

IEEE Standard for Information Technology—X Window System— Modular Toolkit Environment (MTE)

Circuits and Devices

Communications Technology

IEEE Computer Society

Sponsored by the
Standards Activities Board

*Electromagnetics and
Radiation*

Energy and Power

Industrial Applications

*Signals and
Applications*

*Standards
Coordinating
Committees*

IEEE Std 1295-1993



Published by the Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA.

September 8, 1994

SH16956

IEEE Standard for Information Technology—X Window System—Modular Toolkit Environment (MTE)

Sponsor

**Standards Activities Board
of the
IEEE Computer Society**

Approved December 2, 1993

IEEE Standards Board

Approved June 3, 1994

American National Standards Institute

Abstract: Graphical user interface (GUI) applications and user interfaces to open systems are covered. This standard defines the application interface to a group of “widgets” built upon the X Window System X Toolkit Intrinsics.

Keywords: API, application portability, data processing, graphical user interface, open systems, operating system, portable application, X Window System

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1994 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1994. Printed in the United States of America.

ISBN 1-55937-387-3

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE standards documents may involve the use of patented technology. Their approval by the Institute of Electrical and Electronics Engineers does not mean that using such technology for the purpose of conforming to such standards is authorized by the patent owner. It is the obligation of the user of such technology to obtain all necessary permissions.

Contents

	PAGE
Introduction	xiii
Section 1: Overview	1
1.1 Scope	1
1.2 Normative References	2
1.3 Conformance	2
1.3.1 Implementation Conformance	2
1.3.2 Application Conformance	3
Section 2: General Requirements	5
2.1 Typographic Conventions	5
2.2 Definitions	7
2.2.1 Conformance Terms	7
2.2.2 Technical Terms	8
2.3 Abbreviations	14
2.3.1 C Standard:	14
2.3.2 MTE:	14
2.3.3 POSIX.1:	14
2.3.4 Xlib:	14
2.3.5 Xt:	14
2.4 C-Language Datatypes	14
2.5 C-Language Definitions	17
2.5.1 Symbols from the C Standard	17
2.5.2 MTE Symbols	17
2.5.3 Header and Function Prototypes	17
2.6 Resource Names	18
2.6.1 MTE Resource Names	18
2.6.2 Xm to Xt Resource Correspondence	18
Section 3: Input Control	21
3.1 Input Model	21
3.1.1 Model Keyboard	21
3.1.2 Model Mouse	23
3.1.3 The Focus Model	24
3.2 Navigation	25
3.2.1 Pointer Navigation	25
3.2.2 Keyboard-Based Navigation	27
3.2.3 Menu Traversal	30
3.2.4 Scrollable Component Navigation	32
3.3 Selection	33
3.3.1 Selection Models	33
3.3.2 Mouse-Based Discontiguous Selection	36
3.3.3 Autoscrolling and Selection	39

3.3.4	Selection Actions	40
3.3.5	Transfer Models	43
3.4	Activation	46
3.4.1	Basic Activation	47
3.4.2	Accelerators	47
3.4.3	Mnemonics	48
3.4.4	Help Activation	48
3.4.5	Default Activation	48
3.4.6	Expert Activation	49
3.4.7	Cancel Activation	49
Section 4: Shell Widget Classes		51
4.1	Vendor Shell	51
4.1.1	VendorShell	51
4.1.2	XmActivateProtocol	55
4.1.3	XmAddProtocols	56
4.1.4	XmAddProtocolCallback	57
4.1.5	XmDeactivateProtocol	58
4.1.6	XmRemoveProtocols	59
4.1.7	XmRemoveProtocolCallback	60
4.1.8	XmSetProtocolHooks	61
4.2	Dialog Shell	62
4.2.1	XmDialogShell	62
4.2.2	XmCreateDialogShell	65
4.3	Menu Shell	66
4.3.1	XmMenuShell	66
4.3.2	XmCreateMenuShell	69
4.4	Display	70
4.4.1	XmDisplay	70
4.4.2	XmGetXmDisplay	71
4.5	Screen	72
4.5.1	XmScreen	72
4.5.2	XmGetXmScreen	73
Section 5: Manager Widget Classes		75
5.1	Manager Widget Class	75
5.1.1	XmManager	75
5.2	Bulletin Board	79
5.2.1	XmBulletinBoard	79
5.2.2	XmCreateBulletinBoard	85
5.2.3	XmCreateBulletinBoardDialog	85
5.3	Command	86
5.3.1	XmCommand	86
5.3.2	XmCreateCommand	92
5.3.3	XmCommandAppendValue	92
5.3.4	XmCommandError	93
5.3.5	XmCommandSetValue	93
5.4	Drawing Area	94
5.4.1	XmDrawingArea	94
5.4.2	XmCreateDrawingArea	98

5.5	File Selection Box	99
5.5.1	XmFileSelectionBox	99
5.5.2	XmCreateFileSelectionBox	112
5.5.3	XmCreateFileSelectionDialog	112
5.5.4	XmFileSelectionDoSearch	113
5.6	Form	114
5.6.1	XmForm	114
5.6.2	XmCreateForm	124
5.6.3	XmCreateFormDialog	124
5.7	Frame	125
5.7.1	XmFrame	125
5.7.2	XmCreateFrame	128
5.8	Main Window	129
5.8.1	XmMainWindow	129
5.8.2	XmCreateMainWindow	134
5.9	Message Box	135
5.9.1	XmMessageBox	135
5.9.2	XmCreateErrorDialog	141
5.9.3	XmCreateInformationDialog	141
5.9.4	XmCreateMessageBox	142
5.9.5	XmCreateMessageDialog	143
5.9.6	XmCreateQuestionDialog	144
5.9.7	XmCreateWarningDialog	145
5.9.8	XmCreateWorkingDialog	145
5.10	Paned Window	146
5.10.1	XmPanedWindow	146
5.10.2	XmCreatePanedWindow	152
5.11	Row Column Widget Class	153
5.11.1	XmRowColumn	153
5.11.2	XmCreateOptionsMenu	163
5.11.3	XmCreatePopupMenu	165
5.11.4	XmCreatePulldownMenu	166
5.11.5	XmCreateMenuBar	167
5.11.6	XmCreateRadioBox	168
5.11.7	XmCreateRowColumn	169
5.11.8	XmCreateWorkArea	170
5.11.9	XmMenuPosition	170
5.12	Scale Window	171
5.12.1	XmScale	171
5.12.2	XmCreateScale	176
5.12.3	XmScaleGetValue	177
5.12.4	XmScaleSetValue	177
5.13	Scrolled Window	178
5.13.1	XmScrolledWindow	178
5.13.2	XmCreateScrolledList	183
5.13.3	XmCreateScrolledText	184
5.13.4	XmCreateScrolledWindow	185
5.14	Selection Box	186
5.14.1	XmSelectionBox	186
5.14.2	XmCreatePromptDialog	194

5.14.3	XmCreateSelectionBox	195
5.14.4	XmCreateSelectionDialog	195
Section 6: Primitive Widget Classes		197
6.1	Primitive	197
6.1.1	XmPrimitive	197
6.2	Arrow Button	203
6.2.1	XmArrowButton	203
6.2.2	XmCreateArrowButton	208
6.3	Cascade Button	209
6.3.1	XmCascadeButton	209
6.3.2	XmCascadeButtonHighlight	215
6.3.3	XmCreateCascadeButton	215
6.4	Drawn Button	216
6.4.1	XmDrawnButton	216
6.4.2	XmCreateDrawnButton	222
6.5	Label	223
6.5.1	XmLabel	223
6.5.2	XmCreateLabel	230
6.6	List	231
6.6.1	XmList	231
6.6.2	XmCreateList	247
6.6.3	XmListAddItem	248
6.6.4	XmListAddItems	248
6.6.5	XmListAddItemUnselected	249
6.6.6	XmListAddItemsUnselected	250
6.6.7	XmListDeleteAllItems	250
6.6.8	XmListDeleteItem	251
6.6.9	XmListDeleteItems	251
6.6.10	XmListDeleteItemsPos	252
6.6.11	XmListDeletePos	252
6.6.12	XmListDeletePositions	253
6.6.13	XmListDeselectAllItems	253
6.6.14	XmListDeselectItem	254
6.6.15	XmListDeselectPos	254
6.6.16	XmListGetKbdItemPos	255
6.6.17	XmListGetMatchPos	256
6.6.18	XmListGetSelectedPos	256
6.6.19	XmListItemExists	257
6.6.20	XmListItemPos	258
6.6.21	XmListPosSelected	258
6.6.22	XmListPosToBounds	259
6.6.23	XmListReplaceItems	260
6.6.24	XmListReplaceItemsPos	260
6.6.25	XmListReplaceItemsPosUnselected	261
6.6.26	XmListReplaceItemsUnselected	262
6.6.27	XmListReplacePositions	263
6.6.28	XmListSelectItem	263
6.6.29	XmListSelectPos	264
6.6.30	XmListSetAddMode	265

6.6.31	XmListSetBottomItem	265
6.6.32	XmListSetBottomPos	266
6.6.33	XmListSetHorizPos	266
6.6.34	XmListSetItem	267
6.6.35	XmListSetKbdItemPos	267
6.6.36	XmListSetPos	268
6.6.37	XmListUpdateSelectedList	268
6.6.38	XmListYToPos	269
6.7	Push Button	270
6.7.1	XmPushButton	270
6.7.2	XmCreatePushButton	278
6.8	Scroll Bar	279
6.8.1	XmScrollBar	279
6.8.2	XmCreateScrollBar	288
6.8.3	XmScrollBarGetValues	288
6.8.4	XmScrollBarSetValues	289
6.9	Separator	290
6.9.1	XmSeparator	290
6.9.2	XmCreateSeparator	293
6.10	Text	294
6.10.1	XmText	294
6.10.2	XmCreateText	317
6.10.3	XmTextClearSelection	318
6.10.4	XmTextCopy	318
6.10.5	XmTextCut	319
6.10.6	XmTextGetBaseline	319
6.10.7	XmTextGetEditable	320
6.10.8	XmTextGetInsertionPosition	321
6.10.9	XmTextGetLastPosition	321
6.10.10	XmTextGetMaxLength	322
6.10.11	XmTextGetSelection	322
6.10.12	XmTextGetSelectionPosition	323
6.10.13	XmTextGetSource	324
6.10.14	XmTextGetString	324
6.10.15	XmTextGetSubstring	325
6.10.16	XmTextGetTopCharacter	326
6.10.17	XmTextInsert	326
6.10.18	XmTextPaste	327
6.10.19	XmTextPosition	328
6.10.20	XmTextPosToXY	328
6.10.21	XmTextRemove	329
6.10.22	XmTextReplace	329
6.10.23	XmTextScroll	330
6.10.24	XmTextSetAddMode	330
6.10.25	XmTextSetEditable	331
6.10.26	XmTextSetHighlight	331
6.10.27	XmTextSetInsertionPosition	332
6.10.28	XmTextSetMaxLength	333
6.10.29	XmTextSetSelection	333
6.10.30	XmTextSetSource	334

6.10.31	XmTextSetString	335
6.10.32	XmTextSetTopCharacter	335
6.10.33	XmTextShowPosition	336
6.10.34	XmTextXYToPos	336
6.11	Text Field	337
6.11.1	XmTextField	337
6.11.2	XmCreateTextField	353
6.11.3	XmTextFieldClearSelection	353
6.11.4	XmTextFieldCopy	354
6.11.5	XmTextFieldCut	354
6.11.6	XmTextFieldGetBaseline	355
6.11.7	XmTextFieldGetEditable	356
6.11.8	XmTextFieldGetInsertionPosition	356
6.11.9	XmTextFieldGetLastPosition	357
6.11.10	XmTextFieldGetMaxLength	357
6.11.11	XmTextFieldGetSelection	358
6.11.12	XmTextFieldGetSelectionPosition	359
6.11.13	XmTextFieldGetString	359
6.11.14	XmTextFieldGetSubstring	360
6.11.15	XmTextFieldInsert	361
6.11.16	XmTextFieldPaste	361
6.11.17	XmTextFieldPosToXY	362
6.11.18	XmTextFieldRemove	363
6.11.19	XmTextFieldReplace	363
6.11.20	XmTextFieldSetAddMode	364
6.11.21	XmTextFieldSetEditable	365
6.11.22	XmTextFieldSetHighlight	365
6.11.23	XmTextFieldSetInsertionPosition	366
6.11.24	XmTextFieldSetMaxLength	366
6.11.25	XmTextFieldSetSelection	367
6.11.26	XmTextFieldSetString	368
6.11.27	XmTextFieldShowPosition	368
6.11.28	XmTextFieldXYToPos	369
6.12	Toggle Button	370
6.12.1	XmToggleButton	370
6.12.2	XmCreateToggleButton	378
6.12.3	XmToggleButtonGetState	379
6.12.4	XmToggleButtonSetState	380
Section 7:	Font and String Management	381
7.1	Fontlist Management	381
7.1.1	XmFontList	381
7.1.2	XmFontContext	381
7.1.3	XmFontListEntry	382
7.1.4	XmFontType	382
7.1.5	XmFontListAppendEntry	383
7.1.6	XmFontListCopy	383
7.1.7	XmFontListEntryCreate	384
7.1.8	XmFontListEntryFree	385
7.1.9	XmFontListEntryGetFont	385

7.1.10	XmFontListEntryGetTag	386
7.1.11	XmFontListEntryLoad	386
7.1.12	XmFontListFree	387
7.1.13	XmFontListFreeFontContext	388
7.1.14	XmFontListInitFontContext	388
7.1.15	XmFontListNextEntry	389
7.1.16	XmFontListRemoveEntry	390
7.2	String and String List Management	391
7.2.1	XmString	391
7.2.2	XmStringCharSet, XmStringCharSetTable	391
7.2.3	XmStringComponentType	392
7.2.4	XmStringContext	392
7.2.5	XmStringBaseline	393
7.2.6	XmStringCompare	394
7.2.7	XmStringConcat	394
7.2.8	XmStringCopy	395
7.2.9	XmStringCreate	395
7.2.10	XmStringCreateLocalized	396
7.2.11	XmStringCreateSimple	397
7.2.12	XmStringDirection	397
7.2.13	XmStringDraw	398
7.2.14	XmStringDrawImage	399
7.2.15	XmStringDrawUnderline	400
7.2.16	XmStringEmpty	402
7.2.17	XmStringExtent	402
7.2.18	XmStringFree	403
7.2.19	XmStringFreeContext	403
7.2.20	XmStringGetNextComponent	404
7.2.21	XmStringGetNextSegment	405
7.2.22	XmStringHasSubstring	406
7.2.23	XmStringHeight	407
7.2.24	XmStringInitContext	407
7.2.25	XmStringLineCount	408
7.2.26	XmStringSegmentCreate	408
7.2.27	XmStringSeparatorCreate	409
7.2.28	XmStringTable	410
7.2.29	XmStringWidth	410
Section 8:	Toolkit Functions	411
8.1	Clipboard Management	411
8.1.1	XmClipboardCancelCopy	411
8.1.2	XmClipboardCopy	412
8.1.3	XmClipboardCopyByName	413
8.1.4	XmClipboardEndCopy	414
8.1.5	XmClipboardEndRetrieve	415
8.1.6	XmClipboardInquireCount	416
8.1.7	XmClipboardInquireFormat	417
8.1.8	XmClipboardInquireLength	418
8.1.9	XmClipboardInquirePendingItems	419
8.1.10	XmClipboardLock	420

8.1.11	XmClipboardRegisterFormat	421
8.1.12	XmClipboardRetrieve	422
8.1.13	XmClipboardStartCopy	423
8.1.14	XmClipboardStartRetrieve	425
8.1.15	XmClipboardUndoCopy	426
8.1.16	XmClipboardUnlock	427
8.1.17	XmClipboardWithdrawFormat	428
8.2	Tab Group Management	429
8.2.1	XmGetFocusWidget	429
8.2.2	XmGetTabGroup	429
8.2.3	XmGetVisibility	430
8.2.4	XmIsTraversable	430
8.2.5	XmProcessTraversal	431
8.3	Atom Management	433
8.3.1	XmGetAtomName	433
8.3.2	XmInternAtom	433
8.4	Units Management	434
8.4.1	XmConvertUnits	434
8.5	Pixmap Management	435
8.5.1	XmDestroyPixmap	435
8.5.2	XmGetPixmapByDepth	435
8.6	Image Management	438
8.6.1	XmInstallImage	438
8.6.2	XmUninstallImage	439
8.7	Miscellaneous Toolkit Functions	439
8.7.1	XmGetColors	439
8.7.2	XmGetDestination	440
8.7.3	XmGetPostedFromWidget	441
8.7.4	XmTrackingEvent	441
8.7.5	XmUpdateDisplay	442
Annex A (normative) Xt and X11 Dependencies		443
A.1	Xt Identifiers Used in MTE	443
A.2	Xm Identifiers Equivalent to Xt	444
A.3	Xlib Identifiers Used in MTE	449
Annex B (informative) Bibliography		451
Annex C (informative) Relationship to IEEE P1201.2/D2		453
Identifier Index		463
Alphabetic Topical Index		475

TABLES

Table 2-1	– Class Name Translation Exceptions	19
Table 4-1	– Vendor Shell: VendorShell Resource Set	52
Table 4-2	– Vendor Shell: WMSHELL Resource Set	53
Table 4-3	– Vendor Shell: Shell Resource Set	54
Table 4-4	– Vendor Shell: Composite Resource Set	54
Table 4-5	– Vendor Shell: Core Resource Set	54
Table 4-6	– Dialog Shell: TransientShell Resource Set	63
Table 4-7	– Dialog Shell: VendorShell Resource Set	63
Table 4-8	– Dialog Shell: WMSHELL Resource Set	63
Table 4-9	– Dialog Shell: Shell Resource Set	64
Table 4-10	– Dialog Shell: Composite Resource Set	64
Table 4-11	– Dialog Shell: Core Resource Set	64
Table 4-12	– Menu Shell: XmMenuShell Resource Set	67
Table 4-13	– Menu Shell: Shell Resource Set	68
Table 4-14	– Menu Shell: Composite Resource Set	68
Table 4-15	– Menu Shell: Core Resource Set	68
Table 4-16	– XmScreen: XmScreen Resource Set	72
Table 5-1	– XmManager: Resource Set	76
Table 5-2	– XmManager: Composite Resource Set	77
Table 5-3	– XmManager: Core Resource Set	78
Table 5-4	– XmBulletinBoard: Resource Set	80
Table 5-5	– XmBulletinBoard: XmManager Resource Set	83
Table 5-6	– XmBulletinBoard: Composite Resource Set	83
Table 5-7	– XmBulletinBoard: Core Resource Set	83
Table 5-8	– XmCommand: Resource Set	87
Table 5-9	– XmCommand: XmSelectionBox Resource Set	89
Table 5-10	– XmCommand: XmBulletinBoard Resource Set	89
Table 5-11	– XmCommand: XmManager Resource Set	90
Table 5-12	– XmCommand: Composite Resource Set	90
Table 5-13	– XmCommand: Core Resource Set	90
Table 5-14	– XmDrawingArea: Resource Set	95
Table 5-15	– XmDrawingArea: XmManager Resource Set	96
Table 5-16	– XmDrawingArea: Composite Resource Set	96
Table 5-17	– XmDrawingArea: Core Resource Set	96
Table 5-18	– XmFileSelectionBox: Resource Set	102
Table 5-19	– XmFileSelectionBox: XmSelectionBox Resource Set	107
Table 5-20	– XmFileSelectionBox: XmBulletinBoard Resource Set	107
Table 5-21	– XmFileSelectionBox: XmManager Resource Set	108
Table 5-22	– XmFileSelectionBox: Composite Resource Set	108
Table 5-23	– XmFileSelectionBox: Core Resource Set	108
Table 5-24	– XmForm: Resource Set	116
Table 5-25	– XmForm: Constraint Resource Set	117
Table 5-26	– XmForm: XmBulletinBoard Resource Set	122
Table 5-27	– XmForm: XmManager Resource Set	123

Table 5-28 – XmForm: Composite Resource Set	123
Table 5-29 – XmForm: Core Resource Set	123
Table 5-30 – XmFrame: Resource Set	126
Table 5-31 – XmFrame: XmManager Resource Set	127
Table 5-32 – XmFrame: Composite Resource Set	127
Table 5-33 – XmFrame: Core Resource Set	128
Table 5-34 – XmMainWindow: Resource Set	131
Table 5-35 – XmMainWindow: XmScrolledWindow Resource Set	132
Table 5-36 – XmMainWindow: XmManager Resource Set	132
Table 5-37 – XmMainWindow: Composite Resource Set	132
Table 5-38 – XmMainWindow: Core Resource Set	133
Table 5-39 – XmMessageBox: Resource Set	136
Table 5-40 – XmMessageBox: XmBulletinBoard Resource Set	138
Table 5-41 – XmMessageBox: XmManager Resource Set	138
Table 5-42 – XmMessageBox: Composite Resource Set	138
Table 5-43 – XmMessageBox: Core Resource Set	139
Table 5-44 – XmPanedWindow: Resource Set	148
Table 5-45 – XmPanedWindow: Constraint Resource Set	149
Table 5-46 – XmPanedWindow: XmManager Resource Set	149
Table 5-47 – XmPanedWindow: Core Resource Set	150
Table 5-48 – XmPanedWindow: Composite Resource Set	150
Table 5-49 – XmRowColumn: Resource Set	155
Table 5-50 – XmRowColumn: XmManager Resource Set	161
Table 5-51 – XmRowColumn: Composite Resource Set	161
Table 5-52 – XmRowColumn: Core Resource Set	161
Table 5-53 – XmScale: Resource Set	172
Table 5-54 – XmScale: XmManager Resource Set	174
Table 5-55 – XmScale: Composite Resource Set	175
Table 5-56 – XmScale: Core Resource Set	175
Table 5-57 – XmScrolledWindow: Resource Set	179
Table 5-58 – XmScrolledWindow: XmManager Resource Set	181
Table 5-59 – XmScrolledWindow: Composite Resource Set	181
Table 5-60 – XmScrolledWindow: Core Resource Set	182
Table 5-61 – XmSelectionBox: Resource Set	188
Table 5-62 – XmSelectionBox: XmBulletinBoard Resource Set	190
Table 5-63 – XmSelectionBox: XmManager Resource Set	191
Table 5-64 – XmSelectionBox: Composite Resource Set	191
Table 5-65 – XmSelectionBox: Core Resource Set	191
Table 6-1 – XmPrimitive: Resource Set	198
Table 6-2 – XmPrimitive: Core Resource Set	200
Table 6-3 – XmArrowButton: Resource Set	203
Table 6-4 – XmArrowButton: XmPrimitive Resource Set	205
Table 6-5 – XmArrowButton: Core Resource Set	205
Table 6-6 – XmCascadeButton: Resource Set	210
Table 6-7 – XmCascadeButton: XmLabel Resource Set	211
Table 6-8 – XmCascadeButton: XmPrimitive Resource Set	212

Table 6-9	– XmCascadeButton: Core Resource Set	212
Table 6-10	– XmDrawnButton: Resource Set	217
Table 6-11	– XmDrawnButton: XmLabel Resource Set	218
Table 6-12	– XmDrawnButton: XmPrimitive Resource Set	219
Table 6-13	– XmDrawnButton: Core Resource Set	219
Table 6-14	– XmLabel: Resource Set	224
Table 6-15	– XmLabel: XmPrimitive Resource Set	227
Table 6-16	– XmLabel: Core Resource Set	228
Table 6-17	– XmList: Resource Set	233
Table 6-18	– XmList: XmPrimitive Resource Set	236
Table 6-19	– XmList: Core Resource Set	236
Table 6-20	– XmPushButton: Resource Set	271
Table 6-21	– XmPushButton: XmLabel Resource Set	273
Table 6-22	– XmPushButton: XmPrimitive Resource Set	273
Table 6-23	– XmPushButton: Core Resource Set	274
Table 6-24	– XmScrollBar: Resource Set	280
Table 6-25	– XmScrollBar: XmPrimitive Resource Set	283
Table 6-26	– XmScrollBar: Core Resource Set	283
Table 6-27	– XmSeparator: XmSeparator Resource Set	291
Table 6-28	– XmSeparator: XmPrimitive Resource Set	292
Table 6-29	– XmSeparator: Core Resource Set	293
Table 6-30	– XmText: Resource Set	295
Table 6-31	– XmText: Input Resource Set	298
Table 6-32	– XmText: Output Resource Set	298
Table 6-33	– XmText: Output Resource Set (ScrolledWindow)	300
Table 6-34	– XmText: XmPrimitive Resource Set	300
Table 6-35	– XmText: Core Resource Set	301
Table 6-36	– XmTextField: Resource Set	339
Table 6-37	– XmTextField: XmPrimitive Resource Set	342
Table 6-38	– XmTextField: Core Resource Set	342
Table 6-39	– XmToggleButton: Resource Set	371
Table 6-40	– XmToggleButton: XmLabel Resource Set	374
Table 6-41	– XmToggleButton: XmPrimitive Resource Set	374
Table 6-42	– XmToggleButton: Core Resource Set	375
Table A-1	– Xm to Xt Identifier Equivalents	444

Introduction

(This introduction is not a part of IEEE Std 1295-1993, IEEE Standard for Information Technology—Window System—Modular Toolkit Environment (MTE), but is included for information only.)

The purpose of this standard is to define a source-code level interface to an X Window System¹⁾ toolkit Graphical User Interface (GUI) environment based on the Motif²⁾ *Application Environment Specification User Environment Volume* {B4} and *OSF/Motif Style Guide* {B5}. This environment includes a C-language application program interface (API) and a specification of appearance and behavior of visual objects consistent with the GUI Drivability recommended practice being developed by IEEE P1201.2 {B3}.

This Modular Toolkit Environment standard, hereinafter referred to as MTE, represents a portion of the interface requirements for a fully functional GUI. The other standards or specifications that serve as foundations for this interface are

- The X Window System protocol, being standardized by Accredited Standards Committee X3H3.6
- The X Library, specified in the *Xlib Specification* {5}
- The X Toolkit and Intrinsics, specified in the *X Toolkit Specification* {4}

All of these interfaces have become de facto standards within the United States by their specification as FIPS PUB 158.

One of the goals of the developers of the standard was to identify and maximize the common elements with other X Window System toolkit GUI environments.

Within this document, the term “MTE” refers to this standard.

Organization of the Standard

This standard is divided into eight sections, as follows:

- Section 1 presents information about this standard, including a description of the scope, normative references, and conformance.
- Section 2 specifies general requirements, including an explanation of conventions and definitions, and concepts and features.
- Section 3 specifies the input model, and describes navigation, selection, and activation of user interface features.
- Section 4 specifies shell widget classes.
- Section 5 specifies manager widget classes.

1) X Window System is a trademark of the Massachusetts Institute of Technology.

2) Motif is a registered trademark of Open Software Foundation, Inc. in the USA and other countries.

- 31 — Section 6 specifies primitive widget classes.
- 32 — Section 7 specifies font and string management.
- 33 — Section 8 specifies toolkit functions.

34 Most of the clauses describe a single service interface. The C-language binding for the service
35 interface is given in the subclause labeled “Synopsis.” The “Description” subclause pro-
36 vides a specification of the operation performed by the service interface. Some examples may
37 be provided to illustrate the interfaces described. In most cases there is also a “Return Value”
38 subclause specifying return values. References are used to direct the reader to other related
39 sections.

40 Normative annexes are a part of this standard and impose requirements. There is one norma-
41 tive annex in this document. Informative annexes are not part of the standard and are provided
42 for information only. They are provided for guidance and to help understanding. There are
43 three informative annexes in this standard.

44 In publishing this standard, its developers intend to provide a yardstick against which various
45 graphical user interface toolkit implementations can be measured for conformance. It is *not*
46 the intent of the developers to measure or rate any products, to reward or sanction any vendors
47 of products for conformance or lack of conformance to this standard, or to attempt to enforce
48 this standard by these or any other means. The responsibility for determining the degree of
49 conformance or lack thereof with this standard rests solely with the individual who is evaluat-
50 ing the product claiming to be in conformance with this standard.

51 This introduction, the foreword, any footnotes or other notes accompanying the text, and the
52 *informative* annexes are not considered part of the standard. Annexes B and C are informative.

53 **Acknowledgments**

54 The base documents for this standard are the *Application Environment Specification, User*
55 *Environment Volume {B4}* and the *OSF/Motif Style Guide {B5}*, copyright 1993 by Open
56 Software Foundation. The IEEE is grateful to Open Software Foundation for permission to use
57 their materials.

58 **Related Standards Activities**

59 There are other standards activities concerning GUIs in progress within the IEEE Computer
60 Society, and similar efforts can be anticipated in the future.

61 The following areas are under active consideration at this time or are expected to become
62 active in the near future:³⁾

- 63 — A standard for a “virtual toolkit interface,” under development by IEEE P1201.1.
- 64 — A recommended practice on user portability and driveability, under development by
65 IEEE P1201.2 {B3}.

66 3) A *Standards Status Report* that lists all current IEEE Computer Society standards projects is available from the
67 IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903; Telephone:
68 +1 202 371-0101; FAX: +1 202 728-9614.

69 Extensions are approved as “amendments” or “revisions” to this document, following IEEE
70 and ISO/IEC procedures.

71 Approved amendments are published separately until the full document is reprinted and such
72 amendments are incorporated in their proper positions.

73 If you have an interest in participating in the working groups addressing these issues, please
74 send your name, address, and phone number to the Secretary, IEEE Standards Board, Institute
75 of Electrical and Electronics Engineers, Inc., P.O. Box 1331, 445 Hoes Lane, Piscataway, NJ
76 08855-1331, and ask to have this forwarded to the chair of the appropriate working group.

77 IEEE Std 1295-1993 was prepared by the IEEE Std 1295-1993 working group, sponsored by
78 the IEEE Computer Society. At the time this standard was approved, the membership of the
79 IEEE Std 1295-1993 working group was as follows:

IEEE Std 1295-1993 Working Group Officials

81 Chair: John S. Morris
82 Vice Chair: John Hurd
83 Editors: Bob May
84 Paul Rabin
85 Secretaries: Hide Horiuchi

Working Group

87 Laurence Cable	Vania Joloboff	Rosalind Radcliffe
88 Ray Farjardo	Michael Kung	Kamesh Ramakrishna
89 Ken Holdaway	Richard Lancaster	James VanGilder

90 The following persons were members of the IEEE Std 1295-1993 balloting group that approved
91 the standard for submission to the IEEE Standards Board.

92 Donald Allen	Yaacov Fenster	Garry M. Paxinos
93 Charles Amacher	Christopher Flake	Satya Prashakar
94 Ray Anderson	Edward N. Gootee	Brian Purdy
95 Bengt Asker	Georges Grinstein	Paul Rabin
96 Larry M. Augustin	Philip Gust	Rosalind Radcliffe
97 Robert H. Bagwill	Jeffrey S. Haemer	Kamesh Ramakrishna
98 John Barr	Charles Hammons	Robert Robillard
99 Andy Bihain	Libby Hanna	Gary S. Robinson
100 Christophe Binot	Dennis Harms	John Sarkesain
101 Brian Boyter	Robert G. Herriot	Martin J. Schedlbauer
102 Phylecia Brandley	Steven Hiebert	Fritz Schulz
103 David J. Bryant	Hide Horiuchi	Leonard W. Seagren
104 E. Loren Buhle	Chris Hughes	Jacob Slonim
105 Elizabeth Buie	Steve Humphrey	Jon F. Spencer
106 Christopher Byrnes	John C. Hurd	James Stallard
107 Laurence P. G. Cable	Vania Joloboff	Dennis C. Stewart
108 George Carson	Michael Jones	Scott Sutter
109 Kilnam Chon	Judy S. Kerner	James G. Tanner
110 Edward Cochran	Lorraine C. Kevra	Keith M. Taylor
111 Mark F. Cook	Kyuh Kim	William Turner
112 James Cunningham	Thomas M. Kurihara	Mark-Rene Uchida
113 Elizabeth J. Dall	Donald Lewine	James H. VanGilder
114 Peter Derr	Kevin Lewis	Daniel Williams
115 Todd Earles	Robert May	Bill Yoder

116	Paul Eastman	John McNeely	Oren Yuen
117	Ray Fajardo	Jon Meads	Janusz Zalewski

118 When the IEEE Standards Board approved this standard on December 2, 1993, it had the fol-
 119 lowing membership:

120	Wallace S. Read, <i>Chair</i>	Donald C. Loughry, <i>Vice Chair</i>
121	Andrew G. Salem, <i>Secretary</i>	

122	Gilles A. Baril	Jim Isaak	Don T. Michael*
123	Jose A. Berrios de la Paz	Ben C. Johnson	Marco W. Migliaro
124	Clyde R. Camp	Walter J. Karplus	L. John Rankine
125	Donald C. Fleckenstein	Lorraine C. Kevra	Arthur K. Reilly
126	Jay Forster*	E. G. "Al" Kiener	Ronald H. Reimer
127	David F. Franklin	Ivor N. Knight	Gary S. Robinson
128	Ramiro Garcia	Joseph L. Koepfinger*	Leonard L. Tripp
129	Donald N. Heirman	D. N. "Jim" Logothetis	Donald W. Zipse

130 *Member Emeritus

131 Also included are the following nonvoting IEEE Standards Board liaisons:

132 Satish K. Aggarwal
 133 James Beall
 134 Richard B. Engelman
 135 David E. Soffrin
 136 Stanley I. Warshaw

137 Mary Lynne Nielsen
 138 *IEEE Standards Project Editor*

IEEE Standard for Information Technology—X Window System—Modular Toolkit Environment (MTE)

Section 1: Overview

1.1 Scope

This standard defines a standard application programming interface (API) to and a standard appearance and behavior of an X Window System toolkit graphical user interface (GUI) environment based on the *OSF/Motif Style Guide* {B5} and the *Application Environment Specification User Environment Volume* {B4}. The standard serves to support application portability at the source-code level and is intended to be used by both application developers and toolkit implementors.

This standard describes the external characteristics and facilities that are of importance to applications developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications. The descriptions of widgets in this standard emphasize their programmatic interfaces and their external functional characteristics and user interface, rather than their exact visual appearance.

This standard defines program interfaces exclusively at the source-code level. The objective is that a Strictly Conforming Modular Toolkit Environment (MTE) Application source program can be translated to execute on a conforming implementation.

This standard has been designed to be used by both application programmers and toolkit implementors. However, it is intended to be a reference document and not a tutorial on the use of the services, the widgets, or the interrelationships between the widgets and the APIs in this document and those in other components of the X Window System or the operating system itself.

1.2 Normative References

The following standards contain provisions that, through references in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

- {1} ISO/IEC 646: 1991,¹⁾ *Information technology—ISO 7-bit coded character set for information interchange*.
- {2} ISO 8859-1: 1987, *Information processing—ISO 8-bit single-byte coded graphic character sets—Part 1: Latin alphabet No. 1*.
- {3} ISO/IEC 9899: 1990, *Programming languages—C*.
- {4} Asente, Paul J., Swick, Ralph R.; with McCormack, Joel, *X Window System X Toolkit Specification*, X Version 11, Release 4. Bedford, MA: Digital Press, 1990.
- {5} Scheifler, Robert W., Gettys, Jim; with Flowers, Jim, and Rosenthal, David, *X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD*, Third Edition. Burlington, MA: Digital Press (X and Motif Series), 1992.

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Conformance Requirements

A *conforming implementation* shall meet all of the following criteria:

- (1) The toolkit shall support all required interfaces defined within this standard. These interfaces shall support the functional behavior described herein.
- (2) The toolkit may provide additional or enhanced widgets, functions, and graphical user interface facilities not required by this standard. Nonstandard extensions should be identified as such in the toolkit documentation. Nonstandard extensions, when used, may change the behavior of widgets, functions, and graphical user interface facilities defined by this standard. In such cases, the conformance document for the implementation (see 2.2.1.2) shall define an execution environment (i.e., shall provide general operating instructions) in which an application can be run with the behavior specified by the standard. In no case shall such an environment require modification of a Strictly Conforming MTE Application.

¹⁾ ISO/IEC documents can be obtained from the ISO office, 1, Rue de Varembe, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse, and in the USA from the American National Standards Institute, 11 West 42nd Street, New York, NY 10036, USA.

55 1.3.1.2 Documentation

56 A conformance document with the following information shall be available for an implementa-
57 tion claiming conformance to this standard. The conformance document shall have the same
58 structure as this standard, with the information presented in the appropriately numbered sec-
59 tions; sections that consist solely of subordinate section titles, with no other information, are
60 not required.

61 The conformance document shall not contain information about extended facilities or capabili-
62 ties outside the scope of this standard unless those extensions affect the behavior of a Strictly
63 Conforming MTE Application; in such cases, the documentation required by the previous sub-
64 clause shall be included.

65 The conformance document shall contain a statement that indicates the full name, number, and
66 date of the standard that applies. The conformance document may also list software standards
67 approved by ISO/IEC or any ISO/IEC member body that are available for use by a Conforming
68 MTE Application. Applicable characteristics where documentation is required by one of these
69 standards, or by standards of government bodies, may also be included.

70 The conformance document shall describe the behavior of the implementation for all
71 implementation-defined features defined in this standard. This requirement shall be met by
72 listing these features and providing either a specific reference to the toolkit documentation or
73 providing full syntax and semantics of these features. When the value or behavior in the
74 implementation is designed to be variable or customizable on each instantiation of the toolkit,
75 the implementation provider shall document the nature and permissible ranges of this
76 variation.

77 The conformance document may specify the behavior of the implementation for those features
78 where this standard states that implementations may vary or where features are identified as
79 undefined or unspecified.

80 No specifications other than those described in this subclause shall be present in the confor-
81 mance document.

82 The phrase “shall be documented” in this standard means that documentation of the feature
83 shall appear in the conformance document, as described previously, unless the toolkit docu-
84 mentation is explicitly mentioned.

85 The toolkit documentation should also contain the information found in the conformance
86 document.

87 1.3.2 Application Conformance

88 All applications claiming conformance to this standard fall within one of the categories
89 described in the following subclauses.

1.3.2.1 Strictly Conforming MTE Applications

A Strictly Conforming MTE Application is an application that requires only the facilities described in this standard (including any required facilities of the X Library and X Toolkit; see the *Xlib Specification* {5} and the *X Toolkit Specification* {4}). Such an application

- (1) Shall accept any implementation behavior that results from actions it takes in areas described in this standard as *implementation defined* or *unspecified*, or where the standard indicates that implementations may vary
- (2) Shall not perform any actions that are described as producing *undefined* results
- (3) Shall accept any value in the range permitted by this standard for symbolic constants, but shall not rely on any value in the range being greater than the minimum listed in this standard
- (4) Shall not use facilities designated as *obsolescent*
- (5) Is required to tolerate, and is permitted to adapt to, the presence or absence of optional facilities that are described using the verb *may*

Within this standard, any restrictions placed upon a Conforming MTE Application shall also restrict a Strictly Conforming MTE Application.

1.3.2.2 Conforming MTE Application

The term Conforming MTE Application is used to describe either of the two application types described in the following subclauses.

1.3.2.2.1 ISO/IEC Conforming MTE Application

An ISO/IEC Conforming MTE Application is an application that uses only the facilities described in this standard (including any required facilities of the X Toolkit; see *X Toolkit Specification* {4}) and approved conforming language bindings for any ISO or IEC standard. Such an application shall include a statement of conformance that documents all options and limit dependencies, and all other ISO or IEC standards used.

1.3.2.2.2 <National Body> Conforming MTE Application

A <National Body> Conforming MTE Application differs from an ISO/IEC Conforming MTE Application in that it may also use specific standards of a single ISO/IEC member body referred to here as “<National Body>.” Such an application shall include a statement of conformance that documents all options and limit dependencies, and all other <National Body> standards used.

1.3.2.3 Conforming MTE Application Using Extensions

A Conforming MTE Application Using Extensions is an application that differs from a Conforming MTE Application only in that it uses nonstandard facilities that are consistent with this standard. Such an application shall fully document its requirements for these extended facilities, in addition to the documentation required of a Conforming MTE Application. A Conforming MTE Application Using Extensions shall be either an ISO/IEC Conforming MTE Application Using Extensions or a <National Body> Conforming MTE Application Using Extensions (see 1.3.2.2.1 and 1.3.2.2.2).

Section 2: General Requirements

2.1 Typographic Conventions

This standard uses the following typographic conventions:

(1) The *italic* font is used for

- Cross-references to defined terms and symbolic parameters that are generally substituted with real values by the application
- Global external variable names

(2) The **bold** font is used with a word in all capital letters, such as

PATH

and to represent an environment variable. It is also used for the term “NULL pointer.”

(3) The constant-width (Courier) font is used

- For C-language datatypes, function names, and literal strings
- To illustrate examples of system input or output where exact usage is depicted
- For references to utility names and C language headers
- For widget action routine names
- For window system events
- For keys
- For mouse buttons
- For actions

(4) Symbolic constants returned by many functions as error numbers are represented as

`[ERRNO]`

(5) Symbolic constants or limits defined in certain headers are represented as

`{LIMIT}`

(6) Modifier keys are represented as *modifier-key-names*, for example:

`Mctrl`

(7) All other keys are represented as *key-names*, for example:

`KTab`

(8) Mouse buttons are represented as *mouse-button-names*, for example:

`BSelect`

(9) When a control has additional behavior, the conditions governing this behavior are specified using `<>`, for example:

`<FocusIn> <OK Button Activated>`

Numbers within braces, such as “POSIX.1 {B1},” represent cross-references to the Normative References clause (see 1.2). If the number is preceded by a B, it represents a Bibliographic entry (see Annex B). Bibliographic entries are for information only.

Defined terms are shown in three styles, depending on context:

- Terms defined in 2.2.1 and 2.2.2 are expressed as subclause titles. Alternative forms of the terms appear in [brackets].
- The initial appearances of other terms, applying to a limited portion of the text, are in *italics*.
- Subsequent appearances of the term are in the roman font.

Literal characters and strings used as input or output are shown in various ways, depending on the context:

- | | |
|--|---|
| <p><code>%</code>, <code>begin</code></p> <p><code>'c'</code></p> <p><code>``string``</code></p> | <p>When no confusion would result, the character or string is rendered in the Courier font and used directly in the text.</p> <p>In some cases a character is enclosed in single quotation marks, similar to a C-language character constant. Unless otherwise noted, the quotation marks shall not be used as input or output.</p> <p>In some cases a string is enclosed in double quotation marks, similar to a C-language string constant. Unless otherwise noted, the quotation marks shall not be used as input or output.</p> |
|--|---|

Defined names that are usually lowercased, particularly argument names, are never used at the beginning of a sentence or anywhere else that regular English usage would require them to be capitalized.

Parenthetical expressions within normative text also contain normative information. The general typographical hierarchy of parenthetical expressions is

{ [()] }

The square brackets are most frequently used to enclose a parenthetical expression that contains a function name [such as *waitpid()*], with its built-in parentheses.

In some cases, tabular information is presented “inline”; in others it is presented in a separately labeled table. This arrangement was employed purely for ease of typesetting and there is no normative difference between these two cases.

Annexes marked as *normative* are parts of the standard that pose requirements, exactly the same as the numbered sections, but that have been moved to the end of the document for clarity of exposition. *Informative* annexes are for information only and pose no requirements. All material preceding page 1 of the document (the “front matter”) and the two indexes at the end are also informative.

Notes that appear in a smaller type size and that are indented have one of two different meanings, depending on their location:

— When they are within the normal text of the document, they are the same as footnotes—informative, posing no requirements on implementations or applications.

— When they are attached to tables or figures, they are normative, posing requirements.

Text marked as examples (including the use of “e.g.”) is for information only.

The typographical conventions listed here are for ease of reading only. Editorial inconsistencies in the use of typography are unintentional and have no normative meaning in this standard.

2.2 Definitions

2.2.1 Conformance Terms

For the purposes of this standard, the following definitions apply.

2.2.1.1 can: Describes a permissible optional feature or behavior available to the application or user; the implementation shall support such features or behaviors as mandatory requirements.

2.2.1.2 conformance document: A document provided by an implementor that contains implementation details as described in 1.3.1.2.

2.2.1.3 implementation: An object providing to applications and users the services defined by this standard. The word *implementation* is to be interpreted to mean that object after it has been modified in accordance with the instructions of the manufacturer to

— Configure it for conformance with this standard

— Select some of the various **optional facilities** described by this standard through customization by local system **administrators or operators**

An exception to this meaning occurs when discussing conformance documentation or using the term *implementation defined*. See 2.2.1.4 and 1.3.1.2.

2.2.1.4 implementation defined: The requirements for correct program construction and correct data in the use of a value or behavior that the implementation provider shall document. When the value or behavior in the implementation is designed to be variable or customizable on each instantiation of the system, the implementation provider shall document the nature and permissible ranges of this variation. See 1.3.1.2.

2.2.1.5 may: Describes an optional feature or behavior of the implementation that is not required by this standard, but against which there is no prohibition. A Strictly Conforming MTE Application is permitted to use such features, but shall not rely on the actions of the implementation in such cases. To avoid ambiguity, the reverse sense of *may* is not expressed as *may not*, but as *need not*.

103 **2.2.1.6 obsolescent:** Describes certain features that may be considered for withdrawal in
104 future revisions of this standard. They are retained in this version because of widespread use.
105 Their use in new applications is discouraged.

106 **2.2.1.7 shall:** A requirement on the implementation or on Strictly Conforming MTE Appli-
107 cations, where appropriate.

108 **2.2.1.8 should:** With respect to implementations, an implementation recommendation, but
109 not a requirement. With respect to applications, a recommended programming practice for
110 applications and a requirement for Strictly Conforming MTE Applications.

111 **2.2.1.9 system documentation:** All documentation provided with an implementation
112 except the conformance document. Electronically distributed documents for an implementa-
113 tion are considered part of the system documentation.

114 **2.2.1.10 undefined:** When this standard imposes no portability requirements on applica-
115 tions for erroneous program construction, erroneous data, or use of an indeterminate value.
116 Implementations (or other standards) may specify the result of using that value or causing that
117 behavior. An application using such behaviors is using extensions, as defined in 1.3.2.3.

118 **2.2.1.11 unspecified:** When this standard imposes no portability requirements on applica-
119 tions for a correct program construction or correct data. Implementations (or other standards)
120 may specify the result of using that value or causing that behavior. An application requiring a
121 specific behavior, rather than tolerating any behavior when using that functionality, is using
122 extensions as defined in 1.3.2.3.

123 **2.2.2 Technical Terms**

124 The terms atom, cursor, depth, display, event, input focus, KeySym, off-the-spot input method,
125 pixmap, pointer, screen, and tear-off menu used in this standard are defined in the *Xlib*
126 *Specification* {5} and in the *X Toolkit Specification* {4}. For the purposes of this standard, the
127 following additional definitions apply:

128 **2.2.2.1 accelerator:** An Xt Intrinsics facility that allows the binding of a widget event to a
129 keyboard action or a series of actions; see the *X Toolkit Specification* {4}.

130 **2.2.2.2 activate:** To invoke the action associated with the control.

131 **2.2.2.3 arrow button:** A visual user interface control that is a button labeled with an arrow
132 pointing in a specified direction that represents an action associated with that direction. The
133 action is invoked if the user clicks the mouse within the button.

134 **2.2.2.4 beep:** A brief audible warning emitted by the terminal.

135 **2.2.2.5 button:** A generic term for any control that initiates an action when pressed.

136 **2.2.2.6 callback:** A function written as part of the application, associated with a specific
137 widget resource, that is invoked as a result of a specific change of state associated with that
138 widget. For example, the `XmNactivateCallback` resource of the `PushButton` widget
139 points to the callback function that is called when the button is pushed. See *X Toolkit*
140 *Specification* {4}.

141 **2.2.2.7 cascade button:** A choice on a menu that, when activated, presents another menu
142 with additional related choices.

143 **2.2.2.8 check box:** A visual user interface control used to set and reset parameters that
144 have only two (binary) values (e.g., True/False, On/Off, Active/Inactive). When the control is
145 set, a visual indication is provided to indicate its state (e.g., the check box is filled). The user
146 can reset the parameter by selecting the check box again. A check box can be within a group
147 of check boxes. Normally a group of check boxes are not mutually exclusive.

148 **2.2.2.9 child [descendant]:** A widget that is the immediate inferior of the current widget
149 in the widget instance hierarchy.

150 **2.2.2.10 class:** The type of an object that defines the operations that can be performed on
151 that object. See *X Toolkit Specification* {4}.

152 **2.2.2.11 class hierarchy:** The tree-structured organization of classes. A widget class in
153 this tree always supports all of the operations supported by its ancestor classes closer to the
154 root of the tree, but might support them with different implementations and might also add
155 new operations. See *X Toolkit Specification* {4}.

156 **2.2.2.12 click:** The act of pressing and releasing a mouse button without moving the mouse
157 pointer.

158 **2.2.2.13 clipboard:** A software storage device that is used to store an object that is cut or
159 copied from the screen and to retrieve an object that is pasted.

160 **2.2.2.14 colormap:** A set of entries (*colorcells*) controlling the display of colors for a pixel
161 by assigning numeric intensity values to each of red, blue, and green attributes of the pixel.

162 **2.2.2.15 composite widget:** A parent widget that physically contains other widgets.

163 **2.2.2.16 container:** A parent widget that defines a region containing zero or more subob-
164 jects of a given type.

165 **2.2.2.17 control:** A visual user interface element that is defined by this standard.

166 **2.2.2.18 copy:** To duplicate text or graphic objects from the screen to the clipboard.

- 167 **2.2.2.19 cut:** To duplicate text or graphic objects from the screen to the clipboard and then
168 delete them from the screen.
- 169 **2.2.2.20 dialogue window:** A window, such as a `DialogBox`, that pops up to perform a
170 specific function and is then dismissed.
- 171 **2.2.2.21 dimmed:** A visual effect on an object, such as a reduction in brightness or color,
172 indicating that it cannot be selected or accept input.
- 173 **2.2.2.22 dismiss:** To remove a menu or popup window from the screen.
- 174 **2.2.2.23 display:** To make a popup menu visible.
- 175 **2.2.2.24 double click:** To press and release a mouse button twice in rapid succession
176 without moving the pointer.
- 177 **2.2.2.25 emphasis:** Highlighting, color change, or another visual indication of the condi-
178 tion of an object or choice, and the effect of that condition on the ability of the user to interact
179 with that object or choice. Emphasis can also give the user additional information about the
180 state of an object or choice.
- 181 **2.2.2.26 focus emphasis:** A type of emphasis that indicates the current location for enter-
182 ing text.
- 183 **2.2.2.27 inheritance:** Using the same method to implement an operation as the immediate
184 superclass in the class hierarchy. Inheritance greatly simplifies the definition of new object
185 classes and is one of the main reasons for organizing classes into a hierarchy. See *X Toolkit*
186 *Specification* {4}.
- 187 **2.2.2.28 Intrinsics [Xt Intrinsics]:** The facilities described in the *X Toolkit Specification*
188 {4}, used to create widgets.
- 189 **2.2.2.29 justify:** Align text to a margin. Right justification adjusts the text so that its end
190 touches the right margin or end of the text field. Left justification adjusts the text so that its
191 beginning touches the left margin or beginning of the text field.
- 192 **2.2.2.30 label:** A visual user interface control that is noneditable text that can be used as a
193 title, a message, to identify a field, or to provide additional information about a field.
- 194 **2.2.2.31 list box:** A visual user interface control that contains a list of items that a user can
195 select.
- 196 **2.2.2.32 location cursor:** A visual cue that indicates where the interaction of the user with
197 the keyboard will occur.

- 198 **2.2.2.33 main window:** The primary window for an application.
- 199 **2.2.2.34 menu:** A rectangular visual user interface control containing a group of controls
200 used to select an action from a group of choices.
- 201 **2.2.2.35 menu bar:** A visual user interface control that is the bounded area near the top of
202 a window, below the title bar, and above the rest of the window that contains cascade buttons
203 that provide access to other menus.
- 204 **2.2.2.36 message box:** A visual user interface control used to display information not
205 requested by the user but displayed in a secondary window by an application in response to an
206 unexpected event or a possibility of something undesirable happening.
- 207 **2.2.2.37 method:** A procedure implementing one of the operations supported by an object
208 class. [*X Toolkit Specification* {4}]
- 209 **2.2.2.38 mnemonic:** An abbreviation or other shortened keyboard notation that is used to
210 substitute for a more complicated action, such as selecting an object or performing an opera-
211 tion on it.
- 212 **2.2.2.39 modal:** A state in which the user has to complete the request of the mode before
213 continuing.
- 214 **2.2.2.40 mode-less:** A state that does not interfere with the user performing any other
215 action.
- 216 **2.2.2.41 motion:** To move the pointer while a mouse button is pressed.
- 217 **2.2.2.42 mouse:** An electronic or mechanical input device that allows manipulation of the
218 screen pointer. Mice also generally have one or more buttons that allow selection and manipu-
219 lation of objects.
- 220 **2.2.2.43 object:** A piece of data that can be defined by the operations performed on it. The
221 Intrinsic represents an object internally as a pointer to a dynamically allocated data structure.
222 The principal objects the X Toolkit supports are widgets. See *X Toolkit Specification* {4}.
- 223 **2.2.2.44 pane:** The rectangular area within a window where an application displays text or
224 graphics.
- 225 **2.2.2.45 paned window:** A window that allows a user to view information in different
226 panes of a window, separated by sashes. Panes can be shrunk or enlarged by moving the sash,
227 but the total screen area occupied by the pane window remains constant.
- 228 **2.2.2.46 parent:** A widget that is the immediate superior of the current widget in the widget
229 instance hierarchy.

- 230 **2.2.2.47 persistent menu:** A menu that popped up and stayed visible for one round of
231 use. Menus stay on the screen until the user chooses an item or dismisses the menu.
- 232 **2.2.2.48 pixel:** An abbreviation for picture element—the smallest unit of display on a video
233 screen.
- 234 **2.2.2.49 point:** The standard typographical unit of measurement, approximately equal to
235 1/72 in.
- 236 **2.2.2.50 pointing device:** A device, such as a mouse, trackball, or joystick, used to move
237 a pointer on the screen.
- 238 **2.2.2.51 popup menu:** A menu that appears outside of menu bar when requested, usually
239 as the result of pressing BMenu or KMenu.
- 240 **2.2.2.52 press:** To push and hold a mouse button.
- 241 **2.2.2.53 primary window:** A window in which the main interaction between the user and
242 an object or application takes place (also known as main window).
- 243 **2.2.2.54 push a button:** The act of moving the pointer to a button widget and then select-
244 ing the button.
- 245 **2.2.2.55 push button:** A visual user interface control containing a button, labeled with
246 text, graphics, or both, that represents the action initiated when a user selects it.
- 247 **2.2.2.56 radio button:** A visual user interface control used to represent one of a group of
248 mutually exclusive settings. When a radio button is selected, a visual indication is provided to
249 indicate it is the selected button.
- 250 **2.2.2.57 release:** To stop pressing a mouse button or keyboard key.
- 251 **2.2.2.58 resource:** An attribute of a widget or widget class, represented by a named data
252 value in the defining structure of the widget. See *X Toolkit Specification* {4}.
- 253 **2.2.2.59 sash:** A visual user interface control that separates areas of a window to allow the
254 user to display alternative information.
- 255 **2.2.2.60 scale:** A visual user interface control that represents a quantity and its relationship
256 to the range of possible values for that quantity. The user can change the value of the quantity.
- 257 **2.2.2.61 scroll bar:** A visual user interface control, associated with a scrollable area, that
258 indicates to a user that more information is available and can be scrolled into view.

- 259 **2.2.2.62 scrolled window:** A window that presents information that exceeds the space
260 available for display. The user uses the scroll bar to bring the contents currently outside the
261 display area into view.
- 262 **2.2.2.63 selection box:** A boxed area on the user interface that presents a list of items to
263 select from and a text field where selected items are displayed. The user can also directly type
264 in an item into the text field.
- 265 **2.2.2.64 selected emphasis:** A visual cue that indicates the current selection.
- 266 **2.2.2.65 sensitive:** A condition of an object that allows it to accept input events.
- 267 **2.2.2.66 separator:** A visual user interface control consisting of a line boundary that pro-
268 vides a visual distinction between two adjacent areas. The line may be drawn using various
269 graphics styles.
- 270 **2.2.2.67 server [X server]:** The facility in the terminal or work station that provides input
271 (keyboard, mouse) and output (screen graphics) services to the application.
- 272 **2.2.2.68 stipple:** To change the appearance of an object by covering it with a pattern or
273 regularly spaced small dots, spaced sufficiently so that the underlying image or text is still
274 recognizable to the user. Stippling is frequently used to denote an inaccessible, or busy,
275 object.
- 276 **2.2.2.69 subclass:** A class that is defined by specializing another class. The subclass typi-
277 cally adds features to the other class, called the superclass, or uses the facilities of the super-
278 class to implement a more specific set of interfaces. See *X Toolkit Specification* {4}.
- 279 **2.2.2.70 superclass:** A class that is the ancestor of another class in the class hierarchy.
280 See *X Toolkit Specification* {4}.
- 281 **2.2.2.71 system:** An implementation of this standard.
- 282 **2.2.2.72 text cursor:** A screen object that indicates the insertion point for text input.
- 283 **2.2.2.73 text field:** A visual user interface control into which a user types or places
284 alphanumeric text on one line. Its boundaries are usually indicated.
- 285 **2.2.2.74 tile:** The pixmap used repetitively to pattern an area on the screen.
- 286 **2.2.2.75 toggle:** A switching action performed on an object with two states.
- 287 **2.2.2.76 toolkit:** An implementation of this standard.

288 **2.2.2.77 translation manager:** A facility that maps X event sequences (such as keyboard
289 actions) into widget-supplied functionality (action procedures).

290 **2.2.2.78 traversable widget:** A widget that can receive input focus.

291 **2.2.2.79 visual:** A visual user interface control that provides a type indicating the
292 specification for the color handling of a screen. A visual type may support multiple variations
293 of depths and colormaps. Examples are monochrome, gray-scale, and several types of color.
294 A visual is specified by a *depth* (for example, 8 b) and a *visual class* (for example, Gray-
295 Scale or PseudoColor). A Shell widget or a DrawArea widget can have a nondefault
296 visual. Other widgets use the visuals of their nearest shell or DrawArea ancestor. An appli-
297 cation in which eligible widgets have nondefault visuals is termed a *multivisual* application.

298 **2.2.2.80 widget:** A specific instance of a widget class, providing a control in the user inter-
299 face, such as a menu, pushbutton, or text fields. See *X Toolkit Specification* {4}.

300 **2.2.2.81 widget class:** A collection of code and data structures that provides a generic
301 implementation of a part of the user interface. See *X Toolkit Specification* {4}.

302 **2.2.2.82 window:** A work area on the screen used by an application.

303 **2.2.2.83 work area:** The region of a window where controls such as buttons, settings, and
304 text fields are displayed.

305 2.3 Abbreviations

306 For the purposes of this standard, the following abbreviations apply:

307 **2.3.1 C Standard:** ISO/IEC 9899: 1990, *Information Processing Systems—Programming*
308 *Languages—C* {3}.

309 **2.3.2 MTE:** This standard (Modular Toolkit Environment).

310 **2.3.3 POSIX.1:** ISO/IEC 9945-1: 1990: *Information technology—Portable Operating System*
311 *Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language]* {B1}.

312 **2.3.4 Xlib:** The X Window System Intrinsics Library Interface, described in the *Xlib*
313 *Specification* {5}.

314 **2.3.5 Xt:** The X Toolkit Intrinsics, described in the *X Toolkit Specification* {4}.

315 2.4 C-Language Datatypes

316 Some datatypes used by the various system functions are not defined as part of this standard,
317 but are defined by the *Xlib Specification* {5} and the *X Toolkit Specification* {4}. These data-
318 types are listed in Annex A: *Boolean, Cardinal, Colormap, Dimension, Display, KeySym,*
319 *Pixel, Pixmap, Position, Screen, String, Time, Visual, Widget, XFontStruct, XImage, XtArg-*
320 *Val, XtCallbackList, XtPointer.*

321 The following datatypes are defined by the standard. These datatypes are defined in one of the
 322 headers listed in the synopsis subclause for the widget or function using the datatype. The
 323 exact header location of these datatypes (and of all symbols used in this standard) is
 324 unspecified, but the implementation shall provide them when all of the headers listed in the
 325 appropriate synopsis are included by the application.

326 **XmClipboardPendingList**

327 An array providing a mapping of Data ID to Private ID. The definition is

```
328 typedef struct {
329     long DataId;
330     long PrivateId;
331 } XmClipboardPendingRec, *XmClipboardPendingList;
```

332 **XmHighlightMode**

333 Specifies the emphasis to be done to some object.

```
334 — XmHIGHLIGHT_NORMAL
335 — XmHIGHLIGHT_SELECTED
336 — XmHIGHLIGHT_SECONDARY_SELECTED
```

337 **XmStringCharSet**

338 **XmStringCharSetTable**

339 XmStringCharSet specifies a font list element tag. This is either a null-
 340 terminated ISO8859-1 string that matches an XLFD {5} font specification or the
 341 value XmFONTLIST_DEFAULT_TAG. XmStringCharSetTable specifies a
 342 table of XmStringCharSet items.

343 **XmStringComponentType**

344 Specifies the type of a control of an XmString and parameters to various
 345 XmString utility functions.

```
346 — XmSTRING_COMPONENT_FONTLIST_ELEMENT_TAG
347 — XmSTRING_COMPONENT_LOCALE_TEXT
348 — XmSTRING_COMPONENT_TAG
349 — XmSTRING_COMPONENT_TEXT
350 — XmSTRING_COMPONENT_DIRECTION
351 — XmSTRING_COMPONENT_SEPARATOR
352 — XmSTRING_COMPONENT_END
353 — XmSTRING_COMPONENT_UNKNOWN
```

354 **XmTextBlock**

355 **XmTextBlockRec**

356 A data structure that specifies a block of text data that will be used for some
 357 operation. Usually this is the type of some data passed in a callback struct.

```

358         typedef struct {
359             char *ptr;                /* Pointer to data. */
360             int length;              /* Number of bytes of data. */
361             XmTextFormat format;     /* Representations format */
362         } XmTextBlockRec, *XmTextBlock;

```

XmTextDirection
 Specifies the search direction in a XmTextSource object.

- XmTEXT_FORWARD
- XmTEXT_BACKWARD

XmTextFormat
 Specifies the format of the text, either XmFMT_8_BIT or XmFMT_16_BIT.
 Shall be an Xatom.

XmTextPosition
 Specifies a position in a XmTextSource object. Shall be a value in the range 0 to the value of XmNmaxLength resource of the XmText or XmTextField widget.

XmTextScanType
 Specifies actions to perform on multiple mouse-button clicks.

- XmSELECT_POSITION
- XmSELECT_WORD
- XmSELECT_LINE
- XmSELECT_ALL

This is usually used to specify the elements of a selection array.

XmTraversalDirection
 Specifies the direction of traversal within a widget. This type takes its values from the following set:

- XmTRAVERSE_CURRENT
- XmTRAVERSE_NEXT
- XmTRAVERSE_PREV
- XmTRAVERSE_HOME
- XmTRAVERSE_NEXT_TAB_GROUP
- XmTRAVERSE_PREV_TAB_GROUP
- XmTRAVERSE_UP
- XmTRAVERSE_DOWN
- XmTRAVERSE_LEFT
- XmTRAVERSE_RIGHT

XmVisibility
 Specifies the visibility state of a widget.

- 396 — XmVISIBILITY_OBSCURED
- 397 — XmVISIBILITY_PARTIALLY_OBSCURED
- 398 — XmVISIBILITY_FULLY_OBSCURED

399 2.5 C-Language Definitions

400 2.5.1 Symbols from the C Standard

401 The following terms and symbols used in this standard are defined in the C Standard {3}:
 402 *NULL, byte, array of char, header, null character, string.*

403 The term *NULL pointer* in this standard shall be equivalent to the term *null pointer* used in the
 404 C standard {3}.

405 Additionally, the reservation of symbols that begin with an underscore applies:

- 406 (1) All external identifiers that begin with an underscore are reserved.
- 407 (2) All other identifiers that begin with an underscore and either an uppercase letter or
 408 another underscore are reserved.
- 409 (3) If the program defines an external identifier with the same name as a reserved external
 410 identifier, even in a semantically equivalent form, the behavior is undefined.

411 Certain other namespaces are reserved by the C Standard {3}. These reservations apply to this
 412 standard as well. Additionally, the C Standard {3} requires that it be possible to include a
 413 header more than once and that a symbol may be defined in more than one header. This
 414 requirement is also made of headers for this standard.

415 2.5.2 MTE Symbols

416 Certain symbols in this standard are defined in headers. This standard provides only general
 417 guidance on the contents of these headers, allowing implementations to select where symbols
 418 are defined and functions prototyped. Use of any of the widgets and functions defined in this
 419 standard requires the application to include one or more headers. The implementation shall
 420 ensure that, when the application includes all of the indicated headers for a widget or a func-
 421 tion description, all symbols indicated in that subclause shall be available.

422 If any header defined by this standard is included, all symbols with the prefix “Xm” are
 423 reserved for use by the implementation, both before and after the `#include` directive. Future
 424 revisions of this standard are likely to use symbols in this reserved space. Furthermore, the
 425 inclusion of any header defined by this standard may make visible any symbol defined in this
 426 standard.

427 In addition, implementations may add members to any structure or union defined in this stan-
 428 dard, as long as a user-defined macro with the same name cannot interfere with the correct
 429 interpretation of the program.

430 2.5.3 Header and Function Prototypes

431 Implementations shall declare function prototypes for all functions defined by this standard.
 432 The exact locations of these prototypes are unspecified, but the prototypes shall be included
 433 when the headers shown with `#include` directives in the appropriate function synopsis sub-
 434 clauses are used by the application.

Neither the names of the formal parameters nor their types, as they appear in an implementation, are specified by this standard. The names are used within this standard as a notational mechanism. However, any declaration provided by an implementation shall accept all actual parameter types that a declaration lexically identical to one in this standard shall accept, including the effects of both type conversion and checking for the number of arguments implied by the presence of a filled-out prototype. The declaration by the implementation shall not cause a syntax error if an application provides a prototype lexically identical to one in this standard. It is not a requirement that nonconforming parameters to functions that may be used by an application be diagnosed by an implementation, except as specifically required by this standard or the C Standard {3}, as applicable. Where the C Standard {3} has a more restrictive requirement for a function defined by that standard, that requirement shall be honored, and this exception does not apply.

2.6 Resource Names

2.6.1 MTE Resource Names

A widget resource is specified by a name of the form *XmNthisResource*. *XmNthisResource* may be associated with a class. The resource class may be used as an alternative way of identifying a collection of resources to be set via the X Resource Manager Database. The class name corresponding to *XmNthisResource* will be an *XmCxxx* name.

By convention, the value of the *XmNthisResource* is the string "thisResource". Also, by convention, the class name is the symbol *XmCThisResource* with value "ThisResource". Exceptions are listed in table 2-1.

2.6.2 Xm to Xt Resource Correspondence

Tables in this standard define a set of widget resources used by the programmer to specify data. The programmer can also set the resource values for the inherited classes to set attributes for this widget. To reference a resource by name or by class in an X Resource Manager Database, remove the *XmN* or *XmC* prefix and use the remaining letters. To specify one of the defined values for a resource in an X Resource Manager Database, remove the *Xm* prefix and use the remaining letters (either lowercased or uppercased, but include any underscores between words). The codes in the access column indicate if the given resource can be set at creation time (C), set by using *XtSetValues* (S), retrieved by using *XtGetValues* (G), or cannot be applicable (N/A).

Table 2-1 – Class Name Translation Exceptions

Resource Name	Resource Class Name
XmNallowResize	XmCBoolean
XmNbottomAttachment	XmCAttachment
XmNbottomOffset	XmCOffset
XmNbottomPosition	XmCPosition
XmNbottomWidget	XmCWidget
XmNcancelButton	XmCWidget
XmNcascadePixmap	XmCPixmap
XmNcommand	XmCTextString
XmNdefaultButton	XmCWidget
XmNeditMode	XmCEditMode
XmNentryAlignment	XmCAlignment
XmNentryVerticalAlignment	XmCVerticalAlignment
XmNfractionBase	XmCMaxValue
XmNhistoryItemCount	XmCItemCount
XmNhistoryItems	XmCItems
XmNhistoryMaxItems	XmCMaxItems
XmNhistoryVisibleItemCount	XmCVisibleItemCount
XmNhorizontalSpacing	XmCSpacing
XmNleftAttachment	XmCAttachment
XmNleftOffset	XmCOffset
XmNleftPosition	XmCPosition
XmNleftWidget	XmCWidget
XmNlistItemCount	XmCItemCount
XmNlistItems	XmCItems
XmNlistVisibleItemCount	XmCVisibleItemCount
XmNmenuAccelerator	XmCAccelerators
XmNmenuCursor	XmCCursor
XmNmenuHelpWidget	XmCMenuWidget
XmNmenuHistory	XmCMenuWidget
XmNmessageAlignment	XmCAlignment
XmNoffsetX	XmCOffset
XmNoffsetY	XmCOffset
XmNrefigureMode	XmCBoolean
XmNresizable	XmCBoolean
XmNrightAttachment	XmCAttachment
XmNrightOffset	XmCOffset
XmNrightPosition	XmCPosition
XmNrightWidget	XmCWidget
XmNsashShadowThickness	XmCShadowThickness
XmNscrollHorizontal	XmCScroll
XmNscrollLeftSide	XmCScrollSide
XmNscrollTopSide	XmCScrollSide
XmNscrollVertical	XmCScroll
XmNskipAdjust	XmCBoolean
XmNsubMenuId	XmCMenuWidget
XmNsymbolPixmap	XmCPixmap
XmNtextColumns	XmCColumns
XmNtextTranslations	XmCTranslations
XmNtopAttachment	XmCAttachment
XmNtopOffset	XmCOffset
XmNtopPosition	XmCPosition
XmNtopWidget	XmCWidget
XmNverticalSpacing	XmCSpacing

Section 3: Input Control

3.1 Input Model

The MTE input model includes

- The Keyboard model, which specifies the set of Keyboard input events.
- The Mouse model, which specifies the set of Mouse input events.
- The Focus model, which specifies which controls on the screen receive which input events.

3.1.1 Model Keyboard

This standard does not specify the characteristics of an actual keyboard device that may be part of a conforming implementation. Instead, this standard specifies a Model Keyboard for the purpose of specifying keyboard events. Conforming implementations shall provide means for generating each type of event defined for the Model Keyboard. The actual means of generating each event is implementation defined.

The Model Keyboard includes

- An implementation-defined set of keys corresponding to graphic characters
- The following modifier keys:
 - MAlt
 - MCtrl
 - MShift
 - MAny (used to represent any of these modifier keys)
- And the following control keys:
 - KActivate
 - KAddMode
 - KBackSpace
 - KBeginData
 - KBeginLine
 - KCancel
 - KClear

- 28 — KCopy
- 29 — KCut
- 30 — KDelete
- 31 — KDeselectAll
- 32 — KDown
- 33 — KEndData
- 34 — KEndLine
- 35 — KExtend
- 36 — KHelp
- 37 — KLeft
- 38 — KMenuBar
- 39 — KMenu
- 40 — KNextField
- 41 — KPageDown
- 42 — KPageLeft
- 43 — KPageRight
- 44 — KPageUp
- 45 — KPaste
- 46 — KPrevField
- 47 — KPrimaryCopy
- 48 — KPrimaryCut
- 49 — KPrimaryMove
- 50 — KPrimaryPaste
- 51 — KRestore
- 52 — KRight
- 53 — KSelect
- 54 — KSelectAll
- 55 — KSpace
- 56 — KTab
- 57 — KUp
- 58 — KAny (used to represent any of these control keys)

59 A keyboard event consists of a list of Modifier Keys, a Graphic Key or Control Key, and an
 60 Action. The list of modifiers may be empty. In a translation table, the symbol MANY matches
 61 any list of modifiers, and the symbol KAny matches any Graphic or Control key. The possible
 actions associated with a keyboard event include

62	Press	Indicates pressing a key without releasing it.
63	Release	Indicates releasing a key after pressing it.
64	Click	Indicates pressing and releasing a key.

65 3.1.2 Model Mouse

66 A pointing device is a tool that lets the user move a pointer on the screen. A pointing device
 67 also has some means of activating the object under the pointer. A pointing device allows the
 68 user to manipulate screen objects directly. Rather than entering a keyboard command for each
 69 action, a user can point to an object and directly manipulate it using the pointing device.

70 The most typical pointing device is a mouse, although a graphics tablet, track ball, joystick,
 71 and other tools also work as pointing devices. Throughout this standard we use the term
 72 mouse to refer to all pointing devices. Any pointing device can be used in place of a mouse.

73 With a mouse, objects on the screen can be manipulated by combining the mouse pointer with
 74 the mouse buttons.

75 This standard does not specify the characteristics of an actual pointing device that may be part
 76 of a conforming implementation. Instead, this standard specifies a Model Mouse for the pur-
 77 pose of specifying mouse events. Conforming implementations shall provide means for gen-
 78 erating each type of event defined for the Model Mouse. The actual means of generating each
 79 event is implementation defined.

80 The Model Mouse includes the following control buttons:

81	BSelect	Used to select, activate, and set the location cursor.
82	BTransfer	Used to move and copy elements.
83	BMenu	Used to pop up Menus.
84	BAny	Used to represent any of these control buttons.

85 Users can combine mouse button press and release actions along with mouse movements to
 86 activate controls. A mouse event consists of a list of Modifier Keys (see 3.1.1), a Mouse But-
 87 ton, and an Action. The list of modifiers may be empty. In a translation table, the symbol
 88 **MAny** matches any list of modifiers, and the symbol **BAny** matches any Mouse Button. The
 89 possible actions associated with a mouse event include

90	Press	Indicates pressing a mouse button without releasing it. This mouse 91 action is often used to select an object for action.
92	Release	Indicates releasing a mouse button after pressing it. This mouse action 93 is often used to perform the action initiated by a press or a drag.
94	Click	Indicates pressing and releasing a mouse button without moving the 95 pointer. This mouse action is used to select an object or perform an 96 action. The activation action actually occurs on the release of the 97 mouse button.
98	Motion	Indicates pressing a mouse button without releasing it and then moving 99 the position of the pointer. The action of a drag operation tracks the 100 position of the mouse pointer. The drag action ends with a release 101 action. This mouse action is commonly used to select a range of objects or move a selected object.

102	MultiClick	Indicates a number of clicks in quick succession. MultiClicks are often
103		referred to by the actual number of clicks, as in double-click or triple-
104		click. A double-click is often used to perform the default action of an
105		object.
106	MultiPress	Indicates a number of clicks in quick succession without a final release.
107		As in MultiClick, MultiPresses are often referred to by the actual
108		number of presses.
109	MultiMotion	Indicates a MultiPress action followed by moving the position of the
110		pointer. The action of a MultiMotion operation tracks the position of
111		the mouse pointer. The MultiMotion action ends with a release action.
112		As in MultiClick, MultiMotions are often referred to by the actual
113		number of presses.

114 3.1.3 The Focus Model

115 A typical screen can contain many windows. Each window shall receive input from the key-
 116 board, the mouse, or both. The window that receives keyboard events has the input focus.
 117 Indeed, when input is directed to a window, it is actually received by some control within the
 118 window. The input focus model determines which window on the screen and which control
 119 within that window receives each keyboard input. The input focus may also be referred to as
 120 the keyboard focus.

121 In order to avoid conflicts, only one window is allowed to have the input focus at a time. The
 122 window with the input focus shall be emphasized in some way, usually by a change in shade or
 123 color to the window border. A toolkit shall allow only one control at a time to have the input
 124 focus within the window that has the input focus.

125 The input focus model is defined by a focus policy. A focus policy is a specific mechanism for
 126 moving the focus among windows and controls. The focus policy can be different between
 127 windows than it is within windows. This section only attempts to describe the focus policies.
 128 Components shall support both explicit and implicit focus policies.

129 3.1.3.1 Implicit Focus

130 In the implicit focus policy, the input focus tracks the mouse pointer. No explicit action (other
 131 than mouse motion) is performed to set the input focus in the implicit focus model. When an
 132 implicit focus policy is used at the window level, keyboard events are sent to the window in
 133 which the mouse pointer is. When an implicit focus policy is used within a window, keyboard
 134 events are sent to the control in which the mouse pointer is. In implicit mode, the input focus
 135 tracks the mouse pointer. Because of this, there is no way to move the input focus from the
 136 keyboard using implicit mode. Implicit focus is sometimes referred to as pointer, track
 137 pointer, or track listener policy, or as being “real-estate” driven. In this focus policy, the
 138 location cursor for keyboard events does not need to be shown except in controls like Text, in
 139 which keyboard input is common.

140 3.1.3.2 Explicit Focus

141 The explicit focus policy requires the user to select explicitly which window or control
 142 receives the input focus. In explicit focus mode at the window level, a user moves the input
 143 focus to a window by pressing `BSelect` while the mouse pointer is over the window. Simply
 144 moving the mouse pointer over a window does not give the window the input focus. Because
 145 of this, explicit mode is often called click-to-type. In explicit mode within a window, a user
 146 generally moves the input focus to a specific control within a window by pressing `BSelect`
 147 over the control. Pressing `BSelect` shall not move focus to a control that is not traversable or
 148 does not accept input. Pressing `BSelect` in a control that is used only to change the visible
 149 portion of another control, such as a `ScrollBar` or `Sash`, should act on that control but should
 150 not move focus to it. In this focus policy, the location of input focus shall be shown by a loca-
 151 tion cursor.

152 In explicit mode, input focus moves explicitly among the controls in a window as well as
 153 among the windows. Moving the input focus among controls in a window using the keyboard
 154 is called control navigation. Keyboard events go to the control in the window with the input
 155 focus. Input focus is moved among controls using `KTab`, `MCtrl KTab`, `MShift KTab`,
 156 `MCtrl MShift KTab`, `KMenuBar`, `KMenu`, `KPrevField`, `KNextField`, `KBeginData`,
 157 `KEndData`, `KDown`, `KLeft`, `KRight`, and `KUP`.

158 If the focus policy is implicit, an implementation can still enable keyboard-based operations
 159 that explicitly move the location cursor. It shall at least implement the special cases for Menu
 160 traversal. `KMenuBar` moves the cursor to the `MenuBar`. `KMenu` pops up a `Popup Menu`.

161 3.2 Navigation

162 Regardless of whether they use a mouse, a keyboard, or both, users need to move the location
 163 cursor to new positions. That is, they will need to navigate around the screen. The model is
 164 simple for mouse users, and more complicated for keyboard users. This clause describes the
 165 general navigation model for mouse and keyboard users, describes the more complicated Menu
 166 navigation model, and then describes navigation within scrollable controls.

167 3.2.1 Pointer Navigation

168 In mouse-based navigation, the mouse is used to move the focus among controls. If an impli-
 169 cit focus policy is in use, the input focus simply follows the mouse pointer, and no other expli-
 170 cit action is required to change the focus.

171 With an explicit focus policy, pressing `BSelect` on a control shall move focus to it, except
 172 for controls that are used to adjust the size and location of other elements, such as `ScrollBars`
 173 and controls that are not traversable. Pressing `BSelect` on these controls need not move the
 174 focus. If not, after the mouse has acted on the control, the focus should remain on the control
 175 that previously had it as long as it is within the same window. Pressing `BSelect` shall also
 176 generally perform some selection or activation operation. Clicking `MCtrl BSelect` on a
 177 control can move focus to it without any other effect.

178 The only exception to the simple model of pointer navigation is a Menu system because
 179 Menus are not generally available on the screen until activated. Activating a Menu causes it to
 180 be shown on the screen. There are three types of Menus: `Pulldown Menus`, `Popup Menus`, and
 Option Menus. The `MenuBar` is also a special kind of Menu. A Menu system consists of all

the Menus cascading from a single CascadeButton or Popup Menu. The MenuBar system consists of the MenuBar and all of its associated Menus.

A Pulldown Menu is generally activated by pressing BSelect on a CascadeButton, which displays the Pulldown Menu. A Popup Menu is generally activated by pressing BMenu in an area with a Popup Menu, which displays the Popup Menu. An Option Menu is generally activated by pressing BSelect on the CascadeButton that displays the Option Menu. A Pulldown Menu pulled down from a CascadeButton within a Pulldown, Popup, or Option Menu is called a CascadeMenu. A MenuBar is generally activated by moving the input focus to the MenuBar. Since a MenuBar is always visible, activating the MenuBar does not change its appearance. Navigating to a Menu is equivalent to activating a Menu.

Menus are activated in one of two ways: spring-loaded or persistent. Spring-loaded means that the Menu is removed when the mouse button that activated it is released. Persistent means that the Menu is generally not removed when the mouse button that activated it is released, but shall be explicitly removed by another user action.

BSelect Press with the pointer on a Menu shall activate the Menu in a spring-loaded manner. BMenu Motion shall activate a Popup Menu in a spring-loaded manner.

Once a Popup Menu is activated, BMenu shall behave just like a BSelect for any Menu system. BSelect shall also be available from within an activated Popup Menu system and shall behave just as in any Menu system. BSelect Release or BMenu Release within a spring-loaded Menu system shall activate the button under the pointer at the time of release. If the pointer release is on a CascadeButton, the Cascade Menu shall be activated in a persistent manner.

Whenever any Menu is activated, the location cursor shall be placed on the default entry for the Menu, or on the first entry in the Menu if there is no default entry. Support for default entries in Menus is optional.

A spring-loaded Menu shall be removed when the mouse button that activated it is released, except when the button is released on a CascadeButton. While a spring-loaded Menu system is activated, moving the pointer within the Menu system shall move the location cursor to follow the pointer. If the pointer rests on a CascadeButton, the Cascade Menu shall be activated and shall also become traversable. The Menu shall be removed, possibly after a short delay, once the pointer moves to a Menu choice that is not part of the Cascade Menu or the CascadeButton. If the Menu is part of a MenuBar, moving the pointer to any other choice on the MenuBar shall remove the current Menu system, including a CascadeButton. Releasing the button during a spring-loaded Menu operation in a CascadeButton, even a CascadeButton in the MenuBar, causes the associated Menu to remain activated.

BMenu Click shall activate the Popup Menu in a persistent manner. BSelect Click on a Cascade Menu choice or OptionButton shall activate the Menu in a persistent manner.

A persistent Menu remains visible until explicitly removed. The focus model shall be forced to be explicit, and the Menu system can be traversed using the keyboard as described later in this section, or the mouse.

If a mouse button is pressed outside of the Menu to unpost the Menu, the button press can also be treated as follows:

- The button press can have no effect other than to remove the Menu.
- The button press can be passed to the underlying control, either always or only if some condition is met. For example, it can be passed to the underlying control only if the control is in the same window as the Menu system.

— Whether or not the button press is passed to the underlying control, it can have some other effect. For example, it can raise and give focus to the underlying window.

If the button press that removes a Menu is not also passed to the underlying control, subsequent events up to and including the button release shall not be passed to the underlying control.

Once a Pulldown or Option Menu is activated, `BSelect Motion` in the Menu system shall cause the Menu to behave as a spring-loaded Menu. Once a Popup Menu is posted, `BSelect Motion` or `BMenu Motion` in the Menu system shall cause the Menu to behave as a spring-loaded Menu. Because of this, the pointer-driven activation behavior of spring-loaded Menus fully describes pointer-driven activation behavior in Menus.

3.2.2 Keyboard-Based Navigation

The navigation model for keyboard users is more complicated than the pointer navigation model. As described earlier, keyboard navigation is only required when the focus policy is explicit. Since controls shall support the explicit focus policy, they shall also support keyboard navigation. The keyboard navigation model is composed of the following:

- A focus identifier (the location cursor)
- Navigating within windows

3.2.2.1 Component Navigation

Component navigation refers to any visual element that indicates the location and the item at that location to which events are sent if and when the window containing that item has active window focus.

Menu systems, including the MenuBar, follow a different traversal model, described later in this subclause.

A field can be an individual control, such as a Text control. In this case, the directional keys are used for internal navigation within the control. Alternately, a field can consist of a group of controls such as a RadioBox, which holds a group of RadioButtons. In this case, the directional keys are used to navigate among the controls of the group.

The fields in a window are ordered. `KTab` generally moves the location cursor to the next field, and `MShift KTab` moves the location cursor to the previous field. However, a field can use `KTab` and `MShift KTab` for internal navigation. For example, within a multiline Text control, `KTab` is used to tab within the text. Consequently, `MCtrl KTab` shall always navigate to the next field, and `MCtrl MShift KTab` shall always navigate to the previous field. The action associated with the next field navigation operation is `TraverseNext`. The action associated with the previous field navigation operation is `TraversePrev`.

`KTab` (if not used for internal navigation) and `MCtrl KTab` shall move the location cursor forward through fields according to the following rules:

- If the next field is a control, `KTab` (if not used for internal navigation) and `MCtrl KTab` shall move the location cursor to that control.
- If the next field is a group, `KTab` (if not used for internal navigation) and `MCtrl KTab` shall move the location cursor to a traversable control within the group. If the field contains a button that currently shows default emphasis, the location cursor should be

placed on that button; otherwise, the first control in the field (the top-leftmost one in a left-to-right environment) should receive the focus.

If the next field contains no traversable controls, `KTab` (if not used for internal navigation) and `MCtrl KTab` shall skip the field. Note that Separators and Labels should not be traversable. The ScrollBars of ScrolledWindows also should not be traversable, particularly if a `KScroll-Lock` key is available on the keyboard.

`MShift KTab` (if not used for internal navigation) and `MCtrl MShift KTab` shall move the location cursor backward through fields in the order opposite to that of `KTab` (if not used for internal navigation) and `MCtrl KTab`.

When a window acquires focus, the location cursor shall be placed on the control that last had focus in the window, providing that all the following conditions are met:

- The window uses an explicit input focus policy.
- The window acquires the focus through keyboard navigation or through a button press other than within the application area of the window.
- The window had the focus at some time in the past.
- The control that last had focus in the window is still traversable.

If the control that last had focus is no longer traversable, or if the window has not previously had the focus, the location cursor should be placed on the control with which the user is most likely to want to interact. In a `DialogBox`, this is often a text control or a default button. If no such control exists, the location cursor should be placed in the first field in the window—in a left-to-right environment, the top-leftmost field.

A toolkit should follow these general rules for field navigation:

- Field navigation shall wrap between the first and last fields of the window as long as the window is not scrollable.
- In a `PanedWindow`, each `Pane` should consist of one or more fields, and each `Sash` should be a field. Fields in a `PanedWindow` should be traversed in the following order:
 - (1) All fields in the topmost `Pane`
 - (2) The topmost `Sash`
 - (3) All fields in the next `Pane` toward the bottom
 - (4) The next `Sash` toward the bottom
 and so on, to the bottom of the `PanedWindow`.

Directional keys are used both for control navigation within a field and for internal purposes, including internal navigation, within a control.

When `KDown` and `KUp` are used for control navigation within a field, they shall behave in the following way:

- In a left-to-right environment, `KDown` shall move the location cursor through all traversable controls in the field, starting at the upper left and ending at the lower right, then wrapping to the upper left if not scrollable. If the controls are aligned in a matrix-like arrangement, `KDown` shall first traverse one column from top to bottom, then traverse the column to its right, and so on. In a right-to-left environment, `KDown` shall move the location cursor through all traversable controls, starting at the upper right and ending at the lower left.

305 — KUp shall move the location cursor through all traversable controls in the field in the
 306 order opposite to that of KDown.

307 When KRight and KLeft are used for control navigation within a field, they shall behave in
 308 the following way:

309 — In a left-to-right environment, KRight shall move the location cursor through all
 310 traversable controls in the field, starting at the upper left and ending at the lower right,
 311 then wrapping to the upper left if not scrollable. If the controls are aligned in a
 312 matrix-like arrangement, KRight shall first traverse one row from left to right, then
 313 traverse the row below it, and so on. In a right-to-left environment, KRight shall
 314 move the location cursor through all traversable controls, starting at the upper right and
 315 ending at the lower left.

316 — KLeft shall move the location cursor through all traversable controls in the field in the
 317 order opposite to that of KRight.

318 Components that use directional keys internally—such as Text, List, DrawingArea, Sash,
 319 ScrollBar, and Scale—should be fields. If a control that uses directional keys internally does
 320 not act like a field, the directional keys shall be used for both internal purposes and control
 321 navigation.

322 In particular, controls can use directional keys in one dimension for internal purposes and in
 323 another dimension for control navigation. For example, a vertically organized group of
 324 single-line Text controls can be grouped together as a single field. KDown and KUp navigate
 325 among the single-line Text controls in the field. KRight, MCtrl KRight, KLeft, and
 326 MCtrl KLeft navigate among characters and words within an individual single-line Text
 327 control.

328 The directional keys modified with MCtrl can also be used for control navigation, following
 329 the same rules specified above. If the directional keys modified with MCtrl are used for con-
 330 trol navigation, the unmodified directional keys can be used for internal purposes.

331 For example, a table can consist of an array of single-word Text controls, with KUp and KDown
 332 used to navigate up and down a column. KRight and KLeft can navigate among characters
 333 within an individual single-word Text control, while MCtrl KRight and MCtrl KLeft
 334 navigate horizontally among the Text controls in a row.

335 When the directional keys cause changes that are based on some unit, the directional keys
 336 modified by MCtrl can cause changes based on a larger unit. For example:

337 — In Text, KRight moves the cursor a character to the right, and MCtrl KRight moves
 338 the cursor a word to the right.

339 — In a Scale, KRight can increment the Scale value by one unit indicated by minor tick
 340 marks, and MCtrl KRight can increment the Scale value by an amount corresponding
 341 to the distance between major tick marks.

342 If a control uses KRight and KLeft for internal navigation, it shall support the following
 343 behavior:

344 KBeginLine 345 346 347	In a left-to-right environment, this action shall move the location cur- sor to the leftmost edge of the data or the leftmost element. In a right-to-left environment, this action shall move the location cursor to the rightmost edge of the data or the rightmost element.
-------------------------------------	--

348 **KEndLine** In a left-to-right environment, this action shall move the location cur-
 349 sor to the rightmost edge of the data or the rightmost element. In a
 350 right-to-left environment, this action shall move the location cursor to
 351 the leftmost edge of the data or the leftmost element.

352 If a control uses **KUp** and **KDown** for internal navigation, it shall support the following
 353 behavior:

354 **MCtrl KBeginLine** This action shall move the location cursor to one of the following:

- 355 — The first element
- 356 — The topmost edge of the data
- 357 — In a left-to-right environment, the topmost left edge of the data; in
 358 a right-to-left environment, the topmost right edge of the data

359 **MCtrl KEndLine** This action shall move the location cursor to one of the following:

- 360 — The last element
- 361 — The bottommost edge of the data
- 362 — In a left-to-right environment, the bottommost right edge of the
 363 data; in a right-to-left environment, the bottommost left edge of
 364 the data.

365 Groups that are fields can also use **KBeginLine**, **KEndLine**, **MCtrl KBeginLine**, and
 366 **MCtrl KEndLine** to move the location cursor to appropriate controls within the group.

367 **3.2.3 Menu Traversal**

368 The Menu traversal model is different from the field traversal model. This allows Menus to be
 369 traversable even when the focus policy is implicit. If a Menu is traversed to while the focus
 370 policy in the application is implicit, the focus policy shall temporarily change to explicit. The
 371 focus policy shall revert to implicit whenever the user traverses out of the Menu system.

372 Traversing to a Menu system is the same as activating the Menu system. If the MenuBar is
 373 inactive, **KMenuBar** shall traverse to, or activate, the MenuBar system. The location cursor
 374 shall be placed on the first traversable **CascadeButton** in the MenuBar. If there are no travers-
 375 able **CascadeButtons** in the MenuBar, **KMenuBar** shall do nothing.

376 If the input focus is on an element with an inactive Popup Menu and the context of the element
 377 allows a Popup Menu to be displayed, **KMenu** shall activate the Popup Menu. The location
 378 cursor shall be placed on the default item of the Menu, or the first traversable item if there is
 379 no default item. Menus activated from the keyboard should be in the context of the insertion
 380 position of the element with the location cursor. Even if there are no traversable items in the
 381 Popup Menu, the menu has to be activated.

382 If the input focus is in a **CascadeButton** that activates an Option Menu, **KSelect** or **KSpace**
 383 shall activate the Option Menu. The location cursor shall be placed on the previously selected
 384 item in the Option Menu. If the Option Menu is activated for the first time, the location cursor
 385 shall be placed on the default item in the Menu. If there is an active Option Menu, **KEnter**,
 386 **KReturn**, **KSelect**, or **KSpace** shall select the current item in the Option Menu, unpost the
 387 active Option Menu system, and return the location cursor to the **CascadeButton**.

388 Once a Menu system is activated, the Menu items can be traversed using `KDown`, `KLeft`,
 389 `KRight`, and `KUp`. A posted Menu system behaves somewhat like a field, with the addition of
 390 traversing among Menus in the system. When a Menu traversal action traverses to the next or
 391 previous control in a Menu or MenuBar, the order of traversal and the wrapping behavior shall
 392 be the same as that of the corresponding control navigation action within a field.

393 The following Menu traversal behavior shall be supported:

- 394 `KDown` This action shall do the following:
- 395 — If the control is in a vertical or two-dimensional Menu, traverse down to the
 - 396 next traversable control, wrapping within the Menu if necessary.
 - 397 — If the control is in a MenuBar, and the control with the keyboard focus is a
 - 398 CascadeButton, post its associated Pulldown Menu and traverse to the
 - 399 default entry in the Menu or, if the Menu has no default, to the first travers-
 - 400 able entry in the Menu.
- 401 `KUp` If the control is in a vertical or two-dimensional Menu, this action shall traverse
- 402 up to the previous traversable control, wrapping within the Menu if necessary
- 403 and proceeding in the order opposite to that of `KDown`.
- 404 `KLeft` This action shall do the following:
- 405 — If the control is in a MenuBar or two-dimensional Menu, but not at the left
 - 406 edge, traverse left to the previous traversable control.
 - 407 — If the control is at the left edge of a MenuBar, wrap within the MenuBar.
 - 408 — If the control is at the left edge of a vertical or two-dimensional Menu that
 - 409 is the child of a vertical or two-dimensional Menu, unpost the current Menu
 - 410 and traverse to the parent CascadeButton.
 - 411 — If the control is at the left edge of a vertical or two-dimensional Menu that
 - 412 is the child of a MenuBar, unpost the current Menu and traverse left to the
 - 413 previous traversable entry in the MenuBar. If that entry is a CascadeButton,
 - 414 post its associated Pulldown Menu and traverse to the default entry in the
 - 415 Menu or, if the Menu has no default, to the first traversable entry in the
 - 416 Menu.
- 417 `KRight` This action shall do the following:
- 418 — If the control is a CascadeButton in a vertical Menu, post its associated Pull-
 - 419 down Menu and traverse to the default entry in the Menu or, if the Menu has
 - 420 no default, to the first traversable entry in the Menu.
 - 421 — If the control is in a MenuBar or two-dimensional Menu, but not at the right
 - 422 edge, traverse right to the next traversable control.
 - 423 — If the control is at the right edge of a MenuBar, wrap within the MenuBar.
 - 424 — If the control is not a CascadeButton and is at the right edge of a vertical or
 - 425 two-dimensional Menu, and if the current Menu has an ancestor Cascade-
 - 426 Button (typically in a MenuBar) from which `KDown` posts its associated
 - 427 Pulldown Menu, unpost the Menu system pulled down from the nearest
 - 428 such ancestor CascadeButton and traverse right from that CascadeButton to
 - 429 the next traversable control. If that control is a CascadeButton, post its
 - associated Pulldown Menu and traverse to the default entry in the Menu or,
 - if the Menu has no default, to the first traversable entry in the Menu.

- 430 The user can use keyboard actions to exit a Menu or a Menu system in the following way:
- 431 — When a MenuBar system is active, KMenuBar should unpost the entire Menu system.
 - 432 — When a Popup Menu system is active, KMenu should unpost the entire Menu system.
 - 433 — In a Pulldown Menu, KCancel shall either dismiss the Menu and move the location
 - 434 cursor to the CascadeButton used to pull down the Menu or unpost the entire Menu
 - 435 system.
 - 436 — In a Popup Menu, Option Menu, TearOff Menu, or MenuBar, KCancel shall unpost
 - 437 the Menu system.
 - 438 — When KMenuBar, KMenu, or KCancel is used to unpost an entire Menu system and
 - 439 an explicit focus policy is in use, the location cursor shall be moved back to the control
 - 440 that had it before the Menu system was posted.

441 3.2.4 Scrollable Component Navigation

442 Certain controls, such as List and Text, have built-in support for scrolling. However, any con-
 443 trol or group of controls can be associated with scrolling controls or placed inside a Scrolled-
 444 Window and made scrollable.

445 A scrollable control generally has ScrollBars or some other type of scrolling control associated
 446 with it. When a control does not have a scrolling control associated with it, it generally should
 447 not be scrollable; however, controls whose visible contents alone indicate that additional items
 448 exist beyond the bounds of the visible area, like Text, can be scrollable even if they do not
 449 have an associated scrolling control.

450 Any scrollable control shall support the appropriate navigation and scrolling operations. It
 451 shall use the page navigation keys KPageUp, KPageDown, KPageLeft or MCtrl KPage-
 452 Up, and KPageRight or MCtrl KPageDown for scrolling the visible region by a page incre-
 453 ment. A page is the portion of data that is visible, not any underlying structure of the data.
 454 Scrolling by a page shall leave at least one unit of overlap between the old and new pages; for
 455 example, a line in a Text control.

456 If the location cursor can be made visible, it shall be. It should be moved within the control so
 457 that it remains as near as possible to its original location in the viewport. However, if a navi-
 458 gation key (including directional and page navigation keys) is pressed while the KScroll-
 459 Lock key is down, the navigation key should be interpreted as specified for an associated
 460 ScrollBar—that is, it causes scrolling, while leaving the position of the cursor within the
 461 scrollable control unchanged.

462 The directional keys, KUp, KDown, KLeft, and KRight, should be used for moving the loca-
 463 tion cursor among elements, moving the location cursor by increments, or scrolling the visible
 464 region by regular increments. In general, keyboard operations should traverse through the
 465 entire scrollable control, not just through the visible portion.

466 When a mouse button is pressed, initiating a selection operation within a scrollable control,
 467 and the pointer is then dragged outside of the scrollable control, the control should scroll
 468 toward the pointer. This is called autoscrolling.

469 When the location cursor is within a scrollable control, scrolling can move the cursor out of
 470 view; however, any keyboard operation that moves the cursor to or in the control, or that
 471 inserts, deletes, or modifies items at the cursor location shall scroll the control so that the

cursor is visible when the operation is complete. When scrolling using the mouse, the location cursor can be allowed to scroll out of the visible region. When scrolling using the keyboard, the location cursor shall be moved so that it remains within the visible region, if that is possible.

If a mouse-based scrolling action is in progress, `KCancel` shall cancel the scrolling action and return the slider to its position prior to the start of the scrolling operation.

3.3 Selection

The selection model determines how elements are selected from a group of elements. This section describes the selection models that are supported by a toolkit. MTE-compliant toolkits support an object-action selection model. In an object-action selection model, the user first selects an object and then performs an action on it. The object-action selection model mimics real life, so it is a familiar process to the user and increases the sense of control by the user over applications.

Objects include not only recognizable objects like windows, `PushButtons`, and `List` elements, but also control elements that are not always recognized as discrete objects, like individual letters of a text file. Groups of elements can be treated as a single element, either in a contiguous range or as a noncontiguous group. This clause describes the following topics:

- Selection models, which determine how elements in a collection are selected for later action
- Selection actions, which perform actions directly on a selection, such as cut, copy, and paste

3.3.1 Selection Models

Selection is used to mark one or more of a group of elements for latter action or so they can be moved, copied, or otherwise manipulated by the user.

Groups of elements can be organized into the following three types of collections:

- List-like collections
- Text-like collections
- Graphics-like collections

In list-like collections, when the pointer or location cursor is in the collection, the pointer or location cursor is considered to be on an element.

In text-like collections, when the pointer or location cursor is in the collection, the pointer or location cursor is considered to be between two elements. The name “text-like” refers to `Text` controls in insert mode where the text cursor is always between two letters. List-like and text-like collections are usually ordered as linear collections; that is, the collections are treated as if the elements were connected as a series of elements in one dimension.

In graphics-like collections, the pointer and location cursor can be either on or between elements. Graphics-like collections are usually ordered as two-dimensional collections. That is, the ordering of elements in the collection depends on both the horizontal and vertical position of the element within each collection.