLX:CARD16 value.

CLX:IMAGE-P Function

CLX:IMAGE-P Function

Returns true if its argument is of type CLX: IMAGE.

Format

CLX:IMAGE-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:IMAGE-RED-MASK Function

Returns the red value of a pixel in an image whose format is :Z-PIXMAP.

Format

CLX:IMAGE-RED-MASK *image*

Argument

image

A CLX: IMAGE object.

Return Value

A CLX:PIXEL value.

CLX:IMAGE-SCANLINE-PAD Function

Returns 8, 16, or 32. Each scanline in the specified image is padded to a multiple of this value.

CLX:IMAGE-SCANLINE-PAD Function

Format

CLX:IMAGE-SCANLINE-PAD *image*

Argument

image

A CLX:IMAGE object.

Return Value

A member of (8 16 32).

CLX:IMAGE-WIDTH Function

Returns the width of the specified image.

Format

CLX:IMAGE-WIDTH *image*

Argument

image

A CLX:IMAGE object.

Return Value

A CLX:CARD16 value.

CLX:INDEX-SIZE Type Specifier

An abstract type representing the size of the buffer used for character translations, that is, 8-bit font or 16-bit font.

Representation

(MEMBER :DEFAULT 8 16)

CLX:INPUT-FOCUS Function

CLX:INPUT-FOCUS Function

Returns the window that currently has the input focus, and where the focus reverts to if that window becomes unviewable.

Format

CLX:INPUT-FOCUS *display*

Argument

display

A CLX:DISPLAY object.

Return Values

Two values:

- The CLX:WINDOW object that has input focus.
- :NONE, :PARENT, or :PARENT-ROOT.

CLX:INSTALL-COLORMAP Function

Loads color values from a virtual colormap into the hardware colormap.

Format

CLX:INSTALL-COLORMAP *colormap*

Argument

colormap

A CLX:COLORMAP object.

Return Value

Unspecified.

CLX:INSTALLED-COLORMAPS Function

CLX:INSTALLED-COLORMAPS Function

Returns the currently installed colormaps.

Format

CLX:INSTALLED-COLORMAPS *window* &**KEY :RESULT-TYPE**

Arguments

window

A CLX:WINDOW object.

:RESULT-TYPE

The LISP type of the return value. The default is 'LIST.

Return Value

A sequence of CLX:COLORMAP objects.

CLX:INTERN-ATOM Function

Allocates a CLX:XATOM for use by the client. If *name* does not name an atom, one is created.

Format

CLX:INTERN-ATOM *display name*

Arguments

display

A CLX:DISPLAY object.

name

A CLX:XATOM value.

Return Value

An integer that uniquely identifies the atom.

CLX:KEYBOARD-CONTROL Function

CLX:KEYBOARD-CONTROL Function

Returns the current keyboard settings.

Format

CLX:KEYBOARD-CONTROL *display*

Argument

display

A CLX:DISPLAY object.

Return Values

Seven values:

<i>key-click-percent</i>	The volume of key clicks between 0 (off) and 100 (loudest).
<i>bell-percent</i>	The volume of a bell ring between 0 (off) and 100 (loudest).
<i>bell-pitch</i>	The pitch (specified in hertz) of the bell.
<i>bell-duration</i>	The duration of a bell ring, in milliseconds.
<i>led-mask</i>	Each 1 bit indicates an LED that is on. The least significant bit corresponds to the first LED.
<i>global-auto-repeat</i>	:ON if auto-repeat is enabled for the keyboard as a whole; :OFF if it is not.
<i>auto-repeats</i>	A bit vector where each 1 bit indicates that auto-repeat is enabled for the corresponding key.

CLX:KEYBOARD-MAPPING Function

Returns the valid keycodes for the specified display. The number of keycodes returned is the difference between :END and :START. If :DATA is specified, the results are put there.

CLX:KEYBOARD-MAPPING Function

Format

CLX:KEYBOARD-MAPPING *display* &KEY :FIRST-KEYCODE :START :END
:DATA

Arguments

display

A CLX:DISPLAY object.

:FIRST-KEYCODE

An integer that specifies which keycode to start at. The default is the :MIN-KEYCODE component of the *display* argument.

:START

An integer that specifies where (in the return value) to put the first keycode. The default is :FIRST-KEYCODE.

:END

An integer that specifies the last keycode returned. The default :END is one more than the :MAX-KEYCODE component of the *display* argument.

:DATA

An array in which to store the returned keycodes.

Return Value

An array of CLX:KEYSYM values.

CLX:KEYCODE->CHARACTER Function

Returns the CHARACTER that corresponds to the specified keycode. The VAX LISP implementation ignores the :KEYSYM-INDEX and :KEYSYM-INDEX-FUNCTION arguments.

Format

CLX:KEYCODE->CHARACTER *display keycode state*
&KEY :KEYSYM-INDEX
:KEYSYM-INDEX-FUNCTION

Arguments

display

A CLX:DISPLAY object.

keycode

A CLX:CARD8 value.

CLX:KEYCODE->CHARACTER Function

state

A CLX:MASK16 value.

:KEYSYM-INDEX

An integer. The default is the result of the :KEYSYM-INDEX-FUNCTION function, which is called with the following parameters:

(*char0 state caps-lock-p keysyms-per-keycode*)

The *char0* argument is the character associated with the first :KEYSYM-INDEX. The *state* argument is an integer that specifies the state of the modifier keys. The *caps-lock-p* argument is non-NIL when the keysym associated with the lock modifier is for caps-lock.

:KEYSYM-INDEX-FUNCTION

The default value is #' default-keysym-index.

Return Values

Three values:

- The CHARACTER that corresponds to *keycode*; or NIL if there is none.
- If more than one character is bound to *keycode*, the second return value is a string and the first return value is the first character in that string.
- If more than one character is bound to *keycode*, the third return value is the number of characters in the string.

CLX:KEYCODE->KEYSYM Function

Converts a keycode to a keysym.

Format

CLX:KEYCODE->KEYSYM *display keycode keysym-index*

Arguments

display

A CLX:DISPLAY object.

keycode

A CLX:CARD8 value.

keysym-index

An integer.

CLX:KEYCODE->KEYSYM Function

Return Value

A CLX:KEYSYM value.

CLX:KEYSYM Type Specifier

A keysym is an encoding of the symbol on the cap of a key. A list of keysyms is associated with each keycode. The first keysym in each list is the one for no modifiers. Objects of type CLX:KEYSYM are returned by the CLX:KEYCODE->KEYSYM function and CLX:KEYSYM macro, and can be passed to the CLX:KEYSYM->CHARACTER and CLX:KEYSYM->KEYCODES functions.

Representation

INTEGER

CLX:KEYSYM->CHARACTER Function

Finds the character associated with a keysym.

Format

CLX:KEYSYM->CHARACTER *display keysym &OPTIONAL state*

Arguments

display

A CLX:DISPLAY object.

keysym

A CLX:KEYSYM value.

state

A CLX:MASK16 value that specifies the state of the modifier keys.

Return Values

Three values:

- The CHARACTER that corresponds to *keysym*; or NIL if there is none.
- If more than one character is bound to *keysym*, the second return value is a string and the first return value is the first character in that string.
- If more than one character is bound to *keysym*, the third return value is the number of characters in the string.

CLX:KEYSYM->KEYCODES Function

CLX:KEYSYM->KEYCODES Function

Finds the keycode mapped from a keysym.

Format

CLX:KEYSYM->KEYCODES *display keysym* &**OPTIONAL state**

Arguments

display

A CLX:DISPLAY object.

keysym

A CLX:KEYSYM value.

state

An integer that specifies the state of the modifier keys, or NIL.

Return Value

A CLX:CARD8 value.

CLX:KILL-CLIENT Function

Disconnects the client that created the specified resource.

Format

CLX:KILL-CLIENT *display resource-id*

Arguments

display

A CLX:DISPLAY object.

resource-id

A CLX:RESOURCE-ID value.

Return Value

Unspecified.

CLX:KILL-TEMPORARY-CLIENTS Function

CLX:KILL-TEMPORARY-CLIENTS Function

Destroys all resources created by clients that disconnected in :RETAIN-TEMPORARY close-down mode.

Format

CLX:KILL-TEMPORARY-CLIENTS *display*

Argument

display

A CLX:DISPLAY object.

Return Value

Unspecified.

CLX:LIST-EXTENSIONS Function

Returns a list of extensions to the X server.

Format

CLX:LIST-EXTENSIONS *display* &KEY :RESULT-TYPE

Arguments

display

A CLX:DISPLAY object.

:RESULT-TYPE

The LISP type of the return value. The default is 'LIST.

Return Value

A sequence of strings.

CLX:LIST-FONT-NAMES Function

CLX:LIST-FONT-NAMES Function

Lists all available font names that match a given search string.

Format

CLX:LIST-FONT-NAMES *display pattern &KEY :MAX-FONTS :RESULT-TYPE*

Arguments

display

A CLX:DISPLAY object.

pattern

A string containing the pattern that font names must match. Both wildcard characters are allowed. An asterisk (*) matches any number of characters; a question mark (?) matches a single character.

:MAX-FONTS

The maximum number of font names that can be returned. The default value is 1024.

:RESULT-TYPE

The LISP type of the first return value. The default value is 'LIST.

Return Values

Two values:

- A sequence of strings containing the names of fonts that match the *pattern* argument.
- An integer indicating the number of matches found.

CLX:LIST-FONTS Function

Returns pseudo fonts that contain basic font metrics and properties, but no per-character metrics and no resource-ids. These pseudo fonts are converted (internally) to real fonts dynamically as needed, by issuing an OpenFont request. However, the OpenFont might fail, in which case the :INVALID-FONT error is signaled.

Format

CLX:LIST-FONTS *display pattern &KEY :MAX-FONTS :RESULT-TYPE*

CLX:LIST-FONTS Function

Arguments

display

A CLX:DISPLAY object.

pattern

A string containing the pattern that font names must match. Both wildcard characters are allowed. An asterisk (*) matches any number of characters; a question mark (?) matches a single character.

:MAX-FONTS

The maximum number of font names that may be returned. The default value is 1024.

:RESULT-TYPE

The LISP type of the first return value. The default value is 'LIST.

Return Values

Two values:

- A sequence of pseudo fonts whose names match the *pattern* argument.
- An integer indicating the number of matches found.

CLX:LIST-PROPERTIES Function

Returns the property list of the specified window.

Format

CLX:LIST-PROPERTIES *window* &**KEY :RESULT-TYPE**

Arguments

window

A CLX:WINDOW object.

:RESULT-TYPE

The LISP type of the return value. The default is 'LIST.

Return Value

A sequence of keywords. Atoms are returned as keywords.

CLX:LOGICAL-OP Type Specifier

CLX:LOGICAL-OP Type Specifier

Logical operations are functions used by the server to combine source and destination bits when displaying graphics objects on a screen. The screen the client is updating is the destination (*dst*). Whatever graphics object the client is drawing on the screen is the source (*src*). CLX:LOGICAL-OP values are specified as keywords in the :FUNCTION component of CLX:GCONTEXT objects.

Representation

A Common LISP BOOLE constant, or one of the following keywords:

Keyword	Description
:GX-CLEAR	0
:GX-AND	<i>src</i> AND <i>dst</i>
:GX-AND-REVERSE	<i>src</i> AND NOT <i>dst</i>
:GX-COPY	<i>src</i>
:GX-AND-INVERTED	(NOT <i>src</i>) AND <i>dst</i>
:GX-NOOP	<i>dst</i>
:GX-XOR	<i>src</i> XOR <i>dst</i>
:GX-OR	<i>src</i> OR <i>dst</i>
:GX-NOR	(NOT <i>src</i>) AND NOT <i>dst</i>
:GX-EQUIV	(NOT <i>src</i>) XOR <i>dst</i>
:GX-INVERT	NOT <i>dst</i>
:GX-OR-REVERSE	<i>src</i> OR NOT <i>dst</i>
:GX-COPY-INVERTED	NOT <i>src</i>
:GX-OR-INVERTED	(NOT <i>src</i>) OR <i>dst</i>
:GX-NAND	(NOT <i>src</i>) OR NOT <i>dst</i>
:GX-SET	1

CLX:LOOKUP-COLOR Function

Finds the RGB values associated with a named color on a particular screen.
Returns both the screen values (what the hardware supports) and the true values of the named color.

Format

CLX:LOOKUP-COLOR *colormap name*

CLX:LOOKUP-COLOR Function

Arguments

colormap

A CLX:COLORMAP object.

name

A CLX:STRINGABLE value.

Return Values

Two CLX:COLOR values: the first is the screen color; the second is the true color.

CLX:MAKE-COLOR Function

Makes an instance of the CLX:COLOR structure.

Format

CLX:MAKE-COLOR &KEY :RED :GREEN :BLUE &ALLOW-OTHER-KEYS

Arguments

:RED :GREEN :BLUE

(NUMBER 0 1)

Return Value

A CLX:COLOR object.

CLX:MAKE-COLORMAP Function

Given a display and a colormap resource-id, makes an instance of the CLX:COLORMAP structure.

Format

CLX:MAKE-COLORMAP *display resource-id*

Arguments

display

A CLX:DISPLAY object.

CLX:MAKE-COLORMAP Function

resource-id

An integer.

Return Value

A CLX:COLORMAP object.

CLX:MAKE-CURSOR Function

Given a display and a cursor resource-id, makes an instance of the CLX:CURSOR structure.

Format

CLX:MAKE-CURSOR *display resource-id*

Arguments

display

A CLX:DISPLAY object.

resource-id

An integer

Return Value

A CLX:CURSOR object.

CLX:MAKE-DRAWABLE Function

Given a display and a drawable resource-id, makes an object of type CLX:DRAWABLE.

Format

CLX:MAKE-DRAWABLE *display resource-id*

Arguments

display

A CLX:DISPLAY object.

resource-id

An integer.

CLX:MAKE-DRAWABLE Function

Return Value

A CLX:DRAWABLE object.

CLX:MAKE-EVENT-KEYS Function

Returns a list of CLX:EVENT-MASK-CLASS keywords that indicate which events are encoded in the event mask. This is defined only for core events.

Format

CLX:MAKE-EVENT-KEYS *event-mask*

Argument

event-mask

A CLX:MASK32 value.

Return Value

A LIST of CLX:EVENT-MASK-CLASS values.

CLX:MAKE-EVENT-MASK Function

This is defined only for core events. Useful for constructing CLX:EVENT-MASK, CLX:POINTER-EVENT-MASK, CLX:DEVICE-EVENT-MASK.

Format

CLX:MAKE-EVENT-MASK &REST *keys*

Arguments

keys

CLX:EVENT-MASK-CLASS keywords that indicate which events the mask contains.

Return Value

A CLX:MASK32 value that can be passed to functions that require an event mask.

CLX:MAKE-FONT Function

CLX:MAKE-FONT Function

Given a display and a font resource-id, makes an instance of the CLX:FONT structure.

Format

CLX:MAKE-FONT *display resource-id*

Arguments

display

A CLX:DISPLAY object.

resource-id

An integer.

Return Value

A CLX:FONT object.

CLX:MAKE-GCONTEXT Function

Given a display and a GContext resource-id, makes an instance of the CLX:GCONTEXT structure.

Format

CLX:MAKE-GCONTEXT *display resource-id*

Arguments

display

A CLX:DISPLAY object.

resource-id

An integer.

Return Value

A CLX:GCONTEXT object.

CLX:MAKE-PIXMAP Function

Given a display and a pixmap resource-id, makes an instance of the CLX:PIXMAP structure.

Format

CLX:MAKE-PIXMAP *display* *resource-id*

Arguments

display

A CLX:DISPLAY object.

resource-id

An integer.

Return Value

A CLX:PIXMAP object.

CLX:MAKE-STATE-KEYS Function

Returns a list of CLX:STATE-MASK-CLASS keywords that indicate which events are encoded in the event mask. This is defined only for core events.

Format

CLX:MAKE-STATE-KEYS *state-mask*

Argument

state-mask

A CLX:MASK16 value.

Return Value

A list of CLX:STATE-MASK-KEY values.

CLX:MAKE-STATE-MASK Function

CLX:MAKE-STATE-MASK Function

Useful for constructing CLX:MODIFIER-MASK and CLX:STATE-MASK.

Format

CLX:MAKE-STATE-MASK &REST keys

Argument

keys

A list of CLX:STATE-MASK-KEYS values.

Return Value

A CLX:MASK16 value that can be passed to functions that require a state mask.

CLX:MAKE-VISUAL-INFO Function

Given a resource-id, makes an instance of the CLX:VISUAL-INFO structure.

Format

CLX:MAKE-VISUAL-INFO resource-id

Argument

resource-id

A CLX:VISUAL value.

Return Value

A CLX:VISUAL-INFO object.

CLX:MAKE-WINDOW Function

Given a display and a resource-id, makes an instance of the CLX:WINDOW structure.

Format

CLX:MAKE-WINDOW display resource-id

CLX:MAKE-WINDOW Function

Arguments

display

A CLX:DISPLAY object.

resource-id

An integer.

Return Value

A CLX:WINDOW object.

CLX:MAKE-WM-HINTS Function

Given a display and a resource-id, makes an instance of the CLX:WM-HINTS structure.

Format

CLX:MAKE-WM-HINTS *display resource-id*

Arguments

display

A CLX:DISPLAY object.

resource-id

An integer.

Return Value

A CLX:WM-HINTS object.

CLX:MAKE-WM-SIZE-HINTS Function

Given a display and a resource-id, makes an instance of the CLX:WM-SIZE-HINTS structure.

Format

CLX:MAKE-WM-SIZE-HINTS *display resource-id*

CLX:MAKE-WM-SIZE-HINTS Function

Arguments

display

A CLX:DISPLAY object.

resource-id

An integer.

Return Value

A CLX:WM-SIZE-HINTS object.

CLX:MAP-SUBWINDOWS Function

Causes all subwindows of the specified window to be mapped or made viewable on the screen. The windows become visible when the output buffer holding requests to the server is flushed (see CLX:DISPLAY-FORCE-OUTPUT).

Format

CLX:MAP-SUBWINDOWS *window*

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:MAP-WINDOW Function

Causes the specified window to be mapped or made viewable on the screen. The window becomes visible when the output buffer holding requests to the server is flushed (see CLX:DISPLAY-FORCE-OUTPUT).

Format

CLX:MAP-WINDOW *window*

CLX:MAP-WINDOW Function

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:MASK16 Type Specifier

An abstract type used to indicate 16-bit masks. State masks are of type CLX:MASK16.

Representation

FIXNUM

CLX:MASK32 Type Specifier

An abstract type used to indicate 32-bit masks. Event masks are of type CLX:MASK32..

Representation

INTEGER

CLX:MAX-CHAR-ASCENT Function

Returns the largest ascent of any character in the font.

Format

CLX:MAX-CHAR-ASCENT *font*

Argument

font

A CLX:FONT object.

CLX:MAX-CHAR-ASCENT Function

Return Value

An integer.

CLX:MAX-CHAR-ATTRIBUTES Function

Returns the largest number of attributes of any character in the font.

Format

CLX:MAX-CHAR-ATTRIBUTES *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MAX-CHAR-DESCENT Function

Returns the largest descent of any character in the font.

Format

CLX:MAX-CHAR-DESCENT *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MAX-CHAR-LEFT-BEARING Function

CLX:MAX-CHAR-LEFT-BEARING Function

Returns the largest left-bearing of any character in the font.

Format

CLX:MAX-CHAR-LEFT-BEARING *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MAX-CHAR-RIGHT-BEARING Function

Returns the largest right-bearing of any character in the font.

Format

CLX:MAX-CHAR-RIGHT-BEARING *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MAX-CHAR-WIDTH Function

Returns the largest width of any character in the font.

Format

CLX:MAX-CHAR-WIDTH *font*

CLX:MAX-CHAR-WIDTH Function

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MIN-CHAR-ASCENT Function

Returns the smallest ascent of any character in the font.

Format

CLX:MIN-CHAR-ASCENT *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MIN-CHAR-ATTRIBUTES Function

Returns the smallest number of attributes owned by any character in the font.

Format

CLX:MIN-CHAR-ATTRIBUTES *font*

Argument

font

A CLX:FONT object.

CLX:MIN-CHAR-ATTRIBUTES Function

Return Value

An integer.

CLX:MIN-CHAR-DESCENT Function

Returns the smallest descent of any character in the font.

Format

CLX:MIN-CHAR-DESCENT *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MIN-CHAR-LEFT-BEARING Function

Returns the smallest left-bearing of any character in the font.

Format

CLX:MIN-CHAR-LEFT-BEARING *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MIN-CHAR-RIGHT-BEARING Function

CLX:MIN-CHAR-RIGHT-BEARING Function

Returns the smallest right-bearing of any character in the font.

Format

CLX:MIN-CHAR-RIGHT-BEARING *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MIN-CHAR-WIDTH Function

Returns the smallest width of any character in the font.

Format

CLX:MIN-CHAR-WIDTH *font*

Argument

font

A CLX:FONT object.

Return Value

An integer.

CLX:MODIFIER-KEY Type Specifier

Objects of type CLX:MODIFIER-KEY are returned by the CLX:MAKE-STATE-KEYS function and may be passed to the CLX:MAKE-STATE-MASK function.

CLX:MODIFIER-KEY Type Specifier

Representation

(MEMBER :SHIFT :LOCK :CONTROL :MOD-1 :MOD-2 :MOD-3 :MOD-4 :MOD-5)

CLX:MODIFIER-MAPPING Function

Returns the key codes currently being used as modifiers. X permits at most eight modifier keys.

Format

CLX:MODIFIER-MAPPING *display*

Argument

display

A CLX:DISPLAY object.

Return Values

Eight values, each of which is a list of CLX:KEYSYM values representing key codes that have the matching modifier bound to them:

shift
lock
control
mod1
mod2
mod3
mod4
mod5

CLX:MODIFIER-MASK Type Specifier

Objects of type CLX:MODIFIER-MASK are returned by the CLX:MAKE-STATE-MASK function and may be passed to the CLX:UNGRAB-KEY and CLX:MAKE-STATE-KEYS functions.

Representation

(OR (MEMBER :ANY) CLX:MASK16 (LIST CLX:MODIFIER-KEY))

CLX:MOTION-EVENTS Function

CLX:MOTION-EVENTS Function

Returns all events in the motion history buffer which fall between the specified start and stop times (inclusive) and which have coordinates that lie within (including borders) the specified window at its present location.

The X and Y coordinates returned are relative to the origin of the window.

If the start time is later than the stop time or if the start time is later than the current time, no events are returned. If the stop time is later than the current time, the current time is used as the stop time.

Format

CLX:MOTION-EVENTS *window* &**KEY** :**START** :**STOP** :**RESULT-TYPE**

Arguments

window

A CLX:WINDOW object.

:START :STOP

Two CLX:TIMESTAMP values.

:RESULT-TYPE

The LISP type of the return value. The default is 'LIST.

Return Value

A CLX:REPEAT-SEQ value whose elements contain three values:

x Integer

y Integer

time CLX:TIMESTAMP

CLX:OPEN-DISPLAY Function

Opens a connection to the X server on the specified host and returns a CLX:DISPLAY structure. You can open a connection to another machine on the network only if you are an authorized user on that machine. See the VMS DECwindows User's Guide for information on controlling network access to your workstation.

Format

CLX:OPEN-DISPLAY *host* &**KEY** :**DISPLAY** :**PROTOCOL**

CLX:OPEN-DISPLAY Function

Arguments

host

A string that names a machine.

:DISPLAY

An integer specifying a screen attached to the given host. The default value is 0, meaning display 0.

:PROTOCOL

Either :DECNET or :TCP. The default value is :DECNET.

Return Value

An object of type CLX:DISPLAY.

CLX:OPEN-FONT Function

Loads a font into LISP memory from a file. The CLX:FONT-PATH function returns the search path used by the server to find font files.

Format

CLX:OPEN-FONT *display name*

Arguments

display

A CLX:DISPLAY object.

name

A string that names the font to be opened.

Return Value

The opened CLX:FONT object.

CLX:PIXARRAY Type Specifier

Pixarrays are arrays of CLX:PIXEL values.

Representation

(ARRAY CLX:PIXEL (* *))

CLX:PIXEL Type Specifier

CLX:PIXEL Type Specifier

Objects of type CLX:PIXEL are returned by the color allocation functions, and can be passed to the color storage and deallocation functions and used in the :FOREGROUND and :BACKGROUND components of a GContext.

Representation

(UNSIGNED-BYTE 32)

CLX:PIXMAP Structure

A pixmap is an area of memory into which clients can either draw objects or temporarily save part of a screen.

Constructor Function

CLX:CREATE-PIXMAP

Information Functions

None of the following functions is a valid SETF place:

Name	Type Definition
CLX:PIXMAP-DISPLAY	CLX:DISPLAY
CLX:PIXMAP-EQUAL	CLX:BOOLEAN
CLX:PIXMAP-ID	CLX:RESOURCE-ID
CLX:PIXMAP-P	CLX:BOOLEAN

Deallocator Function

CLX:FREE-PIXMAP

CLX:PIXMAP-DISPLAY Function

Returns the display on which the specified pixmap was created.

Format

CLX:PIXMAP-DISPLAY *Pixmap*

Argument

Pixmap

A CLX:PIXMAP object.

CLX:PIXMAP-DISPLAY Function

Return Value

A CLX:DISPLAY object.

CLX:PIXMAP-EQUAL Function

Returns true if its two arguments are the same CLX:PIXMAP object.

Format

CLX:PIXMAP-EQUAL *object-1 object-2*

Arguments

object-1 object-2

Any two LISP objects.

Return Value

A CLX:BOOLEAN value.

CLX:PIXMAP-FORMAT Structure

Objects of type CLX:PIXMAP-FORMAT show which formats the display hardware supports.

Constructor Function

CLX:MAKE-PIXMAP-FORMAT

Accessor Functions

All of the following functions can be used with SETF:

Name	Type Definition
CLX:PIXMAP-FORMAT-BITS-PER-PIXEL	(MEMBER 1 4 16 24 32)
CLX:PIXMAP-FORMAT-DEPTH	CLX:IMAGE-DEPTH
CLX:PIXMAP-FORMAT-PAD	(MEMBER 8 16 32)

Deallocator Function

Pixmap-formats are subject to garbage collection.

CLX:PIXMAP-FORMAT-BITS-PER-PIXEL Function

CLX:PIXMAP-FORMAT-BITS-PER-PIXEL Function

Returns the number of bits used to represent each pixel in the specified pixmap-format.

Format

CLX:PIXMAP-FORMAT-BITS-PER-PIXEL *pixmap-format*

Argument

pixmap-format

A CLX:PIXMAP-FORMAT object.

Return Value

A member of (1 4 8 16 24 32).

CLX:PIXMAP-FORMAT-DEPTH Function

Returns the depth, or number of planes, in the specified pixmap-format.

Format

CLX:PIXMAP-FORMAT-DEPTH *pixmap-format*

Argument

pixmap-format

A CLX:PIXMAP-FORMAT object.

Return Value

A CLX:IMAGE-DEPTH value.

CLX:PIXMAP-FORMAT-P Function

CLX:PIXMAP-FORMAT-P Function

Returns T if its argument is a CLX:PIXMAP-FORMAT object.

Format

CLX:PIXMAP-FORMAT-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:PIXMAP-FORMAT-PAD Function

Returns the number of bits to a multiple of which each scanline in the specified pixmap-format is to be padded.

Format

CLX:PIXMAP-FORMAT-PAD *pixmap-format*

Argument

pixmap-format

A CLX:PIXMAP-FORMAT object.

Return Value

A member of (8 16 32).

CLX:PIXMAP-ID Function

CLX:PIXMAP-ID Function

Returns the integer resource-id of the specified pixmap.

Format

CLX:PIXMAP-ID *pixmap*

Argument

pixmap

A CLX:PIXMAP object.

Return Value

The CLX:RESOURCE-ID associated with *pixmap*.

CLX:PIXMAP-P Function

Returns T if its argument is a CLX:PIXMAP object.

Format

CLX:PIXMAP-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:POINTER-CONTROL Function

Returns the pointer cursor acceleration and threshold as multiple values.

Format

CLX:POINTER-CONTROL *display*

CLX:POINTER-CONTROL Function

Argument

display

A CLX:DISPLAY object.

Return Values

Two values:

- A number, possibly a ratio, that indicates acceleration compared to the default speed.
- An integer that indicates the threshold distance of pointer motion before acceleration is applied.

CLX:POINTER-EVENT-MASK Type Specifier

Objects of type CLX:POINTER-EVENT-MASK are returned by the CLX:MAKE-EVENT-MASK function and may be passed to the CLX:MAKE-EVENT-KEYS, CLX:GRAB-POINTER, and CLX:GRAB-BUTTON functions.

Representation

(OR CLX:MASK32 (LIST CLX:POINTER-EVENT-MASK-CLASS))

CLX:POINTER-EVENT-MASK-CLASS Type Specifier

Objects of type CLX:POINTER-EVENT-MASK-CLASS are returned by the CLX:MAKE-EVENT-MASK function and can be passed to the CLX:MAKE-EVENT-KEYS function.

Representation

(MEMBER :BUTTON-PRESS :BUTTON-RELEASE :ENTER-WINDOW :LEAVE-WINDOW
:POINTER-MOTION :POINTER-HINT :BUTTON-1-MOTION :BUTTON-2-MOTION
:BUTTON-3-MOTION :BUTTON-4-MOTION :BUTTON-5-MOTION
:BUTTON-MOTION :KEYMAP-STATE)

CLX:POINTER-MAPPING Function

Returns a mapping list that defines which buttons are enabled for the pointer cursor on the specified display. This function can be used with SETF.

CLX:POINTER-MAPPING Function

Format

CLX:POINTER-MAPPING *display &KEY :RESULT-TYPE*

Arguments

display

A CLX:DISPLAY object.

:RESULT-TYPE

The LISP type of the return value. The default is 'LIST.

Return Value

A sequence of integers.

CLX:POINTER-POSITION Function

Returns the position of the pointer cursor relative to the origin of the specified window. The last return value indicates whether the pointer cursor is on the same screen.

Format

CLX:POINTER-POSITION *window*

Argument

window

A CLX:WINDOW object.

Return Values

Three values:

- | | |
|----------------------|--------------------------|
| <i>x</i> | An integer X coordinate. |
| <i>y</i> | An integer Y coordinate. |
| <i>same-screen-p</i> | A CLX:BOOLEAN value. |

CLX:POINT-SEQ Type Specifier

A CLX:REPEAT-SEQ of X and Y coordinates. Objects of this type may be passed to the CLX:DRAW-POINTS function.

Representation

(CLX:REPEAT-SEQ (FIXNUM x) (FIXNUM y))

CLX:PROCESS-EVENT Function

Reads events from the queue and invokes the :HANDLER function on each event until a non-NIL value is returned. That returned object is then returned by CLX:PROCESS-EVENT. If this function is invoked recursively, the nested invocation begins with the event after the one currently being processed.

Hangs until non-NIL is returned for an event, or for the number of seconds specified by the :TIMEOUT argument.

The arguments to the :HANDLER function are described in detail in Section 12.5.

If a value for the :EVENT argument is provided, the function processes that event rather than reading from the server queue.

Format

CLX:PROCESS-EVENT *display* &**KEY** :HANDLER :TIMEOUT :PEEK-P
:DISCARD-P :FORCE-OUTPUT-P :EVENT

Arguments

display

A CLX:DISPLAY object.

:HANDLER

A function to be executed when an event occurs.

:TIMEOUT

The number of seconds to wait for an event.

:PEEK-P

A CLX:BOOLEAN value. If true, events are left on the queue.

:DISCARD-P

A CLX:BOOLEAN value. If true, events for which the :HANDLER function returns NIL are removed from the queue; otherwise, they are left in place.

:FORCE-OUTPUT-P

A CLX:BOOLEAN value. If true, the function invokes CLX:DISPLAY-FORCE-OUTPUT before it starts reading events. The default value is T.

CLX:PROCESS-EVENT Function

:EVENT

A DWT:EVENT value.

Return Value

The first non-NIL value returned by the :HANDLER function, or NIL if :TIMEOUT is reached.

CLX:PUT-IMAGE Function

Puts an image into a drawable. When :BITMAP-P is true, the function forces the format of the image to be :BITMAP when its depth is 1. This causes *gcontext* to supply the foreground and background pixels.

Format

CLX:PUT-IMAGE *drawable gcontext image &KEY :SRC-X :SRC-Y :X :Y :WIDTH :HEIGHT :BITMAP-P*

Arguments

drawable

A CLX:DRAWABLE object.

gcontext

A CLX:GCONTEXT object.

Image

A CLX:IMAGE object.

:SRC-X :SRC-Y

Integers that specify the X and Y offsets into the image data. The default value for these keywords is 0.

:X :Y

Integers that specify the X and Y coordinates in the drawable for the origin of the inserted image data. These arguments are required.

:WIDTH :HEIGHT

Integers that specify the dimensions of the image to insert.

:BITMAP-P

A CLX:BOOLEAN value.

Return Value

Unspecified.

CLX:PUT-RAW-IMAGE Function

Puts image data into a drawable.

The *data* argument must be a sequence of 8-bit quantities already in the appropriate format for transmission; the client is responsible for all byte and bit swapping and compaction. The :START argument is the starting index in *data*; the end is computed from the other arguments.

Format

CLX:PUT-Raw-IMAGE *drawable gcontext data*
 &KEY :START :DEPTH :X :Y :WIDTH :HEIGHT
 :LEFT-PAD :FORMAT

Arguments

drawable

The CLX:DRAWABLE where the image is displayed.

gcontext

The CLX:GCONTEXT that specifies the graphics characteristics, such as foreground and background colors, of the image.

data

A sequence of 8-bit quantities containing the image.

:START

A CLX:ARRAY-INDEX specifying the starting point within *data*. The default value is 0.

:DEPTH

An integer that specifies the depth of the image.

:X :Y

Integers that specify the X and Y coordinates in the drawable for the origin of the image.

:WIDTH :HEIGHT

Integers that specify the dimensions of the image to insert.

:LEFT-PAD

An integer that specifies the number of pixels offset in the X direction. The default value is 0.

CLX:PUT-RAW-IMAGE Function

:FORMAT

The type of image format, either :XY-PIXMAP or Z-PIXMAP.

Return Value

Unspecified.

CLX:QUERY-BEST-CURSOR Function

Asks the server for the preferred cursor dimensions that are closest to the ones supplied as arguments to the function.

Format

CLX:QUERY-BEST-CURSOR *width height display*

Arguments

width height

Integers that indicate dimensions of a cursor, in pixels.

display

A CLX:DISPLAY object.

Return Values

Two integers that specify the dimensions of the server's nearest cursor dimensions.

CLX:QUERY-BEST-STIPPLE Function

Asks the server for the preferred dimensions for fast stippling that are closest to the ones supplied as arguments to the function.

Format

CLX:QUERY-BEST-STIPPLE *width height drawable*

Arguments

width height

Integers that indicate the requested dimensions.

CLX:QUERY-BEST-STIPPLE Function

drawable

A CLX:DRAWABLE object.

Return Values

Two integers that represent the server's preferred dimensions for fast stipple operations.

CLX:QUERY-BEST-TILE Function

Asks the server for the preferred dimensions for fast tiling that are closest to the ones supplied as arguments to the function.

Format

CLX:QUERY-BEST-TILE *width height drawable*

Arguments

width height

Integers that indicate the requested dimensions.

drawable

A CLX:DRAWABLE object.

Return Values

Two integers that represent the server's preferred dimensions for fast tiling operations.

CLX:QUERY-COLORS Function

Returns the RGB values of specified entries in a colormap.

Format

CLX:QUERY-COLORS *colormap pixels &KEY :RESULT-TYPE*

Arguments

colormap

A CLX:COLORMAP object.

pixels

A sequence of CLX:PIXEL values.

CLX:QUERY-COLORS Function

:RESULT-TYPE

The LISP type of the return value. The default is 'LIST.

Return Value

A sequence of CLX:COLOR objects.

CLX:QUERY-EXTENSION Function

Returns information about the specified extension. For more information, see the *X Window System: C Library and Protocol Reference*.

Format

CLX:QUERY-EXTENSION *display name*

Arguments

display

A CLX:DISPLAY object.

name

A string that names an extension.

Return Values

Three integer values:

- The major operation code, if any.
- The base event type code, if the extension defines additional event types.
- The base error code, if the extension defines additional error codes.

CLX:QUERY-KEYMAP Function

Returns the state of keys on the keyboard.

Format

CLX:QUERY-KEYMAP *display*

CLX:QUERY-KEYMAP Function

Argument

display

A CLX:DISPLAY object.

Return Value

A (BIT-VECTOR 256) value whose 1 bits signify that a key is pressed.

CLX:QUERY-POINTER Function

Asks the server for information about the pointer, relative to the specified window.

Format

CLX:QUERY-POINTER *window*

Argument

window

A CLX:WINDOW object.

Return Values

Eight values:

1. Integer X coordinate of the pointer cursor's location.
2. Integer Y coordinate of the pointer cursor's location.
3. A CLX:BOOLEAN value that indicates whether the pointer cursor is on the same screen as *window*.
4. The CLX:WINDOW object that is a child of *window* and that contains the pointer cursor.
5. A CLX:MASK32 value that indicates the state of buttons on the pointer device.
6. Integer X coordinate of the pointer cursor's location relative to the root of *window*.
7. Integer Y coordinate of the pointer cursor's location relative to the root of *window*.
8. The CLX:WINDOW object that is the root of *window*.

CLX:QUERY-TREE Function

CLX:QUERY-TREE Function

Returns the children, parent, and root of the specified window.

Format

CLX:QUERY-TREE *window* &**KEY** :**RESULT-TYPE**

Arguments

window

A CLX:WINDOW object.

:RESULT-TYPE

The LISP type of the first return value. The default is 'LIST.

Return Values

Three values:

- A sequence of CLX:WINDOW objects that contains the children of *window*.
- The CLX:WINDOW object that is the parent of *window*.
- The CLX:WINDOW object that is the root of *window*.

CLX:QUEUE-EVENT Function

Adds an event to the queue.

The event is put at the head of the queue if :APPEND-P is NIL, else the tail.
Additional arguments depend on the *event-key* argument and are as specified in
Section 12.5.

Format

CLX:QUEUE-EVENT *display* *event-key* &**REST** *args* &**KEY** :**APPEND-P**
&**ALLOW-OTHER-KEYS**

Arguments

display

A CLX:DISPLAY object.

event-key

A CLX:EVENT-KEY value.

CLX:QUEUE-EVENT Function

args

A lambda list of the arguments allowed for *event-key* (see Section 12.5).

:APPEND-P

A CLX:BOOLEAN value. If true, the event is placed at the tail of the queue.

Additional keyword arguments, corresponding to the valid arguments for the *event-key*, are allowed.

Return Value

Unspecified.

CLX:READ-BITMAP-FILE Function

Creates an image or a pixmap from a C include file in standard X11 format.

Format

CLX:READ-BITMAP-FILE pathname &KEY :PIXMAP-P :DRAWABLE

Arguments

pathname

A pathname or string that specifies the file to read.

:PIXMAP-P

A CLX:BOOLEAN value that determines whether the return value is a pixmap or an image.

:DRAWABLE

A CLX:DRAWABLE object. This argument is required if :PIXMAP-P is true.

Return Values

If :PIXMAP-P is false, the function returns a single CLX:IMAGE object. If :PIXMAP-P is true, the function returns three values:

- A CLX:PIXMAP object containing the data from the specified file.
- An INTEGER that specifies the X coordinate of a cursor's hot spot.
- An INTEGER that specifies the Y coordinate of a cursor's hot spot.

CLX:RECOLOR-CURSOR Function

CLX:RECOLOR-CURSOR Function

Changes the foreground and background colors of the specified cursor.

Format

CLX:RECOLOR-CURSOR *cursor foreground background*

Arguments

cursor

A CLX:CURSOR object.

foreground background

Two CLX:COLOR objects that specify the new colors.

Return Value

Unspecified.

CLX:RECT-SEQ Type Specifier

A CLX:REPEAT-SEQ of X and Y coordinates, width and height. Objects of type CLX:RECT-SEQ can be passed to the CLX:DRAW-RECTANGLES function.

Representation

(CLX:REPEAT-SEQ (FIXNUM *x*) (FIXNUM *y*)
(FIXNUM *width*) (FIXNUM *height*))

CLX:REMOVE-ACCESS-HOST Function

Deletes a machine from the access control list of the specified display.

Format

CLX:REMOVE-ACCESS-HOST *display host*

Arguments

display

A CLX:DISPLAY object.

CLX:REMOVE-ACCESS-HOST Function

host

A string that names a machine.

Return Value

Unspecified.

CLX:REMOVE-FROM-SAVE-SET Function

Removes the specified window from your client's saveset so that it is destroyed when its client exits. The window must have been created by some other client or an error occurs.

Format

CLX:REMOVE-FROM-SAVE-SET *window*

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:REARENT-WINDOW Function

Changes the parent and possibly location of the specified window.

Format

CLX:REARENT-WINDOW *window parent x y*

Arguments

window parent

Two CLX:WINDOW objects.

x y

Integer X and Y coordinates of the new origin of *window*, measured in pixels from the origin of *parent*.

CLX:REPARENT-WINDOW Function

Return Value

Unspecified.

CLX:REPEAT-SEQ Type Specifier

CLX:REPEAT-SEQ is an abstract type that denotes sequences composed of repeating patterns of specifically typed objects.

Representation

SEQUENCE

A sequence containing 0 or more elements of the form:

(*type name*)

CLX:RESET-SCREEN-SAVER Function

Resets the screen saver's timeout clock to 0, as if input had just been received.

Format

CLX:RESET-SCREEN-SAVER *display*

Argument

display

A CLX:DISPLAY object.

Return Value

Unspecified.

CLX:RESOURCE-ID Type Specifier

Objects of type CLX:RESOURCE-ID are returned by the CLX:*-ID functions.

Representation

INTEGER

CLX:RGB-VAL Type Specifier

Objects of type CLX:RGB-VAL are returned by the color allocation functions and can be used with SETF and the CLX:COLOR-RED, CLX:COLOR-GREEN, and CLX:COLOR-BLUE functions.

Representation

(FLOAT 0.0 1.0)

CLX:SCREEN Structure

This structure contains information on the hardware and software characteristics of a workstation screen.

Constructor Function

Instances of type CLX:SCREEN are returned by the CLX:DISPLAY-DEFAULT-SCREEN and CLX:DISPLAY-ROOTS functions.

Accessor Functions

None of these functions is a valid SETF place.

Name	Type Definition
CLX:SCREEN-BACKING-STORES	MEMBER :NEVER :WHEN-MAPPED :ALWAYS
CLX:SCREEN-BLACK-PIXEL	CLX:PIXEL
CLX:SCREEN-DEFAULT-COLORMAP	CLX:COLORMAP
CLX:SCREEN-DEPTHES	(ALIST (CLX:IMAGE-DEPTH depth) ((LIST CLX:VISUAL-INFO) visuals))
CLX:SCREEN-DISPLAY	CLX:DISPLAY
CLX:SCREEN-EVENT-MASK-AT-OPEN	CLX:EVENT-MASK
CLX:SCREEN-HEIGHT	INTEGER
CLX:SCREEN-HEIGHT-IN-MILLIMETERS	INTEGER
CLX:SCREEN-MAX-INSTALLED-MAPS	INTEGER
CLX:SCREEN-MIN-INSTALLED-MAPS	INTEGER
CLX:SCREEN-P	CLX:BOOLEAN
CLX:SCREEN-ROOT	CLX:WINDOW
CLX:SCREEN-ROOT-DEPTH	CLX:IMAGE-DEPTH
CLX:SCREEN-ROOT-VISUAL	CLX:VISUAL
CLX:SCREEN-SAVE-UNDERS-P	CLX:BOOLEAN
CLX:SCREEN-WHITE-PIXEL	CLX:PIXEL
CLX:SCREEN-WIDTH	INTEGER
CLX:SCREEN-WIDTH-IN-MILLIMETERS	INTEGER

CLX:SCREEN Structure

Deallocator Function

Instances of the CLX:SCREEN structure are subject to garbage collection.

CLX:SCREEN-BACKING-STORES Function

Returns a keyword that indicates when the server maintains contents of windows.

Format

CLX:SCREEN-BACKING-STORES *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

:NEVER, :WHEN-MAPPED, or :ALWAYS.

CLX:SCREEN-BLACK-PIXEL Function

Returns the pixel value that produces black on the specified screen.

Format

CLX:SCREEN-BLACK-PIXEL *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

A CLX:PIXEL value.

CLX:SCREEN-DEFAULT-COLORMAP Function

CLX:SCREEN-DEFAULT-COLORMAP Function

Returns the default colormap of the specified screen.

Format

CLX:SCREEN-DEFAULT-COLORMAP *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

A CLX:COLORMAP object.

CLX:SCREEN-DEPTHS Function

Returns the depths supported by the display hardware, with a list of the valid visual types supported at each depth.

Format

CLX:SCREEN-DEPTHS *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

An association list whose elements have the format:

(*depth* *visuals*)

where *depth* is a CLX:IMAGE-DEPTH value and *visuals* is a list of CLX:VISUAL-INFO objects.

CLX:SCREEN-DISPLAY Function

CLX:SCREEN-DISPLAY Function

Returns the display associated with the specified screen.

Format

CLX:SCREEN-DISPLAY *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

A CLX:DISPLAY object.

CLX:SCREEN-EVENT-MASK-AT-OPEN Function

Returns the event mask of the specified screen's root window, at the time the connection was established.

Format

CLX:SCREEN-EVENT-MASK-AT-OPEN *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

A CLX:EVENT-MASK value.

CLX:SCREEN-HEIGHT Function

Returns the height of the screen, measured in pixels.

Format

CLX:SCREEN-HEIGHT *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

An integer.

CLX:SCREEN-HEIGHT-IN-MILLIMETERS Function

Returns the height of the screen, measured in millimeters.

Format

CLX:SCREEN-HEIGHT-IN-MILLIMETERS *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

An integer.

CLX:SCREEN-MAX-INSTALLED-MAPS Function

Returns the maximum number of colormaps that can be installed on the specified screen.

CLX:SCREEN-MAX-INSTALLED-MAPS Function

Format

CLX:SCREEN-MAX-INSTALLED-MAPS *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

An integer.

CLX:SCREEN-MIN-INSTALLED-MAPS Function

Returns the number of colormaps that can be guaranteed to be installed on the specified screen regardless of the number of entries allocated in each map.

Format

CLX:SCREEN-MIN-INSTALLED-MAPS *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

An integer.

CLX:SCREEN-P Function

Tests whether its argument is a CLX:SCREEN object.

Format

CLX:SCREEN-P *object*

CLX:SCREEN-P Function

Argument

object

Any LISP object.

Return Value

T if *object* is of type CLX:SCREEN; otherwise, NIL.

CLX:SCREEN-ROOT Function

Returns the root window of the specified screen.

Format

CLX:SCREEN-ROOT *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

A CLX:WINDOW object.

CLX:SCREEN-ROOT-DEPTH Function

Returns the number of planes in the specified screen.

Format

CLX:SCREEN-ROOT-DEPTH *screen*

Argument

screen

A CLX:SCREEN object.

CLX:SCREEN-ROOT-DEPTH Function

Return Value

A CLX:IMAGE-DEPTH value.

CLX:SCREEN-ROOT-VISUAL Function

Returns the visual type of the specified screen.

Format

CLX:SCREEN-ROOT-VISUAL *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

A CLX:VISUAL value.

CLX:SCREEN-SAVER Function

Returns the current screen saver settings for the specified display.

Format

CLX:SCREEN-SAVER *display*

Argument

display

A CLX:DISPLAY object.

Return Values

Four values:

- | | |
|-----------------|---|
| <i>timeout</i> | Number of seconds that must elapse with no input from the keyboard or pointer before the screen saver turns on. A value of 0 means that the screen saver is disabled. |
| <i>interval</i> | Number of seconds between invocations of the screen saver. |

CLX:SCREEN-SAVER Function

<i>blanking</i>	One of three keywords specifying how the screen is cleared:
:YES	Blank the screen. This can be returned only if the display hardware supports video blanking.
:NO	Do not blank the screen. If exposures are allowed (see below), or if the screen can be regenerated without sending exposure events to clients, the screen is tiled with the root window background tile. If exposures are not allowed or the exposure events are sent to clients, the screen does not change.
:DEFAULT	The default blanking method is used.
<i>exposures</i>	One of three keywords specifying whether exposure events are generated:
:YES	Exposures are allowed. This is the default.
:NO	Exposures are not allowed.
:DEFAULT	The default value is used.

CLX:SCREEN-SAVE-UNDERS-P Function

Indicates whether the server saves the contents of windows obscured by client windows.

Format

CLX:SCREEN-SAVE-UNDERS-P *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

A CLX:BOOLEAN value.

CLX:SCREEN-WHITE-PIXEL Function

Returns the pixel value that produces white on the specified screen.

Format

CLX:SCREEN-WHITE-PIXEL *screen*

CLX:SCREEN-WHITE-PIXEL Function

Argument

screen

A CLX:SCREEN object.

Return Value

A CLX:PIXEL value.

CLX:SCREEN-WIDTH Function

Returns the width of the specified screen, measured in pixels.

Format

CLX:SCREEN-WIDTH *screen*

Argument

screen

A CLX:SCREEN object.

Return Value

An integer.

CLX:SCREEN-WIDTH-IN-MILLIMETERS Function

Returns the width of the specified screen, measured in millimeters.

Format

CLX:SCREEN-WIDTH-IN-MILLIMETERS *screen*

Argument

screen

A CLX:SCREEN object.

CLX:SCREEN-WIDTH-IN-MILLIMETERS Function

Return Value

An integer.

CLX:SEG-SEQ Type Specifier

A CLX:REPEAT-SEQUENCE of line segments. Objects of type CLX:SEG-SEQ can be passed to the CLX:DRAW-SEGMENTS function.

Representation

(CLX:REPEAT-SEQ (FIXNUM *x1*) (FIXNUM *y1*) (FIXNUM *x2*) (FIXNUM *y2*))

CLX:SELECTION-OWNER Function

Returns the current owner of the specified selection (global property).

Format

CLX:SELECTION-OWNER *display selection*

Arguments

display

A CLX:DISPLAY object.

selection

A CLX:XATOM value.

Return Value

A CLX:WINDOW object or NIL if no client owns the selection.

CLX:SEND-EVENT Function

Sends an event to a specified window. The function identifies the destination window, determines which client should receive the event, and ignores any active grabs. The contents of the event are specified by additional keyword arguments, and are not checked or altered by the X server except that the :SEND-EVENT component is set to T.

The *window* argument determines the destination window. A value of :POINTER-WINDOW specifies the window that contains the pointer cursor. A value of :INPUT-FOCUS specifies the window that has input focus.

CLX:SEND-EVENT Function

Normally, the event is sent to every client whose event mask for the destination window matches the *event-mask* argument. Under certain circumstances, the event may be sent to other clients, or no event may be sent:

- If *event-mask* is 0, the event is sent to the client that created the destination window. If that client no longer exists (its connection to the display has been closed), no event is sent.
- If :PROPAGATE-P is T but no client has selected any of the event types in *event-mask* for the destination window, the destination window is replaced with its closest ancestor for which some client has selected a matching event type (assuming any intervening windows propagate the event type). If no such ancestor exists, or if the window is an ancestor of the focus window and :INPUT-FOCUS was originally specified as the destination window, then the event is not sent to any client.

Format

CLX:SEND-EVENT *window* *event-key* *event-mask* &REST *args*
&KEY :PROPAGATE-P :DISPLAY &ALLOW-OTHER-KEYS

Arguments

window

A CLX:WINDOW object, :POINTER-WINDOW, or :INPUT-FOCUS.

event-key

A CLX:EVENT-KEY value or NIL.

event-mask

A CLX:EVENT-MASK value.

args

Keyword-value pairs appropriate to the *event-key*. See Section 12.5.

:PROPAGATE-P

A CLX:BOOLEAN value. If true, the event is propagated to other clients.

:DISPLAY

A CLX:DISPLAY object or NIL.

Additional keywords, corresponding to the arguments listed in Section 12.5, are allowed.

Return Value

Unspecified.

CLX:SET-INPUT-FOCUS Function

Sets the input focus to the specified window.

Format

CLX:SET-INPUT-FOCUS *display* *focus* *revert-to* &OPTIONAL *time*

Arguments

display

A CLX:DISPLAY object.

focus

The CLX:WINDOW object that gets input focus.

revert-to

A keyword that specifies the window input focus reverts to if the *focus* window is unviewable: :NONE, :PARENT, or :POINTER-ROOT.

time

The CLX:TIMESTAMP when input focus is set.

Return Value

Unspecified.

CLX:SET-MODIFIER-MAPPING Function

Changes the key codes of the keys (if any) that are used as modifiers. The function returns a keyword that indicates the status of the change.

Format

CLX:SET-MODIFIER-MAPPING *display* &KEY :SHIFT :LOCK :CONTROL :MOD1
:MOD2 :MOD3 :MOD4 :MOD5

Arguments

display

A CLX:DISPLAY object.

:SHIFT :LOCK :CONTROL :MOD1 :MOD2 :MOD3 :MOD4 :MOD5

Sequences of integers that specify the key codes of the modifier keys.

CLX:SET-MODIFIER-MAPPING Function

Return Value

A keyword:

:SUCCESS	The key code is changed.
:BUSY	A modifier whose key codes are being changed is logically down.
:FAILED	The specified key codes are not valid for the modifier.

CLX:SET-SCREEN-SAVER Function

Changes the settings of the screen saver on the specified display.

Format

CLX:SET-SCREEN-SAVER *display timeout interval blanking exposures*

Arguments

display

A CLX:DISPLAY object.

timeout interval

Integers that indicate time in seconds, or :DEFAULT.

blanking exposures

:YES, :NO, or :DEFAULT.

Return Value

Unspecified.

CLX:SET-WM-CLASS Function

Sets the class of a specified window. Note that the name set with this function may differ from the name set with the CLX:WM-NAME function. The *resource-name* is the formal name of the application that should be used when retrieving the application's resources from the resource database.

Format

CLX:SET-WM-CLASS *window resource-name resource-class*

CLX:SET-WM-CLASS Function

Arguments

window

A CLX:WINDOW object.

resource-name resource-class

Two CLX:STRINGABLE values or NIL.

Return Value

Unspecified.

CLX:STATE-MASK-KEY Type Specifier

Objects of type CLX:STATE-MASK-KEY are returned by the CLX:MAKE-STATE-KEYS function and can be passed to the CLX:MAKE-STATE-MASK function.

Representation

(OR CLX:MODIFIER-KEY (MEMBER :BUTTON-1 :BUTTON-2 :BUTTON-3 :BUTTON-4 :BUTTON-5))

CLX:STORE-COLOR Function

Stores an RGB value in the specified colormap. If the *spec* argument is of type CLX:STRINGABLE, a named color is stored.

Format

CLX:STORE-COLOR colormap pixel spec &KEY :RED-P :GREEN-P :BLUE-P

Arguments

colormap

A CLX:COLORMAP object.

pixel

A CLX:PIXEL value.

spec

A CLX:COLOR object or CLX:STRINGABLE value.

:RED-P :GREEN-P :BLUE-P

Three CLX:BOOLEAN values that specify whether the red, green, and blue values, respectively, are stored. The defaults are T.

CLX:STORE-COLOR Function

Return Value

Unspecified.

CLX:STORE-COLORS Function

Stores multiple RGB values in the specified colormap. If the *spec* argument uses CLX:STRINGABLE values, the names are first resolved and then a single StoreColors protocol request is issued.

Format

CLX:STORE-COLORS *colormap* *specs* &KEY :RED-P :GREEN-P :BLUE-P

Arguments

colormap

A CLX:COLORMAP object.

specs

A CLX:REPEAT-SEQ with elements in the format:

(*pixel* *color*)

where *pixel* is a CLX:PIXEL and *color* is either a CLX:COLOR object or a CLX:STRINGABLE value that specifies a named color.

:RED-P :GREEN-P :BLUE-P

Three CLX:BOOLEAN values that specify whether the red, green, and blue values, respectively, are stored. The defaults are T.

Return Value

Unspecified.

CLX:STRINGABLE Type Specifier

Either a string or a symbol. This abstract type is used by functions that need string arguments. Symbols are coerced to strings.

Representation

(OR STRING SYMBOL)

CLX:TEXT-EXTENTS Function

Returns the maximum character metrics of a specified string. The :TRANSLATE function is always called with a 16-bit destination buffer.

Format

CLX:TEXT-EXTENTS *fonts sequence &KEY :START :END :TRANSLATE*

Arguments

fonts

A CLX:FONT or a CLX:GCONTEXT object.

sequence

A sequence of characters (that is, a string).

:START

An integer. The default value is 0.

:END

An integer. The default value is the length of the *sequence*.

:TRANSLATE

A function that translates *sequence* into font indexes. The default function is CLX:TRANSLATE-DEFAULT.

Return Values

Nine values:

1. The total width of the specified string.
2. The largest ascent of any character in the string.
3. The largest descent of any character in the string.
4. The smallest left-bearing of any character in the string.
5. The largest right-bearing of any character in the string.
6. The font-ascent of the specified font.
7. The font-descent of the font.
8. The direction hint of the font, either 0 for left-to-right or 1 for right-to-left.
9. An array index value that indicates the position within the string where the translation failed; if the entire string was translated, this value is NIL.

CLX:TEXT-WIDTH Function

CLX:TEXT-WIDTH Function

Returns the width of the specified string, measured in pixels and calculated using the character information stored in the specified font.

Format

CLX:TEXT-WIDTH *fonts sequence &KEY :START :END :TRANSLATE*

Arguments

fonts

A CLX:FONT or a CLX:GCONTEXT object.

sequence

A sequence of text items; for example, a string.

:START

An index into *sequence*. The default is 0.

:END

An index into *sequence*. The default is the length of *sequence*.

:TRANSLATE

A function that translates *sequence* into font indexes. The default function is CLX:TRANSLATE-DEFAULT. If you provide a different function, it must deal with 16-bit buffers.

Return Values

Two values:

- An integer that gives the width of the sequence.
- A CLX:ARRAY-INDEX value that indicates the position in the sequence where the :TRANSLATE function failed; or NIL if the whole sequence was translated.

CLX:TIMESTAMP Type Specifier

Objects of type CLX:TIMESTAMP are used as arguments in the CLX:EVENT-CASE macro and CLX:PROCESS-EVENT function, and can be passed to the CLX:SEND-EVENT and CLX:GRAB-KEYBOARD functions. NIL stands for the current server time.

Representation

(OR NULL FIXNUM)

CLX:TRANSLATE-COORDINATES Function

CLX:TRANSLATE-COORDINATES Function

Translates X and Y coordinates in one window to those in another, and indicates which subwindow contains the coordinates. If the *src* and *dst* windows are not on the same screen, the values returned are 0 ; 0 ; NIL.

Format

CLX:TRANSLATE-COORDINATES *src* *src-x* *src-y* *dst*

Arguments

src

A CLX:WINDOW object.

src-x* *src-y

Integer X and Y coordinates of a location in the source window.

dst

A CLX:WINDOW object.

Return Values

Three values:

- The X coordinate in the destination window.
- The Y coordinate in the destination window.
- A CLX:WINDOW object if the translated coordinates are contained in a mapped child of the destination window; otherwise NIL.

CLX:TRANSLATE-DEFAULT Function

Translates as many elements of *src* as possible into indexes in the current font, and stores them into *dst*.

The *dst* argument is guaranteed to have room for (*src-end* *src-start*) integer elements, starting at *dst-start*; whether *dst* holds 8-bit or 16-bit elements depends on context. If known, *font* is the current font.

The first return value should be the *src* index of the first untranslated element. If no further elements need to be translated, the second return value should be NIL. If a horizontal motion is required before further translation, the second return value should be the delta in X coordinate. If known, the pixel width of the translated text can be returned as the third value; this can allow for appending of subsequent output to the same protocol request if no overall width has been specified at the higher level.

CLX:TRANSLATE-DEFAULT Function

Format

CLX:TRANSLATE-DEFAULT *src src-start src-end font dst dst-start*

Arguments

src

A sequence of text elements; for example, a string.

src-start src-end

Two CLX:ARRAY-INDEX values that specify the first and last positions within the source.

font

Null or a CLX:FONT object.

dst

A vector.

dst-start

A CLX:ARRAY-INDEX value that specifies the starting position within the destination.

Return Values

Three values:

- A CLX:ARRAY-INDEX value that indicates the first untranslated element in the source.
- NIL if the source was entirely translated.
- An INTEGER that indicates the width of the translated text in pixels if known; or NIL if not.

CLX:UNGRAB-BUTTON Function

Releases a passive grab of a button on the pointing device.

Format

CLX:UNGRAB-BUTTON *window button &KEY :MODIFIERS*

Arguments

window

A CLX:CREATE-WINDOW object.

button

An integer or :ANY.

CLX:UNGRAB-BUTTON Function

:MODIFIERS

A CLX:MODIFIER-MASK value. The default is 0.

Return Value

Unspecified.

CLX:UNGRAB-KEY Function

Releases a passive grab of a key on the keyboard.

Format

CLX:UNGRAB-KEY *window* *key &KEY* :**MODIFIERS**

Arguments

window

A CLX:WINDOW object.

key

An integer or :ANY.

:MODIFIERS

A CLX:MODIFIER-MASK value. The default is 0.

Return Value

Unspecified.

CLX:UNGRAB-KEYBOARD Function

Releases an active grab of the keyboard.

Format

CLX:UNGRAB-KEYBOARD *display &KEY* :**TIME**

Arguments

display

A CLX:DISPLAY object.

:TIME

The CLX:TIMESTAMP when the keyboard is released.

CLX:UNGRAB-KEYBOARD Function

Return Value

Unspecified.

CLX:UNGRAB-POINTER Function

Releases an active grab of the pointing device.

Format

CLX:UNGRAB-POINTER *display* &**KEY :TIME**

Arguments

display

A CLX:DISPLAY object.

:TIME

The CLX:TIMESTAMP when the pointer is released.

Return Value

Unspecified.

CLX:UNGRAB-SERVER Function

Releases an active server grab established by CLX:GRAB-SERVER. Events that occurred while the server was grabbed are released to the event queue.

Format

CLX:UNGRAB-SERVER *display*

Argument

display

A CLX:DISPLAY object.

Return Value

Unspecified.

CLX:UNINSTALL-COLORMAP Function

Removes color values from the hardware lookup table.

Format

CLX:UNINSTALL-COLORMAP *colormap*

Argument

colormap

A CLX:COLORMAP object.

Return Value

Unspecified.

CLX:UNMAP-SUBWINDOWS Function

Causes all children of the specified window to be unmapped or made not viewable on the screen. The windows are actually removed when the output buffer holding requests to the server is flushed (see CLX:DISPLAY-FORCE-OUTPUT).

Format

CLX:UNMAP-SUBWINDOWS *window*

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:UNMAP-WINDOW Function

CLX:UNMAP-WINDOW Function

Causes the specified window to be unmapped or made unviewable on the screen. The window is actually removed when the output buffer holding requests to the server is flushed (see CLX:DISPLAY-FORCE-OUTPUT).

Format

CLX:UNMAP-WINDOW *window*

Argument

window

A CLX:WINDOW object.

Return Value

Unspecified.

CLX:VISUAL Type Specifier

Values of type CLX:VISUAL are pointers to CLX:VISUAL-INFO objects. They are returned by the CLX:SCREEN-ROOT-VISUAL and CLX:WINDOW-VISUAL functions.

Representation

INTEGER

CLX:VISUAL-INFO Structure

This structure contains information on the hardware and software characteristics of a workstation screen.

Constructor Function

The CLX:MAKE-VISUAL-INFO function takes a CLX:VISUAL value and returns a CLX:VISUAL-INFO object.

Accessor Functions

None of the following functions is a valid SETF place:

CLX:VISUAL-INFO Structure

Name	Type Definition
CLX:VISUAL-INFO-BITS-PER-RGB	INTEGER
CLX:VISUAL-INFO-BLUE-MASK	CLX:PIXEL
CLX:VISUAL-INFO-CLASS	(MEMBER :STATIC-GRAY :STATIC-COLOR :TRUE-COLOR :GRAY-SCALE :PSEUDO-COLOR :DIRECT-COLOR)
CLX:VISUAL-INFO-COLORMAP-ENTRIES	INTEGER
CLX:VISUAL-INFO-GREEN-MASK	CLX:PIXEL
CLX:VISUAL-INFO-ID	INTEGER
CLX:VISUAL-INFO-RED-MASK	CLX:PIXEL

Deallocator Function

Instances of the CLX:VISUAL-INFO structure are subject to garbage collection.

Predicate

CLX:VISUAL-INFO-P

CLX:VISUAL-INFO-BITS-PER-RGB Function

Returns the number of bits used to store an RGB value.

Format

CLX:VISUAL-INFO-BITS-PER-RGB *visual-info*

Argument

visual-info

A CLX:VISUAL-INFO object.

Return Value

An integer.

CLX:VISUAL-INFO-BLUE-MASK Function

Returns the mask used for blue indexes.

CLX:VISUAL-INFO-BLUE-MASK Function

Format

CLX:VISUAL-INFO-BLUE-MASK *visual-info*

Argument

visual-info

A CLX:VISUAL-INFO object.

Return Value

A CLX:PIXEL value.

CLX:VISUAL-INFO-CLASS Function

Returns the visual class of the screen.

Format

CLX:VISUAL-INFO-CLASS *visual-info*

Argument

visual-info

A CLX:VISUAL-INFO object.

Return Value

:STATIC-GRAY, :STATIC-COLOR, :TRUE-COLOR, :GRAY-SCALE, :PSEUDO-COLOR, or
:DIRECT-COLOR.

CLX:VISUAL-INFO-COLORMAP-ENTRIES Function

Returns the size of the colormap used by the visual.

Format

CLX:VISUAL-INFO-COLORMAP-ENTRIES *visual-info*

Argument

visual-info

A CLX:VISUAL-INFO object.

CLX:VISUAL-INFO-COLORMAP-ENTRIES Function

Return Value

An integer.

CLX:VISUAL-INFO-GREEN-MASK Function

Returns the mask used for green indexes.

Format

CLX:VISUAL-INFO-GREEN-MASK *visual-info*

Argument

visual-info

A CLX:VISUAL-INFO object.

Return Value

A CLX:PIXEL value.

CLX:VISUAL-INFO-ID Function

Returns the integer resource-id of the specified visual-info.

Format

CLX:VISUAL-INFO-ID *visual-info*

Argument

visual-info

A CLX:VISUAL-INFO object.

Return Value

A CLX:RESOURCE-ID value.

CLX:VISUAL-INFO-P Function

CLX:VISUAL-INFO-P Function

Returns true if its argument is of type CLX:VISUAL-INFO.

Format

CLX:VISUAL-INFO-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:VISUAL-INFO-RED-MASK Function

Returns the mask used for red indexes.

Format

CLX:VISUAL-INFO-RED-MASK *visual-info*

Argument

visual-info

A CLX:VISUAL-INFO object.

Return Value

A CLX:PIXEL value.

CLX:WARP-POINTER Function

Moves the pointer cursor to the specified location in the destination window.

Format

CLX:WARP-POINTER *dst dst-x dst-y*

CLX:WARP-POINTER Function

Arguments

dst

A CLX:WINDOW object.

dst-x dst-y

Integer X and Y coordinates of the cursor's destination.

Return Value

Unspecified.

CLX:WARP-POINTER-IF-INSIDE Function

Moves the pointer cursor to the specified location in the destination window if the pointer cursor is within the specified rectangle of the source window.

Passing in a zero *src-width* or *src-height* is a no-op. If *src-width* is null, the current width of the source window minus *src-x* is used. If *src-height* is null, the current height of the source window minus *src-y* is used.

Format

CLX:WARP-POINTER-IF-INSIDE *dst dst-x dst-y src src-x src-y*
 &OPTIONAL *src-width src-height*

Arguments

dst

A CLX:WINDOW object.

dst-x dst-y

Integer X and Y coordinates of the cursor's destination.

src

A CLX:WINDOW object.

src-x src-y

Integer X and Y coordinates of the origin of a rectangle within the source window.

src-width src-height

Integer dimensions of a rectangle within the source window.

Return Value

Unspecified.

CLX:WARP-POINTER-RELATIVE Function

CLX:WARP-POINTER-RELATIVE Function

Moves the pointer cursor a specified distance relative to its current position.

Format

CLX:WARP-POINTER-RELATIVE *display x-off y-off*

Arguments

display

A CLX:DISPLAY object.

x-off y-off

Integer distances (offsets) to move the pointer cursor.

Return Value

Unspecified.

CLX:WARP-POINTER-RELATIVE-IF-INSIDE Function

Moves the pointer cursor a specified distance relative to its current position if the pointer cursor is within the specified rectangle of the source window.

Passing in a zero *src-width* or *src-height* is a no-op. If *src-width* is null, the current width of the source window minus *src-x* is used. If *src-height* is null, the current height of the source window minus *src-y* is used.

Format

CLX:WARP-POINTER-RELATIVE-IF-INSIDE *x-off y-off src src-x src-y &OPTIONAL src-width src-height*

Arguments

x-off y-off

Integer distances (offsets) to move the pointer cursor.

src

A CLX:WINDOW object.

src-x src-y

Integer X and Y coordinates of the origin of a rectangle in the source window.

CLX:WARP-POINTER-RELATIVE-IF-INSIDE Function

src-width src-height

Integer dimensions of a rectangle in the source window.

Return Value

Unspecified.

CLX:WINDOW Structure

The CLX:WINDOW structure contains the visible (size, color) and invisible (event masks) characteristics of a window. Not all the components of a window structure have an accessor function: Use the DRAWABLE-X, DRAWABLE-Y, DRAWABLE-WIDTH, DRAWABLE-HEIGHT, DRAWABLE-DEPTH, DRAWABLE-BORDER-WIDTH, and DRAWABLE-ROOT functions to access the corresponding attributes of a window. Use the CLX:QUERY-TREE function to find the parent of a window.

Windows have no foreground component. The foreground color of graphics objects is determined by the foreground component of the GContext used when they are drawn.

Constructor Function

CLX:CREATE-WINDOW.

Predicates

CLX:WINDOW-P and CLX:WINDOW-EQUAL.

Accessor Functions

Unless otherwise noted, the following functions can be used with SETF:

Name	Type Definition
CLX:WINDOW-ALL-EVENT-MASKS	CLX:MASK32 This function is not a valid SETF place.
CLX:WINDOW-BACKING-PIXEL	CLX:PIXEL
CLX:WINDOW-BACKING-PLANES	CLX:PIXEL
CLX:WINDOW-BACKING-STORE	(MEMBER :NOT-USEFUL :WHEN-MAPPED :ALWAYS)
CLX:WINDOW-BIT-GRAVITY	CLX:WIN-GRAVITY
CLX:WINDOW-CLASS	(MEMBER :INPUT-OUTPUT :INPUT-ONLY) This function is not a valid SETF place.
CLX:WINDOW-COLORMAP	(OR (MEMBER :COPY) CLX:COLORMAP)
CLX:WINDOW-COLORMAP-INSTALLED-P	CLX:BOOLEAN This function is not a valid SETF place.
CLX:WINDOW-DISPLAY	CLX:DISPLAY This function is not a valid SETF place.
CLX:WINDOW-DO-NOT-PROPAGATE-MASK	CLX:MASK32

CLX:WINDOW Structure

Name	Type Definition
CLX:WINDOW-EVENT-MASK	CLX:MASK32
CLX:WINDOW-GRAVITY	CLX:WIN-GRAVITY
CLX:WINDOW-ID	CLX:RESOURCE-ID
	This function is not a valid SETF place.
CLX:WINDOW-MAP-STATE	(MEMBER :UNMAPPED :UNVIEWABLE :VIEWABLE)
	This function is not a valid SETF place.
CLX:WINDOW-OVERRIDE-REDIRECT	(MEMBER :ON :OFF)
CLX:WINDOW-SAVE-UNDER	(MEMBER :ON :OFF)
CLX:WINDOW-VISUAL	CLX:VISUAL
	This function is not a valid SETF place.

SETF Forms

The following functions can be used with SETF but cannot be called:

Name	Description
CLX:WINDOW-BACKGROUND	Changes the background color of the specified window. This does not change the window's contents until it is cleared. (SETF (CLX:WINDOW-BACKGROUND <i>window</i>) <i>color</i>) <i>window</i> A CLX:WINDOW object. <i>color</i> NIL, :NONE, :PARENT-RELATIVE, a CLX:PIXEL value, or a CLX:PIXMAP object. If the background is :PARENT-RELATIVE, the origin of the window's background tile aligns with the origin of the parent's background tile.
CLX:WINDOW-BORDER	Changes the color of a window's border. (SETF (CLX:WINDOW-BORDER <i>window</i>) <i>color</i>) <i>window</i> A CLX:WINDOW object. <i>color</i> (OR NULL (MEMBER :COPY) CLX:PIXEL CLX:PIXMAP) Use a CLX:PIXEL value or CLX:PIXMAP object for the color of the window's border, or :COPY if the border was copied from the window's parent when it was created.
CLX:WINDOW-CURSOR	Changes the cursor used when the pointer is in the specified window. (SETF (CLX:WINDOW-CURSOR <i>window</i>) <i>cursor</i>) <i>window</i> A CLX:WINDOW object. <i>cursor</i> A CLX:CURSOR object or :NONE or NIL if no cursor shape is defined for the window. Changing the cursor of a root window to :NONE restores the default cursor for that window and all its subwindows.

CLX:WINDOW Structure

Name	Description									
CLX:WINDOW-PRIORITY	Changes the hierarchical position of a window in relation to all windows in the stack or to a specified sibling. (SETF (CLX:WINDOW-PRIORITY <i>window</i> [<i>sibling</i>]) <i>mode</i>)									
<i>window</i>	Two CLX:WINDOW objects.									
<i>sibling</i>										
<i>mode</i>	A keyword that names the stacking mode of the specified window. Possible values are: <table border="1"><thead><tr><th>Keyword</th><th>Relative to All Windows</th><th>Relative to Sibling</th></tr></thead><tbody><tr><td>:ABOVE</td><td>Top of stack.</td><td>Above <i>sibling</i>.</td></tr><tr><td>:BELOW</td><td>Bottom of stack.</td><td>Below <i>sibling</i>.</td></tr></tbody></table>	Keyword	Relative to All Windows	Relative to Sibling	:ABOVE	Top of stack.	Above <i>sibling</i> .	:BELOW	Bottom of stack.	Below <i>sibling</i> .
Keyword	Relative to All Windows	Relative to Sibling								
:ABOVE	Top of stack.	Above <i>sibling</i> .								
:BELOW	Bottom of stack.	Below <i>sibling</i> .								
:TOP-IF	If any sibling obscures <i>window</i> , the server places <i>window</i> on top of the stack.	If <i>sibling</i> obscures <i>window</i> , the server places <i>window</i> on top of the stack.								
:BOTTOM-IF	If <i>window</i> obscures any sibling, the server places <i>window</i> at the bottom of the stack.	If <i>window</i> obscures <i>sibling</i> , the server places <i>window</i> at the bottom of the stack.								
:OPPOSITE	If any sibling obscures <i>window</i> , the server pops <i>window</i> to the top of the stack. If <i>window</i> obscures any other window, the server pushes <i>window</i> to the bottom of the stack.	If <i>sibling</i> obscures <i>window</i> , the server pops <i>window</i> to the top of the stack. If <i>window</i> obscures <i>sibling</i> , the server pushes <i>window</i> to the bottom of the stack.								

Deallocator Function

CLX:DESTROY-WINDOW

CLX:WINDOW-ALL-EVENT-MASKS Function

Returns the inclusive-OR of all event masks selected on the specified window by clients.

Format

CLX:WINDOW-ALL-EVENT-MASKS *window*

CLX:WINDOW-ALL-EVENT-MASKS Function

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:MASK32 value.

CLX:WINDOW-BACKING-PIXEL Function

Indicates which pixel value to use in planes not covered by the backing-planes component.

This function can be used with SETF, but changing the backing-pixel of a mapped window may have no immediate effect.

Format

CLX:WINDOW-BACKING-PIXEL *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:PIXEL value or NIL if *window* uses the default backing-pixel of its parent.

CLX:WINDOW-BACKING-PLANES Function

Returns the planes of the specified window's contents that are kept in the backing store by the server. This function can be used with SETF.

Format

CLX:WINDOW-BACKING-PLANES *window*

Argument

window

A CLX:WINDOW object.

CLX:WINDOW-BACKING-PLANES Function

Return Value

A CLX:PIXEL value or NIL if maintaining contents is :NOT-USEFUL.

CLX:WINDOW-BACKING-STORE Function

Returns the status of the backing store for the specified window. This function can be used with SETF.

Format

CLX:WINDOW-BACKING-STORE *window*

Argument

window

A CLX:WINDOW object.

Return Value

:NOT-USEFUL, :WHEN-MAPPED, or :ALWAYS; or NIL if no backing store has been defined for *window*.

CLX:WINDOW-BIT-GRAVITY Function

Returns the bit gravity for the specified window, indicating which region of *window* is retained if it is resized. This function can be used with SETF.

Format

CLX:WINDOW-BIT-GRAVITY *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:BIT-GRAVITY value or NIL if no bit gravity is defined for *window*.

CLX:WINDOW-CLASS Function

CLX:WINDOW-CLASS Function

Returns the class of the specified window, that is, indicates whether the window can display output as well as receive input.

Format

CLX:WINDOW-CLASS *window*

Argument

window

A CLX:WINDOW object.

Return Value

:COPY, :INPUT-OUTPUT, or :INPUT-ONLY.

CLX:WINDOW-COLORMAP Function

Returns the colormap that reflects the "true" colors of the specified window.

This function can be used with SETF. The colormap must have the same visual type as the window or an error occurs. If :COPY is specified, the colormap is copied from the parent, but the window must have the same visual type as the parent and the parent must not have a colormap of :NONE or an error occurs.

Changing the colormap of a visible window may have no immediate effect on the screen; see CLX:INSTALL-COLORMAP. Also, changes to the parent's colormap do not affect the child, even if the window's colormap has previously been set to :COPY.

Format

CLX:WINDOW-COLORMAP *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:COLORMAP object, or :COPY if *window*'s colormap was copied from its parent when *window* was created.

CLX:WINDOW-COLORMAP-INSTALLED-P Function

CLX:WINDOW-COLORMAP-INSTALLED-P Function

Returns T if the specified window has a colormap installed.

Format

CLX:WINDOW-COLORMAP-INSTALLED-P *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:BOOLEAN value.

CLX:WINDOW-DISPLAY Function

Returns the display on which the specified window appears when it is mapped.
This function is not a valid SETF place.

Format

CLX:WINDOW-DISPLAY *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:DISPLAY object.

CLX:WINDOW-DO-NOT-PROPAGATE-MASK Function

CLX:WINDOW-DO-NOT-PROPAGATE-MASK Function

Returns the events that should not be propagated to ancestor windows when no client has selected the event type in the specified window. There is only one do-not-propagate mask per window, not one per client for each window. This function can be used with SETF.

Format

CLX:WINDOW-DO-NOT-PROPAGATE-MASK *window*

Argument

window

A CLX:WINDOW object.

Return Value

An integer or CLX:DEVICE-EVENT-MASK value; or NIL if no do-not-propagate mask has been defined for the specified window.

CLX:WINDOW-EQUAL Function

Returns true if its two arguments are the same CLX:WINDOW object.

Format

CLX:WINDOW-EQUAL *object-1 object-2*

Arguments

object-1 object-2

Any two LISP objects.

Return Value

A CLX:BOOLEAN value.

CLX:WINDOW-EVENT-MASK Function

CLX:WINDOW-EVENT-MASK Function

Returns the events selected by the client in the specified window. This function can be used with SETF.

Format

CLX:WINDOW-EVENT-MASK *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:EVENT-MASK value or NIL if no event mask is defined for *window*.

CLX:WINDOW-GRAVITY Function

Indicates how the specified window is repositioned after its parent is resized. This function can be used with SETF.

Format

CLX:WINDOW-GRAVITY *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:WIN-GRAVITY value or NIL if no window gravity is defined for *window*.

CLX:WINDOW-ID Function

CLX:WINDOW-ID Function

Returns the resource-id of the specified window. This function is not a valid SETF place.

Format

CLX:WINDOW-ID *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:RESOURCE-ID value.

CLX:WINDOW-MAP-STATE Function

Returns the map state of the specified window. This function is not a valid SETF place.

Format

CLX:WINDOW-MAP-STATE *window*

Argument

window

A CLX:WINDOW object.

Return Value

:UNMAPPED, :UNVIEWABLE, or :VIEWABLE.

CLX:WINDOW-OVERRIDE-REDIRECT Function

CLX:WINDOW-OVERRIDE-REDIRECT Function

Indicates whether map and configure requests on the specified window should override a SubstructureRedirect protocol request on its parent. Typically, this function is used to tell the window manager not to tamper with the window. This function can be used with SETF.

Format

CLX:WINDOW-OVERRIDE-REDIRECT *window*

Argument

window

A CLX:WINDOW object.

Return Value

:ON or :OFF.

CLX:WINDOW-P Function

Returns T if the argument is a CLX:WINDOW object.

Format

CLX:WINDOW-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN value.

CLX:WINDOW-SAVE-UNDER Function

CLX:WINDOW-SAVE-UNDER Function

Indicates whether the server saves the contents of windows obscured by the specified window. This function can be used with SETF.

Format

CLX:WINDOW-SAVE-UNDER *window*

Argument

window

A CLX:WINDOW object.

Return Value

:ON or :OFF.

CLX:WINDOW-VISUAL Function

Returns a pointer to the CLX:VISUAL-INFO object associated with the specified window.

Format

CLX:WINDOW-VISUAL *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:VISUAL value or :COPY if the visual info was copied from the parent when *window* was created.

CLX:WIN-GRAVITY Type Specifier

Gravity is the direction in which the children or contents of a window move when it is resized. Objects of type CLX:WIN-GRAVITY are returned by the CLX:WINDOW-BIT-GRAVITY and CLX:WINDOW-GRAVITY functions and can be passed to SETF forms of these functions.

Representation

```
(MEMBER :FORGET :STATIC :CENTER :NORTH :NORTH-EAST :EAST :SOUTH-EAST  
      :SOUTH :SOUTH-WEST :WEST :NORTH-WEST)
```

CLX:WITH-DISPLAY Macro

This macro is for use in a multiprocess environment. It provides exclusive access to the local display object for multiple request generation. It need not provide immediate exclusive access for replies; that is, if another process is waiting for a reply (while not in a CLX:WITH-DISPLAY), synchronization need not (but can) occur immediately.

Nested uses of this macro work correctly. This macro does not prevent concurrent event processing; see WITH-EVENT-QUEUE.

Format

CLX:WITH-DISPLAY *display* &**BODY** *body*

Arguments

display

A CLX:DISPLAY object.

body

LISP forms.

Return Value

Unspecified.

CLX:WITH-EVENT-QUEUE Macro

CLX:WITH-EVENT-QUEUE Macro

Grants exclusive access to event queue.

Format

CLX:WITH-EVENT-QUEUE *display &BODY body*

Arguments

display

A CLX:DISPLAY object.

body

LISP forms to be executed while the client has exclusive access to the event queue.

Return Value

Unspecified.

CLX:WITH-GCONTEXT Macro

Changes GCONTEXT components within the dynamic scope of the body (that is, indefinite scope and dynamic extent). The body is not surrounded by a WITH-DISPLAY. If :CACHE-P is NIL or some component states are unknown, this implements save/restore by creating a temporary GContext and doing CLX:COPY-GCONTEXT-COMPONENTS to and from it.

Format

```
CLX:WITH-GCONTEXT gcontext
  &KEY :FUNCTION :PLANE-MASK :FOREGROUND
        :BACKGROUND :LINE-WIDTH :LINE-STYLE
        :CAP-STYLE :JOIN-STYLE :FILL-STYLE
        :FILL-RULE :ARC-MODE :TILE :STIPPLE
        :TS-X :TS-Y :FONT :SUBWINDOW-MODE
        :EXPOSURES :CLIP-X :CLIP-Y :CLIP-MASK
        :CLIP-ORDERING :DASHES :DASH-OFFSET
  &BODY body
```

Arguments***gcontext***

A CLX:GCONTEXT object.

:FUNCTION

A CLX:LOGICAL-OP value.

:PLANE-MASK

Integer or NIL.

:FOREGROUND :BACKGROUND

Two CLX:PIXEL values or NIL.

:LINE-WIDTH

Integer or NIL.

:LINE-STYLE

NIL or :SOLID, :DASH, or :DOUBLE-DASH.

:CAP-STYLE

NIL or one of :NOT-LAST, :BUTT, :ROUND, or :PROJECTING.

:JOIN-STYLE

NIL or one of :MITER, :ROUND, or :BEVEL.

:FILL-STYLE

NIL or one of :SOLID, :TILED, :OPAQUE-STIPPLED, or :STIPPLED.

:FILL-RULE

NIL or either :EVEN-ODD or :WINDING.

:ARC-MODE

NIL or either :CHORD or :PIE-SLICE.

:TILE :STIPPLE

A CLX:PIXMAP object.

:TS-X :TS-Y

Integers.

CLX:WITH-GCONTEXT Macro

:FONT

A CLX:FONT object.

:SUBWINDOW-MODE

NIL or either :CLIP-BY-CHILDREN or :INCLUDE-INFERIORS.

:EXPOSURES

NIL or either :ON or :OFF.

:CLIP-X CLIP-Y

Integers.

:CLIP-MASK

NIL or :NONE, or either a CLX:PIXMAP or a CLX:RECT-SEQ.

:CLIP-ORDERING

NIL or one of :UNSORTED, :Y-SORTED, :YX-SORTED, or :YX-BANDED.

:DASHES

An integer or a sequence of integers.

:DASH-OFFSET

An integer.

body

LISP forms.

Return Value

Unspecified.

CLX:WITH-SERVER-GRABBED Macro

Invokes CLX:GRAB-SERVER before executing the *body*, and CLX:UNGRAB-SERVER afterward. The client therefore has exclusive possession of the server, on the specified display, while the *body* is executed.

The *body* is not surrounded by an implicit CLX:WITH-DISPLAY.

Format

CLX:WITH-SERVER-GRABBED *display &BODY body*

Arguments

display

A CLX:DISPLAY object.

body

LISP forms to be executed while the client has possession of the server.

CLX:WITH-SERVER-GRABBED Macro

Return Value

Unspecified.

CLX:WITH-STATE Macro

Batches multiple window queries into a single protocol GetWindowAttributes or GetGeometry request; and multiple window settings into a single protocol ChangeWindowAttributes or ConfigureWindow request.

Table CLX-1 lists the accessor functions and SETF places batched by the CLX:WITH-STATE macro. Within the indefinite scope of the body, the first call within an Accessor Group on the specified drawable causes the complete results of the protocol request to be retained, and returned in any subsequent accessor calls. Calls within a SETF Group are delayed, and are executed in a single request on exit from the body. In addition, if a call on a function within an Accessor Group follows a call on a function in the corresponding SETF Group, then all delayed SETFS for that group are executed; any retained accessor information for that group is discarded; the corresponding protocol request is (re)issued; and the results are (again) retained and returned in any subsequent accessor calls.

Table CLX-1: WITH-STATE Accessor and SETF Groups

Accessor Group A (for GetWindowAttributes)	
CLX:WINDOW-ALL-EVENT-MASKS	CLX:WINDOW-DO-NOT-PROPAGATE-MASK
CLX:WINDOW-BACKING-PIXEL	CLX:WINDOW-EVENT-MASK
CLX:WINDOW-BACKING-PLANES	CLX:WINDOW-GRAVITY
CLX:WINDOW-BACKING-STORE	CLX:WINDOW-MAP-STATE
CLX:WINDOW-BIT-GRAVITY	CLX:WINDOW-OVERRIDE-REDIRECT
CLX:WINDOW-CLASS	CLX:WINDOW-SAVE-UNDER
CLX:WINDOW-COLORMAP	CLX:WINDOW-VISUAL
CLX:WINDOW-COLORMAP-INSTALLED-P	

SETF Group A (for ChangeWindowAttributes)	
CLX:WINDOW-BACKING-PIXEL	CLX:WINDOW-DO-NOT-PROPAGATE-MASK
CLX:WINDOW-BACKING-PLANES	CLX:WINDOW-EVENT-MASK
CLX:WINDOW-BACKING-STORE	CLX:WINDOW-GRAVITY
CLX:WINDOW-BIT-GRAVITY	CLX:WINDOW-OVERRIDE-REDIRECT
CLX:WINDOW-COLORMAP	CLX:WINDOW-SAVE-UNDER
CLX:WINDOW-CURSOR	

Accessor Group G (for GetGeometry)	
CLX:DRAWABLE-BORDER-WIDTH	CLX:DRAWABLE-WIDTH
CLX:DRAWABLE-DEPTH	CLX:DRAWABLE-X
CLX:DRAWABLE-HEIGHT	CLX:DRAWABLE-Y
CLX:DRAWABLE-ROOT	

(continued on next page)

CLX:WITH-STATE Macro

Table CLX-1 (Cont.): WITH-STATE Accessor and SETF Groups

SETF Group G (for ConfigureWindow)	
CLX:DRAWABLE-BORDER-WIDTH	CLX:DRAWABLE-X
CLX:DRAWABLE-HEIGHT	CLX:DRAWABLE-Y
CLX:DRAWABLE-WIDTH	CLX:WINDOW-PRIORITY

The *body* is not surrounded by a CLX:WITH-DISPLAY.

Format

CLX:WITH-STATE *drawable* &**BODY** *body*

Arguments

drawable

A CLX:DRAWABLE object.

body

LISP forms.

Return Value

Unspecified.

CLX:WM-HINTS Function

Returns the window manager hints of the specified window.

Format

CLX:WM-HINTS *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:WM-HINTS object.

CLX:WM-HINTS Structure

The CLX:WM-HINTS structure contains the client's hints to the window manager as to a window's status and its icon. The window manager is not obliged to follow the client's hints.

Constructor Function

CLX:MAKE-WM-HINTS

Accessor Functions

All of the following functions are valid SETF places:

Name	Type Definition
CLX:WM-HINTS-FLAGS	CARD32
CLX:WM-HINTS-ICON-MASK	(OR NULL CLX:PIXMAP)
CLX:WM-HINTS-ICON-PIXMAP	(OR NULL CLX:PIXMAP)
CLX:WM-HINTS-ICON-WINDOW	(OR NULL CLX:WINDOW)
CLX:WM-HINTS-ICON-X	(OR NULL CARD16)
CLX:WM-HINTS-ICON-Y	(OR NULL CARD16)
CLX:WM-HINTS-INITIAL-STATE	(OR NULL (MEMBER :DONT-CARE :NORMAL :ZOOM :ICONIC :INACTIVE))
CLX:WM-HINTS-INPUT	(OR NULL (MEMBER :OFF :ON))
CLX:WM-HINTS-WINDOW-GROUP	(OR NULL CLX:RESOURCE-ID)

Deallocator Function

Instances of the CLX:WM-HINTS structure are subject to garbage collection.

CLX:WM-HINTS-FLAGS Function

Indicates which components of a CLX:WM-HINTS object have been defined. This function can be used with SETF.

Format

CLX:WM-HINTS-FLAGS *hints*

Argument

hints

A CLX:WM-HINTS object.

CLX:WM-HINTS-FLAGS Function

Return Value

An integer.

CLX:WM-HINTS-ICON-MASK Function

Indicates which pixels of the icon pixmap are used to create the icon associated with a window. This function can be used with SETF.

Format

CLX:WM-HINTS-ICON-MASK *hints*

Argument

hints

A CLX:WM-HINTS object.

Return Value

A CLX:PIXMAP object.

CLX:WM-HINTS-ICON-PIXMAP Function

Returns the pixmap used to create the icon associated with a window. This function can be used with SETF.

Format

CLX:WM-HINTS-ICON-PIXMAP *hints*

Argument

hints

A CLX:WM-HINTS object.

Return Value

A CLX:PIXMAP object.

CLX:WM-HINTS-ICON-WINDOW Function

CLX:WM-HINTS-ICON-WINDOW Function

Returns the window used for the icon associated with a window. This function can be used with SETF.

Format

CLX:WM-HINTS-ICON-WINDOW *hints*

Argument

hints

A CLX:WM-HINTS object.

Return Value

A CLX:WINDOW object.

CLX:WM-HINTS-ICON-X Function

Returns the X coordinate of the icon associated with a window. This function can be used with SETF.

Format

CLX:WM-HINTS-ICON-X *hints*

Argument

hints

A CLX:WM-HINTS object.

Return Value

An integer.

CLX:WM-HINTS-ICON-Y Function

CLX:WM-HINTS-ICON-Y Function

Returns the Y coordinate of the icon associated with a window. This function can be used with SETF.

Format

CLX:WM-HINTS-ICON-Y *hints*

Argument

hints

A CLX:WM-HINTS object.

Return Value

An integer.

CLX:WM-HINTS-INITIAL-STATE Function

Indicates how the window should appear in its initial state. This function can be used with SETF.

Format

CLX:WM-HINTS-INITIAL-STATE *hints*

Argument

hints

A CLX:WM-HINTS object.

Return Value

A keyword:

- :DONT-CARE Client is not interested in the initial state.
- :NORMAL Initial state used most often.
- :ZOOM Window starts zoomed.
- :ICONIC Window starts as an icon.
- :INACTIVE Window is seldom used.

CLX:WM-HINTS-INPUT Function

Indicates whether the client relies on the window manager to get keyboard input. This function can be used with SETF.

Format

CLX:WM-HINTS-INPUT *hints*

Argument

hints

A CLX:WM-HINTS object.

Return Value

Returns :ON if the client relies on the window manager to get keyboard input, :OFF if it does not.

CLX:WM-HINTS-P Function

Returns true if its argument is a CLX:WM-HINTS object.

Format

CLX:WM-HINTS-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN object.

CLX:WM-HINTS-WINDOW-GROUP Function

CLX:WM-HINTS-WINDOW-GROUP Function

Indicates whether a window belongs to a group. This function can be used with SETF.

Format

CLX:WM-HINTS-WINDOW-GROUP *hints*

Argument

hints

A CLX:WM-HINTS object.

Return Value

Returns the CLX:RESOURCE-ID of the group to which a window belongs.

CLX:WM-ICON-NAME Function

Returns the title of the icon associated with the specified window. This function can be used with SETF.

Format

CLX:WM-ICON-NAME *window*

Argument

window

A CLX:WINDOW object.

Return Value

A string containing the name in the window's icon.

CLX:WM-NAME Function

Returns the title of the specified window. This function can be used with SETF.

Format

CLX:WM-NAME *window*

Argument

window

A CLX:WINDOW object.

Return Value

A string containing the name in the window's title bar.

CLX:WM-NORMAL-HINTS Function

Returns the suggested values for the size and location for a window in the normal state.

Format

CLX:WM-NORMAL-HINTS *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:WM-SIZE-HINTS object.

CLX:WM-SIZE-HINTS Structure

CLX:WM-SIZE-HINTS Structure

The CLX:WM-SIZE-HINTS structure contains the client's suggestions to the window manager as to how a window should be sized. The window manager is not guaranteed to follow the client's suggestions.

Constructor Function

CLX:MAKE-WM-SIZE-HINTS

Accessor Functions

All of the following functions are valid SETF places. After setting values in a CLX:WM-SIZE-HINTS object you can use it as the value to SETF the CLX:WM-NORMAL-HINTS or CLX:WM-ZOOM-HINTS function.

Name	Type Definition
CLX:WM-SIZE-HINTS-HEIGHT	(OR NULL CARD16)
CLX:WM-SIZE-HINTS-HEIGHT-INC	(OR NULL CARD16)
CLX:WM-SIZE-HINTS-MAX-ASPECT	(OR NULL NUMBER)
CLX:WM-SIZE-HINTS-MAX-HEIGHT	(OR NULL CARD16)
CLX:WM-SIZE-HINTS-MAX-WIDTH	(OR NULL CARD16)
CLX:WM-SIZE-HINTS-MIN-ASPECT	(OR NULL NUMBER)
CLX:WM-SIZE-HINTS-MIN-HEIGHT	(OR NULL CARD16)
CLX:WM-SIZE-HINTS-MIN-WIDTH	(OR NULL CARD16)
CLX:WM-SIZE-HINTS-USER-SPECIFIED-POSITION-P	CLX:BOOLEAN
CLX:WM-SIZE-HINTS-USER-SPECIFIED-SIZE-P	CLX:BOOLEAN
CLX:WM-SIZE-HINTS-WIDTH	(OR NULL CARD16)
CLX:WM-SIZE-HINTS-WIDTH-INC	(OR NULL CARD16)
CLX:WM-SIZE-HINTS-X	(OR NULL INTEGER)
CLX:WM-SIZE-HINTS-Y	(OR NULL INTEGER)

Deallocator Function

Instances of the CLX:WM-SIZE-HINTS structure are subject to garbage collection.

CLX:WM-SIZE-HINTS-HEIGHT Function

Indicates the width of the window. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-HEIGHT *size-hints*

CLX:WM-SIZE-HINTS-HEIGHT Function

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-SIZE-HINTS-HEIGHT-INC Function

Indicates the increment by which the height of a window may be resized. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-HEIGHT-INC *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-SIZE-HINTS-MAX-ASPECT Function

Indicates the largest aspect ratio of a window. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-MAX-ASPECT *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

CLX:WM-SIZE-HINTS-MAX-ASPECT Function

Return Value

A number.

CLX:WM-SIZE-HINTS-MAX-HEIGHT Function

Indicates the largest useful height of a window. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-MAX-HEIGHT *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-SIZE-HINTS-MAX-WIDTH Function

Indicates the largest useful width of a window. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-MAX-WIDTH *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-SIZE-HINTS-MIN-ASPECT Function

CLX:WM-SIZE-HINTS-MIN-ASPECT Function

Indicates the smallest aspect ratio of a window. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-MIN-ASPECT *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

A number.

CLX:WM-SIZE-HINTS-MIN-HEIGHT Function

Indicates the smallest useful height of a window. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-MIN-HEIGHT *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-SIZE-HINTS-MIN-WIDTH Function

CLX:WM-SIZE-HINTS-MIN-WIDTH Function

Indicates the smallest useful width of a window. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-MIN-WIDTH *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-SIZE-HINTS-P Function

Returns true if its argument is a CLX:WM-SIZE-HINTS object.

Format

CLX:WM-SIZE-HINTS-P *object*

Argument

object

Any LISP object.

Return Value

A CLX:BOOLEAN object.

CLX:WM-SIZE-HINTS-USER-SPECIFIED-POSITION-P Function

CLX:WM-SIZE-HINTS-USER-SPECIFIED-POSITION-P Function

Indicates whether the user wants to move the window to the X and Y location specified in *size-hints*. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-USER-SPECIFIED-POSITION-P *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

T if the user has moved the window, NIL if not.

CLX:WM-SIZE-HINTS-USER-SPECIFIED-SIZE-P Function

Indicates whether the user has resized the window. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-USER-SPECIFIED-SIZE-P *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

T if the user has resized the window, NIL if not.

CLX:WM-SIZE-HINTS-WIDTH Function

CLX:WM-SIZE-HINTS-WIDTH Function

Indicates the width of a window. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-WIDTH *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-SIZE-HINTS-WIDTH-INC Function

Indicates the increment by which the width of a window may be resized. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-WIDTH-INC *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-SIZE-HINTS-X Function

Indicates the X coordinate of the window's top-left corner. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-X *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-SIZE-HINTS-Y Function

Indicates the Y coordinate of the window's top-left corner. This function can be used with SETF.

Format

CLX:WM-SIZE-HINTS-Y *size-hints*

Argument

size-hints

A CLX:WM-SIZE-HINTS object.

Return Value

An integer.

CLX:WM-ZOOM-HINTS Function

CLX:WM-ZOOM-HINTS Function

Returns the suggested size and location of a window in the zoomed state. This function can be used with SETF.

Format

CLX:WM-ZOOM-HINTS *window*

Argument

window

A CLX:WINDOW object.

Return Value

A CLX:WM-SIZE-HINTS object.

CLX:WRITE-BITMAP-FILE Function

Writes an image or a pixmap to a C include file in standard X11 format. The *name* argument is used for variable prefixes and defaults to "IMAGE-".

Format

CLX:WRITE-BITMAP-FILE *pathname image*
 &OPTIONAL *name*
 &KEY :PIXMAP-P :WIDTH :HEIGHT :X-HOT :Y-HOT
 :DRAWABLE

Arguments

pathname

The pathname of the file to write.

image

A CLX:IMAGE object.

name

A CLX:STRINGABLE value.

:PIXMAP-P

A CLX:BOOLEAN value that specifies whether the *image* argument is a pixmap or an integer.

CLX:WRITE-BITMAP-FILE Function

:WIDTH :HEIGHT

Integers that specify the dimensions of *image*.

:X-HOT :Y-HOT

Integers that specify the X and Y coordinates of a cursor's hot spot. The defaults are 0.

:DRAWABLE

A CLX:DRAWABLE object that provides information about the screen and display.

Return Value

Unspecified.

CLX:XATOM Type Specifier

Atoms are names of properties. Objects of type CLX:XATOM can be passed to the CLX:CHANGE-PROPERTY, CLX:GET-PROPERTY, CLX:LIST-PROPERTIES, and CLX:ROTATE-PROPERTIES functions.

Representation

(OR STRING SYMBOL)

RECORDED IN THE OFFICE OF THE CLERK

CLERK OF THE COURT OF APPEALS

FOR THE STATE OF WASHINGTON

IN THE MATTER OF THE APPOINTMENT OF

THE ATTORNEY FOR THE DEFENDANT

IN THE CRIMINAL CASE

OF RICHARD ROBERTSON

APPOINTED ATTORNEY

FOR THE DEFENDANT

IN THE CRIMINAL CASE

OF RICHARD ROBERTSON

Appendix A

DECwindows Constants

VAX LISP defines the DECwindows constants listed in Table A-1.

Table A-1: DECwindows Constants Defined by VAX LISP

Name	Value
Alignments	
DWT:ALIGNMENT-CENTER	1
DWT:ALIGNMENT-BEGINNING	2
DWT:ALIGNMENT-END	3
Label Type	
DWT:TEXT	1
DWT:PIXMAP	2
DWT:TEXT16	3
DWT:TEXT-WITH-ACCELERATOR	4
DWT:TEXT16-WITH-ACCELERATOR	5
Toggle Button Shapes	
DWT:RECTANGULAR	1
DWT:OVAL	2
Orientations	
DWT:ORIENTATION-VERTICAL	1
DWT:ORIENTATION-HORIZONTAL	2
Dialog Styles	
DWT:MODAL	1
DWT:MODELESS	2
DWT:WORKAREA	3

(continued on next page)

Table A-1 (Cont.): DECwindows Constants Defined by VAX LISP

Name	Value
Menu Styles	
DWT:MENU-BAR-TYPE	0
DWT:MENU-WORK-AREA	1
DWT:MENU-PULL-DOWN	2
DWT:MENU-POPUP	3
DWT:MENU-OPTION	5
Menu-Packing Styles	
DWT:MENU-PACKING-TIGHT	0
DWT:MENU-PACKING-COLUMN	1
DWT:MENU-PACKING-NONE	2
File Selection	
DWT:VMS	1
DWT:UNIX	2
DWT:MSDOS	3
Caution Box	
DWT:YES-BUTTON	0
DWT:NO-BUTTON	1
DWT:CANCEL-BUTTON	2
Font Unit Positioning	
DWT:FONT-UNITS	0
DWT:PIXEL-UNITS	0
Cut and Paste	
DWT:CLIPBOARD-SUCCESS	1
DWT:CLIPBOARD-TRUNCATE	2
DWT:CLIPBOARD-FAIL	3
DWT:CLIPBOARD-LOCKED	4
DWT:CLIPBOARD-NON-DATA	5

(continued on next page)

Table A-1 (Cont.): DECwindows Constants Defined by VAX LISP

Name	Value
All Conversion Strings	
DWT:C-NULL	0
DWT:C-FALSE	0
DWT:C-TRUE	1
DWT:NO-CALLBACKLIST	1
DWT:NO-CALLBACKS	2
DWT:YES-CALLBACKS	3
DWT:GEOMETRY-YES	1
DWT:GEOMETRY-NO	2
DWT:GEOMETRY-ALMOST	3
Event Management	
DWT:C-PASS	1
DWT:C-IGNORE	2
DWT:C-REMAP	3
DWT:C-IS-SENSITIVE	1
DWT:C-NOT-SENSITIVE	0
Event-Gathering Routines	
DWT:C-INPUT-NONE-MASK	1
DWT:C-INPUT-READ-MASK	2
DWT:C-INPUT-WRITE-MASK	4
DWT:C-INPUT-EXCEPT-MASK	8
Geometry Management	
DWT:CSM-DONT-CHANGE	5
Pop-up Handling	
DWT:GRAB-NONE	1
DWT:GRAB-NONEXCLUSIVE	2
DWT:GRAB-EXCLUSIVE	3
DWT:OVERRIDE	1
DWT:AUGMENT	2

Table A-2: DRM Constants

Name	Value	Description
DWT:DRM-SUCCESS	1	Operation completed without error
DWT:DRM-CREATENEW	3	New {file} created

(continued on next page)

Table A-2 (Cont.): DRM Constants

Name	Value	Description
DWT:DRM-INDEXRETRY	5	Retry on entering index required
DWT:DRM-INDEXGT	7	Index orders greater-than entry
DWT:DRM-INDEXLT	9	Index orders less-than entry
DWT:DRM-FAILURE	0	Operation not successful
DWT:DRM-NOTFOUND	2	{File, Record, etc.} not found
DWT:DRM-EXISTS	4	{File, Record, etc.} does not exist
DWT:DRM-NULGROUP	6	Null group field
DWT:DRM-NULTYPE	8	Null type field
DWT:DRM-WRONGGROUP	10	Wrong group field
DWT:DRM-WRONGTYPE	12	Wrong type field
DWT:DRM-OUTOFRANGE	14	Out of range
DWT:DRM-BADRECORD	16	Record number wrong type
DWT:DRM-NULLDATA	18	No data for entry
DWT:DRM-BADDATAINDEX	20	Data index in RID out of range
DWT:DRM-BADORDER	22	Bad ordering specifier
DWT:DRM-BADCONTEXT	24	Invalid DRM context
DWT:DRM-NOTVALID	26	Validation failure
DWT:DRM-BADBTREE	28	GT/LT pointer error in BTree
DWT:DRM-BADWIDGETREC	30	Validation failure on widget record
DWT:DRM-BADCLASSTYPE	32	Class type not a valid DRMwc... value
DWT:DRM-NOCLASSNAME	34	User class name is null
DWT:DRM-TOOMANY	36	Too many entries requested in some list
DWT:DRM-BADIFMODULE	38	Invalid interface module
DWT:DRM-NULLDESC	40	Arglist or children descriptor null
DWT:DRM-OUTOFCOMMITS	42	Argument index out of arglist bounds
DWT:DRM-BADCOMPRESS	44	Invalid compression code
DWT:DRM-BADARGTYPE	46	Invalid type, not in RGMrType...
DWT:DRM-NOTIMP	48	Not yet implemented
DWT:DRM-NULLINDEX	50	Empty index string
DWT:DRM-BADKEYTYPE	52	Key must be DRMrIndex or DRMrRID
DWT:DRM-BADCALLBACK	54	Invalid callback descriptor
DWT:DRM-NULLROUTINE	56	Empty callback routine name string
DWT:DRM-VECTOOBIG	58	Too many elements in vector
DWT:DRM-BADHIERARCHY	60	Invalid DRM file hierarchy
DWT:DRM-BADCLASSCODE	62	Class code not found in DRMwc...

(continued on next page)

Table A-2 (Cont.): DRM Constants

Name	Value	Description
DWT:DRM-BADMSGTYPE	64	Message type not in RGMmt...
DWT:DRM-BADMESSAGE	66	Message record invalid
DWT:DRM-OPEN-PARAM-LEN	12	Length of OPEN-PARAM structure
DWT:DRM-OPEN-PARAM-VERSION	1	Current version of this record is 1

Table A-3: Callback Reasons

Name	Value
DWT:CRUNIT-INC	0
DWT:CRUNIT-DEC	1
DWT:CRPAGE-INC	2
DWT:CRPAGE-DEC	3
DWT:CRSCROLL-START	4
DWT:CRDRAG	5
DWT:CRSCROLL-END	6
DWT:CRTO-TOP	7
DWT:CRTO-BOTTOM	8
DWT:CRCANCEL	9
DWT:CRACTIVATE	10
DWT:CRARM	11
DWT:CRDISARM	12
DWT:CRVALUE-CHANGE	13
DWT:CRNO	14
DWT:CRTO-POSITION	15
DWT:CRMAP	16
DWT:CRUNMAP	17
DWT:CRPULL-DOWN	18
DWT:CFOCUS	19
DWT:SINGLE	20
DWT:SINGLE-CONFIRM	21
DWT:EXTEND	22
DWT:EXTEND-CONFIRM	23
DWT:CRCOMMAND-ENTERED	24
DWT:CRCREATE	25
DWT:CRHELP-REQUESTED	26
DWT:CRHELP	26
DWT:CRSELECTION	27
DWT:CRCLIPBOARD-DATA-REQUEST	28
DWT:CRCLIPBOARD-DATA-DELETE	29

(continued on next page)

Table A-3 (Cont.): Callback Reasons

Name	Value
DWT:CRYES	30
DWT:CREXPOSE	31

Appendix B

CLX to X11 Mappings

The following tables show the mappings between CLX functions and X11 requests. Table B-1 lists the X11 protocol requests in alphabetical order on the left, with the corresponding CLX function, macro, or SETF form on the right. Table B-2 lists the CLX functions and macros in alphabetical order on the left, with the corresponding X11 request, if any, on the right. The Xlib functions that match the protocol requests are in the middle column in both tables. Due to space limitations, the package prefix `CLX:` is not shown in these tables.

Table B–1: Mapping X11 Requests to CLX Functions

Protocol Request	Xlib Function(s)	CLX Function(s) or Equivalent
AllocColor	XAllocColor	ALLOC-COLOR
AllocColorCells	XAllocColorCells	ALLOC-COLOR-CELLS
AllocColorPlanes	XAllocColorPlanes	ALLOC-COLOR-PLANES
AllocNamedColor	XAllocNamedColor	ALLOC-COLOR
AllowEvents	XAllowEvents	ALLOW-EVENTS
Bell	XBell	BELL
ChangeActivePointerGrab	XChangeActivePointerGrab	CHANGE-ACTIVE-POINTER-GRAB
ChangeGC	XChangeGC XSetArcMode XSetBackground XSetClipMask XSetClipOrigin XSetGraphicsExposures XSetFillRule XSetFillStyle XSetFont XSetForeground XSetFunction XSetLineAttributes XSetPlaneMask XSetState XSetStipple XSetSubwindowMode XSetTile XSetTSSource	(setf (gcontext-arc-mode <i>gc</i>) <i>mode</i>) (setf (gcontext-background <i>gc</i>) <i>integer</i>) (setf (gcontext-cap-style <i>gc</i>) <i>keyword</i>) (setf (gcontext-clip-mask <i>gc</i> [<i>ordering</i>]) <i>mask</i>)† (setf (gcontext-clip-ordering <i>gc</i>) <i>keyword</i>) (setf (gcontext-clip-x <i>gc</i>) <i>integer</i>) (setf (gcontext-clip-y <i>gc</i>) <i>integer</i>) (setf (gcontext-dash-offset <i>gc</i>) <i>integer</i>) (setf (gcontext-dashes <i>gc</i>) <i>dash-list</i>) (setf (gcontext-exposures <i>gc</i>) <i>state</i>) (setf (gcontext-fill-rule <i>gc</i>) <i>keyword</i>) (setf (gcontext-fill-style <i>gc</i>) <i>keyword</i>) (setf (gcontext-font <i>gc</i> [<i>metrics-pj</i>]) <i>font</i>)† (setf (gcontext-foreground <i>gc</i>) <i>integer</i>) (setf (gcontext-function <i>gc</i>) <i>logical-op</i>) (setf (gcontext-join-style <i>gc</i>) <i>keyword</i>) (setf (gcontext-line-style <i>gc</i>) <i>keyword</i>) (setf (gcontext-line-width <i>gc</i>) <i>integer</i>) (setf (gcontext-plane-mask <i>gc</i>) <i>integer</i>) (setf (gcontext-stipple <i>gc</i>) <i> pixmap</i>) (setf (gcontext-subwindow-mode <i>gc</i>) <i>keyword</i>) (setf (gcontext-tile <i>gc</i>) <i> pixmap</i>) (setf (gcontext-ts-x <i>gc</i>) <i>integer</i>) (setf (gcontext-ts-y <i>gc</i>) <i>integer</i>) <i>See also</i> FORCE-GCONTEXT-CHANGES, WITH-GCONTEXT
ChangeHosts	XAddHost XAddHosts XRemoveHost XRemoveHosts	ADD-ACCESS-HOST REMOVE-ACCESS-HOST
ChangeKeyboardControl	XChangeKeyboardControl XAutoRepeatOff XAutoRepeatOn	CHANGE-KEYBOARD-CONTROL
ChangeKeyboardMapping	XChangeKeyboardMapping	CHANGE-KEYBOARD-MAPPING
ChangePointerControl	XChangePointerControl	CHANGE-POINTER-CONTROL

†Note that [brackets] represent optional arguments.

(continued on next page)

Table B-1 (Cont.): Mapping X11 Requests to CLX Functions

Protocol Request	Xlib Function(s)	CLX Function(s) or Equivalent
ChangeProperty	XChangeProperty XSetCommand XSetIconName XSetIconSizes XSetNormalHints XSetSizeHints XSetStandardProperties XSetWMHints XSetZoomHints XStoreBuffer XStoreBytes XStoreName	CHANGE-PROPERTY (setf (wm-icon-name <i>window</i>) <i>string</i>) (setf (icon-sizes <i>window</i>) <i>size-hints</i>) (setf (wm-normal-hints <i>window</i>) <i>size-hints</i>) (setf (wm-size-hints-xxx <i>window</i>) <i>value</i>)‡ (setf (wm-hints-xxx <i>window</i>) <i>value</i>)‡ (setf (wm-zoom-hints <i>window</i>) <i>size-hints</i>)
ChangeSaveSet	XChangeSaveSet XAddToSaveSet XRemoveFromSaveSet	ADD-TO-SAVE-SET REMOVE-FROM-SAVE-SET
ChangeWindowAttributes	XChangeWindowAttributes XSetWindowBackground XSetWindowBackgroundPixmap XSetWindowBorder XSetWindowBorderPixmap XSetWindowColormap XDefineCursor XUndefineCursor XSelectInput	See also WITH-STATE (setf (window-background <i>window</i>) <i>pixel</i>) (setf (window-backing-pixel <i>window</i>) <i>pixel</i>) (setf (window-backing-planes <i>window</i>) <i>pixel</i>) (setf (window-backing-store <i>window</i>) <i>keyword</i>) (setf (window-bit-gravity <i>window</i>) <i>gravity</i>) (setf (window-border <i>window</i>) <i>pixel</i>) (setf (window-colormap <i>window</i>) <i>colormap</i>) (setf (window-cursor <i>window</i>) <i>cursor</i>) (setf (window-cursor <i>window</i>) :none) (setf (window-do-not-propagate-mask <i>w</i>) <i>mask</i>) (setf (window-event-mask <i>window</i>) <i>mask</i>) (setf (window-gravity <i>window</i>) <i>gravity</i>) (setf (window-override-redirect <i>window</i>) <i>keyword</i>) (setf (window-save-under <i>window</i>) <i>keyword</i>) CIRCULATE-WINDOW-DOWN CIRCULATE-WINDOW-UP
CirculateWindow	XCirculateSubwindows XCirculateSubwindowsDown XCirculateSubwindowsUp XLowerWindow XRaiseWindow	
ClearArea	XClearArea XClearWindow	CLEAR-AREA
CloseFont	XFreeFont XUnloadFont	CLOSE-FONT

‡See Table B-2, Table 6-3, or Part IV for a complete list of accessor functions for CLX:WM-SIZE-HINTS objects.

(continued on next page)

Table B-1 (Cont.): Mapping X11 Requests to CLX Functions

Protocol Request	Xlib Function(s)	CLX Function(s) or Equivalent
ConfigureWindow	XConfigureWindow XLowerWindow XMapRaised XMoveResizeWindow XMoveWindow XRaiseWindow XResizeWindow XRestackWindow XSetWindowBorderWidth	<i>See also</i> WITH-STATE (setf (drawable-border-width <i>drawable</i>) <i>integer</i>) (setf (drawable-depth <i>drawable</i>) <i>integer</i>) (setf (drawable-height <i>drawable</i>) <i>integer</i>) (setf (drawable-width <i>drawable</i>) <i>integer</i>) (setf (drawable-x <i>drawable</i>) <i>integer</i>) (setf (drawable-y <i>drawable</i>) <i>integer</i>) (setf (window-priority <i>window</i> [<i>sibling</i>]) <i>integer</i>)†
ConvertSelection	XConvertSelection	CONVERT-SELECTION
CopyArea	XCopyArea	COPY-AREA
CopyColormapAndFree	XCopyColormapAndFree	COPY-COLORMAP-AND-FREE
CopyGC	XCopyGC	COPY-GCONTEXT COPY-GCONTEXT-COMPONENTS
CopyPlane	XCopyPlane	COPY-PLANE
CreateColormap	XCreateColormap	CREATE-COLORMAP
CreateCursor	XCreatePixmapCursor	CREATE-CURSOR
CreateGC	XCreateGC	CREATE-GCONTEXT
	XCreateBitmapFromData XCreatePixmapFromData XReadBitmapFile	READ-BITMAP-FILE
CreateGlyphCursor	XCreateGlyphCursor XCreateFontCursor	CREATE-GLYPH-CURSOR
CreatePixmap	XCreatePixmap	CREATE-Pixmap
	XCreateBitmapFromData XCreatePixmapFromData XReadBitmapFile	READ-BITMAP-FILE
CreateWindow	XCreateWindow XCreateSimpleWindow	CREATE-WINDOW
DeleteProperty	XDeleteProperty	DELETE-PROPERTY
DestroySubwindows	XDestroySubwindows	DESTROY-SUBWINDOWS
DestroyWindow	XDestroyWindow	DESTROY-WINDOW
FillPoly	XFillPolygon	DRAW-LINES
ForceScreenSaver	XForceScreenSaver XActivateScreenSaver XResetScreenSaver	ACTIVATE-SCREEN-SAVER RESET-SCREEN-SAVER
FreeColormap	XFreeColormap	FREE-COLORMAP
FreeColors	XFreeColors	FREE-COLORS
FreeCursor	XFreeCursor	FREE-CURSOR

†Note that [brackets] represent optional arguments.

(continued on next page)

Table B-1 (Cont.): Mapping X11 Requests to CLX Functions

Protocol Request	Xlib Function(s)	CLX Function(s) or Equivalent
FreeGC	XFreeGC XCreateBitmapFromData XCreatePixmapFromData XReadBitmapFile	FREE-GCONTEXT
FreePixmap	XFreePixmap	FREE-Pixmap
GetAtomName	XGetAtomName	ATOM-NAME
GetFontPath	XGetFontPath	FONT-PATH
GetGeometry	XGetGeometry XGetWindowAttributes	<i>See also</i> WITH-STATE DRAWABLE-BORDER-WIDTH DRAWABLE-DEPTH DRAWABLE-HEIGHT DRAWABLE-ROOT DRAWABLE-WIDTH DRAWABLE-X DRAWABLE-Y
GetImage	XGetImage	GET-IMAGE GET-RAW-IMAGE
GetInputFocus	XGetInputFocus XSync	INPUT-FOCUS DISPLAY-FINISH-OUTPUT
GetKeyboardControl	XGetKeyboardControl	KEYBOARD-CONTROL
GetKeyboardMapping	XGetKeyboardMapping	KEYBOARD-MAPPING
GetModifierMapping	XGetModifierMapping	MODIFIER-MAPPING
GetMotionEvents	XGetMotionEvents	MOTION-EVENTS
GetPointerControl	XGetPointerControl	POINTER-CONTROL
GetPointerMapping	XGetPointerMapping	POINTER-MAPPING
GetProperty	XFetchBytes XFetchName XGetIconSizes XGetNormalHints XGetSizeHints XGetWMHints XGetWindowProperty XGetZoomHints	GET-PROPERTY
GetSelectionOwner	XGetSelectionOwner	SELECTION-OWNER

(continued on next page)

Table B-1 (Cont.): Mapping X11 Requests to CLX Functions

Protocol Request	Xlib Function(s)	CLX Function(s) or Equivalent
GetWindowAttributes	XGetWindowAttributes	<i>See also</i> WITH-STATE WINDOW-ALL-EVENT-MASKS WINDOW-BACKING-PIXEL WINDOW-BACKING-PLANES WINDOW-BACKING-STORE WINDOW-BIT-GRAVITY WINDOW-CLASS WINDOW-COLORMAP WINDOW-COLORMAP-INSTALLED-P WINDOW-DO-NOT-PROPAGATE-MASK WINDOW-EVENT-MASK WINDOW-GRAVITY WINDOW-MAP-STATE WINDOW-OVERRIDE-REDIRECT WINDOW-SAVE-UNDER WINDOW-VISUAL
GrabButton	XGrabButton	GRAB-BUTTON
GrabKey	XGrabKey	GRAB-KEY
GrabKeyboard	XGrabKeyboard	GRAB-KEYBOARD
GrabPointer	XGrabPointer	GRAB-POINTER
GrabServer	XGrabServer	GRAB-SERVER
ImageText16	XDrawImageString16	DRAW-IMAGE-GLYPH DRAW-IMAGE-GLYPHS
ImageText8	XDrawImageString	DRAW-IMAGE-GLYPHS
InstallColormap	XInstallColormap	INSTALL-COLORMAP
InternAtom	XInternAtom	INTERN-ATOM FIND-ATOM
KillClient	XKillClient	KILL-CLIENT KILL-TEMPORARY-CLIENTS
ListExtensions	XListExtensions	LIST-EXTENSIONS
ListFonts	XListFonts	LIST-FONT-NAMES
ListFontsWithInfo	XListFontsWithInfo	LIST-FONTS
ListHosts	XListHosts	ACCESS-CONTROL ACCESS-HOSTS
ListInstalledColormaps	XListInstalledColormaps	INSTALLED-COLORMAPS
ListProperties	XListProperties	LIST-PROPERTIES
LookupColor	XLookupColor XParseColor	LOOKUP-COLOR
MapSubwindows	XMapSubwindows	MAP-SUBWINDOWS
MapWindow	XMapWindow XMapRaised	MAP-WINDOW
NoOperation	XNoOp	
OpenFont	XLoadFont XLoadQueryFont	OPEN-FONT

(continued on next page)

Table B-1 (Cont.): Mapping X11 Requests to CLX Functions

Protocol Request	Xlib Function(s)	CLX Function(s) or Equivalent
PolyArc	XDrawArcs XDrawArc	DRAW-ARCS DRAW-ARC
PolyFillArc	XFillArcs XFillArc	DRAW-ARCS DRAW-ARC
PolyFillRectangle	XFillRectangles XFillRectangle	DRAW-RECTANGLES DRAW-RECTANGLE
PolyLine	XDrawLines	DRAW-LINES DRAW-LINE
PolyPoint	XDrawPoints XDrawPoint	DRAW-POINTS DRAW-POINT
PolyRectangle	XDrawRectangles XDrawRectangle	DRAW-RECTANGLES DRAW-RECTANGLE
PolySegment	XDrawSegments XDrawLine	DRAW-SEGMENTS
PolyText16	XDrawText16 XDrawString16	DRAW-GLYPH DRAW-GLYPHS
PolyText8	XDrawText XDrawString	DRAW-GLYPHS
PutImage	XPutImage	PUT-IMAGE PUT-RAW-IMAGE
	XCreateBitmapFromData XCreatePixmapFromData XReadBitmapFile	READ-BITMAP-FILE
QueryBestSize	XQueryBestSize XQueryBestCursor XQueryBestStipple XQueryBestTile	QUERY-BEST-CURSOR QUERY-BEST-STIPPLE QUERY-BEST-TILE
QueryColors	XQueryColor XQueryColors	QUERY-COLORS
QueryExtension	XQueryExtension XInitExtension	QUERY-EXTENSION
QueryFont	XQueryFont XLoadQueryFont	CHAR-ASCENT CHAR-ATTRIBUTES CHAR-DESCENT CHAR-LEFT-BEARING CHAR-RIGHT-BEARING CHAR-WIDTH

(continued on next page)

Table B–1 (Cont.): Mapping X11 Requests to CLX Functions

Protocol Request	Xlib Function(s)	CLX Function(s) or Equivalent
		FONT-ALL-CHARS-EXIST-P FONT-ASCENT FONT-DEFAULT-CHAR FONT-DESCENT FONT-DIRECTION FONT-MAX-BYTE1 FONT-MAX-BYTE2 FONT-MAX-CHAR FONT-MIN-BYTE1 FONT-MIN-BYTE2 FONT-MIN-CHAR FONT-NAME FONT-PROPERTIES FONT-PROPERTY
		MAX-CHAR-ASCENT MAX-CHAR-ATTRIBUTES MAX-CHAR-DESCENT MAX-CHAR-LEFT-BEARING MAX-CHAR-RIGHT-BEARING MAX-CHAR-WIDTH
		MIN-CHAR-ASCENT MIN-CHAR-ATTRIBUTES MIN-CHAR-DESCENT MIN-CHAR-LEFT-BEARING MIN-CHAR-RIGHT-BEARING MIN-CHAR-WIDTH
QueryKeymap	XQueryKeymap	QUERY-KEYMAP
QueryPointer	XQueryPointer	QUERY-POINTER GLOBAL-POINTER-POSITION POINTER-POSITION
QueryTextExtents	XQueryTextExtents XQueryTextExtents16	TEXT-EXTENTS TEXT-WIDTH
QueryTree	XQueryTree	QUERY-TREE
RecolorCursor	XRecolorCursor	RECOLOR-CURSOR
ReparentWindow	XReparentWindow	REPARENT-WINDOW
RotateProperties	XRotateWindowProperties XRotateBuffers	ROTATE-PROPERTIES
SendEvent	XSendEvent	SEND-EVENT
SetAccessControl	XSetAccessControl XDisableAccessControl XEnableAccessControl	(setf (access-control <i>display</i>) <i>boolean</i>)

(continued on next page)

Table B–1 (Cont.): Mapping X11 Requests to CLX Functions

Protocol Request	Xlib Function(s)	CLX Function(s) or Equivalent
SetClipRectangles	XSetClipRectangles	(setf (gcontext-clip-x <i>gc</i>) <i>integer</i>) (setf (gcontext-clip-y <i>gc</i>) <i>integer</i>) (setf (gcontext-clip-mask <i>gc</i> [<i>ordering</i>]) <i>mask</i>)† (setf (gcontext-clip-ordering <i>gc</i>) <i>keyword</i>) <i>See also</i> FORCE-GCONTEXT-CHANGES, WITH-GCONTEXT
SetCloseDownMode	XSetCloseDownMode	(setf (close-down-mode <i>display</i>) <i>mode</i>)
SetDashes	XSetDashes	(setf (gcontext-dash-offset <i>gc</i>) <i>integer</i>) (setf (gcontext-dashes <i>gc</i>) <i>dash-list</i>) <i>See also</i> FORCE-GCONTEXT-CHANGES, WITH-GCONTEXT
SetFontPath	XSetFontPath	(setf (font-path <i>font</i>) <i>paths</i>)
SetInputFocus	XSetInputFocus	SET-INPUT-FOCUS
SetModifierMapping	XSetModifierMapping	(setf (modifier-mapping <i>display</i>) <i>keycodes</i>)
SetPointerMapping	XSetPointerMapping	(setf (pointer-mapping <i>display</i>) <i>list</i>)
SetScreenSaver	XSetScreenSaver XGetScreenSaver	SET-SCREEN-SAVER
SetSelectionOwner	XSetSelectionOwner	(setf (selection-owner <i>display</i> <i>selection</i>) <i>window</i>)
StoreColors	XStoreColors	STORE-COLORS
	XStoreColor	STORE-COLOR
StoreNamedColor	XStoreNamedColor	STORE-COLOR STORE-COLORS
TranslateCoordinates	XTranslateCoordinates	TRANSLATE-COORDINATES
UngrabButton	XUngrabButton	UNGRAB-BUTTON
UngrabKey	XUngrabKey	UNGRAB-KEY
UngrabKeyboard	XUngrabKeyboard	UNGRAB-KEYBOARD
UngrabPointer	XUngrabPointer	UNGRAB-POINTER
UngrabServer	XUngrabServer	UNGRAB-SERVER
UninstallColormap	XUninstallColormap	UNINSTALL-COLORMAP
UnmapSubwindows	XUnmapSubWindows	UNMAP-SUBWINDOWS
UnmapWindow	XUnmapWindow	UNMAP-WINDOW
WarpPointer	XWarpPointer	WARP-POINTER WARP-POINTER-IF-INSIDE WARP-POINTER-RELATIVE WARP-POINTER-RELATIVE-IF-INSIDE

†Note that [brackets] represent optional arguments.

Table B–2: Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
ACCESS-CONTROL	XListHosts	ListHosts
ACCESS-HOSTS	XListHosts	ListHosts
ACTIVATE-SCREEN-SAVER	XForceScreenSaver XActivateScreenSaver XResetScreenSaver	ForceScreenSaver
ADD-ACCESS-HOST	XAddHost XAddHosts	ChangeHosts
ADD-TO-SAVE-SET	XAddToSaveSet XChangeSaveSet	ChangeSaveSet
ALLOC-COLOR	XAllocColor	AllocColor AllocNamedColor
ALLOC-COLOR-CELLS	XAllocColorCells	AllocColorCells
ALLOC-COLOR-PLANES	XAllocColorPlanes	AllocColorPlanes
ALLOW-EVENTS	XAllowEvents	AllowEvents
ATOM-NAME	XGetAtomName	GetAtomName
BELL	XBell	Bell
BITMAP-FORMAT-LSB-FIRST-P		
BITMAP-FORMAT-P		
BITMAP-FORMAT-PAD		
BITMAP-FORMAT-UNIT		
CHANGE-ACTIVE-POINTER-GRAB	XChangeActivePointerGrab	ChangeActivePointerGrab
CHANGE-KEYBOARD-CONTROL	XChangeKeyboardControl XAutoRepeatOff XAutoRepeatOn	ChangeKeyboardControl
CHANGE-KEYBOARD-MAPPING	XChangeKeyboardMapping	ChangeKeyboard Mapping
CHANGE-POINTER-CONTROL	XChangePointerControl	ChangePointerControl
CHANGE-PROPERTY	XChangeProperty	ChangeProperty
	XSetCommand XSetIconName XSetIconSizes XSetNormalHints XSetSizeHints XSetStandardProperties XSetWMHints XSetZoomHints XStoreBuffer XStoreBytes XStoreName	
CHARACTER->KEYSYMS	XStringToKeysym	
CHAR-ASCENT	XQueryFont XLoadQueryFont	QueryFont

(continued on next page)

Table B–2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
CHAR-ATTRIBUTES	XQueryFont XLoadQueryFont	QueryFont
CHAR-DESCENT	XQueryFont XLoadQueryFont	QueryFont
CHAR-LEFT-BEARING	XQueryFont XLoadQueryFont	QueryFont
CHAR-RIGHT-BEARING	XQueryFont XLoadQueryFont	QueryFont
CHAR-WIDTH	XQueryFont XLoadQueryFont	QueryFont
CIRCULATE-WINDOW-DOWN		CirculateWindow
CIRCULATE-WINDOW-UP	XCirculateSubwindows XCirculateSubwindowsDown	CirculateWindow
CLEAR-AREA	XClearArea XClearWindow	ClearArea
CLOSE-DISPLAY	XCloseDisplay	
CLOSE-DOWN-MODE†	XSetCloseDownMode	SetCloseDownMode
CLOSE-FONT	XFreeFont XUnloadFont	CloseFont
COLOR-BLUE†		
COLOR-GREEN†		
COLOR-P		
COLOR-RED†		
COLOR-RGB		
COLORMAP-DISPLAY		
COLORMAP-EQUAL		
COLORMAP-ID		
COLORMAP-P		
CONVERT-SELECTION	XConvertSelection	ConvertSelection
COPY-AREA	XCopyArea	CopyArea
COPY-COLORMAP-AND-FREE	XCopyColormapAndFree	CopyColormapAndFree
COPY-GCONTEXT	XCopyGC	CopyGC
COPY-GCONTEXT-COMPONENTS	XCopyGC	CopyGC
COPY-IMAGE		
COPY-PLANE	XCopyPlane	CopyPlane
CREATE-COLORMAP	XCreateColormap	CreateColormap
CREATE-CURSOR	XCreatePixmapCursor	CreateCursor

†This function can be used with SETF.

(continued on next page)

Table B–2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
CREATE-GCONTEXT	XCreateGC XCreateBitmapFromData XCreatePixmapFromData XOpenDisplay XReadBitmapFromFile	CreateGC
CREATE-GLYPH-CURSOR	XCreateGlyphCursor XCreateFontCursor	CreateGlyphCursor
CREATE-IMAGE	XCreateImage	
CREATE-PIXMAP	XCreatePixmap XCreateBitmapFromData XCreatePixmapFromData XReadBitmapFromFile	CreatePixmap
CREATE-WINDOW	XCreateWindow XCreateSimpleWindow	CreateWindow
CURSOR-DISPLAY		
CURSOR-EQUAL		
CURSOR-ID		
CURSOR-P		
DELETE-PROPERTY	XDeleteProperty	DeleteProperty
DESTROY-SUBWINDOWS	XDestroySubwindows	DestroySubwindows
DESTROY-WINDOW	XDestroyWindow	DestroyWindow
DISCARD-CURRENT-EVENT	XNextEvent	
DISCARD-FONT-INFO		
DISPLAY-AFTER-FUNCTION†		
DISPLAY-BITMAP-FORMAT		
DISPLAY-DEFAULT-SCREEN		
DISPLAY-FINISH-OUTPUT	XSync .	
DISPLAY-FORCE-OUTPUT	XFlush	
DISPLAY-IMAGE-LSB-FIRST-P		
DISPLAY-MAX-KEYCODE		
DISPLAY-MAX-REQUEST-LENGTH		
DISPLAY-MIN-KEYCODE		
DISPLAY-MOTION-BUFFER-SIZE		
DISPLAY-P		
DISPLAY-PIXMAP-FORMATS		
DISPLAY-PROTOCOL-MAJOR-VERSION		
DISPLAY-PROTOCOL-MINOR-VERSION		
DISPLAY-RELEASE-NUMBER		
DISPLAY-ROOTS		
DISPLAY-VENDOR-NAME		
DRAWABLE-BORDER-WIDTH†	XSetWindowBorderWidth	GetGeometry

†This function can be used with SETF.

(continued on next page)

Table B–2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
DRAWABLE-DEPTH	XGetWindowAttributes	GetGeometry
DRAWABLE-DISPLAY		
DRAWABLE-EQUAL		
DRAWABLE-HEIGHT†	XGetWindowAttributes	GetGeometry
DRAWABLE-ID		
DRAWABLE-P		
DRAWABLE-ROOT	XGetWindowAttributes	GetGeometry
DRAWABLE-WIDTH†	XGetWindowAttributes	GetGeometry
DRAWABLE-X†	XGetWindowAttributes	GetGeometry
DRAWABLE-Y†	XGetWindowAttributes	GetGeometry
DRAW-ARC	XDrawArc XFillArc	PolyArc PolyFillArc
DRAW-ARCS	XDrawArcs XFillArcs	PolyArcs PolyFillArcs
DRAW-GLYPH	XDrawText16 XDrawString16	PolyText16
DRAW-GLYPHS	XDrawText XDrawText16 XDrawString XDrawString16	PolyText16 PolyText8
DRAW-IMAGE-GLYPH	XDrawImageString XDrawImageString16	ImageText8 ImageText16
DRAW-IMAGE-GLYPHS	XDrawImageString XDrawImageString16	ImageText8 ImageText16
DRAW-LINE	XDrawLines	PolyLine
DRAW-LINES	XDrawLines XFillPolygon	PolyLine FillPoly
DRAW-POINT	XDrawPoints	PolyPoint
DRAW-POINTS	XDrawPoints	PolyPoint
DRAW-RECTANGLE	XDrawRectangle XFillRectangle	PolyRectangle PolyFillRectangle
DRAW-RECTANGLES	XDrawRectangles XFillRectangles	PolyRectangle PolyFillRectangle
DRAW-SEGMENTS	XDrawSegments	PolySegment
EVENT-CASE		
EVENT-LISTEN	XEventsQueued	
FIND-ATOM	XInternAtom	InternAtom
FONT-ALL-CHARS-EXIST-P	XQueryFont XLoadQueryFont	QueryFont
FONT-ASCENT	XQueryFont XLoadQueryFont	QueryFont
FONT-DEFAULT-CHAR	XQueryFont XLoadQueryFont	QueryFont

†This function can be used with SETF.

(continued on next page)

Table B–2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
FONT-DESCENT	XQueryFont XLoadQueryFont	QueryFont
FONT-DIRECTION	XQueryFont XLoadQueryFont	QueryFont
FONT-DISPLAY		
FONT-EQUAL		
FONT-ID		
FONT-MAX-BYTE1	XQueryFont XLoadQueryFont	QueryFont
FONT-MAX-BYTE2	XQueryFont XLoadQueryFont	QueryFont
FONT-MAX-CHAR	XQueryFont XLoadQueryFont	QueryFont
FONT-MIN-BYTE1	XQueryFont XLoadQueryFont	QueryFont
FONT-MIN-BYTE2	XQueryFont XLoadQueryFont	QueryFont
FONT-MIN-CHAR	XQueryFont XLoadQueryFont	QueryFont
FONT-NAME	XQueryFont XLoadQueryFont	QueryFont
FONT-P		
FONT-PATH	XGetFontPath	GetFontPath
FONT-PROPERTIES	XQueryFont XLoadQueryFont	QueryFont
FONT-PROPERTY	XQueryFont XLoadQueryFont	QueryFont
FORCE-GCONTEXT-CHANGES	XChangeGC XSetClipRectangles XSetDashes	ChangeGC SetClipRectangles SetDashes
FREE-COLORMAP	XFreeColormap	FreeColormap
FREE-COLORS	XFreeColors	FreeColors
FREE-CURSOR	XFreeCursor	FreeCursor
FREE-GCONTEXT	XFreeGC	FreeGC
	XCreateBitmapFromData XCreatePixmapFromData XReadBitmapFromFile	
FREE-PIXMAP	XFreePixmap	FreePixmap
GCONTEXT-ARC-MODE†	XChangeGC	ChangeGC
GCONTEXT-BACKGROUND†	XChangeGC	ChangeGC
GCONTEXT-CACHE-P		
GCONTEXT-CAP-STYLE†	XChangeGC	ChangeGC
GCONTEXT-CLIP-X†	XChangeGC	ChangeGC

†This function can be used with SETF.

(continued on next page)

Table B-2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
GCONTEXT-CLIP-Y†	XChangeGC	ChangeGC
GCONTEXT-DASHES†	XChangeGC	ChangeGC
GCONTEXT-DASH-OFFSET†	XChangeGC	ChangeGC
GCONTEXT-DISPLAY		
GCONTEXT-EQUAL		
GCONTEXT-EXPOSURE†	XChangeGC	ChangeGC
GCONTEXT-FILL-RULE†	XChangeGC	ChangeGC
GCONTEXT-FILL-STYLE†	XChangeGC	ChangeGC
GCONTEXT-FONT†	XChangeGC	ChangeGC
GCONTEXT-FOREGROUND†	XChangeGC	ChangeGC
GCONTEXT-FUNCTION†	XChangeGC	ChangeGC
GCONTEXT-ID		
GCONTEXT-JOIN-STYLE†	XChangeGC	ChangeGC
GCONTEXT-LINE-STYLE†	XChangeGC	ChangeGC
GCONTEXT-LINE-WIDTH†	XChangeGC	ChangeGC
GCONTEXT-P		
GCONTEXT-PLANE-MASK†	XChangeGC	ChangeGC
GCONTEXT-STIPPLE†	XChangeGC	ChangeGC
GCONTEXT-SUBWINDOW-MODE†	XChangeGC	ChangeGC
GCONTEXT-TILE†	XChangeGC	ChangeGC
GCONTEXT-TS-X†	XChangeGC	ChangeGC
GCONTEXT-TS-Y†	XChangeGC	ChangeGC
GET-IMAGE	XGetImage	GetImage
GET-PROPERTY		GetProperty
	XFetchBytes XFetchName XGetIconSizes XGetNormalHints XGetSizeHints XGetWMHints XGetWindowProperty XGetZoomHints	
GET-RAW-IMAGE	XGetImage	GetImage
GET-WM-CLASS		
GLOBAL-POINTER-POSITION	XQueryPointer	QueryPointer
GRAB-BUTTON	XGrabButton	GrabButton
GRAB-KEY	XGrabKey	GrabKey
GRAB-KEYBOARD	XGrabKeyboard	GrabKeyboard
GRAB-POINTER	XGrabPointer	GrabPointer
GRAB-SERVER	XGrabServer	GrabServer
ICON-SIZES		

†This function can be used with SETF.

(continued on next page)

Table B-2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
IMAGE-BIT-LSB-FIRST-P†		
IMAGE-BITS-PER-PIXEL†		
IMAGE-BLUE-MASK†		
IMAGE-BYTE-LSB-FIRST-P†		
IMAGE-BYTES-PER-LINE†		
IMAGE-DEPTH		
IMAGE-FORMAT†		
IMAGE-GREEN-MASK†		
IMAGE-HEIGHT†		
IMAGE-P		
IMAGE-RED-MASK†		
IMAGE-SCANLINE-PAD†		
IMAGE-WIDTH†		
IMAGE-XY-BITMAP-LIST		
IMAGE-Z-BITS-PER-PIXEL		
IMAGE-Z-PIXARRAY		
INPUT-FOCUS	XGetInputFocus	GetInputFocus
INSTALL-COLORMAP	XInstallColormap	InstallColormap
INSTALLED-COLORMAPS	XListInstalledColormaps	ListInstalledColormaps
INTERN-ATOM	XInternAtom	InternAtom
KEYBOARD-CONTROL	XGetKeyboardControl	GetKeyboardControl
KEYBOARD-MAPPING	XGetKeyboardMapping	GetKeyboardMapping
KEYCODE->CHARACTER	XLookupString	
KEYCODE->KEYSYM	XLookupKeysym	
KEYSYM->CHARACTER	XKeysymToString	
KEYSYM->KEYCODES	XKeysymToKeycodes	
KILL-CLIENT	XKillClient	KillClient
KILL-TEMPORARY-CLIENTS	XKillClient	KillClient
LIST-EXTENSIONS	XListExtensions	ListExtensions
LIST-FONT-NAMES	XListFonts	ListFonts
LIST-FONTS	XListFontsWithInfo	ListFontsWithInfo
LIST-PROPERTIES	XListProperties	ListProperties
LOOKUP-COLOR	XLookupColor XParseColor	LookupColor
MAKE-COLOR		
MAKE-COLORMAP		
MAKE-CURSOR		
MAKE-DRAWABLE		
MAKE-EVENT-KEYS		

†This function can be used with SETF.

(continued on next page)

Table B-2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
MAKE-EVENT-MASK		
MAKE-FONT		
MAKE-GCONTEXT		
MAKE-PIXMAP		
MAKE-STATE-KEYS		
MAKE-STATE-MASK		
MAKE-VISUAL-INFO		
MAKE-WINDOW		
MAKE-WM-HINTS		
MAKE-WM-SIZE-HINTS		
MAP-SUBWINDOWS	XMapSubwindows	MapSubwindows
MAP-WINDOW	XMapWindow XMapRaised	MapWindow
MAX-CHAR-ASCENT	XQueryFont XLoadQueryFont	QueryFont
MAX-CHAR-ATTRIBUTES	XQueryFont XLoadQueryFont	QueryFont
MAX-CHAR-DESCENT	XQueryFont XLoadQueryFont	QueryFont
MAX-CHAR-LEFT-BEARING	XQueryFont XLoadQueryFont	QueryFont
MAX-CHAR-RIGHT-BEARING	XQueryFont XLoadQueryFont	QueryFont
MAX-CHAR-WIDTH	XQueryFont XLoadQueryFont	QueryFont
MIN-CHAR-ASCENT	XQueryFont XLoadQueryFont	QueryFont
MIN-CHAR-ATTRIBUTES	XQueryFont XLoadQueryFont	QueryFont
MIN-CHAR-DESCENT	XQueryFont XLoadQueryFont	QueryFont
MIN-CHAR-LEFT-BEARING	XQueryFont XLoadQueryFont	QueryFont
MIN-CHAR-RIGHT-BEARING	XQueryFont XLoadQueryFont	QueryFont
MIN-CHAR-WIDTH	XQueryFont XLoadQueryFont	QueryFont
MODIFIER-MAPPING	XGetModifierMapping	GetModifierMapping
MOTION-EVENTS	XGetMotionEvents	GetMotionEvents
OPEN-DISPLAY	XOpenDisplay	
OPEN-FONT	XLoadFont XLoadQueryFont	OpenFont
PIXMAP-DISPLAY		

(continued on next page)

Table B–2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
PIXMAP-EQUAL		
PIXMAP-FORMAT-BITS-PER-PIXEL†		
PIXMAP-FORMAT-DEPTH†		
PIXMAP-FORMAT-P		
PIXMAP-FORMAT-PAD†		
PIXMAP-ID		
PIXMAP-P		
POINTER-CONTROL	XGetPointerControl	GetPointerControl
POINTER-MAPPING†	XGetPointerMapping XSetPointerMapping	GetPointerMapping SetPointerMapping
POINTER-POSITION	XQueryPointer	QueryPointer
PROCESS-EVENT		
PUT-IMAGE	XPutImage	PutImage
PUT-RAW-IMAGE	XPutImage	PutImage
QUERY-BEST-CURSOR	XQueryBestCursor	QueryBestSize
QUERY-BEST-STIPPLE	XQueryBestStipple	QueryBestSize
QUERY-BEST-TILE	XQueryBestTile	QueryBestSize
QUERY-COLORS	XQueryColors XQueryColor	QueryColors
QUERY-EXTENSION	XQueryExtension XInitExtension	QueryExtension
QUERY-KEYMAP	XQueryKeymap	QueryKeymap
QUERY-POINTER	XQueryPointer	QueryPointer
QUERY-TREE	XQueryTree	QueryTree
QUEUE-EVENT	XPutBackEvent	
READ-BITMAP-FILE	XReadBitmapFile	PutImage
RECOLOR-CURSOR	XRecolorCursor	RecolorCursor
REMOVE-ACCESS-HOST	XRemoveHost XRemoveHosts	ChangeHosts
REMOVE-FROM-SAVE-SET	XRemoveFromSaveSet XChangeSaveSet	ChangeSaveSet
REPARENT-WINDOW	XReparentWindow	ReparentWindow
RESET-SCREEN-SAVER	XResetScreenSaver	ForceScreenSaver
ROTATE-PROPERTIES	XRotateProperties	RotateProperties
SCREEN-BACKING-STORES		
SCREEN-BLACK-PIXEL		
SCREEN-DEFAULT-COLORMAP		
SCREEN-DEPTHES		
SCREEN-DISPLAY		
SCREEN-EVENT-MASK-AT-OPEN		

†This function can be used with SETF.

(continued on next page)

Table B-2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
SCREEN-HEIGHT		
SCREEN-HEIGHT-IN-MILLIMETERS		
SCREEN-MAX-INSTALLED-MAPS		
SCREEN-MIN-INSTALLED-MAPS		
SCREEN-NDEPTHS		
SCREEN-P		
SCREEN-ROOT		
SCREEN-ROOT-DEPTH		
SCREEN-ROOT-VISUAL		
SCREEN-SAVE-UNDERS-P		
SCREEN-SAVER		
SCREEN-WHITE-PIXEL		
SCREEN-WIDTH		
SCREEN-WIDTH-IN-MILLIMETERS		
SELECTION-OWNER†	XGetSelectionOwner XSetSelectionOwner	GetSelectionOwner SetSelectionOwner
SEND-EVENT	XSendEvent	SendEvent
SET-INPUT-FOCUS	XSetInputFocus	SetInputFocus
SET-MODIFIER-MAPPING	XSetModifierMapping	SetModifierMapping
SET-SCREEN-SAVER	XSetScreenSaver	SetScreenSaver
SET-WM-CLASS	XSetClassHints	
STORE-COLOR	XStoreColor XStoreNamedColor	StoreColors StoreNamedColor
STORE-COLORS	XStoreColors XStoreNamedColors	StoreColors StoreNamedColor
TEXT-EXTENTS	XQueryTextExtents XQueryTextExtents16	QueryTextExtents
TEXT-WIDTH	XQueryTextWidth XQueryTextWidth16	QueryTextExtents
TRANSLATE-COORDINATES	XTranslateCoords	TranslateCoords
TRANSLATE-DEFAULT		
UNGRAB-BUTTON	XUngrabButton	UngrabButton
UNGRAB-KEY	XUngrabKey	UngrabKey
UNGRAB-KEYBOARD	XUngrabKeyboard	UngrabKeyboard
UNGRAB-POINTER	XUngrabPointer	UngrabPointer
UNGRAB-SERVER	XUngrabServer	UngrabServer
UNINSTALL-COLORMAP	XUninstallColormap	UninstallColormap
UNMAP-SUBWINDOWS	XUnmapSubwindows	UnmapSubwindows
UNMAP-WINDOW	XUnmapWindow	UnmapWindow
VISUAL-INFO-BITS-PER-RGB		

†This function can be used with SETF.

(continued on next page)

Table B–2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
VISUAL-INFO-BLUE-MASK		
VISUAL-INFO-CLASS		
VISUAL-INFO-COLORMAP-ENTRIES		
VISUAL-INFO-GREEN-MASK		
VISUAL-INFO-ID		
VISUAL-INFO-P		
VISUAL-INFO-RED-MASK		
WARP-POINTER	XWarpPointer	WarpPointer
WARP-POINTER-IF-INSIDE	XWarpPointer	WarpPointer
WARP-POINTER-RELATIVE	XWarpPointer	WarpPointer
WARP-POINTER-RELATIVE-IF-INSIDE	XWarpPointer	WarpPointer
WINDOW-ALL-EVENT-MASKS	XGetWindowAttributes	GetWindowAttributes
WINDOW-BACKING-PIXEL †	XGetWindowAttributes	GetWindowAttributes
WINDOW-BACKGROUND‡	XChangeWindowAttributes	ChangeWindow Attributes
WINDOW-BACKING-PLANES †	XGetWindowAttributes	GetWindowAttributes
WINDOW-BACKING-STORE†	XGetWindowAttributes	GetWindowAttributes
WINDOW-BIT-GRAVITY †	XGetWindowAttributes	GetWindowAttributes
WINDOW-BORDER‡	XChangeWindowAttributes	ChangeWindow Attributes
WINDOW-CLASS	XGetWindowAttributes	GetWindowAttributes
WINDOW-COLORMAP†	XGetWindowAttributes	GetWindowAttributes
WINDOW-COLORMAP-INSTALLED-P	XGetWindowAttributes	GetWindowAttributes
WINDOW-CURSOR‡	XChangeWindowAttributes	ChangeWindow Attributes
WINDOW-DISPLAY		
WINDOW-DO-NOT-PROPAGATE-MASK†	XGetWindowAttributes	GetWindowAttributes
WINDOW-EQUAL		
WINDOW-EVENT-MASK†	XGetWindowAttributes	GetWindowAttributes
WINDOW-GRAVITY†	XGetWindowAttributes	GetWindowAttributes
WINDOW-ID		
WINDOW-MAP-STATE	XGetWindowAttributes	GetWindowAttributes
WINDOW-P		
WINDOW-PRIORITY‡		
WINDOW-OVERRIDE-REDIRECT†	XGetWindowAttributes	GetWindowAttributes
WINDOW-SAVE-UNDER†	XGetWindowAttributes	GetWindowAttributes
WINDOW-VISUAL	XGetWindowAttributes	GetWindowAttributes
WITH-DISPLAY		
WITH-EVENT-QUEUE		

†This function can be used with SETF.

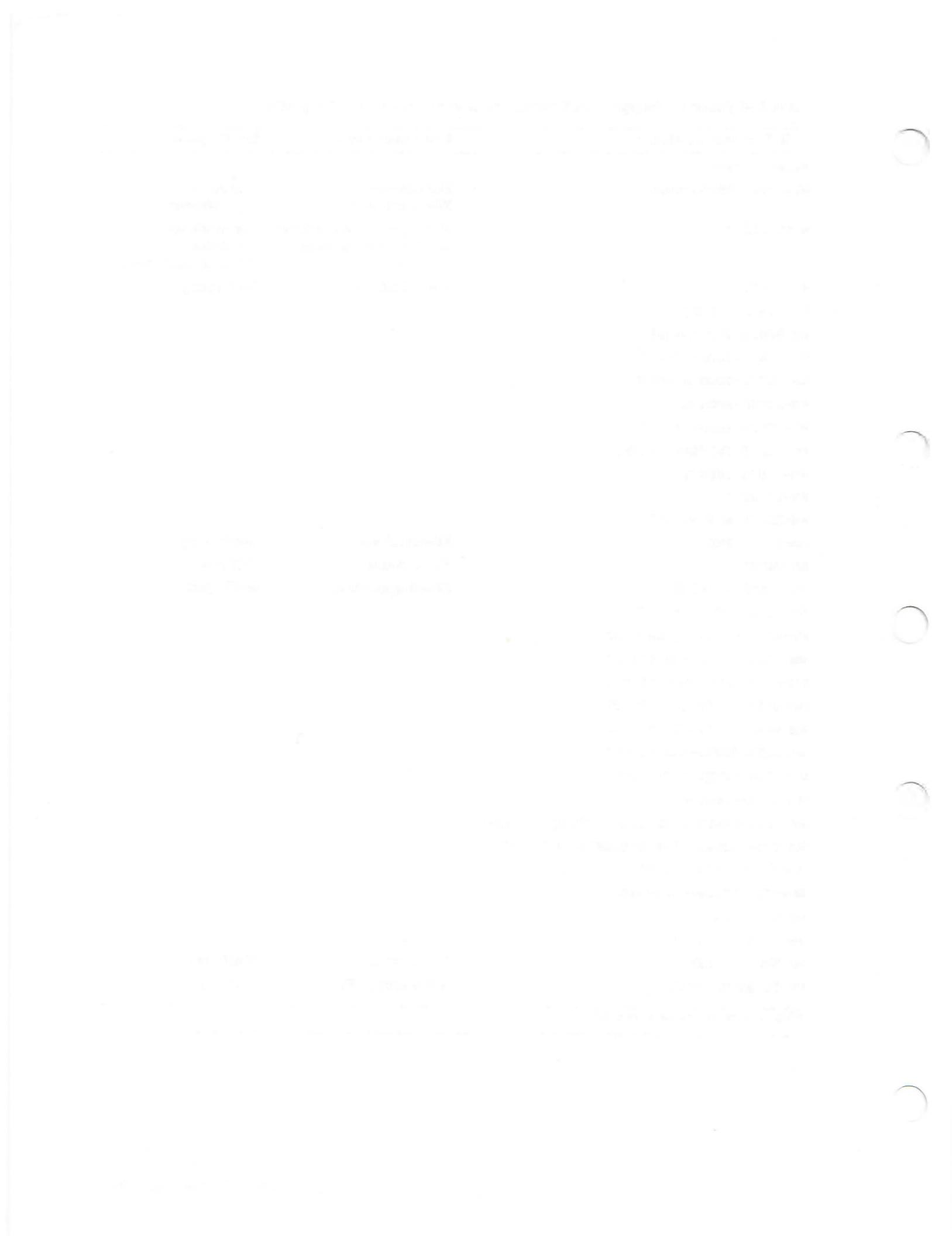
‡This function is not callable. It can be used with SETF only.

(continued on next page)

Table B–2 (Cont.): Mapping CLX Functions and Macros to X11 Requests

CLX Function or Macro	Xlib Function(s)	X11 Request(s)
WITH-GCONTEXT		
WITH-SERVER-GRABBED	XGrabServer XUngrabServer	GrabServer UngrabServer
WITH-STATE	XChangeWindowAttributes XGetWindowAttributes	ChangeWindow Attributes GetWindowAttributes
WM-HINTS	XGetWMHints	GetProperty
WM-HINTS-FLAGS†		
WM-HINTS-ICON-MASK†		
WM-HINTS-ICON-PIXMAP†		
WM-HINTS-ICON-WINDOW†		
WM-HINTS-ICON-X†		
WM-HINTS-ICON-Y†		
WM-HINTS-INITIAL-STATE†		
WM-HINTS-INPUT†		
WM-HINTS-P		
WM-HINTS-WINDOW-GROUP†		
WM-ICON-NAME†	XGetIconName	GetProperty
WM-NAME†	XFetchName	GetProperty
WM-NORMAL-HINTS†	XGetNormalHints	GetProperty
WM-SIZE-HINTS-HEIGHT†		
WM-SIZE-HINTS-HEIGHT-INC†		
WM-SIZE-HINTS-MAX-ASPECT†		
WM-SIZE-HINTS-MAX-HEIGHT†		
WM-SIZE-HINTS-MAX-WIDTH†		
WM-SIZE-HINTS-MIN-ASPECT†		
WM-SIZE-HINTS-MIN-HEIGHT†		
WM-SIZE-HINTS-MIN-WIDTH†		
WM-SIZE-HINTS-P		
WM-SIZE-HINTS-USER-SPECIFIED-POSITION-PT†		
WM-SIZE-HINTS-USER-SPECIFIED-SIZE-PT†		
WM-SIZE-HINTS-WIDTH†		
WM-SIZE-HINTS-WIDTH-INC†		
WM-SIZE-HINTS-X†		
WM-SIZE-HINTS-Y†		
WM-ZOOM-HINTS†	XGetZoomHints	GetProperty
WRITE-BITMAP-FILE	XWriteBitmapFile	GetImage

†This function can be used with SETF.



Index

A

Access from the network, 13–8 to 13–9
ACTIVATE attribute keyword
 User interface language, 2–7
Arcs, 9–7 to 9–9
Areas, 9–9 to 9–10
ARGUMENTS section
 User interface language, 2–5
Atom
 definition, 6–5

B

:BUTTON-PRESS event-key, 12–16
:BUTTON-RELEASE event-key, 12–16
Bell, 13–2
 duration, 13–6
 pitch, 13–6
 volume, 13–2, 13–5
Bitmap formats, 10–2 to 10–3
Box widget, 1–4
 See also Widgets
Button widget, 1–4
 See also Widgets

C

:CIRCULATE-NOTIFY event-key, 12–17
:CIRCULATE-REQUEST event-key, 12–17
:CLIENT-MESSAGE event-key, 12–17
:COLORMAP-NOTIFY event-key, 12–18
:CONFIGURE-NOTIFY event-key, 12–18
:CONFIGURE-REQUEST event-key, 12–19
:CREATE-NOTIFY event-key, 12–19
Callback facility
 calling DECwindows application functions, 1–7
Callback function
 See DECwindows callback function
Callback reasons, A–5
Callback structure, 3–9
CALLBACKS section
 User interface language, 2–5
Characters, 11–1 to 11–3
Clipboard
 accessing with DECwindows toolkit, 1–2
Closing the connection, 13–10 to 13–11
CLX data types, 4–2 to 4–4
CLX:ACCESS-CONTROL function, 13–9, CLX-1
CLX:ACCESS-HOSTS function, 13–9, CLX-1

CLX:ACTIVATE-SCREEN-SAVER function, 13–8, CLX-2
CLX:ADD-ACCESS-HOST function, 13–9, CLX-2
CLX:ADD-TO-SAVE-SET function, 6–4, 13–11, CLX-3
CLX:ALIST type specifier, 4–2, CLX-3
CLX:ALLOC-COLOR function, 8–3, CLX-3
CLX:ALLOC-COLOR-CELLS function, 8–5, CLX-4
CLX:ALLOC-COLOR-PLANES function, 8–5, CLX-5
CLX:ALLOW-EVENTS function, 12–14, CLX-6
CLX:ANGLE type specifier, 4–2, CLX-7
CLX:ARC-SEQ type specifier, 4–2, CLX-7
CLX:ARRAY-INDEX type specifier, 4–2, CLX-7
CLX:ATOM-NAME function, 6–6, CLX-7
CLX:BELL function, 13–2, CLX-8
CLX:BIT-GRAVITY type specifier, 4–2, CLX-8
CLX:BITMAP type specifier, 4–2, CLX-9
CLX:BITMAP-FORMAT structure, CLX-9
CLX:BITMAP-FORMAT-LSB-FIRST-P function, CLX-9
CLX:BITMAP-FORMAT-P function, CLX-10
CLX:BITMAP-FORMAT-PAD function, CLX-10
CLX:BITMAP-FORMAT-UNIT function, CLX-11
CLX:BOOLEAN type specifier, 4–2, CLX-11
CLX:CARD type specifier, CLX-12
CLX:CARD16 type specifier, 4–2, CLX-11
CLX:CARD29 type specifier, 4–2
CLX:CARD8 type specifier, 4–2, CLX-11
CLX:CHANGE-ACTIVE-POINTER-GRAB function, 12–12, CLX-12
CLX:CHANGE-KEYBOARD-CONTROL function, 13–5, CLX-12
CLX:CHANGE-KEYBOARD-MAPPING function, 13–2, CLX-13
CLX:CHANGE-POINTER-CONTROL function, 13–6, CLX-14
CLX:CHANGE-PROPERTY function, 6–5, CLX-15
CLX:CHAR-ASCENT function, CLX-16
CLX:CHAR-ATTRIBUTES function, CLX-17
CLX:CHAR-DESCENT function, CLX-17
CLX:CHAR-LEFT-BEARING function, CLX-18
CLX:CHAR-RIGHT-BEARING function, CLX-18
CLX:CHAR-WIDTH function, CLX-19
CLX:CHARACTER->KEYSYMS function, 13–4, CLX-16
CLX:CIRCULATE-WINDOW-DOWN function, 6–10, CLX-19
CLX:CIRCULATE-WINDOW-UP function, 6–10, CLX-20
CLX:CLEAR-AREA function, 9–9, CLX-20

CLX:CLOSE-DISPLAY function, 5-4, CLX-21
CLX:CLOSE-DOWN-MODE function, 13-10, CLX-21
CLX:CLOSE-FONT function, 11-4, CLX-22
CLX:COLOR structure, CLX-22
CLX:COLOR-BLUE function, CLX-23
CLX:COLOR-GREEN function, CLX-23
CLX:COLOR-P function, CLX-26
CLX:COLOR-RED function, CLX-26
CLX:COLOR-RGB function, CLX-26
CLX:COLORMAP structure, CLX-23
CLX:COLORMAP-DISPLAY function, CLX-24
CLX:COLORMAP-EQUAL function, CLX-24
CLX:COLORMAP-ID function, CLX-25
CLX:COLORMAP-P function, CLX-25
CLX:CONVERT-SELECTION function, 6-9,
 CLX-27
CLX:COPY-AREA function, 9-9, CLX-28
CLX:COPY-COLORMAP-AND-FREE function, 8-6,
 CLX-28
CLX:COPY-GCONTEXT function, 7-5, CLX-29
CLX:COPY-GCONTEXT-COMPONENTS function,
 7-5, CLX-29
CLX:COPY-IMAGE function, CLX-30
CLX:COPY-PLANE function, 9-10, CLX-30
CLX:CREATE-COLORMAP function, 8-5, CLX-31
CLX:CREATE-CURSOR function, 9-10, CLX-32
CLX:CREATE-GCONTEXT function, 7-1, CLX-32
CLX:CREATE-GLYPH-CURSOR function, 9-10,
 CLX-34
CLX:CREATE-IMAGE function, 10-4, CLX-35
CLX:CREATE-PIXMAP function, 10-1, CLX-36
CLX:CREATE-WINDOW function, 6-1, CLX-36
CLX:CURSOR-DISPLAY function, CLX-38
CLX:CURSOR-EQUAL function, CLX-39
CLX:CURSOR-ID function, CLX-39
CLX:CURSOR-P function, CLX-40
CLX:DELETE-PROPERTY function, 6-6, CLX-40
CLX:DESTROY-SUBWINDOWS function, 6-4,
 CLX-41
CLX:DESTROY-WINDOW function, 6-4, CLX-41
CLX:DEVICE-EVENT-MASK type specifier, 4-2,
 CLX-42
CLX:DEVICE-EVENT-MASK-CLASS type specifier,
 4-2, CLX-42
CLX:DISCARD-CURRENT-EVENT function, 12-9,
 CLX-42
CLX:DISCARD-FONT-INFO function, 11-4,
 CLX-43
CLX:DISPLAY structure, CLX-43
CLX:DISPLAY-AFTER-FUNCTION function, 5-4,
 CLX-44
CLX:DISPLAY-BITMAP-FORMAT function,
 CLX-44
CLX:DISPLAY-DEFAULT-SCREEN function,
 CLX-45
CLX:DISPLAY-FINISH-OUTPUT function, 5-4,
 CLX-45
CLX:DISPLAY-FORCE-OUTPUT function, 5-4,
 CLX-46
CLX:DISPLAY-IMAGE-LSB-FIRST-P function,
 CLX-46
CLX:DISPLAY-MAX-KEYCODE function, CLX-47
CLX:DISPLAY-MAX-REQUEST-LENGTH function,
 CLX-47
CLX:DISPLAY-MIN-KEYCODE function, CLX-48
CLX:DISPLAY-MOTION-BUFFER-SIZE function,
 CLX-48

CLX:DISPLAY-P function, CLX-49
CLX:DISPLAY-PIXMAP-FORMATS function,
 CLX-49
CLX:DISPLAY-PROTOCOL-MAJOR-VERSION
 function, CLX-49
CLX:DISPLAY-PROTOCOL-MINOR-VERSION
 function, CLX-50
CLX:DISPLAY-RELEASE-NUMBER function,
 CLX-50
CLX:DISPLAY-ROOTS function, CLX-51
CLX:DISPLAY-VENDOR-NAME function, CLX-51
CLX:DRAW-ARC function, 9-7, CLX-57
CLX:DRAW-ARCS function, 9-9, CLX-58
CLX:DRAW-DIRECTION type specifier, 4-2,
 CLX-59
CLX:DRAW-GLYPH function, 11-6, CLX-59
CLX:DRAW-GLYPHS function, 11-6, CLX-60
CLX:DRAW-IMAGE-GLYPH function, 11-7, CLX-61
CLX:DRAW-IMAGE-GLYPHS function, 11-7,
 CLX-62
CLX:DRAW-LINE function, 9-4, CLX-63
CLX:DRAW-LINES function, 9-5, CLX-64
CLX:DRAW-POINT function, 9-1, CLX-64
CLX:DRAW-POINTS function, 9-1, CLX-65
CLX:DRAW-RECTANGLE function, 9-6, CLX-66
CLX:DRAW-RECTANGLES function, 9-7, CLX-66
CLX:DRAW-SEGMENTS function, 9-5, CLX-67
CLX:DRAWABLE type specifier, 4-2, CLX-52
CLX:DRAWABLE-BORDER-WIDTH function,
 CLX-52
CLX:DRAWABLE-DEPTH function, CLX-52
CLX:DRAWABLE-DISPLAY function, CLX-53
CLX:DRAWABLE-EQUAL function, CLX-53
CLX:DRAWABLE-HEIGHT function, CLX-54
CLX:DRAWABLE-ID function, CLX-54
CLX:DRAWABLE-P function, CLX-55
CLX:DRAWABLE-ROOT function, CLX-55
CLX:DRAWABLE-WIDTH function, CLX-56
CLX:DRAWABLE-X function, CLX-56
CLX:DRAWABLE-Y function, CLX-57
CLX:EVENT-CASE macro, 12-7, CLX-68
CLX:EVENT-KEY type specifier, 4-3, CLX-69
CLX:EVENT-LISTEN function, 12-6, CLX-69
CLX:EVENT-MASK type specifier, 4-3, CLX-70
CLX:EVENT-MASK-CLASS type specifier, 4-3,
 CLX-70
CLX:FIND-ATOM function, 6-6, CLX-70
CLX:FONT structure, CLX-71
CLX:FONT-ALL-CHARS-EXIST-P function,
 CLX-72
CLX:FONT-ASCENT function, CLX-72
CLX:FONT-DEFAULT-CHAR function, CLX-72
CLX:FONT-DESCENT function, CLX-73
CLX:FONT-DIRECTION function, CLX-73
CLX:FONT-DISPLAY function, CLX-74
CLX:FONT-EQUAL function, CLX-74
CLX:FONT-ID function, CLX-75
CLX:FONT-MAX-BYTE1 function, CLX-75
CLX:FONT-MAX-BYTE2 function, CLX-75
CLX:FONT-MAX-CHAR function, CLX-76
CLX:FONT-MIN-BYTE1 function, CLX-76
CLX:FONT-MIN-BYTE2 function, CLX-77
CLX:FONT-MIN-CHAR function, CLX-77
CLX:FONT-NAME function, CLX-78
CLX:FONT-P function, CLX-78
CLX:FONT-PATH function, 11-5, CLX-79
CLX:FONT-PROPERTIES function, CLX-79

CLX:FONT-PROPERTY function, CLX-80
 CLX:FONT-PROPS type specifier, 4-3, CLX-80
 CLX:FONTABLE type specifier, 4-3, CLX-71
 CLX:FORCE-GCONTEXT-CHANGES function, 7-6,
 CLX-80
 CLX:FREE-COLORMAP function, 8-7, CLX-81
 CLX:FREE-COLORS function, 8-7, CLX-81
 CLX:FREE-CURSOR function, 9-12, CLX-82
 CLX:FREE-GCONTEXT function, 7-5, CLX-82
 CLX:FREE-PIXMAP function, 10-2, CLX-83
 CLX:GCONTEXT structure, CLX-83
 CLX:GCONTEXT-ARC-MODE function, CLX-84
 CLX:GCONTEXT-BACKGROUND function, CLX-85
 CLX:GCONTEXT-CACHE-P function, CLX-85
 CLX:GCONTEXT-CAP-STYLE function, CLX-86
 CLX:GCONTEXT-CLIP-MASK function, CLX-86
 CLX:GCONTEXT-CLIP-X function, CLX-87
 CLX:GCONTEXT-CLIP-Y function, CLX-87
 CLX:GCONTEXT-DASH-OFFSET function, CLX-88
 CLX:GCONTEXT-DASHES function, CLX-88
 CLX:GCONTEXT-DISPLAY function, CLX-89
 CLX:GCONTEXT-EQUAL function, CLX-89
 CLX:GCONTEXT-EXPOSURES function, CLX-90
 CLX:GCONTEXT-FILL-RULE function, CLX-90
 CLX:GCONTEXT-FILL-STYLE function, CLX-91
 CLX:GCONTEXT-FONT function, CLX-91
 CLX:GCONTEXT-FOREGROUND function, CLX-92
 CLX:GCONTEXT-FUNCTION function, CLX-92
 CLX:GCONTEXT-ID function, CLX-93
 CLX:GCONTEXT-JOIN-STYLE function, CLX-93
 CLX:GCONTEXT-KEY type specifier, 4-3, CLX-94
 CLX:GCONTEXT-LINE-STYLE function, CLX-94
 CLX:GCONTEXT-LINE-WIDTH function, CLX-94
 CLX:GCONTEXT-P function, CLX-95
 CLX:GCONTEXT-PLANE-MASK function, CLX-95
 CLX:GCONTEXT-STIPPLE function, CLX-96
 CLX:GCONTEXT-SUBWINDOW-MODE function,
 CLX-96
 CLX:GCONTEXT-TILE function, CLX-97
 CLX:GCONTEXT-TS-X function, CLX-97
 CLX:GCONTEXT-TS-Y function, CLX-98
 CLX:GET-IMAGE function, 10-6, CLX-98
 CLX:GET-PROPERTY function, 6-5, CLX-99
 CLX:GET-RAW-IMAGE function, 10-6, CLX-100
 CLX:GET-WM-CLASS function, CLX-101
 CLX:GLOBAL-POINTER-POSITION function,
 CLX-101
 CLX:GRAB-BUTTON function, 12-11, CLX-102
 CLX:GRAB-KEY function, 12-13, CLX-103
 CLX:GRAB-KEYBOARD function, 12-12, CLX-104
 CLX:GRAB-POINTER function, 12-11, CLX-104
 CLX:GRAB-SERVER function, 12-14, CLX-105
 CLX:GRAB-STATUS type specifier, 4-3, CLX-106
 CLX:ICON-SIZES function, CLX-106
 CLX:IMAGE structure, CLX-106
 CLX:IMAGE-BIT-LSB-FIRST-P function,
 CLX-107
 CLX:IMAGE-BITS-PER-PIXEL function,
 CLX-108
 CLX:IMAGE-BLUE-MASK function, CLX-108
 CLX:IMAGE-BYTE-LSB-FIRST-P function,
 CLX-109
 CLX:IMAGE-BYTES-PER-LINE function,
 CLX-109
 CLX:IMAGE-DEPTH function, CLX-110
 CLX:IMAGE-DEPTH type specifier, 4-3, CLX-110
 CLX:IMAGE-FORMAT function, CLX-110
 CLX:IMAGE-GREEN-MASK function, CLX-111
 CLX:IMAGE-HEIGHT function, CLX-111
 CLX:IMAGE-P function, CLX-112
 CLX:IMAGE-RED-MASK function, CLX-112
 CLX:IMAGE-SCANLINE-PAD function, CLX-112
 CLX:IMAGE-WIDTH function, CLX-113
 CLX:INDEX-SIZE type specifier, 4-3, CLX-113
 CLX:INPUT-FOCUS function, CLX-114
 CLX:INSTALL-COLORMAP function, 8-2, CLX-114
 CLX:INSTALLED-COLORMAPS function, 8-2,
 CLX-115
 CLX:INTERN-ATOM function, 6-7, CLX-115
 CLX:KEYBOARD-CONTROL function, 13-5,
 CLX-116
 CLX:KEYBOARD-MAPPING function, 13-2,
 CLX-116
 CLX:KEYCODE->CHARACTER function, 13-4,
 CLX-117
 CLX:KEYCODE->KEYSYM function, 13-4, CLX-118
 CLX:KEYSYM type specifier, 4-3, CLX-119
 CLX:KEYSYM->CHARACTER function, 13-4,
 CLX-119
 CLX:KEYSYM->KEYCODES function, 13-4,
 CLX-120
 CLX:KILL-CLIENT function, 13-10, CLX-120
 CLX:KILL-TEMPORARY-CLIENTS function,
 13-10, CLX-121
 CLX:LIST-EXTENSIONS function, 13-11,
 CLX-121
 CLX:LIST-FONT-NAMES function, 11-3, CLX-122
 CLX:LIST-FONTS function, 11-4, CLX-122
 CLX:LIST-PROPERTIES function, 6-6, CLX-123
 CLX:LOGICAL-OP type specifier, 4-3, CLX-124
 CLX:LOOKUP-COLOR function, 8-8, CLX-124
 CLX:MAKE-COLOR function, 8-4, CLX-125
 CLX:MAKE-COLORMAP function, CLX-125
 CLX:MAKE-CURSOR function, CLX-126
 CLX:MAKE-DRAWABLE function, CLX-126
 CLX:MAKE-EVENT-KEYS function, 12-4, CLX-127
 CLX:MAKE-EVENT-MASK function, 12-4, CLX-127
 CLX:MAKE-FONT function, CLX-128
 CLX:MAKE-GCONTEXT function, CLX-128
 CLX:MAKE-PIXMAP function, CLX-129
 CLX:MAKE-STATE-KEYS function, 12-10,
 CLX-129
 CLX:MAKE-STATE-MASK function, 12-10,
 CLX-130
 CLX:MAKE-VISUAL-INFO function, CLX-130
 CLX:MAKE-WINDOW function, CLX-130
 CLX:MAKE-WM-HINTS function, CLX-131
 CLX:MAKE-WM-SIZE-HINTS function, CLX-131
 CLX:MAP-SUBWINDOWS function, 6-4, CLX-132
 CLX:MAP-WINDOW function, 6-4, CLX-132
 CLX:MASK16 type specifier, 4-3, CLX-133
 CLX:MASK32 type specifier, 4-4, CLX-133
 CLX:MAX-CHAR-ASCENT function, CLX-133
 CLX:MAX-CHAR-ATTRIBUTES function, CLX-134
 CLX:MAX-CHAR-DESCENT function, CLX-134
 CLX:MAX-CHAR-LEFT-BEARING function,
 CLX-135
 CLX:MAX-CHAR-RIGHT-BEARING function,
 CLX-135
 CLX:MAX-CHAR-WIDTH function, CLX-135
 CLX:MIN-CHAR-ASCENT function, CLX-136
 CLX:MIN-CHAR-ATTRIBUTES function, CLX-136
 CLX:MIN-CHAR-DESCENT function, CLX-137

CLX:MIN-CHAR-LEFT-BEARING function,
 CLX-137
 CLX:MIN-CHAR-RIGHT-BEARING function,
 CLX-138
 CLX:MIN-CHAR-WIDTH function, CLX-138
 CLX:MODIFIER-KEY type specifier, 4-4, CLX-138
 CLX:MODIFIER-MAPPING function, 13-3,
 CLX-139
 CLX:MODIFIER-MASK type specifier, 4-4,
 CLX-139
 CLX:MOTION-EVENTS function, CLX-140
 CLX:OPEN-DISPLAY function, 5-1, CLX-140
 CLX:OPEN-FONT function, 11-4, CLX-141
 CLX:PIXARRAY type specifier, 4-4, CLX-141
 CLX:PIXEL type specifier, 4-4, CLX-142
 CLX:PIXMAP structure, CLX-142
 CLX:PIXMAP-DISPLAY function, CLX-142
 CLX:PIXMAP-EQUAL function, CLX-143
 CLX:PIXMAP-FORMAT structure, CLX-143
 CLX:PIXMAP-FORMAT-BITS-PER-PIXEL
 function, CLX-144
 CLX:PIXMAP-FORMAT-DEPTH function, CLX-144
 CLX:PIXMAP-FORMAT-P function, CLX-145
 CLX:PIXMAP-FORMAT-PAD function, CLX-145
 CLX:PIXMAP-ID function, CLX-146
 CLX:PIXMAP-P function, CLX-146
 CLX:POINT-SEQ type specifier, 4-4, CLX-149
 CLX:POINTER-CONTROL function, 13-6, CLX-146
 CLX:POINTER-EVENT-MASK type specifier, 4-4,
 CLX-147
 CLX:POINTER-EVENT-MASK-CLASS type
 specifier, 4-4, CLX-147
 CLX:POINTER-MAPPING function, 13-3, CLX-147
 CLX:POINTER-POSITION function, CLX-148
 CLX:PROCESS-EVENT function, 12-6, CLX-149
 CLX:PUT-IMAGE function, 10-6, CLX-150
 CLX:PUT-RAW-IMAGE function, 10-7, CLX-151
 CLX:QUERY-BEST-CURSOR function, 9-12,
 CLX-152
 CLX:QUERY-BEST-STIPPLE function, CLX-152
 CLX:QUERY-BEST-TILE function, CLX-153
 CLX:QUERY-COLORS function, 8-8, CLX-153
 CLX:QUERY-EXTENSION function, 13-11,
 CLX-154
 CLX:QUERY-KEYMAP function, CLX-154
 CLX:QUERY-POINTER function, CLX-155
 CLX:QUERY-TREE function, 6-10, CLX-156
 CLX:QUEUE-EVENT function, 12-9, CLX-156
 CLX:READ-BITMAP-FILE function, 10-3,
 CLX-157
 CLX:RECOLOR-CURSOR function, 9-12, CLX-158
 CLX:RECT-SEQ type specifier, 4-4, CLX-158
 CLX:REMOVE-ACCESS-HOST function, 13-9,
 CLX-158
 CLX:REMOVE-FROM-SAVE-SET function, 6-4,
 13-11, CLX-159
 CLX:REPARTENT-WINDOW function, 6-10, 13-1,
 CLX-159
 CLX:REPEAT-SEQ type specifier, 4-4, CLX-160
 CLX:RESET-SCREEN-SAVER function, 13-8,
 CLX-160
 CLX:RESOURCE-ID type specifier, 4-4, CLX-160
 CLX:RGB-VAL type specifier, 4-4, CLX-161
 CLX:ROTATE-PROPERTIES function, 6-6
 CLX:SCREEN structure, CLX-161
 CLX:SCREEN-BACKING-STORES function,
 CLX-162

CLX:SCREEN-BLACK-PIXEL function, CLX-162
 CLX:SCREEN-DEFAULT-COLORMAP function,
 CLX-163
 CLX:SCREEN-DEPTH function, CLX-163
 CLX:SCREEN-DISPLAY function, CLX-164
 CLX:SCREEN-EVENT-MASK-AT-OPEN function,
 CLX-164
 CLX:SCREEN-HEIGHT function, CLX-165
 CLX:SCREEN-HEIGHT-IN-MILLIMETERS
 function, CLX-165
 CLX:SCREEN-MAX-INSTALLED-MAPS function,
 CLX-165
 CLX:SCREEN-MIN-INSTALLED-MAPS function,
 CLX-166
 CLX:SCREEN-P function, CLX-166
 CLX:SCREEN-ROOT function, CLX-167
 CLX:SCREEN-ROOT-DEPTH function, CLX-167
 CLX:SCREEN-ROOT-VISUAL function, CLX-168
 CLX:SCREEN-SAVE-UNDERS-P function,
 CLX-169
 CLX:SCREEN-SAVER function, 13-7, CLX-168
 CLX:SCREEN-WHITE-PIXEL function, CLX-169
 CLX:SCREEN-WIDTH function, CLX-170
 CLX:SCREEN-WIDTH-IN-MILLIMETERS
 function, CLX-170
 CLX:SEG-SEQ type specifier, 4-4, CLX-171
 CLX:SELECTION-OWNER function, 6-9, CLX-171
 CLX:SEND-EVENT function, 12-15, CLX-171
 CLX:SET-INPUT-FOCUS function, CLX-173
 CLX:SET-MODIFIER-MAPPING function, 13-3,
 CLX-173
 CLX:SET-SCREEN-SAVER function, 13-7,
 CLX-174
 CLX:SET-WM-CLASS function, CLX-174
 CLX:STATE-MASK-KEY type specifier, 4-4,
 CLX-175
 CLX:STORE-COLOR function, 8-6, CLX-175
 CLX:STORE-COLORS function, 8-7, CLX-176
 CLX:STRINGABLE type specifier, 4-4, CLX-176
 CLX:TEXT-EXTENTS function, 11-5, CLX-177
 CLX:TEXT-WIDTH function, 11-5, CLX-178
 CLX:TIMESTAMP type specifier, 4-4, CLX-178
 CLX:TRANSLATE-COORDINATES function,
 CLX-179
 CLX:TRANSLATE-DEFAULT function, 11-5,
 CLX-179
 CLX:UNGRAB-BUTTON function, 12-12, CLX-180
 CLX:UNGRAB-KEY function, 12-13, CLX-181
 CLX:UNGRAB-KEYBOARD function, 12-13,
 CLX-181
 CLX:UNGRAB-POINTER function, 12-12, CLX-182
 CLX:UNGRAB-SERVER function, 12-14, CLX-182
 CLX:UNINSTALL-COLORMAP function, 8-2,
 CLX-183
 CLX:UNMAP-SUBWINDOWS function, 6-5, CLX-183
 CLX:UNMAP-WINDOW function, 6-4, CLX-184
 CLX:VISUAL type specifier, 4-4, CLX-184
 CLX:VISUAL-INFO structure, CLX-184
 CLX:VISUAL-INFO-BITS-PER-RGB function,
 CLX-185
 CLX:VISUAL-INFO-BLUE-MASK function,
 CLX-185
 CLX:VISUAL-INFO-CLASS function, CLX-186
 CLX:VISUAL-INFO-COLORMAP-ENTRIES
 function, CLX-186
 CLX:VISUAL-INFO-GREEN-MASK function,
 CLX-187

CLX:VISUAL-INFO-ID function, CLX-187
 CLX:VISUAL-INFO-P function, CLX-188
 CLX:VISUAL-INFO-RED-MASK function,
 CLX-188
 CLX:WARP-POINTER function, CLX-188
 CLX:WARP-POINTER-IF-INSIDE function,
 CLX-189
 CLX:WARP-POINTER-RELATIVE function,
 CLX-190
 CLX:WARP-POINTER-RELATIVE-IF-INSIDE
 function, CLX-190
 CLX:WIN-GRAVITY type specifier, 4-4, CLX-203
 CLX:WINDOW structure, CLX-191
 CLX:WINDOW-ALL-EVENT-MASKS function,
 CLX-193
 CLX:WINDOW-BACKING-PIXEL function,
 CLX-194
 CLX:WINDOW-BACKING-PLANES function,
 CLX-194
 CLX:WINDOW-BACKING-STORE function,
 CLX-195
 CLX:WINDOW-BIT-GRAVITY function, CLX-195
 CLX:WINDOW-CLASS function, CLX-196
 CLX:WINDOW-COLORMAP function, CLX-196
 CLX:WINDOW-COLORMAP-INSTALLED-P
 function, CLX-197
 CLX:WINDOW-DISPLAY function, CLX-197
 CLX:WINDOW-DO-NOT-PROPAGATE-MASK
 function, CLX-198
 CLX:WINDOW-EQUAL function, CLX-198
 CLX:WINDOW-EVENT-MASK function, CLX-199
 CLX:WINDOW-GRAVITY function, CLX-199
 CLX:WINDOW-ID function, CLX-200
 CLX:WINDOW-MAP-STATE function, CLX-200
 CLX:WINDOW-OVERRIDE-REDIRECT function,
 CLX-201
 CLX:WINDOW-P function, CLX-201
 CLX:WINDOW-SAVE-UNDER function, CLX-202
 CLX:WINDOW-VISUAL function, CLX-202
 CLX:WITH-DISPLAY macro, CLX-203
 CLX:WITH-EVENT-QUEUE macro, 12-6, CLX-204
 CLX:WITH-GCONTEXT macro, 7-6, CLX-204
 CLX:WITH-SERVER-GRABBED macro, 12-14,
 CLX-206
 CLX:WITH-STATE macro, 6-13, CLX-207
 CLX:WM-HINTS function, CLX-208
 CLX:WM-HINTS structure, CLX-209
 CLX:WM-HINTS-FLAGS function, CLX-209
 CLX:WM-HINTS-ICON-MASK function, CLX-210
 CLX:WM-HINTS-ICON-PIXMAP function,
 CLX-210
 CLX:WM-HINTS-ICON-WINDOW function,
 CLX-211
 CLX:WM-HINTS-ICON-X function, CLX-211
 CLX:WM-HINTS-ICON-Y function, CLX-212
 CLX:WM-HINTS-INITIAL-STATE function,
 CLX-212
 CLX:WM-HINTS-INPUT function, CLX-213
 CLX:WM-HINTS-P function, CLX-213
 CLX:WM-HINTS-WINDOW-GROUP function,
 CLX-214
 CLX:WM-ICON-NAME function, CLX-214
 CLX:WM-NAME function, CLX-215
 CLX:WM-NORMAL-HINTS function, CLX-215
 CLX:WM-SIZE-HINTS structure, CLX-216
 CLX:WM-SIZE-HINTS-HEIGHT function,
 CLX-216

CLX:WM-SIZE-HINTS-HEIGHT-INC function,
 CLX-217
 CLX:WM-SIZE-HINTS-MAX-ASPECT function,
 CLX-217
 CLX:WM-SIZE-HINTS-MAX-HEIGHT function,
 CLX-218
 CLX:WM-SIZE-HINTS-MAX-WIDTH function,
 CLX-218
 CLX:WM-SIZE-HINTS-MIN-ASPECT function,
 CLX-219
 CLX:WM-SIZE-HINTS-MIN-HEIGHT function,
 CLX-219
 CLX:WM-SIZE-HINTS-MIN-WIDTH function,
 CLX-220
 CLX:WM-SIZE-HINTS-P function, CLX-220
 CLX:WM-SIZE-HINTS-USER-SPECIFIED-
 POSITION-P function, CLX-221
 CLX:WM-SIZE-HINTS-USER-SPECIFIED-
 SIZE-P function, CLX-221
 CLX:WM-SIZE-HINTS-WIDTH function, CLX-222
 CLX:WM-SIZE-HINTS-WIDTH-INC function,
 CLX-222
 CLX:WM-SIZE-HINTS-X function, CLX-223
 CLX:WM-SIZE-HINTS-Y function, CLX-223
 CLX:WM-ZOOM-HINTS function, CLX-224
 CLX:WRITE-BITMAP-FILE function, 10-2,
 CLX-224
 CLX:XATOM type specifier, 4-4, CLX-225
 Colormap
 definition, 8-1
 Colors, 8-1 to 8-8
 for exclusive use, 8-4 to 8-7
 freeing, 8-7
 named, 8-3
 querying, 8-8
 sharing resources, 8-3 to 8-4
 storing, 8-6 to 8-7
 Composite widget, 1-7
 Compound string function
 DWT:ADD-FONT-LIST, DWT-3
 DWT:CREATE-FONT-LIST, DWT-21
 DWT:CS-STRING, DWT-23
 DWT:CSBYTECMP, DWT-22
 DWT:CSEMPTRY, DWT-23
 DWT:CSTRCAT, DWT-23
 DWT:CSTRCPY, DWT-24
 DWT:CSTRLEN, DWT-24
 DWT:CSTRNCAT, DWT-24
 DWT:CSTRNCPY, DWT-25
 DWT:GET-NEXT-SEGMENT, DWT-40
 DWT:INIT-GET-SEGMENT, DWT-45
 DWT:LATIN1-STRING, DWT-51
 DWT:XTSTRING, DWT-104
 Constants
 callback reasons, A-5
 DECwindows resource manager (DRM), A-3
 DECwindows toolkit, A-1
 CONTROLS attribute keyword
 User interface language, 2-6
 CONTROLS section
 User interface language, 2-5
 Convenience function
 DWT:CALLBACK-NONEXCLUSIVE, DWT-11
 DWT:CHILDREN, DWT-14
 DWT:DISPLAY-CS-MESSAGE, DWT-29
 DWT:DISPLAY-VMS-MESSAGE, DWT-30
 DWT:GET-DISPLAY, DWT-39

Convenience function (Cont.)

DWT:GET-SCREEN, DWT-41
DWT:NUMBER-CHILDREN, DWT-65
DWT:VMS-CLEAR-STRING, DWT-100
DWT:VMS-FREE-ARGNAMES, DWT-100
DWT:VMS-SET-ARG, DWT-100
DWT:VMS-SET-CALLBACK-ARG, DWT-101
DWT:VMS-SET-DESC-ARG, DWT-101
Cursors, 9-10 to 9-12
 wait, 9-11
Cut-and-paste function, 1-2
 DWT:BEGIN-COPY-TO-CLIPBOARD, DWT-9
 DWT:CANCEL-COPY-FORMAT, DWT-12
 DWT:CANCEL-COPY-TO-CLIPBOARD,
 DWT-18
 DWT:CLIPBOARD-LOCK, DWT-15
 DWT:CLIPBOARD-UNLOCK, DWT-15
 DWT:COPY-FROM-CLIPBOARD, DWT-19
 DWT:COPY-TO-CLIPBOARD, DWT-19
 DWT:END-COPY-TO-CLIPBOARD, DWT-34
 DWT:INQUIRE-NEXT-PASTE-COUNT,
 DWT-46
 DWT:INQUIRE-NEXT-PASTE-FORMAT,
 DWT-47
 DWT:INQUIRE-NEXT-PASTE-LENGTH,
 DWT-47
 DWT:LIST-PENDING-ITEMS, DWT-56
 DWT:RECOPY-TO-CLIPBOARD, DWT-75
 DWT:UNDO-COPY-TO-CLIPBOARD, DWT-98

D

:DESTROY-NOTIFY event-key, 12-20
Data types
 in CLX:, 4-2 to 4-4
Debugging
 CLX programs, 5-4
DECnet, 4-1
DECW\$CURSOR font, in CLX, 9-11
DECW-PUBLIC.LSP functions
 DWT+:CREATE-ARGLIST, 2-9
 DWT+:CREATE-WIDGET-LIST, 3-3
 DWT+:GET-CALLBACK-TAG, 3-7
 DWT+:GET-CALLBACK-WIDGET, 3-6
 DWT+:MAKE-REGISTER-LIST, 3-5
DECwindows callback function, 3-4
 access callback argument list
 callback structure, 3-9
 accessing callback argument list, 3-6 to 3-9
 user-defined data, 3-7
 widget ID, 3-6
 declaring, 3-4 to 3-5
DECwindows Resource Manager (DRM), 1-2
 constants, A-3
DECwindows Session Manager
 keyboard control, 13-2
 reparents windows, 6-3
 screen saver, 13-7
 security, 13-8
DECwindows toolkit
 building an application, 3-1
 accessing the user interface, 3-1 to 3-4
 writing callback functions, 3-4 to 3-9
 calling application functions, 1-7 to 1-8
 components, 1-2 to 1-3
 constants, A-1 to A-6
 creating a user interface, 1-7, 2-1

DECwindows toolkit

creating a user interface (Cont.)
 with high-level functions, 2-7 to 2-9
 with low-level functions, 2-9 to 2-10
 with the UIL, 2-3 to 2-7
initializing, 3-1
overview, 1-1
program examples
 DECW-PUBLIC.LSP functions
 DWT+:CREATE-ARGLIST, 2-9
 DWT+:CREATE-WIDGET-LIST, 3-3
 DWT+:GET-CALLBACK-TAG, 3-7
 DWT+:GET-CALLBACK-WIDGET, 3-6
 DWT+:MAKE-REGISTER-LIST, 3-5
 Hello World!, 1-5
 recipe application, 2-1
VAX LISP interface to, 1-3
widgets, 1-3 to 1-6
DECwindows window manager, 6-1, 6-8
DECwindows X server
 GContext limitations, 7-6
Depth
 definition, 8-1
Display
 closing, 5-4 to 5-5
 information about, 5-2 to 5-3
 opening, 5-1 to 5-2
 requests, 5-4
Drawing, 9-1 to 9-12
 arcs, 9-7 to 9-9
 areas, 9-9 to 9-10
 cursors, 9-10 to 9-12
 images, 10-3 to 10-7
 lines, 9-4 to 9-6
 points, 9-1 to 9-4
 rectangles, 9-6 to 9-7
 text, 11-6 to 11-9
DRM function
 DWT:CLOSE-HIERARCHY, DWT-16
 DWT:DRM-FREE-RESOURCE-CONTEXT,
 DWT-30
 DWT:DRM-GET-RESOURCE-CONTEXT,
 DWT-31
 DWT:DRM-HGET-INDEXED-LITERAL,
 DWT-31
 DWT:DRM-RC-BUFFER, DWT-32
 DWT:DRM-RC-SET-TYPE, DWT-32
 DWT:DRM-RC-SIZE, DWT-32
 DWT:DRM-RC-TYPE, DWT-33
 DWT:FETCH-INTERFACE-MODULE, DWT-35
 DWT:FETCH-SET-VALUES, DWT-35
 DWT:FETCH-WIDGET, DWT-36
 DWT:FETCH-WIDGET-OVERRIDE, DWT-36
 DWT:INITIALIZE-DRM, DWT-46
 DWT:OPEN-HIERARCHY, DWT-66
 DWT:REGISTER-CLASS, DWT-76
 DWT:REGISTER-DRM-NAMES, DWT-76
 using
 DWT:FETCH-WIDGET, 3-3
 DWT:INITIALIZE-DRM, 3-2
 DWT:OPEN-HIERARCHY, 3-2
 DWT:REGISTER-DRM-NAMES, 3-2, 3-4
 DWT+:CREATE-ARGLIST LISP function, 2-9
 DWT+:CREATE-WIDGET-LIST LISP function, 3-3
 DWT+:GET-CALLBACK-TAG LISP function, 3-7
 DWT+:GET-CALLBACK-WIDGET LISP function,
 3-6

DWT+:MAKE-REGISTER-LIST LISP function, DWT-5
 DWT:ADD-ACTIONS function, DWT-1
 DWT:ADD-CALLBACK function, DWT-1
 DWT:ADD-CALLBACKS function, DWT-1
 DWT:ADD-CONVERTER function, DWT-2
 DWT:ADD-EVENT-HANDLER function, DWT-2
 DWT:ADD-EXPOSURE-TO-REGION function,
 DWT-2
 DWT:ADD-FONT-LIST function, DWT-3
 DWT:ADD-GRAB function, DWT-3
 DWT:ADD-INPUT function, DWT-3
 DWT:ADD-RAW-EVENT-HANDLER function,
 DWT-4
 DWT:ADD-TIME-OUT function, DWT-4
 DWT:ADD-WORK-PROC function, DWT-4
 DWT:APP-ADD-INPUT function, DWT-5
 DWT:APP-ADD-TIME-OUT function, DWT-5
 DWT:APP-ADD-WORK-PROC function, DWT-5
 DWT:APP-CREATE-SHELL function, DWT-6
 DWT:APP-MAIN-LOOP function, DWT-6
 DWT:APP-NEXT-EVENT function, DWT-6
 DWT:APP-PEEK-EVENT function, DWT-7
 DWT:APP-PENDING function, DWT-7
 DWT:APP-PROCESS-EVENT function, DWT-7
 DWT:ATTACHED-DB function, DWT-8
 DWT:ATTACHED-DB-CREATE function, DWT-8
 DWT:ATTACHED-DB-POPUP-CREATE function,
 DWT-8
 DWT:AUGMENT-TRANSLATIONS function, DWT-9
 DWT:BEGIN-COPY-TO-CLIPBOARD function,
 DWT-9
 DWT:BUILD-EVENT-MASK function, DWT-10
 DWT:CALL-ACCEPT-FOCUS function, DWT-10
 DWT:CALL-CALLBACKS function, DWT-10
 DWT:CALLBACK-EXCLUSIVE function, DWT-11
 DWT:CALLBACK-NONE function, DWT-11
 DWT:CALLBACK-NONEEXCLUSIVE function,
 DWT-11
 DWT:CALLBACK-POPDOWN function, DWT-12
 DWT:CALLOC function, DWT-12
 DWT:CANCEL-COPY-FORMAT function, DWT-12
 DWT:CANCEL-COPY-TO-CLIPBOARD function,
 DWT-13
 DWT:CAUTION-BOX function, DWT-13
 DWT:CAUTION-BOX-CREATE function, DWT-14
 DWT:CHILDREN function, DWT-14
 DWT:CLASS function, DWT-14
 DWT:CLIPBOARD-LOCK function, DWT-15
 DWT:CLIPBOARD-UNLOCK function, DWT-15
 DWT:CLOSE-DISPLAY function, DWT-15
 DWT:CLOSE-HIERARCHY function, DWT-16
 DWT:COMMAND-APPEND function, DWT-16
 DWT:COMMAND-ERROR-MESSAGE function,
 DWT-16
 DWT:COMMAND-SET function, DWT-17
 DWT:COMMAND-WINDOW function, DWT-17
 DWT:COMMAND-WINDOW-CREATE function,
 DWT-17
 DWT:CONFIGURE-WIDGET function, DWT-18
 DWT:CONVERT function, DWT-18
 DWT:CONVERT-CASE function, DWT-18
 DWT:COPY-FROM-CLIPBOARD function, DWT-19
 DWT:COPY-TO-CLIPBOARD function, DWT-19
 DWT:CREATE-APPLICATION-CONTEXT function,
 DWT-20
 DWT:CREATE-APPLICATION-SHELL function,
 DWT-20
 DWT:CREATE-FONT-LIST function, DWT-21
 DWT:CREATE-MANAGED-WIDGET function,
 DWT-21
 DWT:CREATE-POPUP-SHELL function, DWT-21
 DWT:CREATE-WIDGET function, DWT-22
 DWT:CREATE-WINDOW function, DWT-22
 DWT:CS-STRING function, DWT-23
 DWT:CSBYTECMP function, DWT-23
 DWT:CSEMPTRY function, DWT-23
 DWT:CSTRCAT function, DWT-23
 DWT:CSTRCPY function, DWT-24
 DWT:CSTRLEN function, DWT-24
 DWT:CSTRNCAT function, DWT-24
 DWT:CSTRNCPY function, DWT-25
 DWT:DATABASE function, DWT-25
 DWT:DESTROY-APPLICATION-CONTEXT
 function, DWT-25
 DWT:DESTROY-GC function, DWT-26
 DWT:DESTROY-WIDGET function, DWT-26
 DWT:DIALOG-BOX function, DWT-26
 DWT:DIALOG-BOX-CREATE function, DWT-27
 DWT:DIALOG-BOX-POPUP-CREATE function,
 DWT-27
 DWT:DIRECT-CONVERT function, DWT-27
 DWT:DISOWN-SELECTION function, DWT-28
 DWT:DISPATCH-EVENT function, DWT-28
 DWT:DISPLAY function, DWT-28
 DWT:DISPLAY-CS-MESSAGE function, DWT-29
 DWT:DISPLAY-INITIALIZE function, DWT-29
 DWT:DISPLAY-VMS-MESSAGE function, DWT-30
 DWT:DRM-FREE-RESOURCE-CONTEXT function,
 DWT-30
 DWT:DRM-GET-RESOURCE-CONTEXT function,
 DWT-31
 DWT:DRM-HGET-INDEXED-LITERAL function,
 DWT-31
 DWT:DRM-RC-BUFFER function, DWT-32
 DWT:DRM-RC-SET-TYPE function, DWT-32
 DWT:DRM-RC-SIZE function, DWT-32
 DWT:DRM-RC-TYPE function, DWT-33
 DWT:DWTHELP function, DWT-33
 DWT:DWTWINDOW function, DWT-33
 DWT:END-COPY-TO-CLIPBOARD function,
 DWT-34
 DWT:ERROR-MSG function, DWT-34
 DWT:FETCH-INTERFACE-MODULE function,
 DWT-35
 DWT:FETCH-SET-VALUES function, DWT-35
 DWT:FETCH-WIDGET function, 3-3, DWT-36
 DWT:FETCH-WIDGET-OVERRIDE function,
 DWT-36
 DWT:FILE-SELECTION function, DWT-37
 DWT:FILE-SELECTION-CREATE function,
 DWT-37
 DWT:FILE-SELECTION-DO-SEARCH function,
 DWT-38
 DWT:FREE function, DWT-38
 DWT:GET-APPLICATION-RESOURCES function,
 DWT-38
 DWT:GET-DISPLAY function, DWT-39
 DWT:GET-ERROR-DATABASE function, DWT-39
 DWT:GET-ERROR-DATABASE-TEXT function,
 DWT-39
 DWT:GET-GC function, DWT-40
 DWT:GET-NEXT-SEGMENT function, DWT-40
 DWT:GET-RESOURCE-LIST function, DWT-41
 DWT:GET-SCREEN function, DWT-41

DWT:GET-SELECTION-TIMEOUT function,
DWT-41
DWT:GET-SELECTION-VALUE function, DWT-41
DWT:GET-SELECTION-VALUE-INCR function,
DWT-42
DWT:GET-SELECTION-VALUES function,
DWT-42
DWT:GET-SELECTION-VALUES-INCR function,
DWT-43
DWT:GET-SUBRESOURCES function, DWT-43
DWT:GET-SUBVALUES function, DWT-44
DWT:GET-VALUES function, DWT-44
DWT:HAS-CALLBACKS function, DWT-44
DWT:HELP function
 See XTHELP function
DWT:HELP-CREATE function, DWT-45
DWT:INIT-GET-SEGMENT function, DWT-45
DWT:INITIALIZE function, 3-1, DWT-46
DWT:INITIALIZE-DRM function, 3-2, DWT-46
DWT:INQUIRE-NEXT-PASTE-COUNT function,
DWT-46
DWT:INQUIRE-NEXT-PASTE-FORMAT function,
DWT-47
DWT:INQUIRE-NEXT-PASTE-LENGTH function,
DWT-47
DWT:INSTALL-ACCELERATORS function,
DWT-48
DWT:INSTALL-ALL-ACCELERATORS function,
DWT-48
DWT:IS-COMPOSITE function, DWT-49
DWT:IS-MANAGED function, DWT-49
DWT:IS-REALIZED function, DWT-49
DWT:IS-SENSITIVE function, DWT-49
DWT:IS-SUBCLASS function, DWT-50
DWT:LABEL function, DWT-50
DWT:LABEL-CREATE function, DWT-50
DWT:LABEL-GADGET-CREATE function, DWT-51
DWT:LATIN1-STRING function, DWT-51
DWT:LIST-BOX function, DWT-51
DWT:LIST-BOX-ADD-ITEM function, DWT-52
DWT:LIST-BOX-CREATE function, DWT-52
DWT:LIST-BOX-DELETE-ITEM function,
DWT-53
DWT:LIST-BOX-DELETE-POS function, DWT-53
DWT:LIST-BOX-DESELECT-ALL-ITEMS
function, DWT-53
DWT:LIST-BOX-DESELECT-ITEM function,
DWT-54
DWT:LIST-BOX-DESELECT-POS function,
DWT-54
DWT:LIST-BOX-ITEM-EXISTS function,
DWT-54
DWT:LIST-BOX-SELECT-ITEM function,
DWT-55
DWT:LIST-BOX-SELECT-POS function, DWT-55
DWT:LIST-BOX-SET-HORIZ-POS function,
DWT-55
DWT:LIST-BOX-SET-ITEM function, DWT-56
DWT:LIST-BOX-SET-POS function, DWT-56
DWT:LIST-PENDING-ITEMS function, DWT-56
DWT:MAIN-LOOP function, 3-4, DWT-57
DWT:MAIN-WINDOW function, 2-8
DWT:MAIN-WINDOW function, DWT-57
DWT:MAIN-WINDOW-CREATE function, DWT-57
DWT:MAIN-WINDOW-SET-AREAS function,
DWT-58
DWT:MAKE-CALL-BACK-ROUTINE function, 3-4

DWT:MAKE-GEOMETRY-REQUEST function,
DWT-58
DWT:MAKE-RESIZE-REQUEST function, DWT-59
DWT:ALLOC function, DWT-59
DWT:MANAGE-CHILD function, 3-2, 3-3, DWT-59
DWT:MANAGE-CHILDREN function, 3-3, DWT-60
DWT:MAP-WIDGET function, DWT-60
DWT:MENU function, 2-9, DWT-60
DWT:MENU-BAR function, 2-9, DWT-61
DWT:MENU-BAR-CREATE function, DWT-61
DWT:MENU-CREATE function, DWT-61
DWT:MENU-POPUP-CREATE function, DWT-62
DWT:MENU-POSITION function, DWT-62
DWT:MENU-PULL-DOWN function, 2-9
DWT:MENU-PULLDOWN-CREATE function,
DWT-62
DWT:MERGE-ARG-LISTS function, DWT-63
DWT:MESSAGE-BOX function, DWT-63
DWT:MESSAGE-BOX-CREATE function, 2-10,
DWT-64
DWT:MOVE-WIDGET function, DWT-64
DWT:NAME-TO-WIDGET function, DWT-64
DWT:NEXT-EVENT function, DWT-65
DWT:NUMBER-CHILDREN function, DWT-65
DWT:OPEN-DISPLAY function, DWT-65
DWT:OPEN-HIERARCHY function, 3-2, DWT-66
DWT:OPTION-MENU function, DWT-66
DWT:OPTION-MENU-CREATE function, DWT-66
DWT:OVERRIDE-TRANSLATIONS function,
DWT-67
DWT:OWN-SELECTION function, DWT-67
DWT:OWN-SELECTION-INCREMENTAL function,
DWT-68
DWT:PARENT function, DWT-68
DWT:PARSE-ACCELERATOR-TABLE function,
DWT-68
DWT:PARSE-TRANSLATION-TABLE function,
DWT-69
DWT:PEEK-EVENT function, DWT-69
DWT:PENDING function, DWT-69
DWT:POPDOWN function, DWT-70
DWT:POPUP function, DWT-70
DWT:PROCESS-EVENT function, DWT-70
DWT:PULL-DOWN-MENU-ENTRY function, 2-9,
DWT-71
DWT:PULL-DOWN-MENU-ENTRY-CREATE
function, DWT-71
DWT:PULL-DOWN-MENU-ENTRY-HILITE
function, DWT-71
DWT:PUSH-BUTTON function, 2-9, DWT-72
DWT:PUSH-BUTTON-CREATE function, DWT-72
DWT:PUSH-BUTTON-GADGET-CREATE function,
DWT-73
DWT:QUERY-GEOMETRY function, DWT-73
DWT:RADIO-BOX function, DWT-73
DWT:RADIO-BOX-CREATE function, DWT-74
DWT:REALIZE-WIDGET function, 3-3, DWT-74
DWT:REALLOC function, DWT-74
DWT:RECOPY-TO-CLIPBOARD function, DWT-75
DWT:REGISTER-CASE-CONVERTER function,
DWT-75
DWT:REGISTER-CLASS function, DWT-76
DWT:REGISTER-DRM-NAMES function, 3-2, 3-4,
DWT-76
DWT:REMOVE-ALL-CALLBACKS function,
DWT-76
DWT:REMOVE-CALLBACK function, DWT-77

DWT:REMOVE-CALLBACKS function, DWT-77
DWT:REMOVE-EVENT-HANDLER function,
 DWT-77
DWT:REMOVE-GRAB function, DWT-78
DWT:REMOVE-INPUT function, DWT-78
DWT:REMOVE-RAW-EVENT-HANDLER function,
 DWT-78
DWT:REMOVE-TIME-OUT function, DWT-79
DWT:REMOVE-WORK-PROC function, DWT-79
DWT:RESIZE-WIDGET function, DWT-79
DWT:RESIZE-WINDOW function, DWT-80
DWT:S-SET-TEXT-STRING function, 3-7
DWT:S-TEXT function, 2-9, DWT-91
DWT:S-TEXT-CLEAR-SELECTION function,
 DWT-91
DWT:S-TEXT-CREATE function, DWT-91
DWT:S-TEXT-GET-EDITABLE function, DWT-92
DWT:S-TEXT-GET-MAX-LENGTH function,
 DWT-92
DWT:S-TEXT-GET-SELECTION function,
 DWT-92
DWT:S-TEXT-GET-STRING function, DWT-93
DWT:S-TEXT-REPLACE function, DWT-93
DWT:S-TEXT-SET-EDITABLE function, DWT-93
DWT:S-TEXT-SET-MAX-LENGTH function,
 DWT-94
DWT:S-TEXT-SET-SELECTION function,
 DWT-94
DWT:S-TEXT-SET-STRING function, DWT-94
DWT:SCALE function, DWT-80
DWT:SCALE-CREATE function, DWT-81
DWT:SCALE-GET-SLIDER function, DWT-81
DWT:SCALE-SET-SLIDER function, DWT-81
DWT:SCREEN function, DWT-82
DWT:SCROLL-BAR function, DWT-82
DWT:SCROLL-BAR-CREATE function, DWT-83
DWT:SCROLL-BAR-GET-SLIDER function,
 DWT-83
DWT:SCROLL-BAR-SET-SLIDER function,
 DWT-84
DWT:SCROLL-WINDOW function, DWT-84
DWT:SCROLL-WINDOW-CREATE function,
 DWT-84
DWT:SCROLL-WINDOW-SET-AREAS function,
 DWT-85
DWT:SELECTION function, DWT-85
DWT:SELECTION-CREATE function, DWT-86
DWT:SEPARATOR function, DWT-86
DWT:SEPARATOR-CREATE function, DWT-86
DWT:SEPARATOR-GADGET-CREATE function,
 DWT-87
DWT:SET-ERROR-HANDLER function, DWT-87
DWT:SET-ERROR-MSG-HANDLER function,
 DWT-87
DWT:SET-KEY-TRANSLATOR function, DWT-88
DWT:SET-KEYBOARD-FOCUS function, DWT-88
DWT:SET-MAPPED-WHEN-MANAGED function,
 DWT-88
DWT:SET-SELECTION-TIMEOUT function,
 DWT-89
DWT:SET-SENSITIVE function, DWT-89
DWT:SET-SUBVALUES function, DWT-89
DWT:SET-VALUES function, DWT-90
DWT:SET-WARNING-HANDLER function, DWT-90
DWT:SET-WARNING-MSG-HANDLER function,
 DWT-90

DWT:STRING function
 See XTSTRING function
DWT:SUPERCLASS function, DWT-95
DWT:TOGGLE-BUTTON function, DWT-95
DWT:TOGGLE-BUTTON-CREATE function,
 DWT-96
DWT:TOGGLE-BUTTON-GADGET-CREATE
 function, DWT-96
DWT:TOGGLE-BUTTON-GET-STATE function,
 DWT-96
DWT:TOGGLE-BUTTON-SET-STATE function,
 DWT-97
DWT:TOOLKIT-INITIALIZE function, DWT-97
DWT:TRANSLATE-COORDS function, DWT-97
DWT:TRANSLATE-KEYCODE function, DWT-98
DWT:UNDO-COPY-TO-CLIPBOARD function,
 DWT-98
DWT:UNINSTALL-TRANSLATIONS function,
 DWT-98
DWT:UNMANAGE-CHILD function, DWT-99
DWT:UNMANAGE-CHILDREN function, DWT-99
DWT:UNREALIZE-WIDGET function, DWT-99
DWT:VMS-CLEAR-STRING function, DWT-100
DWT:VMS-FREE-ARGNAMES function, DWT-100
DWT:VMS-SET-ARG function, DWT-100
DWT:VMS-SET-CALLBACK-ARG function,
 DWT-101
DWT:VMS-SET-DESC-ARG function, DWT-101
DWT:WARNING function
 See DWT:XTWARNING function
DWT:WARNING-MSG function, DWT-101
DWT:WIDGET-TO-APPLICATION-CONTEXT
 function, DWT-102
DWT:WINDOW function, DWT-102
DWT:WINDOW-CREATE function, DWT-102
DWT:WINDOW-TO-WIDGET function, DWT-103
DWT:WORK-BOX function, DWT-103
DWT:WORK-BOX-CREATE function, DWT-104
DWT:XTERROR function, DWT-104
DWT:XTSTRING function, DWT-104
DWT:XTWARNING function, DWT-105

E

:ENTER-NOTIFY event-key, 12-20
:EXPOSURE event-key, 12-21
Event window, in CLX, 12-1
Events
 allowing, 12-14 to 12-15
 controlling, 12-9 to 12-15
 keys, 12-16 to 12-29
 masks, 12-2
 processing, 12-5 to 12-9
 selecting, 12-1 to 12-5
 sending, 12-15 to 12-16
Extensions to CLX, 13-11

F

:FOCUS-IN event-key, 12-21
:FOCUS-OUT event-key, 12-21
Fonts, 11-1 to 11-5

G

:GRAPHICS-EXPOSURE event-key, 12-22
:GRAVITY-NOTIFY event-key, 12-22

Gadget function

DWT:LABEL-GADGET-CREATE, DWT-51
DWT:PUSH-BUTTON-GADGET-CREATE,
 DWT-73
DWT:SEPARATOR-GADGET-CREATE, DWT-87
DWT:TOGGLE-BUTTON-GADGET-CREATE,
 DWT-96
Gadgets, 1-5
 See also Widgets
Grabbing
 keyboard, 12-12 to 12-13
 pointer, 12-11 to 12-12
 server, 12-14
Graphics contexts, 7-1 to 7-6
 changing, 7-4 to 7-5
 copying, 7-5
 creating, 7-1 to 7-3
 freeing, 7-5
 using efficiently, 7-6

H

Hardware, 4-1

Hello World! DECwindows application, 1-5

High-level function, 1-2, 2-7 to 2-9

DWT:ATTACHED-DB, DWT-8
DWT:CAUTION-BOX, DWT-13
DWT:COMMAND-APPEND, DWT-16
DWT:COMMAND-ERROR-MESSAGE, DWT-16
DWT:COMMAND-SET, DWT-17
DWT:COMMAND-WINDOW, DWT-17
DWT:DIALOG-BOX, DWT-26
DWT:DWTHELP, DWT-33
DWT:FILE-SELECTION, DWT-37
DWT:FILE-SELECTION-DO-SEARCH,
 DWT-38
DWT:LABEL, DWT-50
DWT:LIST-BOX, DWT-51
DWT:LIST-BOX-ADD-ITEM, DWT-52
DWT:LIST-BOX-DELETE-ITEM, DWT-53
DWT:LIST-BOX-DELETE-POS, DWT-53
DWT:LIST-BOX-DESELECT-ALL-ITEMS,
 DWT-53
DWT:LIST-BOX-DESELECT-ITEM, DWT-54
DWT:LIST-BOX-DESELECT-POS, DWT-54
DWT:LIST-BOX-ITEM-EXISTS, DWT-54
DWT:LIST-BOX-SELECT-ITEM, DWT-55
DWT:LIST-BOX-SELECT-POS, DWT-55
DWT:LIST-BOX-SET-HORIZ-POS, DWT-55
DWT:LIST-BOX-SET-ITEM, DWT-56
DWT:LIST-BOX-SET-POS, DWT-56
DWT:MAIN-WINDOW, DWT-57
DWT:MAIN-WINDOW-SET-AREAS, DWT-58
DWT:MENU, DWT-60
DWT:MENU-BAR, DWT-61
DWT:MESSAGE-BOX, DWT-63
DWT:OPTION-MENU, DWT-66
DWT:PULL-DOWN-MENU-ENTRY, DWT-71
DWT:PULL-DOWN-MENU-ENTRY-HILITE,
 DWT-71
DWT:PUSH-BUTTON, DWT-72
DWT:RADIO-BOX, DWT-73
DWT:S-TEXT, DWT-91
DWT:S-TEXT-CLEAR-SELECTION, DWT-91
DWT:S-TEXT-GET-EDITABLE, DWT-92
DWT:S-TEXT-GET-MAX-LENGTH, DWT-92

High-level function (Cont.)

DWT:S-TEXT-GET-SELECTION, DWT-92
DWT:S-TEXT-GET-STRING, DWT-93
DWT:S-TEXT-REPLACE, DWT-93
DWT:S-TEXT-SET-EDITABLE, DWT-93
DWT:S-TEXT-SET-MAX-LENGTH, DWT-94
DWT:S-TEXT-SET-SELECTION, DWT-94
DWT:S-TEXT-SET-STRING, DWT-94
DWT:SCALE, DWT-80
DWT:SCALE-GET-SLIDER, DWT-81
DWT:SCALE-SET-SLIDER, DWT-81
DWT:SCROLL-BAR, DWT-82
DWT:SCROLL-BAR-GET-SLIDER, DWT-83
DWT:SCROLL-BAR-SET-SLIDER, DWT-84
DWT:SCROLL-WINDOW, DWT-84
DWT:SCROLL-WINDOW-SET-AREAS, DWT-85
DWT:SELECTION, DWT-85
DWT:SEPARATOR, DWT-86
DWT:TOGGLE-BUTTON, DWT-95
DWT:TOGGLE-BUTTON-GET-STATE, DWT-96
DWT:TOGGLE-BUTTON-SET-STATE, DWT-97
DWT:WORK-BOX, DWT-103
using
 DWT:LATIN1-STRING, 2-9
 DWT:MAIN-WINDOW, 2-8
 DWT:MENU, 2-9
 DWT:MENU-BAR, 2-9
 DWT:MENU-PULL-DOWN, 2-9
 DWT:MESSAGE-BOX-CREATE, 2-10
 DWT:PULL-DOWN-MENU-ENTRY, 2-9
 DWT:PUSH-BUTTON, 2-9
 DWT:S-TEXT, 2-9
 DWT:S-TEXT-SET-STRING, 3-7

Images, 10-3 to 10-7

Initializing DECwindows, 3-1

Intrinsic function, 1-2

ADD-CALLBACKS, DWT-1
DWT:ADD-ACTIONS, DWT-1
DWT:ADD-CALLBACK, DWT-1
DWT:ADD-CONVERTER, DWT-2
DWT:ADD-EVENT-HANDLER, DWT-2
DWT:ADD-EXPOSURE-TO-REGION, DWT-2
DWT:ADD-GRAB, DWT-3
DWT:ADD-INPUT, DWT-3
DWT:ADD-RAW-EVENT-HANDLER, DWT-4
DWT:ADD-TIME-OUT, DWT-4
DWT:ADD-WORK-PROC, DWT-4
DWT:APP-ADD-INPUT, DWT-5
DWT:APP-ADD-TIME-OUT, DWT-5
DWT:APP-ADD-WORK-PROC, DWT-5
DWT:APP-CREATE-SHELL, DWT-6
DWT:APP-MAIN-LOOP, DWT-6
DWT:APP-NEXT-EVENT, DWT-6
DWT:APP-PEEK-EVENT, DWT-7
DWT:APP-PENDING, DWT-7
DWT:APP-PROCESS-EVENT, DWT-7
DWT:AUGMENT-TRANSLATIONS, DWT-9
DWT:BUILD-EVENT-MASK, DWT-10
DWT:CALL-ACCEPT-FOCUS, DWT-10
DWT:CALL-CALLBACKS, DWT-10
DWT:CALLBACK-EXCLUSIVE, DWT-11
DWT:CALLBACK-NONE, DWT-11
DWT:CALLBACK-PODPDOWN, DWT-12
DWT:CALLOC, DWT-12

Intrinsic function (Cont.)

DWT:CLASS, DWT-14
DWT:CLOSE-DISPLAY, DWT-15
DWT:CONFIGURE-WIDGET, DWT-18
DWT:CONVERT, DWT-18
DWT:CONVERT-CASE, DWT-18
DWT:CREATE-APPLICATION-CONTEXT,
 DWT-20
DWT:CREATE-APPLICATION-SHELL,
 DWT-20
DWT:CREATE-MANAGED-WIDGET, DWT-21
DWT:CREATE-POPUP-SHELL, DWT-21
DWT:CREATE-WIDGET, DWT-22
DWT:CREATE-WINDOW, DWT-22
DWT:DATABASE, DWT-25
DWT:DESTROY-APPLICATION-CONTEXT,
 DWT-25
DWT:DESTROY-GC, DWT-26
DWT:DESTROY-WIDGET, DWT-26
DWT:DIRECT-CONVERT, DWT-27
DWT:DISOWN-SELECTION, DWT-28
DWT:DISPATCH-EVENT, DWT-28
DWT:DISPLAY, DWT-28
DWT:DISPLAY-INITIALIZE, DWT-29
DWT:DWTWINDOW, DWT-33
DWT:ERROR-MSG, DWT-34
DWT:FREE, DWT-38
DWT:GET-APPLICATION-RESOURCES,
 DWT-38
DWT:GET-ERROR-DATABASE, DWT-39
DWT:GET-ERROR-DATABASE-TEXT, DWT-39
DWT:GET-GC, DWT-40
DWT:GET-RESOURCE-LIST, DWT-41
DWT:GET-SELECTION-TIMOUT, DWT-41
DWT:GET-SELECTION-VALUE, DWT-41
DWT:GET-SELECTION-VALUE-INCR,
 DWT-42
DWT:GET-SELECTION-VALUES, DWT-42
DWT:GET-SELECTION-VALUES-INCR,
 DWT-43
DWT:GET-SUBRESOURCES, DWT-43
DWT:GET-SUBVALUES, DWT-44
DWT:GET-VALUES, DWT-44
DWT:HAS-CALLBACKS, DWT-44
DWT:INITIALIZE, DWT-46
DWT:INSTALL-ACCELERATORS, DWT-48
DWT:INSTALL-ALL-ACCELERATORS,
 DWT-48
DWT:IS-COMPOSITE, DWT-49
DWT:IS-MANAGED, DWT-49
DWT:IS-REALIZED, DWT-49
DWT:IS-SENSITIVE, DWT-49
DWT:IS-SUBCLASS, DWT-50
DWT:MAIN-LOOP, DWT-57
DWT:MAKE-GEOMETRY-REQUEST, DWT-58
DWT:MAKE-RESIZE-REQUEST, DWT-59
DWT:MALLOC, DWT-59
DWT:MANAGE-CHILD, DWT-59
DWT:MANAGE-CHILDREN, DWT-60
DWT:MAP-WIDGET, DWT-60
DWT:MERGE-ARG-LISTS, DWT-63
DWT:MOVE-WIDGET, DWT-64
DWT:NAME-TO-WIDGET, DWT-64
DWT:NEXT-EVENT, DWT-65
DWT:OPEN-DISPLAY, DWT-65
DWT:OVERRIDE-TRANSLATIONS, DWT-67
DWT:OWN-SELECTION, DWT-67

Intrinsic function (Cont.)

DWT:OWN-SELECTION-INCREMENTAL,
 DWT-68
DWT:PARENT, DWT-68
DWT:PARSE-ACCELERATOR-TABLE, DWT-68
DWT:PARSE-TRANSLATION-TABLE, DWT-69
DWT:PEEK-EVENT, DWT-69
DWT:PENDING, DWT-69
DWT:PODPDOWN, DWT-70
DWT:POPUP, DWT-70
DWT:PROCESS-EVENT, DWT-70
DWT:QUERY-GEOMETRY, DWT-73
DWT:REALIZE-WIDGET, DWT-74
DWT:REALLOC, DWT-74
DWT:REGISTER-CASE-CONVERTER, DWT-75
DWT:REMOVE-ALL-CALLBACKS, DWT-76
DWT:REMOVE-CALLBACK, DWT-77
DWT:REMOVE-CALLBACKS, DWT-77
DWT:REMOVE-EVENT-HANDLER, DWT-77
DWT:REMOVE-GRAB, DWT-78
DWT:REMOVE-INPUT, DWT-78
DWT:REMOVE-RAW-EVENT-HANDLER,
 DWT-78
DWT:REMOVE-TIME-OUT, DWT-79
DWT:REMOVE-WORK-PROC, DWT-79
DWT:RESIZE-WIDGET, DWT-79
DWT:RESIZE-WINDOW, DWT-80
DWT:SCREEN, DWT-82
DWT:SET-ERROR-HANDLER, DWT-87
DWT:SET-ERROR-MSG-HANDLER, DWT-87
DWT:SET-KEY-TRANSLATOR, DWT-88
DWT:SET-KEYBOARD-FOCUS, DWT-88
DWT:SET-MAPPED-WHEN-MANAGED, DWT-88
DWT:SET-SELECTION-TIMEOUT, DWT-89
DWT:SET-SENSITIVE, DWT-89
DWT:SET-SUBVALUES, DWT-89
DWT:SET-VALUES, DWT-90
DWT:SET-WARNING-HANDLER, DWT-90
DWT:SET-WARNING-MSG-HANDLER, DWT-90
DWT:SUPERCLASS, DWT-95
DWT:TOOLKIT-INITIALIZE, DWT-97
DWT:TRANSLATE-COORDS, DWT-97
DWT:TRANSLATE-KEYCODE, DWT-98
DWT:UNINSTALL-TRANSLATIONS, DWT-98
DWT:UNMANAGE-CHILD, DWT-99
DWT:UNMANAGE-CHILDREN, DWT-99
DWT:UNREALIZE-WIDGET, DWT-99
DWT:WARNING-MSG, DWT-101
DWT:WIDGET-TO-APPLICATION-CONTEXT,
 DWT-102
DWT:WINDOW, DWT-102
DWT:WINDOW-TO-WIDGET, DWT-103
DWT:XERROR, DWT-104
DWT:XTWARNING, DWT-105
using
 DWT:INITIALIZE, 3-1
 DWT:MAIN-LOOP, 3-4
 DWT:MANAGE-CHILD, 3-2, 3-3
 DWT:MANAGE-CHILDREN, 3-3
 DWT:REALIZE-WIDGET, 3-3

K

:KEY-PRESS event-key, 12-23
:KEY-RELEASE event-key, 12-24
:KEYMAP-NOTIFY event-key, 12-23
Key codes, 13-2

Keyboard controls, 13-5 to 13-6
Keycodes, 13-3 to 13-5
Keysyms, 13-2

L

:LEAVE-NOTIFY event-key, 12-24
Label widget, 1-4
See also Widgets
LABEL_LABEL attribute keyword
User interface language, 2-7
Lines, 9-4 to 9-6
LIST section
User interface language, 2-4
Low-level function, 1-2
creating a user interface, 2-9
DWT:ATTACHED-DB-CREATE, DWT-8
DWT:ATTACHED-DB-POPUP-CREATE, DWT-8
DWT:CAUTION-BOX-CREATE, DWT-14
DWT:COMMAND-WINDOW-CREATE, DWT-17
DWT:DIALOG-BOX-CREATE, DWT-27
DWT:DIALOG-BOX-POPUP-CREATE, DWT-27
DWT:FILE-SELECTION-CREATE, DWT-37
DWT:HELP-CREATE, DWT-45
DWT:LABEL-CREATE, DWT-50
DWT:LIST-BOX-CREATE, DWT-52
DWT:MAIN-WINDOW-CREATE, DWT-57
DWT:MENU-BAR-CREATE, DWT-61
DWT:MENU-CREATE, DWT-61
DWT:MENU-POPUP-CREATE, DWT-62
DWT:MENU-POSITION, DWT-62
DWT:MENU-PULLDOWN-CREATE, DWT-62
DWT:MESSAGE-BOX-CREATE, DWT-64
DWT:OPTION-MENU-CREATE, DWT-66
DWT:PULL-DOWN-MENU-ENTRY-CREATE,
DWT-71
DWT:PUSH-BUTTON-CREATE, DWT-72
DWT:RADIO-BOX-CREATE, DWT-74
DWT:S-TEXT-CREATE, DWT-91
DWT:SCALE-CREATE, DWT-81
DWT:SCROLL-BAR-CREATE, DWT-83
DWT:SCROLL-WINDOW-CREATE, DWT-84
DWT:SELECTION-CREATE, DWT-86
DWT:SEPARATOR-CREATE, DWT-86
DWT:TOGGLE-BUTTON-CREATE, DWT-96
DWT:WINDOW-CREATE, DWT-102
DWT:WORK-BOX-CREATE, DWT-104

M

:MAP-NOTIFY event-key, 12-25
:MAP-REQUEST event-key, 12-26
:MAPPING-NOTIFY event-key, 12-25
:MOTION-NOTIFY event-key, 12-26
Main loop, 3-4
Main window
creating with DWT:MAIN-WINDOW, 2-8
Mapping, 13-2 to 13-3
pointer, 13-3
Menu bar widget
creating with DWT:MENU-BAR, 2-9
Menu widget, 1-4
See also Widgets
creating with DWT:MENU, 2-9

Message box widget

creating with DWT:MESSAGE-BOX-CREATE,
2-10
MicroVAX I, 4-1
MODULE statement
User Interface language, 2-4

N

:NO-EXPOSURE event-key, 12-26
NAMES statement
User interface language, 2-4
Network access, 13-8 to 13-9

O

Off-screen graphics
see images, pixmaps
ORIENTATION attribute keyword
User interface language, 2-6

P

:PROPERTY-NOTIFY event-key, 12-27
Pixel
definition, 8-1
Pixmaps, 10-1 to 10-2
Plane
definition, 8-1
Pointer controls, 13-6 to 13-7
Pointer mapping, 13-3
Points, 9-1 to 9-4
Polygons, filling
see CLX:DRAW-LINES
Primitive widget, 1-7
PROCEDURE section
User interface language, 2-4
Property
definition, 6-5
Pull-down menu entry widget
creating with DWT:PULL-DOWN-MENU-ENTRY,
2-9
Pull-down menu widget
creating with DWT:MENU-PULL-DOWN, 2-9
Push button widget
creating with DWT:PUSH-BUTTON, 2-9

R

:REPARENT-NOTIFY event-key, 12-27
:RESIZE-REQUEST event-key, 12-27
Realizing DECwindows widgets, 3-3
Reason
assigning with the UIL, 2-7
constants, A-5
widget attribute, 1-7
Rectangles, 9-6 to 9-7
Reparenting windows, 13-1

S

:SELECTION-CLEAR event-key, 12-28
:SELECTION-NOTIFY event-key, 12-28
:SELECTION-REQUEST event-key, 12-28
Saveset, 6-4, 13-11
Screen saver, 13-7 to 13-8

Selections
definition, 6-9
Separator widget, 1-4

See also Widgets

Simple text widget
creating with DWT:S-TEXT, 2-9
writing to, 3-7

Source window, of CLX event, 12-1
State masks, 12-10

T

TCP/IP, 4-1
Text, 11-5 to 11-9

see also Glyphs

Type specifiers
in CLX:, 4-2 to 4-4

U

:UNMAP-NOTIFY event-key, 12-29
UIL

See User Interface Language (UIL)

UIL command, 2-7

User interface, 1-7, 2-1

accessing from DECwindows applications, 3-1
creating widgets, 3-2
defining widgets

 with high-level routines, 2-7
 with low-level routines, 2-9
 with UIL, 2-3

realizing top-level widget, 3-3

User Interface Language (UIL), 1-2, 2-3 to 2-7
attribute keywords

 ACTIVATE, 2-7
 CONTROLS, 2-6
 LABEL_LABEL, 2-7
 ORIENTATION, 2-6

sections

 ARGUMENTS, 2-5
 CALLBACKS, 2-5
 CONTROLS, 2-5
 LIST, 2-4
 PROCEDURE, 2-4
 VALUE, 2-4

statements

 MODULE, 2-4
 NAMES, 2-4
 VERSION, 2-4

User-defined data, 3-7

V

:VISIBILITY-NOTIFY event-key, 12-29

VALUE section

 User interface language, 2-4

VAX 11/725, 4-1

VERSION statement

 User interface language, 2-4

Visual info

 definition, 8-2

W

Wait cursor, in CLX, 9-11

Widget ID

accessing in callback argument list, 3-6

Widgets, 1-1

calling back to application functions, 1-7 to 1-8

creating, 3-2

 main window, 2-8

 menu, 2-9

 menu bar, 2-9

 message box, 2-10

 pull-down menu, 2-9

 push button, 2-9

 simple text, 2-9

hierarchy, 1-7

realizing, 3-3

types of, 1-3 to 1-6

Window manager

hints from CLX, 6-7 to 6-9

Window widget, 1-4

See also Widgets

creating with DWT:MAIN-WINDOW, 2-8

Windows, 6-1 to 6-13

 attributes, 6-2 to 6-3

 changing attributes, 6-10

 creating, 6-1 to 6-3

 destroying, 6-3 to 6-4

 getting information, 6-10 to 6-13

 mapping, 6-4 to 6-5

 properties, 6-5 to 6-9

 reparenting, in CLX, 13-1

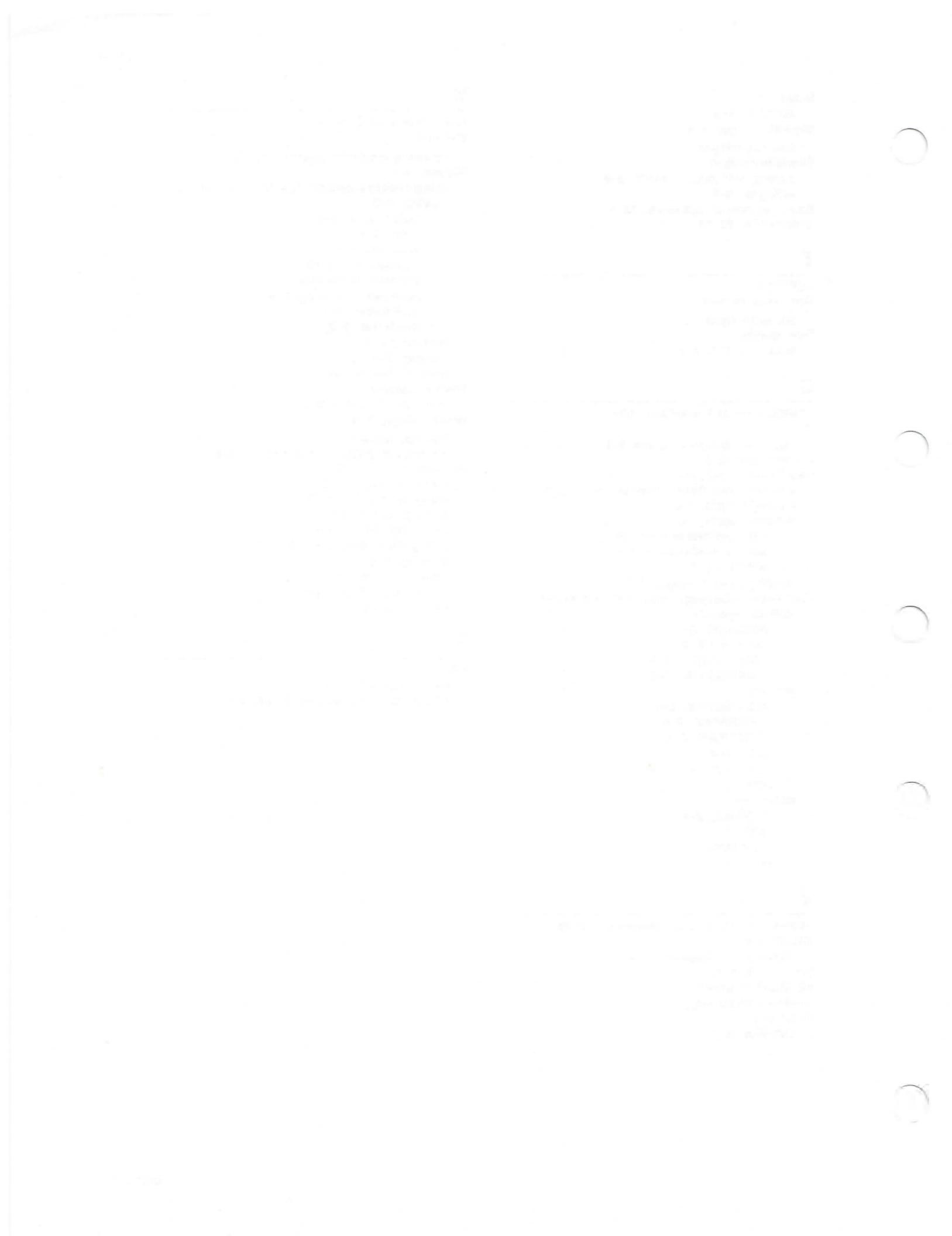
 stacking, 6-10

X

Xlib

relationship to CLX, 4-1

relationship to DECwindows toolkit, 1-1



HOW TO ORDER ADDITIONAL DOCUMENTATION

From	Call	Write
Alaska, Hawaii, or New Hampshire	603-884-6660	Digital Equipment Corporation P.O. Box CS2008 Nashua NH 03061
Rest of U.S.A. and Puerto Rico ¹	800-DIGITAL	
<hr/> ¹ Prepaid orders from Puerto Rico, call Digital's local subsidiary (809-754-7575)		
Canada	800-267-6219 (for software documentation) 613-592-5111 (for hardware documentation)	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6 Attn: Direct Order Desk
Internal orders (for software documentation)	—	Software Supply Business (SSB) Digital Equipment Corporation Westminster MA 01473
Internal orders (for hardware documentation)	DTN: 234-4323 508-351-4323	Publishing & Circulation Services (P&CS) NRO3-1/W3 Digital Equipment Corporation Northboro MA 01532

Reader's Comments

VAX LISP/VMS DECwindows Programming Guide
AA-MK71A-TE

Your comments and suggestions will help us improve the quality of our future documentation. Please note that this form is for comments on documentation only.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (product works as described)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Completeness (enough information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Clarity (easy to understand)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization (structure of subject matter)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Figures (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Examples (useful)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Index (ability to find topic)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Page layout (easy to find information)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

What I like best about this manual: _____

What I like least about this manual: _____

I found the following errors in this manual:

Page Description

My additional comments or suggestions for improving this manual:

Please indicate the type of user/reader that you most nearly represent:

- | | |
|---|---|
| <input type="checkbox"/> Administrative Support | <input type="checkbox"/> Scientist/Engineer |
| <input type="checkbox"/> Computer Operator | <input type="checkbox"/> Software Support |
| <input type="checkbox"/> Educator/Trainer | <input type="checkbox"/> System Manager |
| <input type="checkbox"/> Programmer/Analyst | <input type="checkbox"/> Other (please specify) _____ |
| <input type="checkbox"/> Sales | |

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

— — Do Not Tear — Fold Here and Tape — — — — —

digital™



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

**DIGITAL EQUIPMENT CORPORATION
CORPORATE USER PUBLICATIONS
PKO3-1/30D
129 PARKER STREET
MAYNARD, MA 01754-2198**



— — Do Not Tear — Fold Here — — — — —

Cut Along Dotted Line

