VMS DECwindows Toolkit Routines Reference Manual

Order Number: Part II: AA–MK88A–TE

December 1988

This manual describes the VMS DECwindows Toolkit routines, an implementation of the XUI Toolkit by DIGITAL.

Revision/Update Information:	This is a new manual.
Software Version:	VMS Version 5.1

digital equipment corporation
maynard, massachusetts

December 1988

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

ZK4731

3 Convenience Routines

This chapter contains XUI Toolkit convenience routines and message routines.

Table 3–1 lists the supported convenience routines.

Table 3–1 Convenience Routines

Routine Name	Description
CHILDREN	Returns a list of the widget's children.
GET DISPLAY	Returns the widget display.
GET SCREEN	Returns the widget screen.
GET WINDOW	Returns the widget window.
NUMBER CHILDREN	Returns the number of children that the widget has.
RESOLVE PART OFFSETS	Allows writing of upward-compatible applications and widgets.
VMS CLEAR STRING	Frees a string in an argument list.
VMS FREE ARGNAMES	Frees memory allocated for argument names.
VMS GET DESC VALUE	Retrieves a descriptor value.
VMS SET ARG	Places an argument in the argument list.
VMS SET CALLBACK ARG	Places a callback in the argument list.
VMS SET DESC ARG	Places a descriptor in the argument list.

Table 3–2 lists the supported message routines.

Table 3–2 Message Routines

Routine Name	Description
DISPLAY CS MESSAGE	Displays a compound string message.
DISPLAY VMS MESSAGE	Accepts and displays a VMS message.

3.1 Convenience Routines

Convenience routines provide an easy method for obtaining commonly required information. Some routines do not have MIT C bindings.

The following pages describe the XUI Toolkit convenience routines.

CHILDREN

Returns a list of the widget's children.

VAX FORMAT

widget_list = **DWT$CHILDREN** *(widget)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget_list	array	uns longword	read	reference
widget	widget	uns longword	read	reference

MIT C FORMAT

widget_list = **DwtChildren** *(widget)*

argument information

```
widget_list DwtChildren (widget)
     CompositeWidget  widget;
```

RETURNS

widget_list
A list of the widget's children.

ARGUMENTS

widget
A pointer to the widget data structure.

DESCRIPTION

CHILDREN returns a list of the widget's children. Children must request geometry management changes from their parent. A parent widget can resize its children.

GET DISPLAY

Returns the widget display.

VAX FORMAT

display = **DWT$GET_DISPLAY** *(widget)*

argument information

Argument	Usage	Data Type	Access	Mechanism
display	uns longword	uns longword	write	value
widget	widget	uns longword	read	reference

MIT C FORMAT

display = **DwtGetDisplay** *(widget)*

argument information

```
display DwtGetDisplay (widget)
     Widget  widget;
```

RETURNS

display
The widget display.

ARGUMENTS

widget
A pointer to the widget data structure.

DESCRIPTION

GET DISPLAY returns information about the physical device on which the widget is displayed. A display is the identifier of the connection between the client and the server.

GET SCREEN

Returns the widget screen.

VAX FORMAT

screen = **DWT$GET_SCREEN** *(widget)*

argument information

Argument	Usage	Data Type	Access	Mechanism
screen	uns longword	uns longword	write	value
widget	widget	uns longword	read	reference

MIT C FORMAT

screen = **DwtGetScreen** *(widget)*

argument information

```
Screen *DwtGetScreen(widget)
     Widget  widget;
```

RETURNS

screen
The widget screen.

ARGUMENTS

widget
A pointer to the widget data structure.

DESCRIPTION

GET SCREEN returns the widget screen. The screen is the physical hardware on which the widget is being displayed.

GET WINDOW

Returns the widget window.

VAX FORMAT

window = **DWT$GET_WINDOW** *(widget)*

argument information

Argument	Usage	Data Type	Access	Mechanism
window	uns longword	uns longword	write	value
widget	widget	uns longword	read	reference

MIT C FORMAT

window = **DwtGetWindow** *(widget)*

argument information

```
window DwtGetWindow(widget)
     Widget  widget;
```

RETURNS

window
The widget window.

ARGUMENTS

widget
A pointer to the widget data structure.

DESCRIPTION

GET WINDOW returns the widget window. The window is a rectangular portion of the screen that has event handling and dispatching capabilities.

NUMBER CHILDREN

Returns the number of children that the widget has.

VAX FORMAT

cardinal = **DWT$NUMBER_CHILDREN** *(widget)*

argument information

Argument	Usage	Data Type	Access	Mechanism
cardinal	uns longword	uns longword	write	value
widget	widget	uns longword	read	reference

MIT C FORMAT

cardinal = **DwtNumberChildren** *(widget)*

argument information

```
Cardinal DwtNumberChildren(widget)
     CompositeWidget  widget;
```

RETURNS

cardinal
The number of children; a positive integer.

ARGUMENTS

widget
A pointer to the widget data structure.

DESCRIPTION

NUMBER CHILDREN returns the number of children that the widget has.

RESOLVE PART OFFSETS

Allows writing of upward-compatible applications and widgets.

VAX FORMAT

void = **DWT$RESOLVE_PART_OFFSETS**
(widget_class, offset)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget_class	widget class	uns longword	read	reference
offset	uns longword	uns longword	write	reference

MIT C FORMAT

void = **DwtResolvePartOffsets**
(widget_class, offset)

argument information

```
void DwtResolvePartOffsets(widget_class, offset)
     WidgetClass   widget_class;
     DwtOffsetPtr  *offset;
```

ARGUMENTS

widget_class
The widget class for which to resolve the part offset.

offset
The variable that contains a pointer to the class offset array.

DESCRIPTION

RESOLVE PART OFFSETS allows you to write upward-compatible applications and widgets. The widget instance records defined by the X Toolkit are not inherently upward compatible. That is, applications and widgets built outside of the X Toolkit cannot use hard-coded offsets in the widget instance records, and they cannot expect to run without recompilation as subsequent releases of the XUI toolkit are made. There is, however, a method for writing upward-compatible applications and widgets.

Applications and widgets must never read or write another widget's fields directly. The only exception is that the parent of a widget can write the widget's geometry fields as its geometry manager. A widget not guaranteed to be recompiled with the intrinsic routines must use an **offset record** to access the fields within its instance record.

The use of offset records requires one extra global variable per widget class. The variable consists of a pointer to an array of offsets in the widget record for each part of the widget structure. These offsets are allocated by RESOLVE PART OFFSETS and are used by the widget to access all of the widget's variables.

A widget needs to perform the following actions:

- Instead of creating a resource list, the widget creates an offset resource list (C macros DwtPartResource and DwtPartOffset are provided as an aid). The data structure for this looks just like a resource list, but instead of having one integer for its offset, it has two 16-bit unsigned quantities. These quantities are put into the class record as if it were a normal resource list. Instead of using XtOffset for the offset, it uses DwtPartOffset.
- Instead of putting the widget size in the class record, the widget puts the widget part size in the same field.
- Instead of putting the symbol XtVersion in the class record, the widget puts the symbol XtVersionDontCheck in the class record.
- The widget defines a variable to point to the offset record. This can be part of the widget's class record or a separate global variable.
- In class initialization, the widget calls the RESOLVE PART OFFSETS, and passes it the address of the offset variable and the class record. This operation does several things:
  - — Adds the superclass (which, by definition, has already been initialized) size field to the part size field.
  - — Allocates an array based upon the number of superclasses.
  - — Fills in the offsets of all the widget parts with the appropriate values, determined by examining the size fields of all superclass records.
  - — Uses the part offset array to modify the offset entries in the resource list to be real offsets, in place.
- Instead of accessing fields directly, the widget must always go through the offset array. The entries contain the offsets within the widget instance record of the various widget parts. The widget must add the offset of a particular field within a part to the offset of the part to get the address of the field.

  You can define macros for each field to make this easier. Assume an integer field member *xyz*:

```
#define BarXyz(w)                       \
        (*(int *) (((char *) w) +       \
        offset [BarIndex] +             \
        XtOffset (BarPart,xyz)))
```

  A C macro DwtField has been provided. Because the DwtPartOffset and DwtField macros concatenate arguments, make sure there is no space before or after the part argument; for example:

```
DwtField(w, offset,Label, text, char *)
```

Note that there are no spaces placed before or after *Label*.

Because of the space before the part argument, the following example does not work:

```
DwtField(w, offset, Label, text, char *)
```

VMS CLEAR STRING

Frees a string in an argument list.

VAX FORMAT

void = **DWT$VMS_CLEAR_STRING**
(arglist, argnumber)

argument information

Argument	Usage	Data Type	Access	Mechanism
arglist	uns longword	uns longword	read	reference
argnumber	uns longword	uns longword	read	reference

ARGUMENTS

arglist
The argument list.

argnumber
The index to the argument list where the argument is to be placed.

DESCRIPTION

VMS CLEAR STRING frees a string in an argument list.

VMS FREE ARGNAMES

Frees memory allocated for argument names.

VAX FORMAT

void = **DWT$VMS_FREE_ARGNAMES**
(arglist, argcount)

argument information

Argument	Usage	Data Type	Access	Mechanism
arglist	uns longword	uns longword	read	reference
argcout	uns longword	uns longword	read	reference

ARGUMENTS

arglist
The argument list.

argcount
The index to the argument list where the argument is to be placed.

DESCRIPTION

VMS FREE ARGNAMES frees memory allocated for argument names in the argument list. This memory is allocated by VMS SET ARG, converting the name descriptors into C null-terminated strings.

VMS GET DESC VALUE

Gets a descriptor value.

VAX FORMAT

void = **DWT$VMS_GET_DESC_VALUE**
(value, buffer, length)

argument information

Argument	Usage	Data Type	Access	Mechanism
value	uns longword	uns longword	read	reference
buffer	uns longword	uns longword	read	reference
length	uns longword	uns longword	read	reference

ARGUMENTS

value
The address of the null-terminated string.

buffer
The descriptor into which the argument is written.

length
The length of the descriptor.

DESCRIPTION

VMS GET DESC VALUE retrieves a descriptor from an argument list.

VMS SET ARG

Places an argument in the argument list.

VAX FORMAT

void = **DWT$VMS_SET_ARG**
(arg, arglist, argcount, argname)

argument information

Argument	Usage	Data Type	Access	Mechanism
arg	longword	longword	read	reference
arglist	uns longword	uns longword	read	reference
argcount	uns longword	uns longword	read	reference
argname	char string	char string	read	descriptor

ARGUMENTS

arg
The argument to be added to the argument list.

arglist
The argument list.

argcount
The index to the argument list where the argument is to be placed.

argname
The name of the resource.

DESCRIPTION

VMS SET ARG places any argument except the callback pointer or descriptor in the argument list. Note that the argument name descriptor is converted into a C null-terminated string, which involves memory allocation. Release this memory after using the argument list by calling VMS FREE ARGNAMES.

VMS SET CALLBACK ARG

Places a callback in the argument list.

VAX FORMAT

void = **DWT$VMS_SET_CALLBACK_ARG**
(callback_arg, arglist, argnumber, argname)

argument information

Argument	Usage	Data Type	Access	Mechanism
callback_arg	uns longword	uns longword	write	reference
arglist	uns longword	uns longword	read	reference
argnumber	uns longword	uns longword	read	reference
argname	char_string	char_string	read	descriptor

ARGUMENTS

callback_arg
The callback list.

arglist
The argument list.

argnumber
The index to the argument list where the argument is to be placed.

argname
The name of the argument.

DESCRIPTION

VMS SET CALLBACK ARG places a callback in the argument list. Note that the argument name descriptor is converted into a C null-terminated string, which involves memory allocation. Release this memory after using the argument list by calling VMS FREE ARGNAMES.

VMS SET DESC ARG

Places a descriptor in the argument list.

VAX FORMAT

void = **DWT$VMS_SET_DESC_ARG**
(arg, arglist, argcount, argname)

argument information

Argument	Usage	Data Type	Access	Mechanism
arg	longword	longword	read	reference
arglist	uns longword	uns longword	read	reference
argcount	uns longword	uns longword	read	reference
argname	char_string	char_string	read	descriptor

ARGUMENTS

arg
The descriptor to be added to the argument list.

arglist
The argument list.

argcount
The index to the argument list where the argument is to be placed.

argname
The name of the argument.

DESCRIPTION

VMS SET DESC ARG places a descriptor in the argument list. Note that the argument name descriptor is converted into a C null-terminated string, which involves memory allocation. Release this memory after using the argument list by calling VMS FREE ARGNAMES.

3.2 Message Routines

The message routines allow messages to be formatted using the $FAO utility, and to be displayed in a message box. Messages such as those from the operating system appear in a message box. The messages themselves are either stored in a VMS message file or are supplied by the application as compound strings.

The message box can be either modal or modeless. The title bar identifies the message box with the title Message and contains a push-to-back icon. The message box contains a message box icon in the upper-left corner, an Acknowledged push button on the bottom, and the text of the message in the middle. The lines of the message are separated by <CR><LF> pairs or "!/" in $FAO. Multiple messages are separated by blank lines.

The user clicks on the Acknowledged push button to erase the message. To receive online help on the message, the user presses the Help key while the message box has input focus.

In addition to the standard $FAO system service flags, the compound string message routines (DWT$DISPLAY_CS_MESSAGE and DwtDisplayCsMessage) accept the flag "!CS". When used, this flag accepts a compound string itself.

ULTRIX applications can make use of the $FAO string substitution utility. Messages defined in the XUI Resource Manager (DRM) database can be supplied to the appropriate message routine to be formatted and displayed.

The following pages describe the XUI Toolkit message routines.

DISPLAY CS MESSAGE

Displays a compound string message

VAX FORMAT

widget = **DWT$DISPLAY_CS_MESSAGE**
(parent_widget, name, default_position, x, y, style, message_vector, widget, convert_proc, ok_callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char_string	char string	read	descriptor
default_position	Boolean	uns longword	read	reference
x	position	uns longword	read	reference
y	position	uns longword	read	reference
style	longword	uns longword	read	reference
message_vector	cntrlblk	longword	read	reference
widget	widget	uns longword	modify	reference
convert_proc	void_proc	proc entry mask	read	reference
ok_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtDisplayCsMessage**
(parent_widget, name, default_position, x, y, style, message_vector, widget, convert_proc, ok_callback, help_callback)

argument information

```
Widget DwtDisplayCsMessage(parent_widget, name,
                           default_position, x, y,
                           style, message_vector,
                           widget, convert_proc,
                           ok_callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Boolean         default_position;
      Position        x;
      Position        y;
      unsigned char   style;
      int             *message_vector;
      Widget          *widget;
      int             (*convert_proc)();
      DwtCallbackPtr  ok_callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget for the created widget.

name
The name of the created widget.

default_position
If true, x and y values are ignored, which forces the default. The default position is centered in the parent window.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window.

y
The placement, in pixels, of the upper side of the widget window relative to the inner upper left corner of the parent window.

style
The style of the dialog box widget. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal type box (default)
DWT$C_MODELESS	DwtModeless	Modeless type box

message_vector
The message argument vector specifying the compound strings and associated information.

The first longword contains the number of longwords in the message blocks to follow. The first longword in each message block contains a pointer to the compound string. The next word consists of the $FAO parameter count. The final remaining longwords in the message block are the $FAO parameters.

In addition to the standard $FAO system service flags, the compound string message routine will accept the new FAO directive "!CS." When used, this directive will insert a compound string itself.

widget

The widget identifier of an already-existing message box widget. If this argument is not zero, a new message box is not created. The intrinsic routine SET VALUES is called on this widget to change the text of the message to match the new message. This variable can be modified, so after the routine is called, the message box widget identifier is filled in by the routine.

convert_proc

A pointer to a routine that is executed after the message is formatted but before it is displayed.

A pointer to the formatted string is passed to the routine as a parameter. In the VAX binding, the parameter is a descriptor. In the C binding, the parameter is a null-terminated character string.

ok_callback

A callback descriptor data structure. The callback is executed when the user clicks on the Acknowledged button. The reason is CRYes.

help_callback

A callback descriptor data structure. The callback is executed when the user requests help. The reason returned is CRHelp.

DESCRIPTION

DISPLAY CS MESSAGE accepts an array of compound strings, formats them, and creates a message box.

If the routine returns a zero, the message is not appended to the messages to be displayed.

DISPLAY VMS MESSAGE

Accepts and displays a VMS message.

VAX FORMAT

widget = **DWT$DISPLAY_VMS_MESSAGE**
(parent_widget, name, default_position, x, y, style, message_vector, widget, convert_proc, ok_callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char_string	char string	read	descriptor
default_position	Boolean	uns longword	read	reference
x	position	uns longword	read	reference
y	position	uns longword	read	reference
style	longword	uns longword	read	reference
message_vector	cntrlblk	longword	read	reference
widget	widget	uns longword	modify	reference
convert_proc	void_proc	proc entry mask	read	reference
ok_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtDisplayVmsMessage**
(parent_widget, name, default_position, x, y, style, message_vector, widget, convert_proc, ok_callback, help_callback)

argument information

```
Widget DwtDisplayVmsMessage(parent_widget, name,
                            default_position, x, y,
                            style, message_vector,
                            widget, convert_proc,
                            ok_callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Boolean         default_position;
      Position        x;
      Position        y;
      unsigned char   style;
      int             *message_vector;
      Widget          *widget;
      int             (*convert_proc)();
      DwtCallbackPtr  ok_callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget for the created widget.

name
The name of the created widget.

default_position
If true, x and y are ignored, which forces the default. The default position is centered in the parent window.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window.

y
The placement, in pixels, of the upper side of the widget window relative to the inner upper left corner of the parent window.

style
The style of the dialog box widget. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal type box (default)
DWT$C_MODELESS	DwtModeless	Modeless type box

message_vector
The message argument vector specifying the message identifier and associated information. This argument is identical to the VMS $PUTMSG system service.

The first longword contains the number of longwords in the message blocks to follow. The first longword in each message block contains a pointer to the VMS message identifier. Message identifiers are passed by value.

If the message is supplied by the application, the next word consists of the $FAO parameter count. The remaining longwords in the message block are the $FAO parameters.

widget

The widget identifier of an already-existing message box widget. If this argument is not zero, a new message box is not created. The intrinsic routine SET VALUES is called on this widget to change the text of the message to match the new message. This variable can be modified, so after the routine is called, the message box widget identifier is filled in by the routine.

convert_proc

A pointer to a routine that is executed after the message is formatted but before it is displayed.

A pointer to the formatted string is passed to the routine as an argument. In the VAX binding, the parameter is a descriptor. In the C binding, the parameter is a null-terminated character string.

ok_callback

A callback descriptor data structure. The callback is executed when the user clicks on the Acknowledged button. The reason is CRYes.

help_callback

A callback descriptor data structure. The callback is executed when the user requests help. The reason returned is CRHelp.

DESCRIPTION

DISPLAY VMS MESSAGE accepts standard VMS message vectors (as defined by the $PUTMSG system service), retrieves the messages, formats them, and creates a message box in which to display the message.

If the routine returns a zero, the message is not appended to the messages to be displayed.

4 XUI Resource Manager (DRM) Routines

The following routines define the application interface to the XUI Resource Manager (DRM). The DRM is responsible for creating widgets based on definitions contained in the User Interface Definition (UID) files created by the User Interface Language (UIL) compiler. The DRM interprets the output of the UIL compiler and generates the appropriate argument lists for the low-level widget routines.

The routines in this chapter allow an application to initialize the DRM, to provide information required by the DRM to successfully interpret information contained in the UID files, and to create widgets using the UID file definitions. The DRM also contains routines that allow an application to read literal definitions from the UID files. These definitions are created by using the EXPORT VALUE definitions in the UIL; the resulting literals may be used for any purpose the application requires.

The representation of widgets in a UID file is not exposed in these routines. All management and translation of these representations is done internally. The interface for reading literals is low-level, and it exposes the definition of both literals themselves and of the DRM mechanisms for accessing the UID file resources.

All definitions required to use the DRM facilities are contained in DwtAppl.h.

For concepts related to DRM routines and information about how to use them, see the *VMS DECwindows User Interface Language Reference Manual*. Table 4–1 lists the supported DRM routines.

Table 4–1 DRM Routines

Routine Name	Description
CLOSE HIERARCHY	Closes a DRM hierarchy.
DRM FREE RESOURCE CONTEXT	Frees a resource context and its buffer.
DRM GET RESOURCE CONTEXT	Gets a new resource context and a buffer.
DRM HGET INDEXED LITERAL	Fetches indexed literals from a DRM hierarchy.
DRM RC BUFFER	Returns a pointer to the resource context buffer.
DRM RC SET TYPE	Modifies the type in the resource context.
DRM RC SIZE	Returns the size of the value in the resource context.
DRM RC TYPE	Returns the type of the value in the resource context buffer.

(continued on next page)

Table 4–1 (Cont.) DRM Routines

Routine Name	Description
FETCH INTERFACE MODULE	Fetches all the widgets defined in an interface module in the DRM hierarchy.
FETCH SET VALUES	Fetches the values to be set from literals stored in UID files.
FETCH WIDGET	Fetches any indexed widget.
FETCH WIDGET OVERRIDE	Fetches any indexed widget. Overrides FETCH WIDGET arguments.
INITIALIZE DRM	Prepares an application to use the DRM facilities.
OPEN HIERARCHY	Opens all the UID files in the DRM hierarchy.
REGISTER CLASS	Provides the DRM with the information about a widget class defined by the application.
REGISTER DRM NAMES	Registers a vector of names of identifiers or callback routines for access in DRM.

4.1 DRM Routines

The following pages describe the XUI Toolkit XUI Resource Manager (DRM) routines.

CLOSE HIERARCHY

Closes a DRM hierarchy.

VAX FORMAT

status = **DWT$CLOSE_HIERARCHY** *(hierarchy_id)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
hierarchy_id	identifier	longword	read	value

MIT C FORMAT

status = **DwtCloseHierarchy** *(hierarchy_id)*

argument information

```
int DwtCloseHierarchy(hierarchy_id)
      DRMHierarchy  hierarchy_id;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMFailure	The operation failed.

ARGUMENTS

hierarchy_id

The identifier of an open DRM hierarchy.

DESCRIPTION

CLOSE HIERARCHY closes a DRM hierarchy. A DRM hierarchy is a collection of open UID files.

DRM FREE RESOURCE CONTEXT

Frees a resource context and its buffer.

VAX FORMAT

status = **DWT$DRM_FREE_RESOURCE_CONTEXT** *(context_id)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
context_id	identifier	longword	read	value

MIT C FORMAT

status = **DwtDrmFreeResourceContext** *(context_id)*

argument information

```
int DwtDrmFreeResourceContext(context_id)
      DRMResourceContextPtr  context_id;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMBadContext	The resource context is invalid.

ARGUMENTS

context_id

The resource context to be freed.

DESCRIPTION

DRM FREE RESOURCE CONTEXT frees the resource context and its associated buffer.

DRM GET RESOURCE CONTEXT

Gets a new resource context and a buffer.

VAX FORMAT

status = **DWT$DRM_GET_RESOURCE_CONTEXT**
(alloc_func, free_func, size, context_id_return)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
alloc_func	procedure	procedure entry mask	read	value
free_func	void proc	procedure entry mask	read	value
size	word	word	read	reference
context_id_return	identifier	longword	write	reference

MIT C FORMAT

status = **DwtDrmGetResourceContext**
(alloc_func, free_func, size, context_id_return)

argument information

```
int DwtDrmGetResourceContext(alloc_func, free_func, size,
                             context_id_return)
      char                   *((*alloc_func)());
      void                   (*free_func)();
      DRMSize                size;
      DRMResourceContextPtr  *context_id_return;
```

RETURNS

status
The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMFailure	The operation failed.

ARGUMENTS

alloc_func
A routine that allocates memory for this resource context. A null pointer activates the default (the intrinsic routine MALLOC).

free_func
A routine that frees memory for this resource context. A null pointer activates the default (the intrinsic routine FREE).

size
The size of the buffer.

context_id_return
The new resource context.

DESCRIPTION

DRM GET RESOURCE CONTEXT allocates a new resource context and a buffer of the requested size. DRM GET RESOURCE CONTEXT then associates the buffer with the resource context.

DRM HGET INDEXED LITERAL

Fetches indexed literals from a DRM hierarchy.

VAX FORMAT

status = **DWT$DRM_HGET_INDEXED_LITERAL** *(hierarchy_id, index, context_id)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
hierarchy_id	identifier	longword	read	value
index	char string	char string	read	descriptor
context_id	identifier	longword	read	value

MIT C FORMAT

status = **DwtDrmHGetIndexedLiteral** *(hierarchy_id, index, context_id)*

argument information

```
int DwtDrmHGetIndexedLiteral(hierarchy_id, index, context_id)
      DRMHierarchy           hierarchy_id;
      String                 index;
      DRMResourceContextPtr  context_id;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMFailure	The operation failed.
DRMNotFound	The literal was not found.

ARGUMENTS

hierarchy_id

The identifier of an open DRM hierarchy.

index

The case-sensitive name of the desired literal.

context_id

The identifier of the resource context into which the literal is read.

DESCRIPTION

DRM HGET INDEXED LITERAL searches a DRM hierarchy for a literal given its index; that is, it gets an exported literal from a DRM search hierarchy. DRM GET INDEXED LITERAL returns the literal as the contents of the resource context buffer. The group that is fetched is always **DRMgLiteral**.

Prior to calling HGET INDEXED LITERAL, the DRM routine RC SET TYPE should be called using both the same resource context identifier as the one used in HGET INDEXED LITERAL, and the constant **DRMtNul**.

The literal type filter is taken from the resource context; if unmodified in the context as obtained from DRM GET RESOURCE CONTEXT, there is no filtering (type=**RGMtNul**). In general, you do not need to set any of the fields in the resource context, except, possibly, type.

The following buffer contents are for some common literal types obtained from a UID file. Note that in some cases that the caller of the routine must cause offsets to be memory pointers.

```
DwtDrmRCType(context_id) == RGMrTypeChar8:
    DwtDrmRCBuffer(context_id) contains a null-terminated ASCII string

DwtDrmRCType(context_id) == RGMrTypeCString:
    DwtDrmRCBuffer(context_id) contains a compound string (DwtCompString)

DwtDrmRCType(context_id) == RGMrTypeChar8Vector:
DwtDrmRCType(context_id) == RGMrTypeCStringVector:
    DwtDrmRCBuffer(context_id) contains an RGM text vector or stringtable
    (RGMTextVector). The items in the text vector contain offsets into the
    buffer that locate either null-terminated ASCII strings or compound
    strings. These can be relocated to memory pointers by adding the buffer
    address to the offset, that is:
```

$$item[n].text\_item.pointer = item[n].text\_item.offset + bufadr$$

DRM RC BUFFER

Returns a pointer to the resource context buffer.

VAX FORMAT

buffer = **DWT$DRM_RC_BUFFER** *(context_id)*

argument information

Argument	Usage	Data Type	Access	Mechanism
buffer	address	longword	write	value
context_id	identifier	longword	read	value

MIT C FORMAT

buffer = **DwtDrmRCBuffer** *(context_id)*

argument information

```
buffer DwtDrmRCBuffer(context_id)
      DRMResourceContextPtr  context_id;
```

RETURNS

buffer
A pointer to the resource context buffer.

ARGUMENTS

context_id
The resource context.

DESCRIPTION

DRM RC BUFFER returns a pointer to the resource context buffer that contains the data for this resource context.

DRM RC SET TYPE

Modifies the type in the resource context.

VAX FORMAT

void = **DWT$DRM_RC_SET_TYPE** *(context_id, type)*

argument information

Argument	Usage	Data Type	Access	Mechanism
context_id	identifier	longword	read	value
type	word	word	read	reference

MIT C FORMAT

void = **DwtDrmRCSetType** *(context_id, type)*

argument information

```
void DwtDrmRCSetType(context_id,type)
      DRMResourceContextPtr  context_id;
      DRMType                type;
```

ARGUMENTS

context_id
The resource context.

type
The new value for the resource type.

DESCRIPTION

DRM RC SET TYPE modifies the type in the resource context. DRM RC SET TYPE is called prior to DRM HGET INDEXED LITERAL.

DRM RC SIZE

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

VAX FORMAT

size = **DWT$DRM_RC_SIZE** *(context_id)*

argument information

Argument	Usage	Data Type	Access	Mechanism
size	integer	word	write	value
context_id	identifier	longword	read	value

MIT C FORMAT

size = **DwtDrmRCSize** *(context_id)*

argument information

```
DRMsize DwtDrmRCSize(context_id)
      DRMResourceContextPtr  context_id;
```

RETURNS

size
The size, in bytes, of the value in the resource context buffer.

ARGUMENTS

context_id
The resource context.

DESCRIPTION

DRM RC SIZE returns the size of value in the resource context buffer.

DRM RC TYPE

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

VAX FORMAT

type = **DWT$DRM_RC_TYPE** *(context_id)*

argument information

Argument	Usage	Data Type	Access	Mechanism
type	integer	word	write	value
context_id	identifier	longword	read	value

MIT C FORMAT

type = **DwtDrmRCType** *(context_id)*

argument information

```
DRMtype DwtDrmRCType(context_id)
      DRMResourceContextPtr  context_id;
```

RETURNS

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

ARGUMENTS

context_id
The resource context.

DESCRIPTION

DRM RC TYPE returns the type of the value in the resource context buffer.

FETCH INTERFACE MODULE

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

VAX FORMAT

status = **DWT$FETCH_INTERFACE_MODULE** *(hierarchy_id, module_name, parent_widget, widget_return)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
hierarchy_id	identifier	longword	read	value
module_name	char string	char string	read	descriptor
parent_widget	identifier	longword	read	reference
widget_return	identifier	longword	write	reference

MIT C FORMAT

status = **DwtFetchInterfaceModule** *(hierarchy_id, module_name, parent_widget, widget_return)*

argument information

```
int DwtFetchInterfaceModule(hierarchy_id, module_name,
                            parent_widget, widget_return)
      DRMHierarchy  hierarchy_id;
      char          *module_name;
      Widget        parent_widget;
      Widget        *widget_return;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMFailure	The operation failed.
DRMNotFound	The interface module or topmost widget was not found.

ARGUMENTS

hierarchy_id
The identifier of the DRM hierarchy that contains the interface definition.

module_name
The name of interface module defining the top level of the interface; by convention, this is usually the generic name of the application.

parent_widget
The parent widget for the topmost widgets being fetched from the module; usually, the top-level widget.

widget_return
Returns the widget identifier of the main window widget for the application.

DESCRIPTION

FETCH INTERFACE MODULE fetches all the widgets defined in an interface module in the DRM hierarchy. Typically, one or more modules define an application interface; each module must be fetched in order to initialize all the widgets the application requires. Applications do not need to define all their widgets in a single module.

If the module defines a main window widget, FETCH INTERFACE MODULE returns its identifier. If no main window widget is contained in the module, a null value is returned. The identifiers of widgets other than the main window widget can be obtained by using creation callbacks.

FETCH SET VALUES

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

VAX FORMAT

status = **DWT$FETCH_SET_VALUES**
(hierachy_id, widget, args, num_args)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
hierarchy_id	identifier	longword	read	value
widget	uns longword	uns longword	read	reference
args	arglist	uns longword	read	reference
num_args	word	longword	read	reference

MIT C FORMAT

status = **DwtFetchSetValues**
(hierarchy_id, widget, args, num_args)

argument information

```
int DwtFetchSetValues(hierarchy_id, widget, args, num_args)
      DRMHierarchy  hierarchy_id;
      Widget        widget;
      ArgList       args;
      Cardinal      num_args;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMFailure	The operation failed.

ARGUMENTS

hierarchy_id

The identifier of the DRM hierarchy that is searched for literal definitions.

widget

The widget that is modified.

args
An argument list that specifies the widget arguments to be modified. The name part of each argument must be the DwtN... string that identifies the argument tag. The value part must be a string that gives the index of the literal. All literals must be literals accessed by index.

num_args
The number of entries in **args**.

DESCRIPTION

FETCH SET VALUES is a cover routine for the intrinsic routine SET VALUES that fetches the values to be set from literals stored in UID files.

FETCH SET VALUES sets the values on a widget and evaluates the values as public literal resource references resolvable from a DRM hierarchy. Each literal is fetched from the hierarchy, and its value is modified and converted as required. This value is then placed in the argument list and is used as the actual value for a call to the intrinsic routines SET VALUES.

FETCH SET VALUES allows a widget to be modified after creation using UID file values, exactly as is done in FETCH WIDGET. As in FETCH WIDGET, each argument whose value can be evaluated from the UID hierarchy is set in the widget. Values that are not found or values in which conversion errors occur are not modified.

Each entry in the argument list identifies an argument to be modified in the widget. The name part of each argument must be the DwtN... string that identifies the argument tag. The value part must be a string that gives the index of the literal. Consequently, the following code would modify the label resource of the widget to have the value of the literal accessed by index *OK_button_label* in the hierarchy:

```
args[n].name = "label"          (DwtNlabel)
args[n].value = "OK_button_label"
```

FETCH WIDGET

Fetches any indexed widget.

VAX FORMAT

status = **DWT$FETCH_WIDGET**
(hierarchy_id, index, parent_widget, widget_return, class_return)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
hierarchy_id	identifier	longword	read	value
index	char string	char string	read	descriptor
parent_widget	uns longword	uns longword	read	reference
widget_return	identifier	longword	write	reference
class_return	word	word	write	reference

MIT C FORMAT

status = **DwtFetchWidget**
(hierarchy_id, index, parent_widget, widget_return, class_return)

argument information

```
int DwtFetchWidget(hierarchy_id, index, parent_widget,
                   widget_return, class_return)
      DRMHierarchy  hierarchy_id;
      String        index;
      Widget        parent_widget;
      Widget        *widget_return;
      DRMType       *class_return;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMNotFound	The widget was not found in hierarchy.
DRMFailure	The operation failed.

ARGUMENTS

hierarchy_id
The identifier of the DRM hierarchy that contains the interface definition.

index
The name of the widget to fetch.

parent_widget
The identifier of the parent widget.

widget_return
The identifier of the created widget.

class_return
The code identifying the widget class. This argument is used to distinguish the main window from the other XUI Toolkit widgets. This argument must be supplied, but its value is undefined and should not be used by the application.

DESCRIPTION

FETCH WIDGET fetches any indexed widget. The index that identifies the widget must be known to the application. In fetch operations, the fetched widget's subtree is also fetched. This widget must not appear as the child of a widget within its own subtree (in other words, there can be no cycles in the subtree graph). FETCH WIDGET does not execute the intrinsic routine MANAGE CHILD for the newly created widget.

All widgets that FETCH WIDGET fetches must meet the following requirements:

- Must not be referenced as the child of any widget in the database
- Must be indexed

FETCH WIDGET fetches widgets where FETCH INTERFACE MODULE is not used. Any named widget in the DRM hierarchy can be fetched using this routine. FETCH WIDGET can be called at any time to fetch a widget that was not fetched at application startup. FETCH WIDGET can be used to defer fetching pop-up widgets until they are first referenced (presumably in a callback), and then can be used to fetch a widget one time.

FETCH WIDGET can also create multiple instances of a widget (and its subtree). In this case, the UID definition functions as a template; a widget definition can be fetched any number of times. This can be used to make multiple instances of a widget, for example, in a dialog box or menu.

FETCH WIDGET OVERRIDE

Fetches any indexed widget. Overrides FETCH WIDGET arguments.

VAX FORMAT

status = **DWT$FETCH_WIDGET_OVERRIDE**
(hierarchy_id, index, parent_widget, override_name, override_args, override_num_args, widget_return, class_return)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
hierarchy_id	identifier	longword	read	value
index	char string	char string	read	descriptor
parent_widget	uns longword	uns longword	read	reference
override_name	char string	char string	read	descriptor
override_args	arglist	uns longword	read	reference
override_num_args	integer	longword	read	reference
widget_return	identifier	longword	write	reference
class_return	word	word	write	reference

MIT C FORMAT

status = **DwtFetchWidgetOverride**
(hierarchy_id, index, parent_widget, override_name, override_args, override_num_args, widget_return, class_return)

argument information

```
int DwtFetchWidgetOverride(hierarchy_id, index, parent_widget,
                           override_name, override_args,
                           override_num_args,
                           widget_return, class_return)
      DRMHierarchy  hierarchy_id;
      String        index;
      Widget        parent_widget;
      String        override_name;
      ArgList       override_args;
      int           override_num_args;
      Widget        *widget_return;
      DRMType       *class_return;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMNotFound	The widget was not found in hierarchy.
DRMFailure	The operation failed.

ARGUMENTS

hierarchy_id

The identifier of the hierarchy that contains the interface definition.

index

The name of the widget to fetch.

parent_widget

The identifier of the parent widget.

override_name

The name to override the widget name. Use a null value if you do not want to override the widget name.

override_args

The override argument list that is the same as the override argument list for the intrinsic routine CREATE WIDGET. Use a null value if you do not want to specify the override argument list.

The override argument list is appended to the existing argument list; the resulting argument list is passed to the widget creation routine.

override_num_args

The number of arguments in **override_args**. Use zero if you do not want to override the argument list.

widget_return

The identifier of the created widget.

class_return

The code identifying the widget class. This argument is used to distinguish the main window from the other XUI Toolkit widgets. This argument must be supplied, but its value is undefined and should not be used by the application.

DESCRIPTION

FETCH WIDGET OVERRIDE is identical to FETCH WIDGET in all respects, except that it allows the caller to override the widget's name and any arguments that FETCH WIDGET would otherwise retrieve from the DRM hierarchy or one of the default mechanisms. The override argument list is not limited to those arguments in the DRM hierarchy.

The override arguments apply only to the widget fetched and returned by this routine; its children do not receive any override arguments.

INITIALIZE DRM

Prepares an application to use the DRM facilities.

VAX FORMAT

void = **DWT$INITIALIZE_DRM** *()*

MIT C FORMAT

void = **DwtInitializeDRM** *()*

argument information

```
void DwtInitializeDRM ()
```

DESCRIPTION

INITIALIZE DRM prepares an application to use the DRM facilities. INITIALIZE DRM must be called before the following operations:

- Widget creation, whether by the DRM or directly by the application
- Calls to the intrinsic routine INITIALIZE or REGISTER CLASS

INITIALIZE can be called more than once; however, all calls after the first have no effect.

The toolkit class records that INITIALIZE DRM uses to initialize its facilities must be uninitialized. INITIALIZE DRM must be called before any calls to the intrinsic routines INITIALIZE, TOOLKIT INITIALIZE, DISPLAY INITIALIZE, and APPLICATION CREATE SHELL.

OPEN HIERARCHY

Opens all the UID files in the DRM hierarchy.

VAX FORMAT

status = **DWT$OPEN_HIERARCHY**
(num_files, file_names_list, ancillary_structures_list, hierarchy_id_return)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
num_files	word	longword	read	reference
file_names_list	array	uns longword	read	reference
ancillary_structures_list	record	uns longword	read	reference
hierarchy_id_return	identifier	longword	write	reference

MIT C FORMAT

status = **DwtOpenHierarchy**
(num_files, file_names_list, ancillary_structures_list, hierarchy_id_return)

argument information

```
int DwtOpenHierarchy(num_files, file_names_list,
                     ancillary_structures_list,
                     hierarchy_id_return)
      DRMCount            num_files;
      String              *file_names_list;
      IDBOSOpenParamPtr   *ancillary_structures_list;
      DRMHierarchy        *hierarchy_id_return;
```

RETURNS

status
The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMNotFound	One or more of the specified files were not found.
DRMFailure	The operation failed.

ARGUMENTS

num_files
The number of file names in file names list.

file_names_list
An array of pointers to the character strings.

ancillary_structures_list
A list of data structures that can be used to specify system-dependent information about the files in **file_names_list**. This argument is optional; a null value should be used if it is to be omitted.

If this argument is specified, each data structure corresponds to a file name in the file names list. In the VMS operating system, you can use this argument to specify an RMS default file name or related file name to be applied to each file name in the file names list. See the DwtAppl.h file for the definition of this data structure (IDBOSOpenParam).

hierarchy_id_return
The identifier of the DRM hierarchy.

DESCRIPTION

OPEN HIERARCHY opens all the UID files in the DRM hierarchy. It initializes the internal data structures that the DRM uses to manage the hierarchy. All files named in **file_names_list** are closed if any errors occur.

REGISTER CLASS

Provides the DRM with the information about a widget class defined by the application.

VAX FORMAT

status = **DWT$REGISTER_CLASS**
(class_code, class_name, create_name, create_proc, class_record)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
class_code	longword	longword	read	reference
class_name	char string	char string	read	descriptor
create_name	char string	char string	read	descriptor
create_proc	void_proc	procedure entry mask	read	value
class_record	record	longword	read	reference

MIT C FORMAT

status = **DwtRegisterClass**
(class_code, class_name, create_name, create_proc, class_record)

argument information

```
int DwtRegisterClass(class_code, class_name, create_name,
                     create_proc, class_record)
      DRMType      class_code;
      String       class_name;
      String       create_name;
      Widget       (* create_proc) ();
      WidgetClass  class_record;
```

RETURNS

status
The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMFailure	The operation failed.

ARGUMENTS

class_code
Defined as DRMwcUnknown for all application-defined widgets.

class_name
The case-sensitive name of the class.

create_name
The case-sensitive name of the low-level widget creation routine as it appears in the UIL module that defines a widget of this widget class.

For application-defined widgets, **create_name** is the name of the creation procedure in the UIL module that defines this widget.

create_proc
The address of the low-level widget creation routine for this widget class.

class_record
The address of the class record for this widget class.

DESCRIPTION

REGISTER CLASS is called when the application uses an application-defined widget class. REGISTER CLASS provides the DRM with the information about an application-defined widget class, such as the class record and class name, that is necessary to create widgets of this class. This information is used by the DRM when a user-defined widget, as defined in a UIL module, is fetched from a DRM hierarchy.

The class and superclass records used by REGISTER CLASS must be uninitialized. REGISTER CLASS must be called before any calls to the intrinsic routines INITIALIZE, TOOLKIT INITIALIZE, DISPLAY INITIALIZE, and APPLICATION CREATE SHELL, and before any widgets in the class are created.

REGISTER DRM NAMES

Registers a vector of names of identifiers or callback routines for access in DRM.

VAX FORMAT

status = **DWT$REGISTER_DRM_NAMES** *(register_list, register_count)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	integer	longword	write	value
register_list	array	uns longword	read	reference
register_count	word	word	read	reference

MIT C FORMAT

status = **DwtRegisterDRMNames** *(register_list, register_count)*

argument information

```
int DwtRegisterDRMNames (register_list, register_count)
      DRMRegisterArglist  register_list;
      DRMCount            register_count;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DRMSuccess	The operation succeeded.
DRMFailure	The operation failed.

ARGUMENTS

register_list

A list of name/value pairs for the names to be registered. Each name is a case-sensitive, null-terminated ASCII string. Each value is a 32-bit quantity, interpreted as a procedure address if the name is a callback routine, and uninterpreted otherwise.

The data structure used to pass the name/value pairs can be found in DwtAppl.h.

register_count

The number of entries in **register_list**.

DESCRIPTION

REGISTER DRM NAMES registers a vector of names and associated values for access in the DRM. The values can be callback routines, pointers to user-defined data, or any other values. The information provided is used to associate symbolic names occurring in UID files to their run-time values. For callbacks, this information provides the procedure address required by the XUI Toolkit to perform callbacks. For names used as identifiers in UIL, this information provides any run-time mapping the application needs.

The names in the list are case sensitive.

Callback routines registered through this REGISTER DRM NAMES may be either regular or creation callbacks. Regular callbacks have declarations determined by the XUI Toolkit. Creation callbacks have the same format as any other callback:

```
void CallBackProc (widget_id, tag, callback_data)
    Widget                  *widget_id;
    Opaque                  tag;
    DwtAnyCallbackStruct    *callback_data;
```

widget_id	The widget identifier (as in any callback routine).
tag	The specified tag value (as in any callback routine).
callback_data	Always DwtCRCreate.

As in any other callback, the widget name and parent are available from the widget record by way of **widget_id**.

REGISTER DRM NAMES can be called at any time, either before or after calls to INITIALIZE DRM or REGISTER CLASS. The same name can be registered more than once if the associated value is changed each time, much like changing the value of a variable.

5 Compound String Routines

The XUI Toolkit provides a set of compound string routines that enables the creation and manipulation of compound strings and font lists.

A **compound string** is a sequence of segments. Each segment consists of a natural language identifier, a text direction identifier, rendition information, a character set identifier, and a counted text string.

A **font list** is an array of font structures indexed by the character set identifier. A **character set identifier** is a constant from the file CDA$DEF.*type*, where *type* is a language's include file type. For example, H is the include file type for the C programming language.

For concepts related to compound string routines and information about how to use them, see the *VMS DECwindows Guide to Application Programming*.

Table 5–1 lists the supported compound string routines.

Table 5–1 Compound String Routines

Routine Name	Description
ADD FONT LIST	Adds an entry to a font list.
CREATE FONT LIST	Creates a new font list.
CS BYTE CMP	Compares two compound strings to determine if they are identical.
CS CAT	Appends a copy of a compound string to the end of another compound string.
CS COPY	Copies a compound string.
CS EMPTY	Determines if the compound string contains any text segments.
CS LEN	Returns the number of bytes in a compound string.
CS STRING	Creates a compound string.
GET NEXT SEGMENT	Gets information about the next segment of the compound string.
INIT GET SEGMENT	Initializes the context needed by GET NEXT SEGMENT.
LATIN1 STRING	Creates a compound string for the LATIN1 character set.
STRING	Creates a compound string by using a simpler interface than CS STRING.

5.1 Compound String Routines

The following pages describe the XUI Toolkit compound string routines.

ADD FONT LIST

Adds an entry to a font list.

VAX FORMAT

font_list = **DWT$ADD_FONT_LIST**
(list, font, charset)

argument information

Argument	Usage	Data Type	Access	Mechanism
font_list	font list	uns longword	write	value
list	font list	uns longword	read	value
font	font	uns longword	read	reference
charset	longword	uns longword	read	reference

MIT C FORMAT

font_list = **DwtAddFontList**
(list, font, charset)

argument information

```
DwtFontList DwtAddFontList(list,font,charset)
      DwtFontList    list;
      XFontStruct    *font;
      unsigned long  charset;
```

RETURNS

font_list
The new font list.

ARGUMENTS

list
A pointer to the font list to which an entry will be added.

font
A pointer to the font structure to be added to the list.

charset
The character set identifier for the font.

DESCRIPTION

ADD FONT LIST adds an entry to a font list and allocates space for a new font list. The space for the resulting font list is allocated within the routine; the space should be freed with the intrinsic routine FREE after use.

CREATE FONT LIST

Creates a new font list.

VAX FORMAT

font_list = **DWT$CREATE_FONT_LIST**
(font, charset)

argument information

Argument	Usage	Data Type	Access	Mechanism
font_list	font list	uns longword	write	value
font	font	uns longword	read	reference
charset	uns longword	uns longword	read	reference

MIT C FORMAT

font_list = **DwtCreateFontList**
(font, charset)

argument information

```
DwtFontList DwtCreateFontList(font,charset)
      XFontStruct    *font;
      unsigned long  charset;
```

RETURNS

font_list
The new font list. Returns a null value if the font specified in **font** is null.

ARGUMENTS

font
A pointer to a font structure for which a new font list is generated.

charset
The character set identifier for the font.

DESCRIPTION

CREATE FONT LIST creates a new font list for the font and character set. CREATE FONT LIST allocates the space for the font list; the space allocated should be freed with the intrinsic routine FREE after use. The end of font list is marked by an element whose character set value is –1.

CS BYTE CMP

Compares two compound strings to determine if they are identical.

VAX FORMAT

status = **DWT$CSBYTECMP** *(compound_string1, compound_string2)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
compound_string1	comp string	uns longword	read	reference
compound_string2	comp string	uns longword	read	reference

MIT C FORMAT

status = **DwtCSbytecmp** *(compound_string1, compound_string2)*

argument information

```
int DwtCSbytecmp(compound_string1, compound_string2)
      DwtCompString  compound_string1, compound_string2;
```

RETURNS

status

The status return value.

The status return value is zero if **compound_string1** and **compound_string2** are exactly the same, or one if they are not the same.

ARGUMENTS

compound_string1

The compound string to be compared with **compound_string2**.

compound_string2

The compound string to be compared with **compound_string1**.

DESCRIPTION

CS BYTE CMP compares two compound strings to determine if they are identical. The value returned is zero if **compound_string1** and **compound_string2** are exactly the same (byte to byte), or one if they are not the same.

CS CAT

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

VAX FORMAT

status = **DWT$CSTRCAT**
(compound_string1, compound_string2, compound_string_result)

status = **DWT$CSTRNCAT**
(compound_string1, compound_string2, compound_string_result, num_chars)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
compound_string1	comp string	uns longword	read	reference
compound_string2	comp string	uns longword	read	reference
compound_string_ result	pointer to comp string	uns longword	write	reference
num_chars	uns longword	uns longword	read	reference

MIT C FORMAT

comp_string = **DwtCStrcat**
(compound_string1, compound_string2)

comp_string = **DwtCStrncat**
(compound_string1, compound_string2, num_chars)

argument information

```
DwtCompString DwtCStrcat(compound_string1, compound_string2)
DwtCompString DwtCStrncat(compound_string1, compound_string2,
                          num_chars)
      DwtCompString  compound_string1, compound_string2;
      int            num_chars;
```

RETURNS

status (VAX only)
The status return value.

DwtSuccess is returned for normal completion.

comp_string (C only)
A pointer to the resulting compound string.

ARGUMENTS

compound_string1
The string to which a copy of **compound_string2** is appended.

compound_string2
The string appended to the end of **compound_string1**.

compound_string_result (VAX only)
The resulting compound string.

num_chars
The maximum number of characters to append.

DESCRIPTION

CS TRCAT appends a copy of a compound string to the end of another compound string. CS TRCAT appends **compound_string2** to the end of **compound_string1** and returns the resulting string. The original strings are preserved. The space for the resulting string is allocated within the routine; the space should be freed with the intrinsic routine FREE after use.

CS TRNCAT appends no more than the number of characters specified in **num_chars**, including the headers and trailers of the compound string.

CS COPY

Copies a compound string.

VAX FORMAT

status = **DWT$CSTRCPY**
(compound_string1, compound_string_result)

status = **DWT$CSTRNCPY**
(compound_string1, compound_string_result, num_chars)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
compound_string1	comp string	uns longword	read	reference
compound_string_ result	pointer to comp string	uns longword	write	reference
num_chars	uns longword	uns longword	read	reference

MIT C FORMAT

comp_string = **DwtCStrcpy**
(compound_string1)

comp_string = **DwtCStrncpy**
(compound_string1, num_chars)

argument information

```
DwtCompString DwtCStrcpy(compound_string1)
DwtCompString DwtCStrncpy(compound_string1, num_chars)
      DwtCompString  compound_string1;
      int            num_chars;
```

RETURNS

status (VAX only)
The status return value.

DwtSuccess is returned for normal completion.

comp_string (C only)
A pointer to the resulting compound string.

ARGUMENTS

compound_string1
A compound string to be copied.

compound_string_result (VAX only)
A pointer to the resulting compound string.

num_chars
The number of characters to be copied.

If the maximum number of characters is fewer than the number of characters in **compound_string2**, the resulting string is not a valid compound string.

DESCRIPTION

CS TRCOPY copies a compound string and returns a pointer to the copy. CS TRNCOPY copies exactly the number of characters in **num_chars**, including the headers and trailers.

The space for the resulting string is allocated within the routine; the space should be freed with the intrinsic routine FREE after use.

CS EMPTY

Determines if the compound string contains any text segments.

VAX FORMAT

status = **DWT$CSEMPTY** *(compound_string)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
compound_string	comp string	uns longword	read	reference

MIT C FORMAT

status = **DwtCSempty** *(compound_string)*

argument information

```
int DwtCSempty(compound_string)
      DwtCompString  compound_string;
```

RETURNS

status
The status return value.

The value is 1 if all segments in **compound_string** have a zero text size; otherwise, the value is zero.

ARGUMENTS

compound_string
The compound string.

DESCRIPTION

CS EMPTY determines if the compound string contains any text segments that are not empty.

CS LEN

Returns the number of bytes in a compound string.

VAX FORMAT

length = **DWT$CSTRLEN** *(compound_string)*

argument information

Argument	Usage	Data Type	Access	Mechanism
length	pointer to comp string	uns longword	write	reference
compound_string	comp string	uns longword	read	reference

MIT C FORMAT

length = **DwtCStrlen** *(compound_string)*

argument information

```
int DwtCStrlen(compound_string)
      DwtCompString  compound_string;
```

RETURNS

length
The number of bytes in **compound_string**, including compound string headers and trailers. If the compound string has an invalid structure, zero is returned.

ARGUMENTS

compound_string
The compound string whose length is being determined.

DESCRIPTION

CS LEN returns the number of bytes in **compound_string**, including compound string headers and trailers.

CS STRING

Creates a compound string.

VAX FORMAT

status = **DWT$CS_STRING**
(text, charset, dir_r_to_l, language, rend, compound_string)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
text	char string	char string	read	descriptor
charset	uns longword	uns longword	read	reference
dir_r_to_l	uns long	uns longword	read	reference
language	uns longword	uns longword	read	reference
rend	uns longword	uns longword	read	reference
compound_string	pointer to comp string	uns longword	write	reference

MIT C FORMAT

comp_string = **DwtCSString**
(text, charset, dir_r_to_l, language, rend)

argument information

```
DwtCompString DwtCSString(text, charset, dir_r_to_l,
                          language, rend)
      char           *text;
      unsigned long  charset;
      int            dir_r_to_l;
      unsigned long  language;
      DwtRendMask    rend;
```

RETURNS

status (VAX only)

The status return value.

DwtSuccess is returned for normal completion.

comp_string (C only)

The resulting compound string. A null pointer is returned if the input string is null.

ARGUMENTS

text
The text string to be converted to a compound string.

charset
The character set identifier. Values for this argument can be found in the require file CDA$DEF with a file extension of the appropriate programming language.

dir_r_to_l
The direction in which the compound string is specified. Has a value of 1 if the text is drawn right to left, or a value of zero if the text is drawn left to right.

language
Reserved by DIGITAL.

rend
Reserved by DIGITAL.

compound_string (VAX only)
The resulting compound string.

DESCRIPTION

CS STRING creates a compound string from information in the argument list.

Space for the resulting string is allocated within the routine; the space should be freed with the intrinsic routine FREE after use.

GET NEXT SEGMENT

Gets information about the next segment in the compound string.

VAX FORMAT

status = **DWT$GET_NEXT_SEGMENT**
(context, text, text_len, charset, dir_r_to_l, lang, rend)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
context	uns longword	uns longword	modify	reference
text	char string	char string	write	descriptor
text_len	uns longword	uns longword	write	reference
charset	uns longword	uns longword	write	reference
dir_r_to_l	uns long	uns longword	write	reference
lang	uns longword	uns longword	write	reference
rend	uns longword	uns longword	write	reference

MIT C FORMAT

status = **DwtGetNextSegment**
(context, text, charset, dir_r_to_l, lang, rend)

argument information

```
int DwtGetNextSegment(context, text, charset,
                      dir_r_to_l, lang, rend)
      DwtCompStringContext  *context;
      char                  **text;
      unsigned long         *charset;
      int                   *dir_r_to_l;
      unsigned long         *lang;
      DwtRendMask           *rend;
```

RETURNS

status
The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DwtEndCS	The context is at the end of the compound string.
DwtFail	The context is not valid.

Return Value Name	Description
DwtSuccess	Normal completion.
DwtTruncate (VAX only)	The text string was truncated to fit in the buffer described by the static descriptor.

ARGUMENTS

context
The context for the call to GET NEXT SEGMENT. Context is initialized by the INIT NEXT SEGMENT call and is updated every time GET NEXT SEGMENT is called.

text
The text in the next segment.

text_len (VAX only)
The length of the text in the next segment.

charset
The character set in the next segment.

dir_r_to_l
The direction for the next segment. Has a value of 1 if the text is drawn right to left, or a value of zero if the text is drawn left to right.

lang
Reserved by DIGITAL.

rend
Reserved by DIGITAL.

DESCRIPTION

GET NEXT SEGMENT gets information about the next segment of the compound string as determined by the context.

For the C format, the space for the returned text string is allocated within the routine. The space should be freed with the intrinsic routine FREE after use.

INIT GET SEGMENT

Initializes the context needed by GET NEXT SEGMENT.

VAX FORMAT

status = **DWT$INIT_GET_SEGMENT**
(context, compound_string)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
context	uns longword	uns longword	write	reference
compound_string	comp string	uns longword	read	reference

MIT C FORMAT

status = **DwtInitGetSegment**
(context, compound_string)

argument information

```
int DwtInitGetSegment(context, compound_string)
      DwtCompStringContext  *context;
      DwtCompString         compound_string;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
DwtSuccess	Normal completion.
DwtEndCS	The string specified is null.
DwtFail	The string specified is not a compound string.

ARGUMENTS

context

The context to be filled by the routine. It should have been previously allocated by the user.

compound_string

The compound string.

DESCRIPTION

INIT GET SEGMENT returns the initialized DwtCompStringContext, **context**, of the compound string, **compound_string**. The returned context is needed for calling GET NEXT SEGMENT.

LATIN1 STRING

Creates a compound string for the LATIN1 character set.

VAX FORMAT

status = **DWT$LATIN1_STRING**
(text, compound_string)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
text	char string	char string	read	descriptor
compound_string	pointer to comp string	uns longword	write	reference

MIT C FORMAT

comp_string = **DwtLatin1String**
(text)

argument information

```
DwtCompString DwtLatin1String(text)
      char  *text;
```

RETURNS

status (VAX only)

The status return value.

DwtSuccess is returned on normal completion.

comp_string (C only)

The resulting compound string. It has the following default values:

character set = CDA$K_ISO_LATIN1
dirRtoL = Zero (LtoR)
Language = DwtLanguageNotSpecified
rendition = DwtRendMaskNone

ARGUMENTS

text

The text string to be converted to a compound string.

compound_string(VAX only)

The resulting compound string. It has the following default values:

character set = CDA$K_ISO_LATIN1
dirRtoL = Zero (LtoR)
Language = DwtLanguageNotSpecified
rendition = DwtRendMaskNone

DESCRIPTION

LATIN1 STRING creates a compound string for the LATIN1 character set. LATIN1 STRING is provided for application programmers who do not need to mix compound strings containing different character sets and directions. LATIN1 STRING assumes the character encoding of the text to be ISO_LATIN1 and the writing direction to be from left to right.

The space for the resulting compound string is allocated within the routine; the space should be freed with the intrinsic routine FREE after use.

STRING

Creates a compound string.

VAX FORMAT

status = **DWT$STRING**
(text, charset, dir_r_to_l, compound_string)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
text	char string	char string	read	descriptor
charset	uns longword	uns longword	read	reference
dir_r_to_l	uns long	uns longword	read	reference
compound_string	pointer to comp string	uns longword	write	reference

MIT C FORMAT

comp_string = **DwtString**
(text, charset, dir_r_to_l)

argument information

```
DwtCompString DwtString(text, charset, dir_r_to_l)
      char           *text;
      unsigned long  charset;
      int            dir_r_to_l;
```

RETURNS

status (VAX only)
The status return value.

DwtSuccess is returned on normal completion.

comp_string (C only)
The resulting compound string. A null pointer is returned if the text is null.

ARGUMENTS

text
The text to be converted to a compound string.

charset
The character set identifier. Values for this argument can be found in the required file CDA$DEF with a file type of the appropriate programming language.

dir_r_to_l

The direction of the compound string. Has a value of 1 if the text is drawn right to left, or a value of zero if the text is drawn left to right.

compound_string (VAX only)

The resulting compound string.

DESCRIPTION

STRING creates a compound string. It has a simpler interface than CS STRING.

STRING assumes the following default values:

- Language = DwtLanguageNotSpecified
- rendition = DwtRendMaskNone

The space for the resulting compound string is allocated within the routine; the space should be freed with the intrinsic routine FREE after use.

6 Cut and Paste Routines

An application can interface to the clipboard through calls to the cut and paste routines. The following menu items represent features normally provided for user access to the clipboard:

Cut	When the user chooses this item, the application calls BEGIN COPY TO CLIPBOARD, COPY TO CLIPBOARD, and END COPY TO CLIPBOARD to copy the data in whatever formats it desires. The application then deletes the cut data from the user's display. The application should save the deleted data if it is providing an UNDO CUT function.
Copy	The same as Cut except the data is not deleted.
Paste	When the user chooses this item, the application calls COPY FROM CLIPBOARD to get the data in some format. The application then allows the user to place the data on the display.

Following is a list of actions that your application can provide to the user:

Undo Cut	The application redraws the deleted data. The application then calls UNDO COPY TO CLIPBOARD to delete the data from the clipboard. The application should not use the clipboard contents for redrawing the deleted data, as the user may have already changed the contents of the clipboard .
Redo Cut	The application performs the same actions as the Cut menu item.
Undo Copy	The application calls UNDO COPY TO CLIPBOARD.
Redo Copy	The application performs the same actions as the Copy menu item.
Undo Paste	The application deletes the pasted data and saves the pasted data for a possible later Redo Paste operation.
Redo Paste	The application pastes the data it saved during the Undo Paste operation.

The clipboard is not involved with the Undo Paste or Redo Paste operations. During the interval between the Paste and the Undo Paste operations or between the Undo Paste and the Redo Paste operations, the clipboard might be changed either directly or indirectly by the user.

For concepts related to cut and paste routines and information about how to use them, see the *VMS DECwindows Guide to Application Programming*.

Table 6–1 lists the supported cut and paste routines.

Table 6–1 Cut and Paste Routines

Routine Name	Description
BEGIN COPY TO CLIPBOARD	Sets up storage and data structures to receive clipboard data.
CANCEL COPY FORMAT	Indicates that the application will no longer supply a data item to the clipboard that the application had previously passed by name.
CANCEL COPY TO CLIPBOARD	Cancels the copy to clipboard that is in progress.
CLIPBOARD LOCK	Locks the clipboard from access by other applications.
CLIPBOARD UNLOCK	Unlocks the clipboard, enabling it to be accessed by other applications.
COPY FROM CLIPBOARD	Retrieves a data item from the clipboard.
COPY TO CLIPBOARD	Copies a data item to the clipboard.
END COPY TO CLIPBOARD	Places data in the clipboard data structure.
INQUIRE NEXT PASTE COUNT	Returns the number of data item formats available for the next-paste item in the clipboard.
INQUIRE NEXT PASTE FORMAT	Returns a specified format name for the next-paste item in the clipboard.
INQUIRE NEXT PASTE LENGTH	Returns the length of the data stored under a specified format name for the next-paste item in the clipboard.
LIST PENDING ITEMS	Returns a list of pending items as data ID/private ID pairs for a specified format name.
RECOPY TO CLIPBOARD	Copies a data item that had been passed by name to the clipboard.
UNDO COPY TO CLIPBOARD	Deletes the last item placed on the clipboard.

6.1 Passing Data by Name

Copying a large piece of data to the clipboard can take time. It is possible that, once copied, no application will ever request that data. An application does not need to actually pass data to the clipboard until the data has been requested by some application. Instead, the application passes format and length information to the clipboard routines, along with a widget identifier and a callback routine address. The widget identifier is needed for communications between the clipboard routines in the application that owns the data and the clipboard routines in the application that requests the data. Your callback routine is responsible for copying the actual data to the clipboard (with the RECOPY TO CLIPBOARD routine call). The callback routine is also called if the data item is removed from the clipboard, and the actual data is therefore no longer needed.

Refer to the BEGIN COPY TO CLIPBOARD, COPY TO CLIPBOARD, and RECOPY TO CLIPBOARD routines for more information on passing data by name.

6.2 Cut and Paste Routines

The following pages describe the XUI Toolkit cut and paste routines.

BEGIN COPY TO CLIPBOARD

Sets up storage and data structures to receive clipboard data.

VAX FORMAT

status = **DWT$BEGIN_COPY_TO_CLIPBOARD**
(display, window, clip_label, widget, callback, item_id)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
clip_label	comp string	uns longword	read	reference
widget	identifier	uns longword	read	reference
callback	uns longword	uns longword	read	reference
item_id	uns longword	uns longword	write	reference

MIT C FORMAT

status = **DwtBeginCopyToClipboard**
(display, window, clip_label, widget, callback, item_id)

argument information

```
int DwtBeginCopyToClipboard (display, window, clip_label,
                             widget, callback, item_id)
      Display         *display;
      Window          window;
      DwtCompString   clip_label;
      Widget          widget;
      VoidProc        callback;
      unsigned long   *item_id;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display

A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual*.

window

The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

clip_label

The label to be associated with the data item. This argument is used to identify the data item, for example, in a clipboard viewer. An example of a label would be the name of the application that places the data in the clipboard.

widget

The identifier of the widget that will receive messages requesting data previously passed by name. This argument must be present in order to pass data by name.

Any valid widget identifier in your application can be used for this purpose and all the message handling is taken care of by the cut and paste routines.

callback

The address of the callback routine that is called when the clipboard needs data that was originally passed by name. This is also the callback to receive the DELETE message for items that were originally passed by name. This argument must be present in order to pass data by name.

The callback format is as follows:

```
routine (widget, data_id, private_id, reason)
      Widget  *widget;
      int     *data_id;
      int     *private_id;
      int     *reason;
```

The arguments in the callback format are defined as follows:

widget	The widget passed to BEGIN COPY TO CLIPBOARD.
data_id	The identifying number returned by COPY TO CLIPBOARD, which identifies the pass-by-name data.
private_id	The private information passed to COPY TO CLIPBOARD.
reason	Either DwtCRClipboardDataDelete or DwtCRClipboardDataRequest.

item_id

The number assigned to this data item. This number is used by the application in calls to COPY TO CLIPBOARD, END COPY TO CLIPBOARD, and CANCEL COPY TO CLIPBOARD.

DESCRIPTION

BEGIN COPY TO CLIPBOARD sets up storage and data structures to receive clipboard data. An application calls BEGIN COPY TO CLIPBOARD during a cut or copy operation. The data item that these structures receive through calls to COPY TO CLIPBOARD then becomes the next item to be pasted (the **next-paste** item) in the clipboard after the call to END COPY TO CLIPBOARD.

The **widget_id** and **callback** arguments must be present in order to pass data by name.

CANCEL COPY FORMAT

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

VAX FORMAT

status = **DWT$CANCEL_COPY_FORMAT** *(display, window, data_id)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
data_id	uns longword	uns longword	read	reference

MIT C FORMAT

status = **DwtCancelCopyFormat** *(display, window, data_id)*

argument information

```
int DwtCancelCopyFormat (display, window, data_id)
      Display        *display;
      Window         window;
      unsigned long  data_id;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display

A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window

The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

data_id

The number that uniquely identifies the data item and format. This was assigned to the item when it was originally passed by COPY TO CLIPBOARD.

DESCRIPTION

CANCEL COPY FORMAT indicates that the application will no longer supply a data item to the clipboard that the application had previously passed by name.

For related information, see the routine LIST PENDING ITEMS.

CANCEL COPY TO CLIPBOARD

Cancels the copy to clipboard that is in progress.

VAX FORMAT

void = **DWT$CANCEL_COPY_TO_CLIPBOARD** *(display, window, item_id)*

argument information

Argument	Usage	Data Type	Access	Mechanism
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
item_id	uns longword	uns longword	read	reference

MIT C FORMAT

void = **DwtCancelCopyToClipboard** *(display, window, item_id)*

argument information

```
void DwtCancelCopyToClipboard(display, window, item_id)
      Display         *display;
      Window          window;
      unsigned long   item_id;
```

ARGUMENTS

display

A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window

The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

item_id

The number assigned to this data item. This number was returned by the previous call to BEGIN COPY TO CLIPBOARD.

DESCRIPTION

CANCEL COPY TO CLIPBOARD cancels the copy to clipboard that is in progress. This routine also frees temporary storage.

CANCEL COPY TO CLIPBOARD

If CANCEL COPY TO CLIPBOARD is called, then END COPY FROM CLIPBOARD does not have to be called. A call to CANCEL COPY TO CLIPBOARD is valid only after a call to BEGIN COPY TO CLIPBOARD and before a call to END COPY TO CLIPBOARD.

CLIPBOARD LOCK

Locks the clipboard from access by other applications.

VAX FORMAT

status = **DWT$CLIPBOARD_LOCK**
(display, window)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference

MIT C FORMAT

status = **DwtClipboardLock**
(display, window)

argument information

```
int DwtClipboardLock(display, window)
      Display  *display;
      Window   window;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display

A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual*.

window

The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

DESCRIPTION

CLIPBOARD LOCK locks the clipboard from access by another application until CLIPBOARD UNLOCK is called. All clipboard routines lock and unlock the clipboard to prevent simultaneous access.

If the clipboard is already locked by another application, CLIPBOARD LOCK returns an error status.

The CLIPBOARD LOCK and CLIPBOARD UNLOCK routines allow the application to keep the clipboard data from changing between calls to inquire routines and other clipboard routines. The application does *not* need to lock the clipboard between calls to BEGIN COPY TO CLIPBOARD and END COPY TO CLIPBOARD.

Multiple calls to CLIPBOARD LOCK by the same application increase the lock level. See the Description section of CLIPBOARD UNLOCK.

CLIPBOARD UNLOCK

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

VAX FORMAT

status = **DWT$CLIPBOARD_UNLOCK**
(display, window, remove_all_locks)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
remove_all_locks	Boolean	uns longword	read	reference

MIT C FORMAT

status = **DwtClipboardUnlock**
(display, window, remove_all_locks)

argument information

```
int DwtClipboardUnlock(display, window, remove_all_locks)
      Display  *display;
      Window   window;
      Boolean  remove_all_locks;
```

RETURNS

status
The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display
A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window

The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

remove_all_locks

A value of true indicates that all nested locks should be removed. A value of False indicates that only one level of lock should be removed.

DESCRIPTION

CLIPBOARD UNLOCK unlocks the clipboard, enabling it to be accessed by other applications.

If multiple calls to CLIPBOARD LOCK have occurred, then the same number of calls to CLIPBOARD UNLOCK is necessary to unlock the clipboard, unless the **remove_all_locks** argument is true.

COPY FROM CLIPBOARD

Retrieves a data item from the clipboard.

VAX FORMAT

status = **DWT$COPY_FROM_CLIPBOARD**
(display, window, format_name, buffer, num_bytes, private_id)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
format_name	char string	char string	read	descriptor
buffer	char string	char string	write	descriptor
num_bytes	uns longword	uns longword	write	reference
private_id	longword	longword	write	reference

MIT C FORMAT

status = **DwtCopyFromClipboard**
(display, window, format_name, buffer, length, num_bytes, private_id)

argument information

```
int DwtCopyFromClipboard(display, window, format_name, buffer,
                         length, num_bytes, private_id)
      Display         *display;
      Window          window;
      char            *format_name;
      char            *buffer;
      unsigned long   length
      unsigned long   *num_bytes;
      int             *private_id;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.
ClipboardTruncate	The data returned is truncated because the user did not provide a buffer that was large enough to hold the data.
ClipboardNoData	The routine could not find data on the clipboard corresponding to the format requested. The following conditions could cause this result: • The clipboard is empty. • There is data on the clipboard but not in the requested format. • The data in the requested format was passed by name and is no longer available.

ARGUMENTS

display

A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window

The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

format_name

The name of a format in which the data is stored on the clipboard.

buffer

The buffer to which the application wants the clipboard to copy the data.

length (C only)

The length of the application buffer.

num_bytes

The number of bytes of data copied into the application buffer.

private_id

The private data stored with the data item by the application that placed the data item on the clipboard. If the application did not store private data with the data item, then this argument returns zero.

DESCRIPTION

COPY FROM CLIPBOARD retrieves the current next-paste item from clipboard storage.

COPY FROM CLIPBOARD returns a warning under the following circumstances:

- The data was truncated because the buffer length is too short.
- The clipboard is locked.
- There is no data on the clipboard in the requested format.

COPY TO CLIPBOARD

Copies a data item to the clipboard.

VAX FORMAT

status = **DWT$COPY_TO_CLIPBOARD**
(display, window, item_id, format_name, buffer, length, private_id, data_id)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
item_id	uns longword	uns longword	read	reference
format_name	char string	char string	read	descriptor
buffer	char string	char string	read	descriptor
length	uns longword	uns longword	read	reference
private_id	longword	longword	read	reference
data_id	uns longword	uns longword	write	reference

MIT C FORMAT

status = **DwtCopyToClipboard**
(display, window, item_id, format_name, buffer, length, private_id, data_id)

argument information

```
int DwtCopyToClipboard(display, window, item_id, format_name,
                       buffer, length, private_id, data_id)
      Display        *display;
      Window         window;
      long           item_id;
      char           *format_name;
      char           *buffer;
      unsigned long  length;
      int            private_id;
      unsigned long  *data_id;
```

RETURNS

status
The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display
A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window
The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

item_id
The number assigned to this data item. This number was returned by the previous call to BEGIN COPY TO CLIPBOARD.

format_name
The name of the format in which the data item is stored.

buffer
The buffer from which the clipboard copies the data.

length
The length of the data being copied to the clipboard.

VAX only

The **length** argument is required only for data that is passed by name.

private_id
The private data stored with the data item by the application that placed the data item on the clipboard.

data_id
An identifying number assigned to the data item that uniquely identifies the data item and the format. This argument is required only for data that is passed by name.

DESCRIPTION

COPY TO CLIPBOARD copies a data item to clipboard storage. The data item is not actually entered in the clipboard data structure until the call to END COPY TO CLIPBOARD. Additional calls to COPY TO CLIPBOARD before a call to END COPY TO CLIPBOARD add data item formats to the same data item or append data to an existing format.

If the **buffer** parameter is null, the data is considered passed by name. If data passed by name is needed later by another application, the application that owns the data receives a callback with a request for the data. The application that owns the data must then transfer the data to the clipboard with the RECOPY TO CLIPBOARD routine. When a data item that was passed by name is deleted from the clipboard, the application that owns the data receives a callback that states that the data is no longer needed.

For information on the callback routine, see the callback argument description in BEGIN COPY TO CLIPBOARD.

END COPY TO CLIPBOARD

Places data in the clipboard data structure.

VAX FORMAT

status = **DWT$END_COPY_TO_CLIPBOARD**
(display, window, item_id)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
item_id	uns longword	uns longword	read	reference

MIT C FORMAT

status = **DwtEndCopyToClipboard**
(display, window, item_id)

argument information

```
int DwtEndCopyToClipboard(display, window, item_id)
      Display   *display;
      Window    window;
      unsigned  long item_id;
```

RETURNS

status
The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display
A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual*.

window

The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

item_id

The number assigned to this data item. This number was returned by the previous call to BEGIN COPY TO CLIPBOARD.

DESCRIPTION

END COPY TO CLIPBOARD locks the clipboard from access by other applications, places data in the clipboard data structure, and unlocks the clipboard. Data items copied to the clipboard by COPY TO CLIPBOARD are not actually entered in the clipboard data structure until the call to END COPY TO CLIPBOARD.

INQUIRE NEXT PASTE COUNT

Returns the number of data item formats available for the next-paste item in the clipboard.

VAX FORMAT

status = **DWT$INQUIRE_NEXT_PASTE_COUNT**
(display, window, count, max_format_name_len)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
count	uns longword	uns longword	write	reference
max_format_name_len	uns longword	uns longword	write	reference

MIT C FORMAT

status = **DwtInquireNextPasteCount**
(display, window, count, max_format_name_len)

argument information

```
int DwtInquireNextPasteCount(display, window, count,
                             max_format_name_len)
      Display         *display;
      Window          window;
      unsigned long   *count;
      unsigned long   *max_format_name_len;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display
A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window
The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

count
The number of data item formats available for the next-paste item in the clipboard. If no formats are available, this argument equals zero. The count includes formats that were passed by name.

max_format_name_len
The maximum length of all format names for the next-paste item in the clipboard.

DESCRIPTION

INQUIRE NEXT PASTE COUNT returns the number of data item formats available for the next-paste item in the clipboard. INQUIRE NEXT PASTE COUNT also returns the maximum name length for all formats in which the next-paste item is stored.

INQUIRE NEXT PASTE FORMAT

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

VAX FORMAT

status = **DWT$INQUIRE_NEXT_PASTE_FORMAT** *(display, window, number, format_name_buf, copied_len)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
number	uns longword	uns longword	read	reference
format_name_buf	char string	char string	write	descriptor
copied_len	uns longword	uns longword	write	reference

MIT C FORMAT

status = **DwtInquireNextPasteFormat** *(display, window, number, format_name_buf, buffer_len, copied_len)*

argument information

```
int DwtInquireNextPasteFormat(display, window, number,
                              format_name_buf, buffer_len,
                              copied_len)
      Display         *display;
      Window          window;
      int             number;
      char            *format_name_buf;
      unsigned long   buffer_len;
      unsigned long   *copied_len;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.
ClipboardTruncate	The data returned is truncated because the user did not provide a buffer that was large enough to hold the data.

ARGUMENTS

display

A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual*.

window

The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

number

The number of the format name to be obtained. If this number is greater than the number of formats for the data item, then INQUIRE NEXT PASTE FORMAT returns a zero in the **copied_len** argument.

format_name_buf

The buffer that receives the format name.

buffer_len (C only)

The number of bytes in the format name buffer.

copied_len

The number of bytes in the string copied to the buffer. If this argument equals zero, then there is no *n*th format for the next-paste item.

DESCRIPTION

INQUIRE NEXT PASTE FORMAT returns a specified format name for the next-paste item in the clipboard.

If the name must be truncated, then a warning status is returned.

INQUIRE NEXT PASTE LENGTH

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

VAX FORMAT

status = **DWT$INQUIRE_NEXT_PASTE_LENGTH**
(display, window, format_name, length)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
format_name	char string	char string	read	descriptor
length	uns longword	uns longword	write	reference

MIT C FORMAT

status = **DwtInquireNextPasteLength**
(display, window, format_name, length)

argument information

```
int DwtInquireNextPasteLength(display, window, format_name,
                              length)
      Display         *display;
      Window          window;
      char            *format_name;
      unsigned long   *length;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display
A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window
The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

format_name
The name of the format for the next-paste item.

length
The length of the next data item in the specified format. This argument equals zero if no data is found for the specified format or if there is no item on the clipboard.

DESCRIPTION

INQUIRE NEXT PASTE LENGTH returns the length of the data stored under a specified format name for the next-paste item in the clipboard.

If no data is found for the specified format, or if there is no item on the clipboard, INQUIRE NEXT PASTE LENGTH returns a value of zero.

Note: Any format passed by name is assumed to have the length passed in the call to COPY TO CLIPBOARD, even though the data has not been transferred yet to the clipboard in that format.

LIST PENDING ITEMS

Returns a list of pending items as data ID/private ID pairs for a specified format name.

VAX FORMAT

status = **DWT$LIST_PENDING_ITEMS**
(display, window, format_name, item_list, item_count)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
format_name	char string	char string	read	descriptor
item_list	uns longword	uns longword	write	reference
item_count	uns longword	uns longword	write	reference

MIT C FORMAT

status = **DwtListPendingItems**
(display, window, format_name, item_list, item_count)

argument information

```
int DwtListPendingItems(display, window, format_name,
                        item_list, item_count)
      Display                  *display;
      Window                   window;
      char                     *format_name;
      DwtClipboardPendingList  *item_list
      unsigned long            *item_count;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display
A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window
The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

format_name
A string containing the name of the format for which the list of data ID/private ID pairs is to be obtained.

item_list
The address of the array of data ID/private ID pairs for the specified format name. This parameter has a type of DwtClipboardPendingList. The application is responsible for freeing the memory provided by this routine for storing the list.

item_count
The number of items returned in the list. If there is no data for the specified format name, or if there is no item on the clipboard, this argument equals zero.

DESCRIPTION

LIST PENDING ITEMS returns a list of pending items as data ID/private ID pairs for a specified format name. For the purposes of this routine, a data item is considered pending if the application originally passed it by name, the application has not yet copied the data, and the item has not been deleted from the clipboard.

The application is responsible for freeing the memory provided by this routine to store the list.

This routine is used by an application when exiting to determine if the data that it passed by name should be sent to the clipboard.

For related information, see the routines RECOPY TO CLIPBOARD and CANCEL COPY FORMAT.

RECOPY TO CLIPBOARD

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

VAX FORMAT

status = **DWT$RECOPY_TO_CLIPBOARD**
(display, window, data_id, buffer, private_id)

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference
data_id	uns longword	uns longword	read	reference
buffer	char string	char string	read	descriptor
private_id	longword	longword	read	reference

MIT C FORMAT

status = **DwtReCopyToClipboard**
(display, window, data_id, buffer, length, private_id)

argument information

```
int DwtReCopyToClipboard(display, window, data_id,
                         buffer, length, private_id)
      Display        *display;
      Window         window;
      unsigned long  data_id;
      char           *buffer;
      unsigned long  length;
      int            private_id;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display
A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window
The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

data_id
An identifying number assigned by COPY TO CLIPBOARD to the data item. This argument uniquely identifies the data item and the format.

buffer
The buffer from which the clipboard copies the data.

length (C only)
The number of bytes in the data item.

private_id
The private data stored with the data item by the application that placed the data item on the clipboard.

DESCRIPTION

RECOPY TO CLIPBOARD copies the actual data for a data item that was previously passed by name to the clipboard. Additional calls to RECOPY TO CLIPBOARD append new data to the existing data.

RECOPY TO CLIPBOARD cannot be used to pass data by name.

For related information, see the routines LIST PENDING ITEMS and CANCEL COPY FORMAT.

UNDO COPY TO CLIPBOARD

Deletes the last item placed on the clipboard.

VAX FORMAT

status = **DWT$UNDO_COPY_TO_CLIPBOARD** *(display, window)*

argument information

Argument	Usage	Data Type	Access	Mechanism
status	uns longword	uns longword	write	value
display	identifier	uns longword	read	reference
window	identifier	uns longword	read	reference

MIT C FORMAT

status = **DwtUndoCopyToClipboard** *(display, window)*

argument information

```
int DwtUndoCopyToClipboard(display, window)
      Display  *display;
      Window   window;
```

RETURNS

status

The status return value. Possible status return values for this routine are as follows:

Return Value Name	Description
ClipboardSuccess	The routine is successful.
ClipboardLocked	The routine failed because the clipboard was locked by another application. The application can continue to try to call the routine again with the same parameters until the lock goes away. This gives the application the opportunity to ask if the user wants to keep trying or to give up on the operation.

ARGUMENTS

display

A reference to the display information originally returned by the Xlib display routine OPEN DISPLAY. For more information about the Xlib routine OPEN DISPLAY, see the *VMS DECwindows Xlib Routines Reference Manual.*

window

The window identifier that relates the application window to the clipboard. The same application instance should pass the same window identifier to each clipboard routine that it calls.

DESCRIPTION

UNDO COPY TO CLIPBOARD deletes the last item placed on the clipboard if the item was placed there by an application with the passed display and window identifiers. Any data item deleted from the clipboard by the original call to COPY TO CLIPBOARD is restored. If the display or window identifiers do not match the last copied item, no action is taken and this routine has no effect.

7 High-Level Widget Routines

The XUI Toolkit contains high-level run-time routines that allow application programmers to create and manipulate widgets.

High-level widget creation routines implement the specific user interface for the application. The high-level routines allow the application to access common attributes needed in screen display, menus, input queries, and command functions. This set of high-level routines provides much of the DECwindows-style conforming screen display and user interface tools in building any application.

The high-level widget manipulation routines allow the application programmer to manipulate widgets or obtain information from existing widgets.

This chapter presents the high-level widget routines in alphabetical order. Table 7–1 lists the supported high-level widget routines.

Table 7–1 High-Level Widget Routines

Routine Name	Description
ATTACHED DIALOG BOX	Creates an attached dialog box widget.
CAUTION BOX	Creates a caution box widget.
COMMAND APPEND	Appends the passed string to the current command line and executes it, if required.
COMMAND ERROR MESSAGE	Writes an error message in the command window and refreshes the command line.
COMMAND SET	Replaces the current command string with the one passed.
COMMAND WINDOW	Creates a command window widget.
DIALOG BOX	Creates a dialog box widget.
FILE SELECTION	Creates a file selection widget.
FILE SELECTION DO SEARCH	Initiates a search with a directory mask option. Otherwise, the current directory mask is used.
HELP	Creates a help widget box.
LABEL	Creates a label widget.
LIST BOX	Creates a list box widget.
LIST BOX ADD ITEM	Adds an item to the list within a list box widget.
LIST BOX DELETE ITEM	Deletes an item from the list within a list box widget.

(continued on next page)

Table 7–1 (Cont.) High-Level Widget Routines

Routine Name	Description
LIST BOX DELETE POS	Deletes an item identified by its position from the list within a list box widget.
LIST BOX DESELECT ALL ITEMS	Cancels the selection of all previously selected items in a list box.
LIST BOX DESELECT ITEM	Cancels the selection of a previously selected item in a list box.
LIST BOX DESELECT POS	Cancels the selection of an item identified by its position in a list box.
LIST BOX ITEM EXISTS	Verifies the existence of a particular item in a list box.
LIST BOX SELECT ITEM	Selects an item in the list box.
LIST BOX SELECT POS	Selects an item identified by its position in the list box.
LIST BOX SET HORIZ POS	Sets the horizontal position to a specified position.
LIST BOX SET ITEM	Makes a specified item (if it exists) the first visible item in a list box, or as close to the top as possible. The item always becomes visible.
LIST BOX SET POS	Makes a specified position (item number in list) the top visible position in a list box, or as close to the top as possible.
MAIN WINDOW	Creates a main window widget.
MAIN WINDOW SET AREAS	Sets up or adds the menu bar, work window, command window, and scroll bar widgets to the main window widget of the application.
MENU	Creates a menu widget.
MENU BAR	Creates a menu bar widget.
MENU POSITION	Positions the pop-up menu when user presses MB2.
MESSAGE BOX	Creates a message box widget.
OPTION MENU	Creates an option menu widget.
PULL DOWN MENU ENTRY	Creates a pull-down menu entry widget.
PULL DOWN MENU ENTRY HILITE	Creates a pull-down menu entry widget.
PUSH BUTTON	Creates a push button widget.
RADIO BOX	Creates a radio box widget.
SCALE	Creates a scale widget.
SCALE GET SLIDER	Gets the current value of the slider position displayed in the scale.
SCALE SET SLIDER	Sets or changes the current value of the slider position displayed in the scale.

(continued on next page)

Table 7–1 (Cont.) High-Level Widget Routines

Routine Name	Description
SCROLL BAR	Creates a scroll bar widget.
SCROLL BAR GET SLIDER	Retrieves the current size and position parameters of the slider in the scroll bar widget.
SCROLL BAR SET SLIDER	Sets or changes the current size/position parameters of the slider in the scroll bar widget.
SCROLL WINDOW	Creates a scroll window widget.
SCROLL WINDOW SET AREAS	Specifies the widgets that are to serve as the work area, horizontal scoll bar, and vertical scroll bar widgets for the scoll window widget.
SELECTION	Creates a selection widget.
SEPARATOR	Creates a separator widget.
S TEXT	Creates a simple text widget.
S TEXT CLEAR SELECTION	Clears the global selection highlighted in the simple text widget.
S TEXT GET EDITABLE	Obtains the current permission state concerning whether the text in the simple text widget can be edited by the user.
S TEXT GET MAX LENGTH	Gets the current maximum allowable length of the text string in the simple text widget.
S TEXT GET SELECTION	Gets the global selection, if any, currently highlighted in the simple text widget.
S TEXT GET STRING	Gets the text string from the simple text widget.
S TEXT REPLACE	Replaces a portion of the current text string in the simple text widget or inserts a new substring in the text.
S TEXT SET EDITABLE	Sets the permission state that determines whether the text in the widget can be edited by the user.
S TEXT SET MAX LENGTH	Sets the maximum allowable length of the text string in the simple text widget.
S TEXT SET SELECTION	Makes specified text in the simple text widget the current global selection and highlights it in the simple text widget.
S TEXT SET STRING	Sets the text string in the simple text widget.

(continued on next page)

Table 7–1 (Cont.) High-Level Widget Routines

Routine Name	Description
TOGGLE BUTTON	Creates a toggle button widget.
TOGGLE BUTTON GET STATE	Gets the current state of the toggle button.
TOGGLE BUTTON SET STATE	Sets or changes the current state of the toggle button.
WINDOW	Creates a window widget.
WORK BOX	Creates a work box widget.

7.1 High-Level Widget Routines

This section provides information about high-level widget routines. See Chapter 1 for a description of the format used in the routines.

ATTACHED DIALOG BOX

Creates an attached dialog box widget.

VAX FORMAT

widget = **DWT$ATTACHED_DB**
(parent_widget, name, default_position, x, y, title, style, map_callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
default_position	Boolean	uns longword	read	reference
x	position	longword	read	reference
y	position	longword	read	reference
title	comp string	uns longword	read	reference
style	uns longword	uns longword	read	reference
map_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtAttachedDB**
(parent_widget, name, default_position, x, y, title, style, map_callback, help_callback)

argument information

```
Widget DwtAttachedDB(parent_widget, name, default_position,
                     x, y, title, style, map_callback,
                     help_callback)
      Widget          parent_widget;
      char            *name;
      Boolean         default_position;
      Position        x;
      Position        y;
      DwtCompString   title;
      unsigned char   style;
      DwtCallbackPtr  map_callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

default_position
A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. This argument sets the **default_position** attribute described in the low-level routine DIALOG BOX CREATE.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

title
The compound string label. The label is given to the window manager for the title bar if the **style** attribute described in the low-level routine DIALOG BOX POPUP CREATE is Modal or Modeless.

style
The style of the dialog box. There are three possible dialog styles: Modal and Modeless for pop-up attached dialog boxes and Workarea for regular attached dialog boxes. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal
DWT$C_MODELESS	DwtModeless	Modeless
DWT$C_WORKAREA	DwtWorkarea	Work area

This argument sets the **style** attribute described in the low-level routines DIALOG BOX CREATE and DIALOG BOX POPUP CREATE.

map_callback
A callback routine or routines called when a dialog box is about to be mapped. This argument is ignored if **style** is Workarea.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine ATTACHED DIALOG BOX CREATE.

CAUTION BOX

Creates a caution box widget for the application to display caution messages.

VAX FORMAT

widget = **DWT$CAUTION_BOX**
(parent_widget, name, default_position, x, y, style, label, yes_label, no_label, cancel_label, default_push_button, callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
default_position	Boolean	uns longword	read	reference
x	position	longword	read	reference
y	position	longword	read	reference
style	uns longword	uns longword	read	reference
label	comp string	uns longword	read	reference
yes_label	comp string	uns longword	read	reference
no_label	comp string	uns longword	read	reference
cancel_label	comp string	uns longword	read	reference
default_push_button	longword	longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtCautionBox**
(parent_widget, name, default_position, x, y, style, label, yes_label, no_label, cancel_label, default_push_button, callback, help_callback)

argument information

```
Widget DwtCautionBox(parent_widget, name, default_position,
                     x, y, style, label, yes_label, no_label,
                     cancel_label, default_push_button,
                     callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Boolean         default_position;
      Position        x;
      Position        y;
      unsigned char   style;
      DwtCompString   label;
      DwtCompString   yes_label;
      DwtCompString   no_label;
      DwtCompString   cancel_label;
      int             default_push_button
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

default_position
A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. This argument sets the **default_position** attribute described in the low-level routine DIALOG BOX CREATE.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

style
The style of the widget. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal type box
DWT$C_MODELESS	DwtModeless	Modeless type box

This argument sets the **style** attribute described in the low-level routine DIALOG BOX POPUP CREATE.

label
The text in the message line or lines. This argument sets the **label** attribute described in the low-level routine CAUTION BOX CREATE.

yes_label
The label for the Yes push button. This argument sets the **yes_label** attribute described in the low-level routine CAUTION BOX CREATE.

no_label
The label for the No push button. This argument sets the **no_label** attribute described in the low-level routine CAUTION BOX CREATE.

cancel_label
The label for the Cancel push button. This argument sets the **cancel_label** attribute described in the low-level routine CAUTION BOX CREATE.

default_push_button
The push button representing the default user action. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_YES_BUTTON	DwtYesButton	Yes button
DWT$C_NO_BUTTON	DwtNoButton	No button
DWT$C_CANCEL_BUTTON	DwtCancelButton	Cancel button

This argument sets the **default_push_button** attribute described in the low-level routine CAUTION BOX CREATE.

callback
The callback routine or routines called when the Yes, No, or Cancel buttons have been activated. This argument sets the **yes_callback**, **no_callback**, and **cancel_callback** attributes described in the low-level routine CAUTION BOX CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION
See the low-level routine CAUTION BOX CREATE.

COMMAND APPEND

Appends the passed string to the current command line.

VAX FORMAT

void = **DWT$COMMAND_APPEND**
(widget, command)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
command	char string	char string	read	descriptor

MIT C FORMAT

void = **DwtCommandAppend**
(widget, command)

argument information

```
void DwtCommandAppend(widget, command)
      Widget  widget;
      char    *command;
```

ARGUMENTS

widget
The identifier of the created widget.

command
The text to be appended to the command line. Lines in a multiline string should be separated by a line-feed character.

DESCRIPTION

Within the command window widget, this routine appends the passed string to the current command line. After the line or lines have been appended, all but the last line is executed and scrolled into the history region. If the last line sent is terminated with a line-feed character, then it too is executed, the command is moved to the command history, and a new prompt is issued. The application-supplied command entered callback is called once for each line executed in a multiline string.

COMMAND ERROR MESSAGE

Writes an error message in the command window.

VAX FORMAT

void = **DWT$COMMAND_ERROR_MESSAGE** *(widget, error)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
error	char string	char string	read	descriptor

MIT C FORMAT

void = **DwtCommandErrorMessage** *(widget, error)*

argument information

```
void DwtCommandErrorMessage(widget, error)
      Widget  widget;
      char    *error;
```

ARGUMENTS

widget

The identifier of the command window widget.

error

Specifies the error message to be placed in the last history line in the command window widget. Lines in a multiline error message should be separated by a line-feed character.

DESCRIPTION

Within the command window widget, this routine writes an error message in the history area. The history is scrolled up as necessary to provide room for the error message line or lines. For recall purposes, error message lines do not become part of the command history.

COMMAND SET

Replaces the current command string with the one passed.

VAX FORMAT

void = **DWT$COMMAND_SET** *(widget, command)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
command	char string	char string	read	descriptor

MIT C FORMAT

void = **DwtCommandSet** *(widget, command)*

argument information

```
void DwtCommandSet(widget, command)
      Widget  widget;
      char    *command;
```

ARGUMENTS

widget

The identifier of the created widget.

command

Specifies the text to replace the text currently on the command line. Lines in multiline commands should be separated by a line-feed character.

DESCRIPTION

Within the command window widget, this routine replaces the current command string with the passed string. A zero-length string can be used to clear the current command line. All but the last line of a multiline command string are executed and scrolled into the history region. If the last line of the command string is terminated by a line-feed character, then it too is executed, the command is moved to the command history, and a new prompt is issued. The application-supplied command entered callback is called once for each line executed in a multiline command string.

COMMAND WINDOW

Creates a command window widget.

VAX FORMAT

widget = **DWT$COMMAND_WINDOW**
(parent_widget, name, prompt, lines, callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
prompt	comp string	uns longword	read	reference
lines	longword	longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtCommandWindow**
(parent_widget, name, prompt, lines, callback, help_callback)

argument information

```
Widget DwtCommandWindow(parent_widget, name, prompt, lines,
                        callback, help_callback)
      Widget          parent_widget;
      char            *name;
      DwtCompString   prompt;
      int             lines;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

prompt
The command line prompt. This argument sets the **prompt** attribute described in the low-level routine COMMAND WINDOW CREATE.

lines
The number of command history lines visible in the command widget window. This argument sets the **lines** attribute described in the low-level routine COMMAND WINDOW CREATE.

callback
The callback routine or routines called when a command is entered or when the contents of a command line change. This argument sets the **command_entered_callback** and the **value_changed_callback** attributes described in the low-level routine COMMAND WINDOW CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION
See the low-level routine COMMAND WINDOW CREATE.

DIALOG BOX

Creates a dialog box widget to contain other subwidgets.

VAX FORMAT

widget = **DWT$DIALOG_BOX**
(parent_widget, name, default_position, x, y, title, style, map_callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
default_position	Boolean	uns longword	read	reference
x	position	longword	read	reference
y	position	longword	read	reference
title	comp string	uns longword	read	reference
style	byte	uns longword	read	reference
map_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtDialogBox**
(parent_widget, name, default_position, x, y, title, style, map_callback, help_callback)

argument information

```
Widget DwtDialogBox(parent_widget, name, default_position,
                    x, y, title, style, map_callback,
                    help_callback)
     Widget          parent_widget;
     char            *name;
     Boolean         default_position;
     Position        x;
     Position        y;
     DwtCompString   title;
     unsigned char   style;
     DwtCallbackPtr  map_callback;
     DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

default_position
A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. This argument sets the **default_position** attribute described in the low-level routine DIALOG BOX CREATE.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

title
The text label. The text label is given to the window manager for the title bar if the **style** attribute is modeless. This argument sets the **title** attribute described in the low-level routine DIALOG BOX POPUP CREATE.

style
The style of the dialog box. There are three possible dialog styles: Modal and Modeless for pop-up attached dialog boxes and Workarea for regular attached dialog boxes. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal
DWT$C_MODELESS	DwtModeless	Modeless
DWT$C_WORKAREA	DwtWorkarea	Work area

This argument sets the **style** attribute described in the low-level routines DIALOG BOX CREATE and DIALOG BOX POPUP CREATE.

map_callback
The callback routine or routines called when the dialog box is mapped.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine DIALOG BOX CREATE.

FILE SELECTION

Creates a file selection box widget for the application to query the user for a file selection.

VAX FORMAT

widget = **DWT$FILE_SELECTION**
(parent_widget, name, x, y, title, value, dirmask, visible_item_count, format, default_position, callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
title	comp string	uns longword	read	reference
value	comp string	uns longword	read	reference
dirmask	comp string	uns longword	read	reference
visible_item_count	longword	longword	read	reference
format	longword	longword	read	reference
default_position	Boolean	uns longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtFileSelection**
(parent_widget, name, x, y, title, value, dirmask, visible_item_count, format, default_position, callback, help_callback)

argument information

```
Widget DwtFileSelection(parent_widget, name, x, y, title,
                        value, dirmask, visible_item_count,
                        format, default_position, callback,
                        help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCompString   title;
      DwtCompString   value;
      DwtCompString   dirmask;
      int             visible_item_count;
      int             format;
      Boolean         default_position;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

title
The text that appears in the banner of the file selection box. This argument sets the **title** attribute described in the low-level routine DIALOG BOX POPUP CREATE.

value
The selected file. The file name appears in the text entry field and is highlighted in the list box, if present. This argument sets the **value** attribute described in the low-level routine SELECTION CREATE.

dirmask
The directory mask used in determining the files displayed in the file selection list box. This argument sets the **dirmask** attribute described in the low-level routine FILE SELECTION CREATE.

visible_item_count
The maximum number of files visible at one time in the file selection list box. This argument sets the **visible_items_count** attribute described in the low-level routine SELECTION CREATE.

format
The style of the pop-up dialog box widget. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal
DWT$C_MODELESS	DwtModeless	Modeless

This sets the **style** attribute described in the low-level routine DIALOG BOX POPUP CREATE.

default_position
A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. This argument sets the **default_position** attribute described in the low-level routine DIALOG BOX CREATE.

callback
The callback routine or routines called when a selection is activated, canceled, or when there is no match for a selected item. This argument sets the **activate_callback**, **cancel_callback**, and **no_match_callback** attributes described in the low-level routine SELECTION CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION
See the low-level routine FILE SELECTION CREATE.

FILE SELECTION DO SEARCH

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

VAX FORMAT

void = **DWT$FILE_SELECTION_DO_SEARCH** *(widget, dirmask)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
dirmask	comp string	uns longword	read	reference

MIT C FORMAT

void = **DwtFileSelectionDoSearch** *(widget, dirmask)*

argument information

```
void DwtFileSelectionDoSearch(widget, dirmask)
      FileSelectionWidget  widget;
      DwtCompString        dirmask;
```

ARGUMENTS

widget

The pointer to the file selection widget data structure.

dirmask

The directory mask for displaying files. If you do not specify a directory mask, the default directory mask is used. This argument sets the **dirmask** attribute described in the low-level routine FILE SELECTION CREATE.

DESCRIPTION

The file selection widget initiates file searches when any of the following occur:

- The file selection widget becomes visible (managed).
- The intrinsic routine SET VALUES is used to change the directory mask.
- The user activates the Apply push button.

- The application calls FILE SELECTION DO SEARCH. Using FILE SELECTION DO SEARCH is another means for applications to initiate a directory search. For example, when the application creates a new file and needs to reflect this change in a mapped file search widget, this routine could be called.

HELP

Creates a help widget.

VAX FORMAT

widget = **DWT$HELP**
(parent_widget, name, default_position, x, y, application_name, library_type, library_spec, first_topic, overview_topic, glossary_topic, unmap_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
default_position	Boolean	uns longword	read	reference
x	position	longword	read	reference
y	position	longword	read	reference
application_name	comp string	uns longword	read	reference
library_type	longword	longword	read	reference
library_spec	comp string	uns longword	read	reference
first_topic	comp string	uns longword	read	reference
overview_topic	comp string	uns longword	read	reference
glossary_topic	comp string	uns longword	read	reference
unmap_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtHelp**
(parent_widget, name, default_position, x, y, application_name, library_type, library_spec, first_topic, overview_topic, glossary_topic, unmap_callback)

argument information

```
Widget DwtHelp(parent_widget, name, default_position, x, y,
               application_name, library_type, library_spec,
               first_topic, overview_topic, glossary_topic,
               unmap_callback)
     Widget          parent_widget;
     DwtCompString   name;
     Boolean         default_position;
     Position        x;
     Position        y;
     DwtCompString   application_name;
     unsigned int    library_type;
     DwtCompString   library_spec;
     DwtCompString   first_topic;
     DwtCompString   overview_topic;
     DwtCompString   glossary_topic;
     DwtCallbackPtr  unmap_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

default_position
A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. This argument sets the **default_position** attribute described in the low-level routine DIALOG BOX CREATE.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

application_name
The application name to be used in the widget title bar.

library_type
The type of help library specified by **library_spec**. The predefined value for this attribute is as follows:

VMS	C	Description
DWT$C_TEXT_ LIBRARY	DwtTextLibrary	VMS help text in a VMS Help Library or ULTRIX Help Directory

library_spec
A host system file specification that identifies the help library.

first_topic
The first help topic to be displayed. If a null string is provided, the **overview_topic** is displayed.

overview_topic
The application overview topic.

glossary_topic
The application glossary topic. If a null string is provided, the Visit Glossary menu item does not appear in the widget's View pull-down menu.

unmap_callback
The callback routine or routines called when the help widget is unmapped.

DESCRIPTION
See the low-level routine HELP CREATE.

LABEL

Creates a label widget.

VAX FORMAT

widget = **DWT$LABEL**
(parent_widget, name, x, y, label, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
label	comp string	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtLabel**
(parent_widget, name, x, y, label, help_callback)

argument information

```
Widget DwtLabel(parent_widget, name, x, y, label,
                help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCompString   label;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x

The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y

The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

label

The text of the label. This argument sets the **label** attribute described in the low-level routine LABEL CREATE.

help_callback

The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine LABEL CREATE.

LIST BOX

Creates a list box widget.

VAX FORMAT

widget = **DWT$LIST_BOX**
(parent_widget, name, x, y, items, item_count, visible_item_count, callback, help_callback, resize, horiz)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
items	array	uns longword	read	reference
item_count	longword	longword	read	reference
visible_item_count	longword	longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference
resize	unsigned longword	uns longword	read	reference
horiz	Boolean	uns longword	read	reference

FORMAT

widget = **DwtListBox**
(parent_widget, name, x, y, items, item_count, visible_item_count, callback, help_callback, resize, horiz)

argument information

```
Widget DwtListBox(parent_widget, name, x, y, items, item_count,
                  visible_item_count, callback, help_callback,
                  resize, horiz)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCompString   *items;
      int             item_count;
      int             visible_items_count;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
      unsigned char   resize;
      Boolean         horiz;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

items
The list of items to be used by the list box widget. This argument sets the **items** attribute described in the low-level routine LIST BOX CREATE.

item_count
The total number of items in the list. This argument sets the **item_count** attribute described in the low-level routine LIST BOX CREATE.

visible_items_count
The maximum number of visible items contained in the list box. For example, if **item_count** is 20, but **visible_items_count** is 5, only 5 items are visible at any one time. This argument sets the **visible_items_count** attribute described in the low-level routine LIST BOX CREATE.

callback
A callback routine or routines called when the single callback, single confirm callback, extend callback, and extend confirm callback functions are activated. This argument sets the **single_callback**,

single_confirm_callback, **extend_callback**, and **extend_confirm_callback** attributes described in the low-level routine LIST BOX CREATE.

help_callback

The callback routine or routines called on a help request.

resize

VAX binding: **DWT$C_NRESIZE**
C binding: **DwtNresize**

How the list box resizes when its children are managed and unmanaged and on geometry requests. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_RESIZE_FIXED	DwtResizeFixed	List box does not change its size when items are added or deleted.
DWT$C_RESIZE_GROW_ONLY	DwtResizeGrowOnly	List box attempts to expand as necessary when items are added.

If **resize** is set to Fixed, DIGITAL suggests that **horiz** be set true.

horiz

A Boolean argument that specifies whether the list box contains a horizontal scroll bar. If true, the list box contains a horizontal scroll bar. If false, the list box does not contain a horizontal scroll bar. This argument sets the **horizontal** attribute described in the low-level routine LIST BOX CREATE.

A horizontal scroll bar cannot be added or deleted to a list box after the list box is created.

DESCRIPTION

See the low-level routine LIST BOX CREATE.

LIST BOX ADD ITEM

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

VAX FORMAT

void = **DWT$LIST_BOX_ADD_ITEM**
(widget, item, position)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
item	comp string	uns longword	read	reference
position	longword	longword	read	reference

MIT C FORMAT

void = **DwtListBoxAddItem**
(widget, item, position)

argument information

```
void DwtListBoxAddItem(widget, item, position)
      Widget         widget;
      DwtCompString  item;
      int            position;
```

ARGUMENTS

widget
The identifier of the list box widget.

item
The text of the item to be added.

position
The placement of the item within the list in terms of cell position. It uses an insert-mode, cell-number scheme with 1 specifying the topmost entry position, and 0 specifying the bottom entry for adding an item to the bottom of the list.

DESCRIPTION

LIST BOX ADD ITEM adds an item to the list within the list box widget.

LIST BOX DELETE ITEM

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

VAX FORMAT

void = **DWT$LIST_BOX_DELETE_ITEM**
(widget, item)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
item	comp string	uns longword	read	reference

MIT C FORMAT

void = **DwtListBoxDeleteItem**
(widget, item)

argument information

```
void DwtListBoxDeleteItem(widget, item)
      Widget         widget;
      DwtCompString  item;
```

ARGUMENTS

widget
The identifier of the list box widget.

item
The text of the item to be deleted.

DESCRIPTION

LIST BOX DELETE ITEM deletes an item from a list within the list box widget. The routine searches the list for the item, removes it, and moves any subsequent entries up one cell position throughout the remaining list. See the related routine LIST BOX ADD ITEM.

LIST BOX DELETE POS

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

VAX FORMAT

void = **DWT$LIST_BOX_DELETE_POS**
(widget, position)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
position	longword	longword	read	reference

MIT C FORMAT

void = **DwtListBoxDeletePos**
(widget, position)

argument information

```
void DwtListBoxDeletePos(widget, position)
      Widget  widget;
      int     position;
```

ARGUMENTS

widget
The identifier of the list box widget.

position
An integer identifying the position of the item to be deleted from the list. The first item in the list is in position 1.

DESCRIPTION

LIST BOX DELETE POS deletes an item from a list within the list box widget. The item to be deleted is identified by its position in the list. The routine searches the list for the specified position, removes the item in that position, and moves any subsequent entries up one cell position throughout the remaining list.

LIST BOX DESELECT ALL ITEMS

Cancels the selection of all previously selected items in a list box.

VAX FORMAT

void = **DWT$LIST_BOX_DESELECT_ALL_ITEMS** *(widget)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference

MIT C FORMAT

void = **DwtListBoxDeselectAllItems** *(widget)*

argument information

```
void DwtListBoxDeselectAllItems(widget)
      Widget  widget;
```

ARGUMENTS

widget
The identifier of the list box widget.

DESCRIPTION

LIST BOX DESELECT ALL ITEMS cancels the selection of all items previously selected, and removes them from the list of selected items.

LIST BOX DESELECT ITEM

Cancels the selection of a previously selected item in a list box.

VAX FORMAT

void = **DWT$LIST_BOX_DESELECT_ITEM**
(widget, item)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
item	comp string	uns longword	read	reference

MIT C FORMAT

void = **DwtListBoxDeselectItem**
(widget, item)

argument information

```
void DwtListBoxDeselectItem(widget,item)
      Widget         widget;
      DwtCompString  item;
```

ARGUMENTS

widget
The identifier of the list box widget.

item
The item in the list box to be deselected.

DESCRIPTION

LIST BOX DESELECT ITEM cancels the selection of an item and removes it from the list of selected items.

LIST BOX DESELECT POS

Cancels the selection of an item identified by its position in a list box.

VAX FORMAT

void = **DWT$LIST_BOX_DESELECT_POS**
(widget, position)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
position	uns longword	uns longword	read	reference

MIT C FORMAT

void = **DwtListBoxDeselectPos**
(widget, position)

argument information

```
void DwtListBoxDeselectPos(widget, position)
      Widget  widget;
      int     position;
```

ARGUMENTS

widget
The identifier of the list box widget.

position
An integer identifying the position of the item to be deselected. The first item in the list box is in position 1.

DESCRIPTION

LIST BOX DESELECT POS cancels the selection of an item based on its position in a list box and removes the item from the list of selected items.

LIST BOX ITEM EXISTS

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

VAX FORMAT

Boolean = **DWT$LIST_BOX_ITEM_EXISTS** *(widget, item)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
item	comp string	uns longword	read	reference

MIT C FORMAT

Boolean = **DwtListBoxItemExists** *(widget, item)*

argument information

```
Boolean DwtListBoxItemExists(widget, item)
      Widget         widget;
      DwtCompString  item;
```

RETURNS

Boolean
If true, a specified item exists in the list box.

ARGUMENTS

widget
The identifier of the list box widget.

item
The item being searched for in the list box.

DESCRIPTION

LIST BOX ITEM EXISTS searches a list box to determine if an item exists.

LIST BOX SELECT ITEM

Selects an item in the list box.

VAX FORMAT

void = **DWT$LIST_BOX_SELECT_ITEM**
(widget, item, notify)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
item	comp string	uns longword	read	reference
notify	Boolean	uns longword	read	reference

MIT C- STYLE FORMAT

void = **DwtListBoxSelectItem**
(widget, item, notify)

argument information

```
void DwtListBoxSelectItem(widget, item, notify)
      Widget         widget;
      DwtCompString  item;
      Boolean        notify;
```

ARGUMENTS

widget
The identifer of the list box widget.

item
The item to be added to the list box.

notify
If true, using this routine results in a callback to the application in the same way that a user selection results in a callback.

DESCRIPTION

LIST BOX SELECT ITEM selects an item in a list box, adds it to a selected items list, and calls back to the application, if **notify** is true.

LIST BOX SELECT POS

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

VAX FORMAT

void = **DWT$LIST_BOX_SELECT_POS**
(widget, position, notify)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
position	uns longword	uns longword	read	reference
notify	Boolean	uns longword	read	reference

MIT C- STYLE FORMAT

void = **DwtListBoxSelectPos**
(widget, position, notify)

argument information

```
void DwtListBoxSelectPos(widget, position, notify)
      Widget   widget;
      int      position;
      Boolean  notify;
```

ARGUMENTS

widget
The identifer of the list box widget.

position
An integer identifying the position of the item to be selected. The first item in the list box is in position 1.

notify
If true, using this routine results in a callback to the application in the same way that a user selection results in a callback.

DESCRIPTION

LIST BOX SELECT POS selects an item in a list box based on its position in the list, adds it to the list of selected items, and calls back to the application, if **notify** is true.

LIST BOX SET HORIZ POS

Sets the horizontal scroll bar slider to a specified position.

VAX FORMAT

void = **DWT$LIST_BOX_SET_HORIZ_POS**
(widget, position)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
position	uns longword	uns longword	read	reference

MIT C FORMAT

void = **DwtListBoxSetHorizPos**
(widget, position)

argument information

```
void DwtListBoxSetHorizPos(widget, position)
      Widget  widget;
      int     position;
```

ARGUMENTS

widget
The identifier of the list box widget.

position
The position in pixels of the horizontal scroll bar slider in the list box.

DESCRIPTION

LIST BOX SET HORIZ POS is used only if the list box has a horizontal scroll bar and the list box contains items too wide to be visible within the current list box width. For example, this routine is used by the file selection widget.

LIST BOX SET ITEM

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

VAX FORMAT

void = **DWT$LIST_BOX_SET_ITEM** *(widget, item)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
item	comp string	uns longword	read	reference

MIT C FORMAT

void = **DwtListBoxSetItem** *(widget, item)*

argument information

```
void DwtListBoxSetItem(widget, item)
      Widget         widget;
      DwtCompString  item;
```

ARGUMENTS

widget

The identifier of the list box widget.

item

The item to be made the first item in the list box.

DESCRIPTION

LIST BOX SET ITEM makes an item in the list box the first visible item.

This routine determines which item in the list box displays at the top of the list box. The choice of which item displays is limited by the **item_count** and **visible_item_count** arguments to the list box widget.

When **visible_item_count** is greater than 1 and less than **item_count**, the list box widget fills the list box with the maximum visible items regardless of the position of the item. For example, if **item_count** is 10 and **visible_item_count** is 5, you cannot make item 8 display at tne top of the list box. Items 6 through 10 would display. Setting **item** to the fourth item in the list would make items 4 through 8 display. If **visible_item_count** is 1, you can make any item in the list display at the top of the list box.

LIST BOX SET POS

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

VAX FORMAT

void = **DWT$LIST_BOX_SET_POS**
(widget, position)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
position	uns longword	uns longword	read	reference

MIT C FORMAT

void = **DwtListBoxSetPos**
(widget, position)

argument information

```
void DwtListBoxSetPos(widget, position)
      Widget  widget;
      int     position;
```

ARGUMENTS

widget
The identifier of the list box widget.

position
The item number in the list displayed in the top position in the list box.

DESCRIPTION

LIST BOX SET POS determines which item in the list box displays at the top of the list box. The choice of which item displays is limited by the **item_count** and **visible_item_count** arguments to the list box widget.

When **visible_item_count** is greater than 1 and less than **item_count**, the list box widget fills the list box with the maximum visible items regardless of the **position** value. For example, if **item_count** is 10 and **visible_item_count** is 5, you cannot make item 8 display at the top of the list box. Items 6 through 10 would display. Setting **position** to 4 would make items 4 through 8 display. If **visible_item_count** is 1, you can make any item in the list display at the top of the list box.

MAIN WINDOW

Creates the main window widget.

VAX FORMAT

widget = **DWT$MAIN_WINDOW**
(parent_widget, name, x, y, width, height)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
width	dimension	uns longword	read	reference
height	dimension	uns longword	read	reference

MIT C FORMAT

widget = **DwtMainWindow**
(parent_widget, name, x, y, width, height)

argument information

```
Widget DwtMainWindow(parent_widget, name, x, y,
                     width, height)
      Widget     parent_widget;
      char       *name;
      Position   x;
      Position   y;
      Dimension  width;
      Dimension  height;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the list box widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

width
The width of the widget window in pixels. This argument sets the core widget **width** attribute described in Section 8.2.

height
The height of the widget window in pixels. This argument sets the core widget **height** attribute described in Section 8.2.

DESCRIPTION
See the low-level routine MAIN WINDOW CREATE.

MAIN WINDOW SET AREAS

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

VAX FORMAT

void = **DWT$MAIN_SET_AREAS**
(widget, menu bar, work_window, command_window, h_scroll, v_scroll)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
menu bar	identifier	uns longword	read	reference
work_window	identifier	uns longword	read	reference
command_window	identifier	uns longword	read	reference
h_scroll	identifier	uns longword	read	reference
v_scroll	identifier	uns longword	read	reference

MIT C FORMAT

void = **DwtMainSetAreas**
(widget, menu bar, work_window, command_window, h_scroll, v_scroll)

argument information

```
void DwtMainSetAreas(widget, menu bar, work_window,
                     command_window, h_scroll, v_scroll)
      Widget  widget;
      Widget  menu bar;
      Widget  work_window;
      Widget  command_window;
      Widget  h_scroll;
      Widget  v_scroll;
```

ARGUMENTS

widget
The identifier of the main window widget.

menu_bar
The identifier of the menu bar widget.

work_window
The identifier of the work window widget.

command_window
The identifier of the command window widget

h_scroll
The identifier of the horizontal scroll bar widget.

v_scroll
The identifier of the vertical scroll bar widget.

DESCRIPTION

MAIN WINDOW SET AREAS sets up or adds the menu bar, work window, command window, and scroll bar widgets to the main window widget for the application. Each area is optional and may be passed as null. The title bar is provided by the window manager.

See the Geometry Management section of MAIN WINDOW CREATE for more detail on setting the areas of a main window widget.

MENU

Creates a menu widget to contain menu items.

VAX FORMAT

widget = **DWT$MENU**
(parent_widget, name, x, y, format, orientation, entry_callback, map_callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
format	identifier	uns longword	read	reference
orientation	uns byte	uns longword	read	reference
entry_callback	callback	uns longword	read	reference
map_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtMenu**
(parent_widget, name, x, y, format, orientation, entry_callback, map_callback, help_callback)

argument information

```
Widget DwtMenu(parent_widget, name, x, y, format, orientation,
               entry_callback, map_callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      int             format;
      unsigned char   orientation;
      DwtCallbackPtr  entry_callback;
      DwtCallbackPtr  map_callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the list box widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

format
The type of menu widget. The predefined values for this argument are as follows:

VAX	C	Description
DWT$C_MENU_POPUP	DwtMenuPopup	Pop-up menu
DWT$C_MENU_PULL_DOWN	DwtMenuPullDown	Pull-down menu
DWT$C_MENU_WORK_AREA	DwtMenuWorkArea	Work area menu

orientation
The orientation of the menu. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ORIENTATION_HORIZONTAL	DwtOrientationHorizontal	Horizontal menu
DWT$C_ORIENTATION_VERTICAL	DwtOrientationVertical	Vertical menu

This argument sets the **orientation** attribute described in the low-level routine MENU CREATE.

entry_callback
The callback routine that causes all menu entry activation callbacks to be revectored to call back through this callback. This argument sets the **entry_callback** attribute described in the low-level routine MENU CREATE.

map_callback
A widget-specific callback routine that enables the application to request a callback because the window is about to be mapped.

help_callback

The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine MENU CREATE.

MENU BAR

Creates a menu bar widget to contain menus.

VAX FORMAT

widget = **DWT$MENU_BAR**
(parent_widget, name, entry_callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
entry_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtMenuBar**
(parent_widget, name, entry_callback, help_callback)

argument information

```
Widget DwtMenuBar(parent_widget, name, entry_callback,
                  help_callback)
      Widget          parent_widget;
      char            *name;
      DwtCallbackPtr  entry_callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

entry_callback

The callback routine that causes all menu entry activation callbacks to be revectored to call back through this callback. This argument sets the **entry_callback** attribute described in the low-level routine MENU CREATE.

help_callback

The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine MENU BAR CREATE.

MENU POSITION

Positions the pop-up menu when user presses MB2.

VAX FORMAT

void = **DWT$MENU_POSITION** *(widget, event)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
event	event	uns longword	read	reference

MIT C FORMAT

void = **DwtMenuPosition** *(widget, event)*

argument information

```
void DwtMenuPosition
      Widget  widget;
      XEvent  *event;
```

ARGUMENTS

widget

The pop-up menu widget to be positioned.

event

The event passed to the action procedure, which manages the pop-up menu.

DESCRIPTION

Positions the pop-up menu when the user presses MB2. This routine must be called before managing the pop-up menu.

MESSAGE BOX

Creates a message box widget.

VAX FORMAT

widget = **DWT$MESSAGE_BOX**
(parent_widget, name, default_position, x, y, style, label, ok_label, callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
default_position	Boolean	uns longword	read	reference
x	position	longword	read	reference
y	position	longword	read	reference
style	uns longword	uns longword	read	reference
label	comp string	uns longword	read	reference
ok_label	comp string	uns longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtMessageBox**
(parent_widget, name, default_position, x, y, style, label, ok_label, callback, help_callback)

argument information

```
Widget DwtMessageBox(parent_widget, name, default_position,
                     x, y, style, label, ok_label, callback,
                     help_callback)
      Widget          parent_widget;
      char            *name;
      Boolean         default_position;
      Position        x;
      Position        y;
      unsigned char   style;
      DwtCompString   label;
      DwtCompString   ok_label;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

default_position
A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. This argument sets the **default_position** attribute described in the low-level routine DIALOG BOX CREATE.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

style
The style of the message box widget. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal
DWT$C_MODELESS	DwtModeless	Modeless

This argument sets the **style** attribute described in the low-level routine DIALOG BOX POPUP CREATE.

label
The text in the message line or lines. This argument sets the **label** attribute described in the low-level routine MESSAGE BOX CREATE.

ok_label
The text of the OK push button. This argument sets the **ok_label** attribute described in the low-level routine MESSAGE BOX CREATE.

callback
The callback routine or routines called when the OK push button is activated. This argument sets the **yes_callback** atttribute described in the low-level routine MESSAGE BOX CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine MESSAGE BOX CREATE.

OPTION MENU

Creates an option menu widget.

VAX FORMAT

widget = **DWT$OPTION_MENU**
(parent_widget, name, x, y, label, entry_callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
label	comp string	uns longword	read	reference
entry_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtOptionMenu**
(parent_widget, name, x, y, label, entry_callback, help_callback)

argument information

```
Widget DwtOptionMenu(parent_widget, name, x, y, label,
                     entry_callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCompString   label;
      DwtCallbackPtr  entry_callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

label
The text in the menu label. This argument sets the **label** attribute described in the low-level routine MENU CREATE.

entry_callback
The callback routine that causes all menu entry activation callbacks to be revectored to call back through this callback. This argument sets the **entry_callback** attribute described in the low-level routine MENU CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine OPTION MENU CREATE.

PULL DOWN MENU ENTRY

Creates a pull-down menu entry widget.

VAX FORMAT

widget = **DWT$PULL_DOWN_MENU_ENTRY** *(parent_widget, name, x, y, label, menu_id, callback, help_callback)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
label	comp string	uns longword	read	reference
menu_id	indentifier	uns longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtPullDownMenuEntry** *(parent_widget, name, x, y, label, menu_id, callback, help_callback)*

argument information

```
Widget DwtPullDownMenuEntry(parent_widget, name, x, y, label,
                            menu_id, callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCompString   label;
      Widget          menu_id;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
Name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

label
The text of the label in the parent menu widget. This argument sets the **label** attribute described in the low-level routine LABEL CREATE.

menu_id
The identifier of the pull-down menu widget.

callback
The callback routine or routines called when a button inside a pull-down menu entry widget is activated. This argument sets the **activate_callback** described in the low-level routine PULL DOWN MENU ENTRY CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine PULL DOWN MENU ENTRY CREATE.

PULL DOWN MENU ENTRY HILITE

Highlights a menu entry.

VAX FORMAT

void = **DWT$PULL_DOWN_MENU_ENTRY_HILITE** *(widget, hilite)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
hilite	longword	longword	read	reference

MIT C FORMAT

void = **DwtPullDownMenuEntryHilite** *(widget, hilite)*

argument information

```
void DwtPullDownMenuEntryHilite
      Widget  widget;
      int     hilite;
```

ARGUMENTS

widget

The pull-down menu entry widget to highlight.

hilite

A Boolean attribute that specifies whether a pull-down menu entry is highlighted. If true, the entry is highlighted. If false, the entry is not highlighted.

DESCRIPTION

This routine is used to manually control the highlighting of pull-down menu entry widgets.

PUSH BUTTON

Creates a push button widget.

VAX FORMAT

widget = **DWT$PUSH_BUTTON**
(parent_widget, name, x, y, label, callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
label	comp string	uns longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtPushButton**
(parent_widget, name, x, y, label, callback, help_callback)

argument information

```
Widget DwtPushButton(parent_widget, name, x, y, label,
                     callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCompString   label;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

label
The text of the button label. This argument sets the **label** attribute described in the low-level routine PUSH BUTTON CREATE.

callback
The callback routine or routines called when a push button is activated. This argument sets the **activate_callback** attribute described in the low-level routine PUSH BUTTON CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine PUSH BUTTON CREATE.

RADIO BOX

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

VAX FORMAT

widget = **DWT$RADIO_BOX**
(parent_widget, name, x, y, entry_callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
entry_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtRadioBox**
(parent_widget, name, x, y, entry_callback, help_callback)

argument information

```
Widget DwtRadioBox(parent_widget, name, x, y, entry_callback,
                   help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCallbackPtr  entry_callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

entry_callback
The callback routine that causes all menu entry activation callbacks to be revectored to call back through this callback. This argument sets the **entry_callback** attribute described in the low-level routine MENU CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION
See the low-level routine RADIO BOX CREATE.

SCALE

Creates a scale widget.

VAX FORMAT

widget = **DWT$SCALE**
(parent_widget, name, x, y, width, height, scale_width, scale_height, title, min_value, max_value, decimal_points, value, orientation, callback, drag_callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
width	dimension	uns longword	read	reference
height	dimension	uns longword	read	reference
scale_width	dimension	uns longword	read	reference
scale_height	dimension	uns longword	read	reference
title	comp string	uns longword	read	reference
min_value	longword	longword	read	reference
max_value	longword	longword	read	reference
decimal_points	longword	longword	read	reference
value	longword	longword	read	reference
orientation	uns byte	uns longword	read	reference
callback	callback	uns longword	read	reference
drag_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtScale**
(parent_widget, name, x, y, width, height, scale_width, scale_height, title, min_value, max_value, decimal_points, value, orientation, callback, drag_callback, help_callback)

argument information

```
Widget DwtScale(parent_widget, name, x, y, width,
                height, scale_width, scale_height, title,
                min_value, max_value, decimal_points, value,
                orientation, callback, drag_callback,
                help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCompString   title;
      Dimension       width;
      Dimension       height;
      Dimension       scale_width;
      Dimension       scale_height;
      int             min_value;
      int             max_value;
      int             decimal_points;
      int             value;
      unsigned char   orientation;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  drag_callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

width
The width of the scale widget. The scale widget width is calculated based on the scale width, the label widths, and orientation. This argument sets the core widget **width** attribute described in Section 8.2.

height
The height of the scale widget. The scale widget height is calculated based on the scale height, the labels, and orientation. This argument sets the core widget **height** attribute described in Section 8.2.

scale_width
The width of the scale (excluding the scale labels). This argument sets the **scale_width** attribute described in the low-level routine SCALE CREATE.

scale_height
The height of the scale (excluding the scale labels). This argument sets the **scale_height** attribute described in the low-level routine SCALE CREATE.

title
The text string to appear in the scale window widget.

min_value
The minimum value of the scale (at the bottom or left end position). This argument sets the **min_value** attribute described in the low-level routine SCALE CREATE.

max_value
The maximum value of the scale (at the top or right end position). This argument sets the **max_value** attribute described in the low-level routine SCALE CREATE.

decimal_points
The number of decimal points to shift the current slider value for display of the next slider position. This argument sets the **decimal_points** attribute described in the low-level routine SCALE CREATE.

value
The value represented by the slider's current position. This argument sets the **value** attribute described in the low-level routine SCALE CREATE.

orientation
The orientation of the scale. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ORIENTATION_ HORIZONTAL	DwtOrientationHorizontal	Horizontal scale
DWT$C_ORIENTATION_VERTICAL	DwtOrientationVertical	Vertical scale

callback
The callback routine or routines called when the value of the scale changes. This argument sets the **value_changed_callback** attribute described in the low-level routine SCALE CREATE.

drag_callback
A callback routine or routines called because the user is dragging the slider.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine SCALE CREATE.

SCALE GET SLIDER

Gets the current value of the slider position displayed in the scale.

VAX FORMAT

void = **DWT$SCALE_GET_SLIDER** *(widget, value)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
value	longword	longword	write	reference

MIT C FORMAT

void = **DwtScaleGetSlider** *(widget, value)*

argument information

```
void DwtScaleGetSlider(widget, value)
      Widget  widget;
      int      *value;
```

ARGUMENTS

widget
The identifier of the scale widget.

value
The current slider position value.

DESCRIPTION

SCALE GET SLIDER retrieves the current slider position for the application. See the related routine SCALE.

SCALE SET SLIDER

Sets or changes the current value of the slider position displayed in the scale.

VAX FORMAT

void = **DWT$SCALE_SET_SLIDER** *(widget, value)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
value	longword	longword	read	reference

MIT C FORMAT

void = **DwtScaleSetSlider** *(widget, value)*

argument information

```
void DwtScaleSetSlider(widget, value)
      Widget  widget;
      int     value;
```

ARGUMENTS

widget

The identifier of the scale widget.

value

The new slider position for the scale. This argument sets the **value** attribute described in the low-level routine SCALE CREATE.

DESCRIPTION

SCALE SET SLIDER sets or changes the current slider position within the scale widget display for the application. See the related routine SCALE and SCALE GET SLIDER.

SCROLL BAR

Creates a scroll bar widget.

VAX FORMAT

widget = **DWT$SCROLL_BAR**
(parent_widget, name, x, y, width, height, inc, page_inc, shown, int_value, min_value, max_value, orientation, callback, help_callback, unit_inc_callback, unit_dec_callback, page_inc_callback, page_dec_callback, to_top_callback, to_bottom_callback, drag_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
width	dimension	uns longword	read	reference
height	dimension	uns longword	read	reference
inc	longword	longword	read	reference
page_inc	longword	longword	read	reference
shown	longword	longword	read	reference
int_value	longword	longword	read	reference
min_value	longword	longword	read	reference
max_value	longword	longword	read	reference
orientation	uns byte	uns longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference
unit_inc_callback	callback	uns longword	read	reference
unit_dec_callback	callback	uns longword	read	reference
page_inc_callback	callback	uns longword	read	reference
page_dec_callback	callback	uns longword	read	reference
to_top_callback	callback	uns longword	read	reference
to_bottom_callback	callback	uns longword	read	reference
drag_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtScrollBar**
(parent_widget, name, x, y, width, height, inc, page_inc, shown, int_value, min_value, max_value, orientation, callback, help_callback, unit_inc_callback, unit_dec_callback, page_inc_callback, page_dec_callback, to_top_callback, to_bottom_callback, drag_callback)

argument information

```
Widget DwtScrollBar(parent_widget, name, x, y, width, height,
                    inc, page_inc, shown, int_value, min_value,
                    max_value, orientation, callback,
                    help_callback, unit_inc_callback,
                    unit_dec_callback, page_inc_callback,
                    page_dec_callback, to_top_callback,
                    to_bottom_callback, drag_callback)
    Widget          parent_widget;
    char            *name;
    Position        x;
    Position        y;
    Dimension       width;
    Dimension       height;
    int             inc;
    int             page_inc;
    int             shown;
    int             int_value;
    int             min_value;
    int             max_value;
    unsigned char   orientation;
    DwtCallbackPtr  callback;
    DwtCallbackPtr  help_callback;
    DwtCallbackPtr  unit_inc_callback;
    DwtCallbackPtr  unit_dec_callback;
    DwtCallbackPtr  page_inc_callback;
    DwtCallbackPtr  page_dec_callback;
    DwtCallbackPtr  to_top_callback;
    DwtCallbackPtr  to_bottom_callback;
    DwtCallbackPtr  drag_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

width
The width of the widget. This argument sets the core widget **width** attribute described in Section 8.2.

height
The height of the widget. This argument sets the core widget **height** attribute described in Section 8.2.

inc
The amount of button increment and decrement. If this argument is not zero, the widget automatically adjusts the slider when a unit increment or unit decrement action occurs. This argument sets the **inc** attribute described in the low-level routine SCROLL BAR CREATE.

page_inc
The amount of page increment and decrement. If this argument is not zero, the widget automatically adjusts the slider when a page increment or page decrement action occurs. This argument sets the **page_inc** attribute described in the low-level routine SCROLL BAR CREATE.

shown
The size of the slider as a value between zero and the absolute value of **max_value** minus **min_value**. The size of the slider varies, depending on how much of the slider scroll area it represents. This argument sets the **shown** attribute described in the low-level routine SCROLL BAR CREATE.

int_value
The position of the top or left end of the scroll bar's slider. This argument sets the **int_value** attribute described in the low-level routine SCROLL BAR CREATE.

min_value
The scroll bar's minimum value. This argument sets the **min_value** attribute described in the low-level routine SCROLL BAR CREATE.

max_value
The scroll bar's maximum value. This argument sets the **max_value** attribute described in the low-level routine SCROLL BAR CREATE.

orientation
The orientation of the scroll bar. The predefined values for this argument are as follows:

VAX	C	Description
DWT$C_ORIENTATION_ HORIZONTAL	DwtOrientationHorizontal	Horizontal scroll bar

VAX	C	Description
DWT$C_ORIENTATION_VERTICAL	DwtOrientationVertical	Vertical scroll bar

This argument sets the **orientation** attribute described in the low-level routine SCROLL BAR CREATE.

callback
The callback routine or routines called when the value of the scroll bar changes. This argument sets the **value_changed_callback** attribute described in the low-level routine SCROLL BAR CREATE.

help_callback
The callback routine or routines called on a help request. This argument sets the **help_callback** attribute described in Section 8.2.

unit_inc_callback
The callback routine or routines called because the user selected the unit increment scroll function. This argument sets the **unit_inc_callback** described in the low-level routine SCROLL BAR CREATE.

unit_dec_callback
A widget-specific callback routine because the user selected the unit decrement function. This argument sets the **unit_dec_callback** described in the low-level routine SCROLL BAR CREATE.

page_inc_callback
The callback routine or routines called because the user selected the page increment function. This argument sets the **page_inc_callback** described in the low-level routine SCROLL BAR CREATE.

page_dec_callback
The callback routine or routines called because the user selected the page decrement scroll function. This argument sets the **page_dec_callback** described in the low-level routine SCROLL BAR CREATE.

to_top_callback
The callback routine or routines called because the user selected the to-top scroll function. This argument sets the **to_top_callback** described in the low-level routine SCROLL BAR CREATE.

to_bottom_callback
The callback routine or routines called because the user selected the to-bottom scroll function. This argument sets the **to_bottom_callback** described in the low-level routine SCROLL BAR CREATE.

drag_callback
The callback routine or routines called because the user is dragging the scroll bar slider. This argument sets the **drag_callback** described in the low-level routine SCROLL BAR CREATE.

DESCRIPTION
See the low-level routine SCROLL BAR CREATE.

SCROLL BAR GET SLIDER

Retrieves the current size and position parameters of the slider in the scroll bar widget.

VAX FORMAT

void = **DWT$SCROLL_BAR_GET_SLIDER**
(widget, value, shown, inc, page_inc)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
value	longword	longword	write	reference
shown	longword	longword	write	reference
inc	longword	longword	write	reference
page_inc	longword	longword	write	reference

MIT C FORMAT

void = **DwtScrollBarGetSlider**
(widget, value, shown, inc, page_inc)

argument information

```
void DwtScrollBarGetSlider(widget, value, shown, inc, page_inc)
      Widget  widget;
      int     *value;
      int     *shown;
      int     *inc;
      int     *page_inc;
```

ARGUMENTS

widget
The identifier of the scroll bar widget.

value
The top or left end of the scroll bar's slider as a value between **min_value** and **max_value**.

shown
The size of the slider as a value between zero and the absolute value of **max_value** minus **min_value**.

inc
The amount of button increment and decrement.

page_inc
The amount of page increment and decrement.

DESCRIPTION

SCROLL BAR GET SLIDER retrieves the currently-displayed scroll bar widget slider size and position parameters for the application. See the related routines SCROLL BAR and SCROLL BAR SET SLIDER.

SCROLL BAR SET SLIDER

Sets or changes the current size/position parameters of the slider in the scroll bar widget.

VAX FORMAT

void = **DWT$SCROLL_BAR_SET_SLIDER**
(widget, value, shown, inc, page_inc, notify)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
value	longword	longword	read	reference
shown	longword	longword	read	reference
inc	longword	longword	read	reference
page_inc	longword	longword	read	reference
notify	Boolean	uns longword	read	reference

MIT C FORMAT

void = **DwtScrollBarSetSlider**
(widget, value, shown, inc, page_inc, notify)

argument information

```
void DwtScrollBarSetSlider(widget, value, shown, inc, page_inc,
                           notify)
      Widget   widget;
      int      value;
      int      shown;
      int      inc;
      int      page_inc;
      Boolean  notify;
```

ARGUMENTS

widget
The identifier of the scroll bar widget.

value
The scroll bar's slider position between the **min_value** and **max_value** arguments to the scroll bar widget. This argument sets the **int_value** attribute described in the low-level routine SCROLL BAR CREATE.

shown
The size of the slider as a value between zero and the absolute value of **max_value** minus **min_value**. The size of the slider varies, depending on how much of the slider scroll area it represents. This argument sets

the **shown** attribute described in the low-level routine SCROLL BAR CREATE.

inc

The amount of button increment and decrement. If this argument is not zero, the widget automatically adjusts the slider when an increment or decrement action occurs. This argument sets the **inc** attribute described in the low-level routine SCROLL BAR CREATE.

page_inc

The amount of page increment and decrement. If this argument is not zero, the widget automatically adjusts the slider when an increment decrement action occurs. This argument sets the **page_inc** attribute described in the low-level routine SCROLL BAR CREATE.

notify

A Boolean attribute that, if true, causes the scroll bar widget to call the value changed callback if its value changes as a result of the SCROLL BAR SET SLIDER routine. If false, the callback does not occur.

DESCRIPTION

SCROLL BAR SET SLIDER sets or changes the currently displayed scroll bar widget slider parameters for the application. See the related routines SCROLL BAR and SCROLL BAR GET SLIDER.

SCROLL WINDOW

Creates a scroll window widget.

VAX FORMAT

widget = **DWT$SCROLL_WINDOW**
(parent_widget, name, x, y, width, height)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
width	dimension	uns longword	read	reference
height	dimension	uns longword	read	reference

MIT C FORMAT

widget = **DwtScrollWindow**
(parent_widget, name, x, y, width, height)

argument information

```
Widget DwtScrollWindow(parent_widget, name, x, y,
                       width, height)
      Widget      parent_widget;
      char       *name;
      Position    x;
      Position    y;
      Dimension   width;
      Dimension   height;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x

The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y

The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

width

The width of the window in pixels. This argument sets the core widget **width** attribute described in Section 8.2.

height

The height of the window in pixels. This argument sets the core widget **height** attribute described in Section 8.2.

DESCRIPTION

See the low-level routine SCROLL WINDOW CREATE.

SCROLL WINDOW SET AREAS

Specifies the widgets that are to serve as the work area, horizontal scroll bar, and vertical scroll bar widgets for the scroll window widget.

VAX FORMAT

void = **DWT$SCROLL_WINDOW_SET_AREAS**
(widget, h_scroll, v_scroll, work_region)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
h_scroll	widget	uns longword	read	reference
v_scroll	widget	uns longword	read	reference
work_region	widget	uns longword	read	reference

MIT C FORMAT

void = **DwtScrollWindowSetAreas**
(widget, h_scroll, v_scroll, work_region)

argument information

```
void DwtScrollWindowSetAreas(widget, h_scroll, v_scroll,
                             work_region)
     Widget  widget;
     Widget  h_scroll;
     Wdiget  v_scroll;
     Widget  work_region;
```

ARGUMENTS

widget
The identifier of the created widget.

h_scroll
The widget identifier for the horizontal scroll bar for the scroll window widget.

v_scroll
The widget identifier for the vertical scroll bar for the scroll window widget.

work_region
The widget identifier for the window to serve as the scroll window work area.

DESCRIPTION

SCROLL WINDOW SET AREAS enables you to specify the window work area, a horizontal scroll bar, and a vertical scroll bar for the scroll window widget. Each area is optional and may be passed as null. See the related routine SCROLL WINDOW.

SELECTION

Creates a selection widget.

VAX FORMAT

widget = **DWT$SELECTION**
(parent_widget, name, x, y, title, value, items, item_count, visible_item_count, style, default_position, callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
title	comp string	uns longword	read	reference
value	comp string	uns longword	read	reference
items	array	uns longword	read	reference
item_count	longword	longword	read	reference
visible_item_count	longword	longword	read	reference
style	uns longword	uns longword	read	reference
default_position	Boolean	uns longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtSelection**
(parent_widget, name, x, y, title, value, items, item_count, visible_item_count, style, default_position, callback, help_callback)

argument information

```
Widget DwtFileSelection(parent_widget, name,
                        x, y, title, value, items,
                        items_count, visible_item_count, style,
                        default_position,
                        callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCompString   title;
      DwtCompString   value;
      DwtCompString   *items;
      int             item_count;
      int             visible_item_count;
      int             style;
      Boolean         default_position;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

title
The text that appears in the window banner. This argument sets the **title** attribute described in the low-level routine DIALOG BOX CREATE.

value
The text in the text edit field. This argument sets the **value** attribute described in the low-level routine SELECTION CREATE.

items
The items in the selection widget. This argument sets the **items** attribute described in the low-level routine SELECTION CREATE.

item_count
The number of items in the selection widget. This argument sets the **item_count** attribute described in the low-level routine SELECTION CREATE.

visible_item_count
The number of items displayed in the list box. This argument sets the **visible_item_count** attribute described in the low-level routine SELECTION CREATE.

style
The style of the pop-up dialog box. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal
DWT$C_MODELESS	DwtModeless	Modeless

This argument sets the **style** attribute described in the low-level routine DIALOG BOX POPUP CREATE.

default_position
A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. This argument sets the **default_position** attribute described in the low-level routine DIALOG BOX CREATE.

callback
A callback routine or routines called when a selection is made, a selection is canceled, or when there is no match for an item selected. This argument sets the **activate_callback**, **cancel_callback**, and **no_match_callback** attributes described in the low-level routine SELECTION CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION
See the low-level routine SELECTION CREATE.

SEPARATOR

Creates a separator widget.

VAX FORMAT

widget = **DWT$SEPARATOR**
(parent_widget, name, x, y, orientation)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
orientation	uns byte	uns longword	read	reference

MIT C FORMAT

widget = **DwtSeparator**
(parent_widget, name, x, y, orientation)

argument information

```
Widget DwtSeparator(parent_widget, name, x, y, orientation)
      Widget         parent_widget;
      char           *name;
      Position       x;
      Position       y;
      unsigned char  orientation;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y

The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

orientation

The orientation of the scroll bar. The predefined values for this argument are as follows:

VAX	C	Description
DWT$C_ORIENTATION_ HORIZONTAL	DwtOrientationHorizontal	Horizontal scroll bar
DWT$C_ORIENTATION_ VERTICAL	DwtOrientationVertical	Vertical scroll bar

This argument sets the **orientation** attribute described in the low-level routine SEPARATOR CREATE.

DESCRIPTION

See the low-level routine SEPARATOR CREATE.

S TEXT

Creates a simple text widget.

VAX FORMAT

widget = **DWT$S_TEXT**
(parent_widget, name, x, y, cols, rows, string_value)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
cols	longword	longword	read	reference
rows	longword	longword	read	reference
string_value	char string	char string	read	descriptor

MIT C FORMAT

widget = **DwtSText**
(parent_widget, name, x, y, cols, rows, string_value)

argument information

```
Widget DwtSText(parent_widget, name, x, y, cols, rows,
                string_value)
      Widget    parent_widget;
      char      *name;
      Position  x;
      Position  y;
      int       cols;
      int       rows;
      char      *string_value;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

cols
The width of the text window measured in character cells. This argument sets the **cols** attribute described in the low-level routine S TEXT CREATE.

rows
The height of the text window measured in character cells or number of lines. This argument sets the **rows** attribute described in the low-level routine S TEXT CREATE.

string_value
The actual text to display. This argument sets the **value** attribute described in the low-level routine S TEXT CREATE.

DESCRIPTION
See the low-level routine S TEXT CREATE.

S TEXT CLEAR SELECTION

Clears the global selection highlighted in the simple text widget.

VAX FORMAT

void = **DWT$S_TEXT_CLEAR_SELECTION**
(widget, time)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
time	date time	uns longword	read	reference

MIT C FORMAT

void = **DwtSTextClearSelection**
(widget, time)

argument information

```
void DwtSTextClearSelection(widget, time)
      Widget  widget;
      Time    time;
```

ARGUMENTS

widget
The identifier of the simple text widget.

time
The time of the event that led to the call to SET SELECTION. You can pass either a timestamping or CurrentTime. However, whenever possible use the timestamping of the event leading to the call.

DESCRIPTION

S TEXT clears the global selection highlighted in the simple text widget. See the related routines S TEXT, S TEXT SET SELECTION, and S TEXT GET SELECTION.

S TEXT GET EDITABLE

Obtains the current permission state concerning whether the text in the simple text widget can be edited by the user.

VAX FORMAT

editable = **DWT$S_TEXT_GET_EDITABLE** *(widget)*

argument information

Argument	Usage	Data Type	Access	Mechanism
editable	Boolean	uns longword	write	write
widget	widget	uns longword	read	reference

MIT C FORMAT

editable = **DwtSTextGetEditable** *(widget)*

argument information

```
Boolean DwtSTextGetEditable(widget)
      Widget  widget;
```

RETURNS

editable
A Boolean attribute that, if true, indicates the user can edit the string text in the simple text widget. When false, the user cannot edit the text string.

ARGUMENTS

widget
The identifier of the simple text widget.

DESCRIPTION

S TEXT GET EDITABLE returns the current permission state concerning whether the text in the simple text widget can be edited by the user. See the related routines S TEXT and S TEXT SET EDITABLE.

S TEXT GET MAX LENGTH

Obtains the current maximum allowable length of the text string in the simple text widget.

VAX FORMAT

max_len = **DWT$S_TEXT_GET_MAX_LENGTH** *(widget)*

argument information

Argument	Usage	Data Type	Access	Mechanism
max_len	uns longword	uns longword	write	reference
widget	widget	uns longword	read	reference

MIT C FORMAT

max_len = **DwtSTextGetMaxLength** *(widget)*

argument information

```
int DwtSTextGetMaxLength(widget)
      Widget  widget;
```

RETURNS

max_len
The maximum length, in characters, of the text string in the simple text widget.

ARGUMENTS

widget
The identifier of the simple text widget.

DESCRIPTION

S TEXT GET MAX LENGTH returns the current maximum allowable length of the text string in the simple text widget. See the related routines S TEXT and S TEXT SET MAX LENGTH.

S TEXT GET SELECTION

Retrieves the global selection, if any, currently highlighted in the simple text widget.

VAX FORMAT

void = **DWT$S_TEXT_GET_SELECTION**
(widget, selection, selection_len)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
selection	identifier	char string	write	descriptor
selection_len	uns longword	uns longword	write	reference

MIT C FORMAT

selection = **DwtSTextGetSelection**
(widget)

argument information

```
char *DwtSTextGetSelection(widget)
      char    *selection;
      Widget  widget;
```

RETURNS

selection (C only)
A pointer to the text string.

ARGUMENTS

widget
The identifier of the simple text widget.

selection (VAX only)
A pointer to the text string selected.

selection_len (VAX only)
The length of the text string.

DESCRIPTION

S TEXT GET SELECTION retrieves the text currently selected in the simple text widget. It returns a null pointer if no text is selected in the widget. The application is responsible for freeing the storage associated with the text string by calling the intrinsic routine FREE.

See the related routines S TEXT, S TEXT SET SELECTION, and S TEXT CLEAR SELECTION.

S TEXT GET STRING

Retrieves the text string from the simple text widget.

VAX FORMAT

void = **DWT$S_TEXT_GET_STRING**
(widget, string_value, string_len)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
string_value	identifier	char string	write	descriptor
string_len	uns longword	uns longword	write	reference

MIT C FORMAT

string_value = **DwtSTextGetString**
(widget)

argument information

```
char *DwtSTextGetString(widget)
      Widget  widget;
```

RETURNS

string_value (C only)
A pointer to the current text string in the simple text widget.

ARGUMENTS

widget
The identifier of the simple text widget.

string_value (VAX only)
The current text string in the simple text widget.

string_len (VAX only)
The length of the text string.

DESCRIPTION

S TEXT GET STRING retrieves the current string from the simple text widget. When using the MIT C binding, the application is responsible for freeing the storage associated with the string by calling the intrinsic routine FREE. See the related routines S TEXT, S TEXT SET STRING, and S TEXT REPLACE.

S TEXT REPLACE

Replaces a portion of the current text string in the simple text widget or inserts a new substring in the text.

VAX FORMAT

void = **DWT$S_TEXT_REPLACE**
(widget, from_pos, to_pos, value)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
from_pos	longword	longword	read	reference
to_pos	longword	longword	read	reference
value	char string	char string	read	descriptor

MIT C FORMAT

void = **DwtSTextReplace**
(widget, from_pos, to_pos, value)

argument information

```
void DwtXTextReplace(widget, from_pos, to_pos, value)
      Widget  widget;
      int     from_pos;
      int     to_pos;
      char    *value;
```

ARGUMENTS

widget
The identifier of the simple text widget.

from_pos
The first character position of the text string being replaced.

to_pos
The last character position of the text string being replaced.

value
The text to replace part of the current text in the simple text widget.

DESCRIPTION

S TEXT REPLACE replaces part of the text string in the simple text widget. Within the widget, positions are numbered starting at 0 and increase sequentially. For example, to replace the second and third characters in the string, **from_pos** should be 1 and **to_pos** should be 3. To insert a string after the fourth character, **from_pos** and **to_pos** should both be 4. See the related routine S TEXT, S TEXT SET STRING, and S TEXT GET STRING.

S TEXT SET EDITABLE

Sets the permission state that determines whether the text in the widget can be edited by the user.

VAX FORMAT

void = **DWT$S_TEXT_SET_EDITABLE**
(widget, editable)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
editable	Boolean	uns longword	read	reference

MIT C FORMAT

void = **DwtSTextSetEditable**
(widget, editable)

argument information

```
void DwtSTextSetEditable(widget, editable)
      Widget   widget;
      Boolean  editable;
```

ARGUMENTS

widget
The identifier of the simple text widget.

editable
A Boolean attribute that, if true, indicates the user can edit the string text in the simple text widget. When false, the user cannot edit the text string.

DESCRIPTION

S TEXT SET EDITABLE sets the permission state concerning whether text in the simple text widget can be edited by the user. See the related routines S TEXT and S TEXT GET EDITABLE.

S TEXT SET MAX LENGTH

Sets the maximum allowable length of the text string in the simple text widget.

VAX FORMAT

void = **DWT$S_TEXT_SET_MAX_LENGTH** *(widget, max_len)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
max_len	longword	longword	read	reference

MIT C FORMAT

void = **DwtSTextSetMaxLength** *(widget, max_len)*

argument information

```
void DwtSTextSetMaxLength(widget, max_len)
      Widget  widget;
      int     max_len;
```

ARGUMENTS

widget

The identifier of the simple text widget.

max_len

The maximum length, in characters, of the text string in the simple text widget. This argument sets the **max_len** attribute described in the low-level routine S TEXT CREATE.

DESCRIPTION

X TEXT SET MAX LENGTH sets the maximum allowable length of the text in the simple text widget. It prohibits the user from entering text strings longer than this limit. See the related routines S TEXT and S TEXT GET MAX LENGTH.

S TEXT SET SELECTION

Makes specified text in the simple text widget the current global selection and highlights it in the simple text widget.

VAX FORMAT

void = **DWT$S_TEXT_SET_SELECTION**
(widget, first, last, time)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
first	longword	longword	read	reference
last	longword	longword	read	reference
time	date time	uns longword	read	reference

MIT C FORMAT

void = **DwtSTextSetSelection**
(widget, first, last, time)

argument information

```
void DwtXTextSetSelection(widget, first, last, time)
      Widget  widget;
      int     first;
      int     last;
      Time    time;
```

ARGUMENTS

widget
The identifier of the simple text widget.

first
The first character position of the text string being selected.

last
The last character position of the text string being selected.

time
Specifies the time of the event that led to the call to SET SELECTION. You can pass either a timestamping or CurrentTime. However, whenever possible use the timestamping of the event leading to the call.

DESCRIPTION

S TEXT SET SELECTION makes specified text in the simple text widget the current global selection and highlights it in the simple text widget. Within the text window, **first** marks the first character position and **last** marks the last position. The field characters start at 0 and increase sequentially. See the related routines S TEXT, S TEXT GET SELECTION, S TEXT CLEAR SELECTION, and S TEXT SET SELECTION.

S TEXT SET STRING

Sets the text string in the simple text widget.

VAX FORMAT

void = **DWT$S_TEXT_SET_STRING**
(widget, string_value)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
string_value	identifier	char string	read	descriptor

MIT C FORMAT

void = **DwtSTextSetString**
(widget, string_value)

argument information

```
void DwtSTextSetString(widget, string_value)
      Widget  widget;
      char    *string_value;
```

ARGUMENTS

widget
The identifier of the simple text widget.

string_value
The text that replaces all text in the current simple text widget.

DESCRIPTION

S TEXT SET STRING completely changes the text string in the simple text widget. See the related routines S TEXT and S TEXT GET STRING.

TOGGLE BUTTON

Creates a toggle button widget.

VAX FORMAT

widget = **DWT$TOGGLE_BUTTON**
(parent_widget, name, x, y, label, value, callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
label	comp string	uns longword	read	reference
value	Boolean	uns longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtToggleButton**
(parent_widget, name, x, y, label, value, callback, help_callback)

argument information

```
Widget DwtToggleButton(parent_widget, name, x, y, label,
                       value, callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      DwtCompString   label;
      Boolean         value;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

label
The text of the button label. This argument sets the **label** attribute described in the low-level routine LABEL CREATE.

value
The value of the toggle button, which can be either true or false. This argument sets the **value** attribute described in the low-level routine TOGGLE BUTTON CREATE.

callback
A callback routine or routines called when the value of the toggle button changes. This argument sets the **value_changed_callback** attribute described in the low-level routine TOGGLE BUTTON CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine TOGGLE BUTTON CREATE.

TOGGLE BUTTON GET STATE

Retrieves the current state of the toggle button.

VAX FORMAT

value = **DWT$TOGGLE_BUTTON_GET_STATE** *(widget)*

argument information

Argument	Usage	Data Type	Access	Mechanism
value	Boolean	uns longword	write	value
widget	widget	uns longword	read	reference

MIT C FORMAT

value = **DwtToggleButtonGetState** *(widget)*

argument information

```
Boolean DwtToggleButtonGetState(widget)
      Widget   widget;
```

RETURNS

value
The value of the toggle button.

ARGUMENTS

widget
The identifier of the toggle button widget.

DESCRIPTION

TOGGLE BUTTON GET STATE retrieves the current state of the toggle button. See the related routine TOGGLE BUTTON.

TOGGLE BUTTON SET STATE

Sets or changes the current state of the toggle button.

VAX FORMAT

void = **DWT$TOGGLE_BUTTON_SET_STATE**
(widget, value, notify)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	read	reference
value	Boolean	uns longword	read	reference
notify	Boolean	uns longword	read	reference

MIT C FORMAT

void = **DwtToggleButtonSetState**
(widget, value, notify)

argument information

```
void DwtToggleButtonSetState(widget, value, notify)
      Widget   widget;
      Boolean  value;
      Boolean  notify;
```

ARGUMENTS

widget
The identifier of the toggle button widget.

value
The value of the toggle button, which can be either true or false. This argument sets the **value** attribute described in the low-level routine TOGGLE BUTTON CREATE.

notify
A Boolean attribute that, if true, indicates that a change in the state of the toggle button has occurred and that the callback should be activated to respond to the recent change. If false, no change in state has occurred and no callback should be activated.

DESCRIPTION

TOGGLE BUTTON SET STATE sets or changes the current state of the toggle button within the display. See the related routine TOGGLE BUTTON.

WINDOW

Creates a window widget.

VAX FORMAT

widget = **DWT$WINDOW**
(parent_widget, name, x, y, width, height, callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
x	position	longword	read	reference
y	position	longword	read	reference
width	dimension	uns longword	read	reference
height	dimension	uns longword	read	reference
callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtWindow**
(parent_widget, name, x, y, width, height, callback)

argument information

```
Widget DwtWindow(parent_widget, name, x, y, width,
                 height, callback)
      Widget          parent_widget;
      char            *name;
      Position        x;
      Position        y;
      Dimension       width;
      Dimension       height;
      DwtCallbackPtr  callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

width
The width of the window in pixels. This argument sets the core widget **width** attribute described in Section 8.2.

height
The height of the window in pixels. This argument sets the core widget **height** attribute described in Section 8.2.

callback
The callback routine or routines called when an expose event occurs. This argument sets the **expose_callback** attribute described in the low-level routine WINDOW CREATE.

DESCRIPTION
See the low-level routine WINDOW CREATE.

WORK BOX

Creates a work box widget.

VAX FORMAT

widget = **DWT$WORK_BOX**
(parent_widget, name, default_position, x, y, style, label, cancel_label, callback, help_callback)

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
default_position	Boolean	uns longword	read	reference
x	position	longword	read	reference
y	position	longword	read	reference
style	uns longword	uns longword	read	reference
label	comp string	uns longword	read	reference
cancel_label	comp string	uns longword	read	reference
callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtWorkBox**
(parent_widget, name, default_position, x, y, style, label, cancel_label, callback, help_callback)

argument information

```
Widget DwtWorkBox(parent_widget, name, default_position,
                  x, y, style, label, cancel_label,
                  callback, help_callback)
      Widget          parent_widget;
      char            *name;
      Boolean         default_position;
      Position        x;
      Position        y;
      unsigned char   style;
      DwtCompString   label;
      DwtCompString   cancel_label;
      DwtCallbackPtr  callback;
      DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

default_position
A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. This argument sets the **default_position** attribute described in the low-level routine DIALOG BOX CREATE.

x
The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **x** attribute described in Section 8.2.

y
The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. This argument sets the core widget **y** attribute described in Section 8.2.

style
The style of the message box widget. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal
DWT$C_MODELESS	DwtModeless	Modeless

This argument sets the **style** attribute described in the low-level routine DIALOG BOX POPUP CREATE.

label
The text of the work box message.

cancel_label
The label for the Cancel push button. If the label is a null string, the button is not displayed.

callback
The callback routine or routines called when the Cancel button is activated. This argument sets the **cancel_callback** attribute described in the low-level routine WORK BOX CREATE.

help_callback
The callback routine or routines called on a help request.

DESCRIPTION

See the low-level routine WORK BOX CREATE.

8 Low-Level Widget Routines

The XUI Toolkit provides a set of low-level routines for widget creation. Unlike the high-level widget creation routines, low-level routines give the programmer access to all widget attributes. Having access to all widget attributes facilitates widget customization. The attributes of a particular widget include those inherited from superclass widgets and those specific to the widget.

Table 8–1 lists the supported low-level widget routines.

Table 8–1 Low-Level Widget Routines

Routine Name	Description
ATTACHED DIALOG BOX CREATE	Creates an attached dialog box widget.
ATTACHED DIALOG POPUP BOX CREATE	Creates an attached pop-up dialog box widget.
CAUTION BOX CREATE	Creates a caution box widget.
COMMAND WINDOW CREATE	Creates a command window widget.
DIALOG BOX CREATE	Creates a dialog box widget.
DIALOG BOX POPUP CREATE	Creates a pop-up dialog box widget.
FILE SELECTION CREATE	Creates a file selection widget.
HELP CREATE	Creates a help widget.
LABEL CREATE	Creates a label widget.
LIST BOX CREATE	Creates a list box widget.
MAIN WINDOW CREATE	Creates a main window widget.
MENU BAR CREATE	Creates a menu bar widget.
MENU CREATE	Creates a menu widget.
MENU POPUP CREATE	Creates a pop-up menu widget.
MENU PULLDOWN CREATE	Creates a pull-down menu.
MESSAGE BOX CREATE	Creates a message box widget.
OPTION MENU CREATE	Creates an option menu widget.
PULL DOWN MENU ENTRY CREATE	Creates a pull-down menu entry widget.
PUSH BUTTON CREATE	Creates a push button widget.
RADIO BOX CREATE	Creates a radio box widget.
SCALE CREATE	Creates a scale widget.
SCROLL BAR CREATE	Creates a scroll bar widget.
SCROLL WINDOW CREATE	Creates a scroll window widget.

(continued on next page)

Table 8–1 (Cont.) Low-Level Widget Routines

Routine Name	Description
SELECTION CREATE	Creates a selection widget.
SEPARATOR CREATE	Creates a separator widget.
S TEXT CREATE	Creates a simple text widget.
TOGGLE BUTTON CREATE	Creates a toggle button widget.
WINDOW CREATE	Creates a window widget.
WORK BOX CREATE	Creates a work box widget.

8.1 Widget Class Hierarchy

All widgets belong to classes that are arranged in a hierarchy. Some classes contain only one widget. For example, the push button class contains only the push button widget. Other classes contain multiple widgets. The menu and dialog classes each contain several widgets. Widget attributes reside in the widget classes.

Inherited attributes are determined by the widget's position in the widget class hierarchy. Within the widget class hierarchy, widgets inherit default values for attributes from all their superclass widgets. Figure 8–1 shows the widget class hierarchy.

Because attributes of several of the top-level widget classes (core, composite, common) are inherited by the majority of widgets, these attributes are termed **common attributes**. Common attributes are described in the next section.

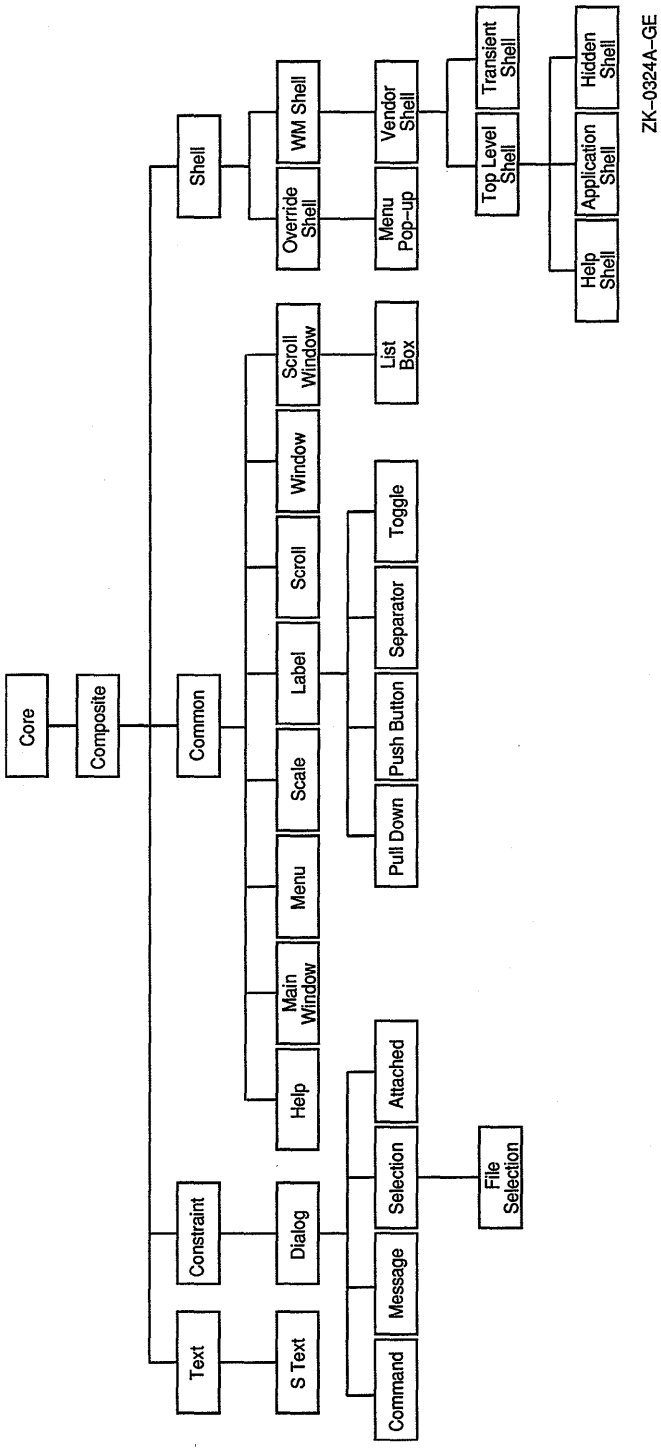
Any exceptions to the rule that widgets inherit attributes from their superclass widgets are described in the Attribute Exceptions section in each routine. Exceptions take two forms: an inherited attribute is not supported by the widget, or the inherited attribute has a different default value from its superclass widget.

Programmers can override the default value of inherited attributes and use widget-specific attributes by using the **override_arglist** argument found in each low-level routine. Each argument in the list is a name/value pair that describes attributes of the created widget.

8.2 Common Attributes

This section describes the common attributes. These are core and common class attributes that are inherited by a majority of widgets. Because the composite class has no user settable attributes, none are listed here.

Figure 8–1 Widget Class Hierarchy



VAX ATTRIBUTE INFORMATION

The following table lists the VAX bindings for the core and common class attributes:

Attribute	Usage	Data Type	Access	Mechanism
Core Widget Attributes				
x	position	word	read	value
y	position	word	read	value
width	dimension	uns word	read	value
height	dimension	uns word	read	value
border_width	dimension	uns word	read	value
border	pixel	uns longword	read	value
border_pixmap	pixmap	uns longword	read	value
background	pixel	uns longword	read	value
background_ pixmap	pixmap	uns longword	read	value
colormap	colormap	uns longword	read	value
sensitive	Boolean	uns byte	read	value
ancestor_sensitive	Boolean	uns byte	read	value
accelerators	translations	uns longword	read	reference
depth	uns longword	uns longword	read	value
translations	translations	uns longword	read	reference
mapped_when_ managed	Boolean	uns longword	read	value
screen	screen	uns longword	read	reference
destroy_callback	callback	uns longword	read	reference
Common Widget Attributes				
foreground	pixel	uns longword	read	value
highlight	pixel	uns longword	read	value
highlight_pixmap	pixmap	uns longword	read	value
user_data	longword	uns longword	read	value
direction_r_to_l	Boolean	uns longword	read	value
font	font list	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C ATTRIBUTE INFORMATION

The following C declarations are used for the core class attributes:

```
Position        x;
Position        y;
Dimension       width;
Dimension       height;
Dimension       border_width;
Pixel           border;
Pixmap          border_pixmap;
Pixel           background;
Pixmap          background_pixmap;
Colormap        colormap;
Boolean         sensitive;
Boolean         ancestor_sensitive;
XtTranslations  accelerators;
int             depth;
XtTranslations  translations;
Boolean         mapped_when_managed;
Screen          *screen;
DwtCallbackPtr  destroy_callback;
```

The following C declarations are used for the common class attributes:

```
Pixel           foreground;
Pixel           highlight;
Pixmap          highlight_pixmap;
Opaque          *user_data;
Boolean         direction_r_to_l;
DwtFontList     font;
DwtCallbackPtr  help_callback;
```

ATTRIBUTES

x

VAX binding: **DWT$C_NX**
C binding: **DwtNx**

The placement, in pixels, of the left side of the widget window relative to the inner upper left corner of the parent window. The default is determined by the geometry manager.

y

VAX binding: **DWT$C_NY**
C binding: **DwtNy**

The placement, in pixels, of the top of the widget window relative to the inner upper left corner of the parent window. The default is determined by the geometry manager.

width

VAX binding: **DWT$C_NWIDTH**
C binding: **DwtNwidth**

The width of the widget window in pixels. The default is widget specific.

height
VAX binding: **DWT$C_NHEIGHT**
C binding: **DwtNheight**

The height of the widget window in pixels. The default is widget specific.

border_width
VAX binding: **DWT$C_NBORDER_WIDTH**
C binding: **DwtNborderWidth**

The widget window border width in pixels. The default is 1 pixel.

border
VAX binding: **DWT$C_NBORDER**
C binding: **DwtNborder**

The widget window border color. The default is the default foreground color.

border_pixmap
VAX binding: **DWT$C_NBORDER_PIXMAP**
C binding: **DwtNborderPixmap**

The widget window border pattern and color. The default is null.

background
VAX binding: **DWT$C_NBACKGROUND**
C binding: **DwtNbackground**

The color of background objects in the widget window. The default is the default background.

background_pixmap
VAX binding: **DWT$C_NBACKGROUND_PIXMAP**
C binding: **DwtNbackgroundPixmap**

The color and pattern of background objects in the widget window. The default is null.

colormap
VAX binding: **DWT$C_NCOLORMAP**
C binding: **DwtNcolormap**

The color map used for the widget's window. The default is the default color map.

sensitive
VAX binding: **DWT$C_NSENSITIVE**
C binding: **DwtNsensitive**

A Boolean attribute that indicates whether the widget reacts to input events. If true, the widget reacts to input events. If false, the widget ignores input events. The default is true.

ancestor_sensitive
VAX binding: **DWT$C_NANCESTOR_SENSITIVE**
C binding: **DwtNancestorSensitive**

A Boolean attribute that indicates whether the parent widget is sensitive. If true, the parent is sensitive. If false, the parent is not sensitive. Applications can get the value of this attribute by using the intrinsic GET VALUES routine. An application should not explicitly set the value of this attribute.

The default is the bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes.

accelerators

VAX binding: **DWT$C_NACCELERATORS**
C binding: **DwtNaccelerators**

A translation table that provides an alternate mode of access to widget functions. Accelerators allow applications to define keystrokes (in addition to clicking on a screen object with the mouse) to activate a function in a widget.

depth

VAX binding: **DWT$C_NDEPTH**
C binding: **DwtNDepth**

The widget window depth. This argument is set at widget creation time and cannot be changed by the intrinsic routine SET VALUES. The default is the depth of the parent window.

translations

VAX binding: **DWT$C_NTRANSLATIONS**
C binding: **DwtNtranslations**

The translation table containing the translation manager syntax for associating particular X events with particular widget events. See the *VMS DECwindows Guide to Application Programming* for information on translation tables.

mapped_when_managed

VAX binding: **DWT$C_NMAPPED_WHEN_MANAGED**
C binding: **DwtNmappedWhenManaged**

A Boolean attribute that, when true, causes the window to be displayed when managed. If false, the window is not displayed when managed. The default is true.

screen

VAX binding: **DWT$C_NSCREEN**
C binding: **DwtNscreen**

Points to the Xlib structure screen. See the display routines in the *VMS DECwindows Xlib Routines Reference Manual* for further information on screen.

destroy_callback

VAX binding: **DWT$C_NDESTROY_CALLBACK**
C binding: **DwtNdestroyCallback**

The callback routine or routines called when the widget is about to be destroyed.

foreground
VAX binding: **DWT$C_NFOREGROUND**
C binding: **DwtNforeground**

The color of foreground objects in the widget window. The default is the default foreground.

highlight
VAX binding: **DWT$C_NHIGHLIGHT**
C binding: **DwtNhighlight**

Color used for highlighting. The default is **foreground**.

highlight_pixmap
VAX binding: **DWT$C_NHIGHLIGHT_PIXMAP**
C binding: **DwtNhighlightPixmap**

The pattern and color used for highlighting. The default is null.

user_data
VAX binding: **DWT$C_NUSER_DATA**
C binding: **DwtNuserData**

Any user private data to be associated with the widget. The default is null.

direction_r_to_l
VAX binding: **DWT$C_NDIRECTION_R_TO_L**
C binding: **DwtNdirectionRToL**

The direction in which the text is drawn. If false, text is drawn from left to right. If true, text is drawn from right to left. The default is false.

font
VAX binding: **DWT$C_NFONT**
C binding: **DwtNfont**

The font of the text used in the widget. The default is the default XUI Toolkit font.

help_callback
VAX binding: **DWT$C_NHELP_CALLBACK**
C binding: **DwtNhelpCallback**

The routine or routines called back on a help request. The default is null.

8.3 Callback Routines

Widgets communicate changes in state to the application that created them by means of callback routines. At creation time (or later using the intrinsic routine SET VALUES), an application specifies the callback routine or routines for a widget instance. Each widget has a (possibly null) set of reasons for issuing callbacks, depending upon how many changes in its state it is willing to communicate. An example of a callback reason is **Value Changed**. The application can specify one or more callback routines for each callback reason.

Most widgets support the common **help_callback** attribute described in Section 8.2.

The basic structure used when specifying callback routines for a callback reason is a null-terminated list of the following entries:

```
typedef struct {
      void    CallbackProc;
      Opaque  tag;
} DwtCallback, *DwtCallbackPtr;
```

CallbackProc	Specifies a pointer to the callback procedure entry point.
tag	Specifies any application-supplied value. This value is usually used by the application to uniquely identify a particular widget instance, and to allow one callback procedure to service multiple widget instances.

The VAX binding names for DwtCallback and DwtCallbackPtr are DWT$CALLBACK and DWT$CALLBACK_PTR, respectively.

By having more than one entry in a callback list for each callback reason supported by a widget, the application can specify more than one routine to be called back when the appropriate widget change in state occurs.

Note: For languages that cannot accept parameters to procedures by immediate value (like VAX PASCAL), the tag should be the address of the value to be returned.

The callback structure for each widget contains at least two fields: *reason* and *event*. Widgets requiring only those two fields can use the standard callback structure (DwtAnyCallbackStruct in the C binding, DWT$ANY_CB_ST in the VAX binding). The format for a callback routine using the standard callback structure follows:

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;

void CallbackProc (widget_id, tag, callback_data)

      Widget                *widget_id;
      Opaque                tag;
      DwtAnyCallbackStruct  *callback_data
```

widget_id	Identifier of the widget doing the callback.
tag	Tag provided when the callback was specified.
callback_data	Identifies a widget-specific data structure. Each XUI Toolkit widget callback data structure has at minimum *reason* and *event* fields.

The *reason* field specifies the reason why this callback procedure was invoked. This field in the data structure is provided so that one callback routine can be called for multiple reasons, if the application desires. Each widget has a different set of callback reasons that are described in the Callback Reasons section in the routine descriptions.

Some widgets have more complex callback structures. These are described in detail under the Callback Data Structure section of the particular routine. Table 8–2 lists the VAX and C callback structure names for all widgets.

Table 8–2 Callback Structure Names

Widget	VAX Structure Name	C Structure Name
Any	DWT$ANY_CB_ST	DwtAnyCallbackStruct
Menu	DWT$MENU_CB_ST	DwtMenuCallbackStruct
Scroll Bar	DWT$SCRL_BAR_CB_ST	DwtScrollBarCallbackStruct
Toggle Button	DWT$TOGGLE_CB_ST	DwtToggleButtonCallbackStruct
Window	DWT$WINDOW_CB_ST	DwtWindowCallbackStruct
Scale	DWT$SCALE_CB_ST	DwtScaleCallbackStruct
List Box	DWT$LISTBOX_CB_ST	DwtListBoxCallbackStruct
Radio Box	DWT$RADIOBOX_CB_ST	DwtRadioBoxCallbackStruct
Selection	DWT$SEL_CB_ST	DwtSelectionCallbackStruct
File Selection	DWT$FILSEL_CB_ST	DwtFileSelectionCallbackStruct
Command Box	DWT$COMWIN_CB_ST	DwtCommandWindowCallbackStruct

After a widget has been created, an application should use the intrinsic routines ADD CALLBACK, ADD CALLBACKS, REMOVE CALLBACK, and REMOVE CALLBACKS to modify a widget callback list. Using the intrinsic routines SET VALUES and GET VALUES sets the entire callback list which may contain callbacks added by the parent widget.

8.4 Low-Level Widget Routines

This section provides information about the low-level routines. See Chapter 1 for an explanation of the format used in the routines.

ATTACHED DIALOG BOX CREATE

Creates an attached dialog box widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
constraint
\
dialog
\
attached dialog box

VAX FORMAT

widget = **DWT$ATTACHED_DB_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
default_horizontal_offset	longword	longword	read	value
default_vertical_offset	longword	longword	read	value
rubber_positioning	Boolean	uns byte	read	value
fraction_base	longword	longword	read	value

MIT C FORMAT

widget = **DwtAttachedDBCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtAttachedDBCreate(parent_widget, name,
                           override_arglist, override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
int      default_horizontal_offset;
int      default_vertical_offset;
Boolean  rubber_positioning;
int      fraction_base;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

default_horizontal_offset

VAX binding: **DWT$C_NDEFAULT_HORIZONTAL_OFFSET**
C binding: **DwtNdefaultHorizontalOffset**

The default horizontal offset for right and left attachments. The offset determines the amount of space between the left or right edge of a child widget and the edge or position to which it is attached. The default offset is zero.

default_vertical_offset

VAX binding: **DWT$C_NDEFAULT_VERTICAL_OFFSET**
C binding: **DwtNdefaultVerticalOffset**

The default vertical offset for top and bottom attachments. The offset determines the amount of space between the top or bottom edge of a child widget and the edge or position to which it is attached. The default offset is zero.

rubber_positioning

VAX binding: **DWT$C_NRUBBER_POSITIONING**
C binding: **DwtNrubberPositioning**

A Boolean attribute that specifies the default attachments of child widget edges. If true, child widget edges default to being attached to themselves. If false, child widget left and top edges default to being attached to the left and top of the attached dialog box. The default is false.

fraction_base

VAX binding: **DWT$C_NFRACTION_BASE**
C binding: **DwtNfractionBase**

The denominator used in specifying fractional positioning. The default is 100.

ATTRIBUTE EXCEPTIONS

None. Inherits all attributes of dialog box and its superclasses.

CONSTRAINT ATTRIBUTES

The following constraint attributes belong to any widget made a child of an attached dialog box widget. These attributes cannot be set on the attached dialog box itself; they must be set on the child widget.

adb_top_attachment

VAX binding: **DWT$C_NADB_TOP_ATTACHMENT**
C binding: **DwtNadbTopAttachment**

Specifies how the top edge of the child widget is attached to its parent attached dialog box widget, another child widget, a position, or itself. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ATTACH_NONE	DwtAttachNone	Do not attach this edge. This attachment can be overridden by other attachments.
DWT$C_ATTACH_ADB	DwtAttachAdb	Attach the top edge of the child widget to the top edge of its parent attached dialog box.
DWT$C_ATTACH_OPP_ADB	DwtAttachOppAdb	Attach the top edge of the child widget to the bottom edge of the parent attached dialog box.

VAX	C	Description
DWT$C_ATTACH_WIDGET	DwtAttachWidget	Attach the top edge of the child widget to the bottom edge of another child widget.
DWT$C_ATTACH_OPP_ WIDGET	DwtAttachOppWidget	Attach the top edge of the child widget to the top edge of another child widget.
DWT$C_ATTACH_POSITION	DwtAttachPosition	Attach the top edge of the child widget to a relative position inside the parent attached dialog box. See the description of the **adb_top_position** attribute.
DWT$C_ATTACH_SELF	DwtAttachSelf	Attach the top edge of the child widget to a relative position corresponding to the edge's initial position in the attached dialog box.

adb_top_widget

VAX binding: **DWT$C_NADB_TOP_WIDGET**
C binding: **DwtNadbTopWidget**

The child widget the top edge is attached to if **adb_top_attachment** is Attach Widget or Attach Opp Widget. This attribute is not used for other types of attachments. The default is null.

adb_top_position

VAX binding: **DWT$C_NADB_TOP_POSITION**
C binding: **DwtNadbTopPosition**

The numerator used with **fraction_base** as denominator to determine the relative positioning of the top edge if **adb_top_attachment** is Attach Position. This attribute is not used for other types of attachments. The default is zero.

adb_top_offset

VAX binding: **DWT$C_NADB_TOP_OFFSET**
C binding: **DwtNadbTopOffset**

The offset of the top edge from the position, widget, or attached dialog box. The default is the value specified with **default_vertical_offset**. If **adb_top_attachment** is Attach Position, one-half the offset value is used for placing the top edge.

adb_bottom_attachment

VAX binding: **DWT$C_NADB_BOTTOM_ATTACHMENT**
C binding: **DwtNadbBottomAttachment**

Specifies how the bottom edge of the widget is attached to the edge of its parent attached dialog box widget, another child widget, a position, or itself. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ATTACH_NONE	DwtAttachNone	Do not attach this edge. This attachment can be overridden by other attachments.
DWT$C_ATTACH_ADB	DwtAttachAdb	Attach this edge to the bottom edge of its parent attached dialog box.
DWT$C_ATTACH_OPP_ADB	DwtAttachOppAdb	Attach this edge to the top edge of the parent attached dialog box.
DWT$C_ATTACH_WIDGET	DwtAttachWidget	Attach this edge to the top edge of another child widget.
DWT$C_ATTACH_OPP_ WIDGET	DwtAttachOppWidget	Attach this edge to the bottom edge of another child widget.
DWT$C_ATTACH_POSITION	DwtAttachPosition	Attach this edge to a relative position inside the parent attached dialog box. See the description of the **adb_bottom_ position** attribute.
DWT$C_ATTACH_SELF	DwtAttachSelf	Attach this edge to a relative position corresponding to the edge's initial position in the parent attached dialog box.

adb_bottom_widget

VAX binding: **DWT$C_NADB_BOTTOM_WIDGET**
C binding: **DwtNadbBottomWidget**

The widget the bottom edge is attached to if **adb_bottom_attachment** is Attach Widget or Attach Opp Widget. This attribute is not used for other attachment types. The default is null.

adb_bottom_position

VAX binding: **DWT$C_NADB_BOTTOM_POSITION**
C binding: **DwtNadbBottomPosition**

The numerator used with **fraction_base** as denominator to determine the relative position of the bottom edge if **adb_bottom_attachment** is Attach Position. This attribute is not used for other attachment types. The default is zero.

adb_bottom_offset

VAX binding: **DWT$C_NADB_BOTTOM_OFFSET**
C binding: **DwtNadbBottomOffset**

The offset of the bottom edge from the position, widget, or attached dialog box. The default is the value specified with **default_vertical_offset**. If **adb_bottom_attachment** is Attach Position, one-half the offset value is used to place the botton edge.

adb_left_attachment

VAX binding: **DWT$C_NADB_LEFT_ATTACHMENT**
C binding: **DwtNadbLeftAttachment**

Specifies how the left edge of the widget is attached to the edge of its parent attached dialog box widget, another child widget, a position, or itself. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ATTACH_NONE	DwtAttachNone	Do not attach this edge. This attachment can be overridden by other attachments.
DWT$C_ATTACH_ADB	DwtAttachAdb	Attach this edge to the left edge of its parent attached dialog box.
DWT$C_ATTACH_OPP_ADB	DwtAttachOppAdb	Attach this edge to the right edge of the parent attached dialog box.
DWT$C_ATTACH_WIDGET	DwtAttachWidget	Attach this edge to the right edge of another child widget.
DWT$C_ATTACH_OPP_ WIDGET	DwtAttachOppWidget	Attach this edge to the left edge of another child widget.
DWT$C_ATTACH_POSITION	DwtAttachPosition	Attach this edge to a relative position inside the parent attached dialog box. See the description of the **adb_left_position** attribute.
DWT$C_ATTACH_SELF	DwtAttachSelf	Attach this edge to a relative position corresponding to the edge's initial position in the parent attached dialog box.

adb_left_widget

VAX binding: **DWT$C_NADB_LEFT_WIDGET**
C binding: **DwtNadbLeftWidget**

The widget the left edge is attached to if **adb_left_attachment** is Attach Widget or Attach Opp Widget. This attribute is not used for other attachment types. The default is null.

adb_left_position

VAX binding: **DWT$C_NADB_LEFT_POSITION**
C binding: **DwtNadbLeftPosition**

The numerator used with **fraction_base** as denominator to determine the relative position of the left edge if **adb_left_attachment** is Attach Position. This attribute is not used for other attachment types. The default is zero.

adb_left_offset

VAX binding: **DWT$C_NADB_LEFT_OFFSET**
C binding: **DwtNadbLeftOffset**

The offset of the left edge from the position, widget, or attached dialog box. The default is the value specified with **default_horizontal_offset**. If **adb_left_attachment** is Attach Position, one-half the offset value is used to place the left edge.

adb_right_attachment

VAX binding: **DWT$C_NADB_RIGHT_ATTACHMENT**
C binding: **DwtNadbRightAttachment**

Specifies how the right edge of the widget is attached to the edge of its parent attached dialog box widget, another child widget, a position, or itself. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ATTACH_NONE	DwtAttachNone	Do not attach this edge. This attachment can be overridden by other attachments.
DWT$C_ATTACH_ADB	DwtAttachAdb	Attach this edge to the right edge of its parent attached dialog box.
DWT$C_ATTACH_OPP_ADB	DwtAttachOppAdb	Attach this edge to the left edge of the parent attached dialog box.
DWT$C_ATTACH_WIDGET	DwtAttachWidget	Attach this edge to the left edge of another child widget.
DWT$C_ATTACH_OPP_ WIDGET	DwtAttachOppWidget	Attach this edge to the right edge of another child widget.

VAX	C	Description
DWT$C_ATTACH_POSITION	DwtAttachPosition	Attach this edge to a relative position inside the parent attached dialog box. See the description of the **adb_right_position** attribute.
DWT$C_ATTACH_SELF	DwtAttachSelf	Attach this edge to a relative position corresponding to the edge's initial position in the parent attached dialog box.

adb_right_widget

VAX binding: **DWT$C_NADB_RIGHT_WIDGET**
C binding: **DwtNadbRightWidget**

The widget the right edge is attached to if **adb_right_attachment** is Attach Widget or Attach Opp Widget. This attribute is not used for other attachment types. The default is null.

adb_right_position

VAX binding: **DWT$C_NADB_RIGHT_POSITION**
C binding: **DwtNadbRightPosition**

The numerator used with the **fraction_base** as denominator to determine the relative position of the right edge if **adb_right_attachment** is Attach Position. This attribute is not used for other attachment types. The default is zero.

adb_right_offset

VAX binding: **DWT$C_NADB_RIGHT_OFFSET**
C binding: **DwtNadbRightOffset**

The offset of the right edge from the position, widget, or attached dialog box. The default is the value specified with **default_horizontal_offset**. If **adb_right_attachment** is Attach Position, one-half the offset value is used to place the right edge.

CALLBACK DATA STRUCTURE

See the low-level routine DIALOG BOX CREATE.

DESCRIPTION

ATTACHED DIALOG BOX CREATE creates an attached dialog box widget to contain child widgets. The attached dialog box acts as a container only. It provides no input semantics over and above the semantics of the widgets that it contains.

The attached dialog box differs from the dialog box in its handling of child widgets. Constraints are placed on each child widget at the time of creation. The default values for the constraint attributes described in ATTACHED DIALOG BOX CREATE are placed on the child unless the programmer specifies values for the constraint attributes. Values are specified either in the **override_arglist** or by using the intrinsic routine SET VALUES.

Using the constraint attributes, the application programmer can attach each of the four edges of a child widget (top, bottom, right, and left) to an edge of the parent attached dialog box, to an edge of another child widget, to a relative position within the attached dialog box, to itself, or to nothing. The possible attachments for each of the four edges are described in the Constraint Attributes section. Specifying these attachments allows the programmer to maintain the position of the child widgets within the attached dialog box as resizing occurs.

In many cases the attachment changes the dimensions of the child widget.

For all attachment types, the programmer can optionally specify an offset in pixels or font units. The offset determines the amount of space between the edge of the child widget and the edge or position to which it is attached. By default, child widgets are positioned in an attached dialog box in terms of font units rather than pixel units. (See the **units** attribute described in DIALOG BOX CREATE.) The x font units are defined to be one-fourth the width of whatever font is supplied for the common attribute **font**. The y font units are defined to be one-eighth the height of whatever font is supplied for the common attribute **font**.

The offsets given are automatically negated when dealing with right and bottom edges. For example, an offset of 5 means that the edge stays 5 units to the right of its attachment if a left edge, and 5 units to the left if a right edge. Offsets default to the value specified for the attached dialog box unless the attachment is Attach Position. In that case, the default offset is one-half the value specified. There are separate horizontal and vertical default offsets.

The application programmer can determine whether the attached dialog box honors resize geometry requests from a given child widget by appropriately setting the **resize** attribute for that child. If it does honor a request, the attached dialog box reconfigures all child widgets based on current attachments of the child widgets.

Child widgets can be added after the attached dialog box widget has been realized. If there is extra room in the attached dialog box, the new child widget is added. If there is not enough room, the attached dialog box requests permission to resize from the geometry manager.

An attached dialog box widget can also be created with the high-level routine ATTACHED DIALOG BOX.

---

geometry management

The attached dialog box widget follows the same rules for geometry management as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

resizing

The attached dialog box widget follows the same rules for resizing as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

ATTACHED DIALOG BOX POPUP CREATE

Creates an attached pop-up dialog box widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
constraint
\
dialog
\
attached dialog box popup

VAX FORMAT

widget = **DWT$ATTACHED_DB_POPUP_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The widget-specific attributes described in the low-level routine ATTACHED DIALOG BOX CREATE can be set in the **override_arglist**.

MIT C FORMAT

widget = **DwtAttachedDBPopupCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtAttachedDBPopupCreate(parent_widget, name,
                                override_arglist,
                                override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The widget-specific attributes described in the low-level routine ATTACHED DIALOG BOX CREATE can be set in the **override_arglist**.

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

See the widget-specific attributes for the low-level routine ATTACHED DIALOG BOX CREATE.

ATTRIBUTE EXCEPTIONS

None. Inherits all attributes of dialog box and its superclasses.

CONSTRAINT ATTRIBUTES

See the constraint attributes described for the low-level routine ATTACHED DIALOG BOX CREATE.

CALLBACK DATA STRUCTURE

See the low-level routine DIALOG BOX POPUP CREATE.

DESCRIPTION

ATTACHED DIALOG BOX POPUP CREATE creates an attached pop-up dialog box widget to contain child widgets. See the description of the low-level routine ATTACHED DIALOG BOX CREATE for details.

geometry management

The attached pop-up dialog box widget follows the same rules for geometry management as its superclass the dialog pop-up widget, described in the low-level routine DIALOG BOX POPUP CREATE.

resizing

The attached pop-up dialog box widget follows the same rules for resizing as its superclass the dialog box pop-up widget, described in the low-level routine DIALOG BOX POPUP CREATE.

CAUTION BOX CREATE

Creates a caution box widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
constraint
\
dialog
\
caution box

VAX FORMAT

widget = **DWT$CAUTION_BOX_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
label	comp string	uns longword	read	reference
yes_label	comp string	uns longword	read	reference
no_label	comp string	uns longword	read	reference
cancel_label	comp string	uns longword	read	reference

Attribute	Usage	Data Type	Access	Mechanism
default_push_button	longword	longword	read	value
yes_callback	callback	uns longword	read	reference
no_callback	callback	uns longword	read	reference
cancel_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtCautionBoxCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtCautionBoxCreate(parent_widget, name,
                           override_arglist, override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
DwtCompString   label;
DwtCompString   yes_label
DwtCompString   no_label;
DwtCompString   cancel_label;
unsigned char   default_push_button;
DwtCallbackPtr  yes_callback;
DwtCallbackPtr  no_callback;
DwtCallbackPtr  cancel_callback;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

label
VAX binding: **DWT$C_NLABEL**
C binding: **DwtNlabel**

The text in the message line or lines. This attribute defaults to the widget name.

yes_label
VAX binding: **DWT$C_NYES_LABEL**
C binding: **DwtNyesLabel**

The label for the Yes push button. If the label has zero length, the button is not displayed. The default is "Yes".

no_label
VAX binding: **DWT$C_NNO_LABEL**
C binding: **DwtNnoLabel**

The label for the No push button. If the label has zero length, the button is not displayed. The default is "No".

cancel_label
VAX binding: **DWT$C_NCANCEL_LABEL**
C binding: **DwtNcancelLabel**

The label for the Cancel push button. If the label is a zero-length string, the button is not displayed. The default is "Cancel".

default_push_button
VAX binding: **DWT$C_NDEFAULT_PUSH_BUTTON**
C binding: **DwtNdefaultPushButton**

The push button that represents the default user action. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_YES_BUTTON	DwtYesButton	Yes button (default)
DWT$C_NO_BUTTON	DwtNoButton	No button
DWT$C_CANCEL_BUTTON	DwtCancelButton	Cancel button

yes_callback
VAX binding: **DWT$C_NYES_CALLBACK**
C binding: **DwtNyesCallback**

The callback routine or routines called when the Yes button is activated. For this routine the callback reason is **Yes**. The default is null.

no_callback
VAX binding: **DWT$C_NNO_CALLBACK**
C binding: **DwtNnoCallback**

The callback routine or routines called when the No button is activated. For this routine the callback reason is **No**. The default is null.

cancel_callback

VAX binding: **DWT$C_NCANCEL_CALLBACK**
C binding: **DwtNcancelCallback**

The callback routine or routines called when the Cancel button is activated. For this routine the callback reason is **Cancel**. The default is null.

ATTRIBUTE EXCEPTIONS

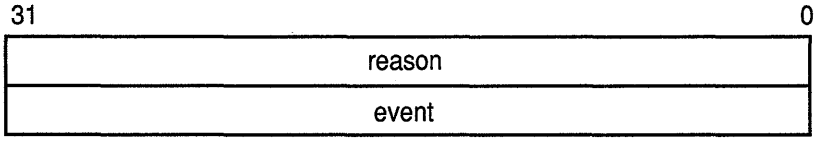
The following attributes described in the low-level routine DIALOG BOX POPUP CREATE are not supported:

- **cancel_button**
- **child_overlap**
- **default_button**
- **text_merge_translation**
- **units**
- **direction_r_to_l**

The following attributes described in the low-level routine DIALOG BOX POPUP CREATE are supported differently by CAUTION BOX CREATE:

- The default for **margin_width** is 12.
- The default for **margin_height** is 10.
- The default for **resize** is Shrink Wrap.
- The default for **style** is Modal.

CALLBACK DATA STRUCTURE



ZK-0091A-GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
     int      reason;
     XEvent   *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Yes

VAX binding: **DWT$C_CRYES**
C binding: **DwtCRYes**

The user activated the Yes button.

No

VAX binding: **DWT$C_CRNO**
C binding: **DwtCRNo**

The user activated the No button.

Cancel

VAX binding: **DWT$C_CRCANCEL**
C binding: **DwtNCRCancel**

The user activated the Cancel button.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help somewhere in the caution box.

DESCRIPTION

CAUTION BOX CREATE creates a caution box widget, a member of the message class of widgets.

The caution box widget is a dialog box that allows the application to display caution messages to the user. The caution message warns the user of the consequences of carrying out an action. When **style** is Modal, execution of the application stops until the user provides input on how to proceed. The box can contain Yes, No, and Cancel push buttons.

If **style** is Modal when the user activates any push button, the widget is cleared from the screen but not destroyed. The widget can be redisplayed using the intrinsic routine MANAGE CHILD.

A caution box can also be created with the high-level routine CAUTION BOX.

geometry management

The caution box widget follows the same rules for geometry management as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

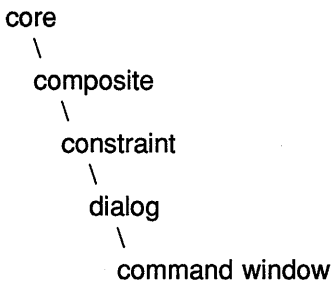
resizing

The caution box widget follows the same rules for resizing as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

COMMAND WINDOW CREATE

Creates a command window widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$COMMAND_WINDOW_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
value	char string	char string	read	reference
prompt	comp string	uns longword	read	reference
lines	uns longword	uns longword	read	value

Attribute	Usage	Data Type	Access	Mechanism
history	char string	char string	read	reference
command_ entered_callback	callback	uns longword	read	reference
value_changed_ callback	callback	uns longword	read	reference
t_translation	translations	uns longword	read	reference

MIT C FORMAT

widget = **DwtCommandWindowCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtCommandWindowCreate(parent_widget, name,
                              override_arglist, override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_ arglist**:

```
char            *value;
DwtCompString   prompt;
int             lines;
char            *history;
DwtCallbackPtr  command_entered_callback;
DwtCallbackPtr  value_changed_callback;
XtTranslations  t_translation;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

value
VAX binding: **DWT$C_NVALUE**
C binding: **DwtNvalue**

The current contents of the command line string. When a command-entered callback is made, this attribute is always null. The default is null.

prompt
VAX binding: **DWT$C_NPROMPT**
C binding: **DwtNprompt**

The command line prompt. The default is ">".

lines
VAX binding: **DWT$C_NLINES**
C binding: **DwtNlines**

The number of command history lines visible in the command widget window. The default is 2 lines.

history
VAX binding: **DWT$C_NHISTORY**
C binding: **DwtNhistory**

The contents of the command line history. Multiple lines should be separated by a linefeed character. The default is the null string.

command_entered_callback
VAX binding: **DWT$C_NCOMMOND_ENTERED_CALLBACK**
C binding: **DwtNcommandEnteredCallback**

The callback routine or routines called when a command is executed. For this routine the callback reason is **Command Entered**. The default is null.

value_changed_callback
VAX binding: **DWT$C_NVALUE_CHANGED_CALLBACK**
C binding: **DwtNvalueChangedCallback**

The callback routine or routines called when the contents of the command line change. For this routine the callback reason is **Value Changed**. The default is null.

t_translation
VAX binding: **DWT$C_NT_TRANSLATION**
C binding: **DwtNtTranslation**

Translations used for the command line text field. The default is null. See the *VMS DECwindows Guide to Application Programming* for information on translation tables.

ATTRIBUTE EXCEPTIONS

The following attributes of the low-level routine DIALOG BOX are not supported:

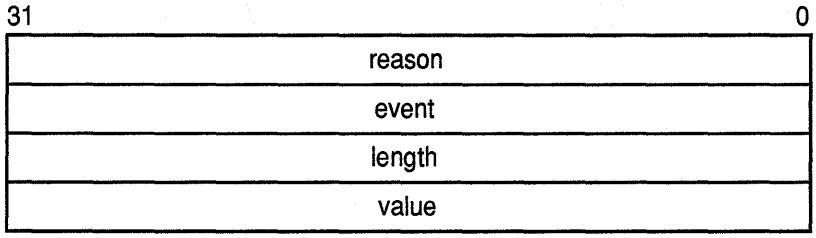
- **child_overlap**
- **resize**

- **text_merge_translation**
- **units**
- **direction_r_to_l**

The following attribute of the low-level routine DIALOG BOX CREATE is supported differently:

- The **default_position** attribute is set to true. This causes the command window to be positioned in the bottom left corner of the parent widget.

CALLBACK DATA STRUCTURE



ZK-0252A-GE

VAX field information

Structure name: DWT$COMWIN_CB_ST

Name	Usage	Data Type	Access	Mechanism
comwindow_reason	callback reason	longword	read	value
comwindow_event	event	uns longword	read	reference
comwindow_length	longword	longword	read	value
comwindow_value	char string	char string	read	reference

MIT C field information

```
typedef struct {
     int      reason;
     XEvent   *event;
     int      length;
     char     *value;
} DwtCommandWindowCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event
A pointer to the X event structure describing the event that generated this callback. For more information about X events, see the *VMS DECwindows Xlib Routines Reference Manual.*

length
The length of the current command line contents.

value
The current command line contents.

CALLBACK REASONS

Command Entered
VAX binding: **DWT$C_CRCOMMAND_ENTERED**
C binding: **DwtCRCommandEntered**

A complete command line is ready to be executed.

Value Changed
VAX binding: **DWT$C_CRVALUE_CHANGED**
C binding: **DwtCRValueChanged**

The contents of the command line have changed.

Focus
VAX binding: **DWT$C_CRFOCUS**
C binding: **DwtCRFocus**

The command widget has received input focus.

Help Requested
VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help in the command window.

DESCRIPTION

COMMAND WINDOW CREATE creates a command window widget. The command window widget handles command line entry, command line history, and command line recall.

A command window widget can also be created with the high-level routine COMMAND WINDOW.

geometry management

The command window widget follows the same rules for geometry management as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

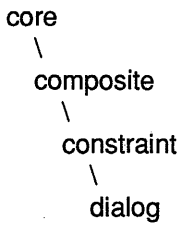
resizing

The command window widget follows the same rules for resizing as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

DIALOG BOX CREATE

Creates a dialog box widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$DIALOG_BOX_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
units	byte	uns byte	read	value
style	byte	uns byte	read	value
focus_callback	callback	uns longword	read	reference
text_merge_ translations	translations	uns longword	read	reference
margin_width	dimension	uns word	read	value
margin_height	dimension	uns word	read	value
default_position	Boolean	uns byte	read	value

Attribute	Usage	Data Type	Access	Mechanism
user_data	longword	uns longwood	read	value
child_overlap	Boolean	uns byte	read	value
resize	byte	uns byte	read	value
foreground	pixel	uns longword	read	value
highlight	pixel	uns longword	read	value
highlight_pixmap	pixmap	uns longword	read	value
direction_r_to_l	Boolean	uns longword	read	value
font	font list	uns longword	read	reference
grab_key_syms	uns longword	uns longword	read	reference
grab_merge_ translations	translations	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtDialogBoxCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtDialogBoxCreate(parent_widget, name, override_arglist,
                          override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_ arglist**:

```
unsigned char   units;
unsigned char   style;
DwtCallbackPtr  focus_callback;
XtTranslations  text_merge_translations;
Dimension       margin_width;
Dimension       margin_height;
Boolean         default_position;
Opaque          *user_data;
Boolean         child_overlap;
unsigned char   resize;
Pixel           foreground;
Pixel           highlight;
Pixmap          highlight_pixmap;
Boolean         direction_r_to_l;
DwtFontList     font;
KeySym          *grab_key_syms;
XtTranslations  grab_merge_translations;
DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created dialog box widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

units
VAX binding: **DWT$C_NUNITS**
C binding: **DwtNunits**

The type of units for **x** and **y** attributes. This attribute cannot be changed after the widget is created. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_PIXEL_UNITS	DwtPixelUnits	Pixel units
DWT$C_FONT_UNITS	DwtFontUnits	Font units (default)

style
VAX binding: **DWT$C_NSTYLE**
C binding: **DwtNstyle**

The style of the widget. This attribute cannot be changed after the widget is created. The predefined value for this attribute is as follows:

VAX	C	Description
DWT$C_WORKAREA	DwtWorkarea	Work area (default)

focus_callback
VAX binding: **DWT$C_NFOCUS_CALLBACK**
C binding: **DwtNfocusCallback**

The callback routine or routines called when the dialog box has accepted the input focus. For this routine, the callback reason is **Focus**. The default is null.

text_merge_translations

VAX binding: **DWT$C_NTEXT_MERGE_TRANSLATIONS**
C binding: **DwtNtextMergeTranslations**

The translation manager syntax to be merged with each text widget. Note that when **text_merge_translations** is changed existing widgets are unaffected. The new value for **text_merge_translations** acts only on widgets that are added after the dialog box is created. The default is null.

See the *VMS DECwindows Guide to Application Programming* for information on translation tables.

margin_width

VAX binding: **DWT$C_NMARGIN_WIDTH**
C binding: **DwtNmarginWidth**

The number of pixels between the maximum right border of a child widget window and the dialog box. The default is 1 pixel.

margin_height

VAX binding: **DWT$C_NMARGIN_HEIGHT**
C binding: **DwtNmarginHeight**

The number of pixels between the maximum bottom border of a child widget window and the dialog box. The default is 1 pixel.

default_position

VAX binding: **DWT$C_NDEFAULT_POSITION**
C binding: **DwtNdefaultPosition**

A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. The default is false.

user_data

VAX binding: **DWT$C_NUSER_DATA**
C binding: **DwtNuserData**

A longword in which the application can store any value. To retrieve the value, use the intrinsic routine GET VALUES. The default is null.

child_overlap

VAX binding: **DWT$C_NCHILD_OVERLAP**
C binding: **DwtNchildOverlap**

Controls whether the dialog box allows its children to overlap on geometry requests. If true, the dialog box allows geometry requests from its children that result in one child overlapping other children. If false, the dialog box disallows these geometry requests. The default is true.

resize

VAX binding: **DWT$C_NRESIZE**
C binding: **DwtNresize**

Controls how the dialog box resizes when its children are managed and unmanaged and on geometry requests. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_RESIZE_FIXED	DwtResizeFixed	Dialog box does not change its size when children are added or deleted, or on geometry requests from its children.
DWT$C_RESIZE_GROW_ ONLY	DwtResizeGrowOnly	Dialog box always attempts to expand as necessary when children are added or deleted or on geometry requests from its children. (default)
DWT$C_RESIZE_SHRINK_ WRAP	DwtResizeShrinkWrap	Dialog box always attempts to expand or shrink to fit its current set of managed children as children are added or deleted or on geometry requests from its children.

foreground

VAX binding: **DWT$C_NFOREGROUND**
C binding: **DwtNforeground**

The color of gadget children in the widget window. The default is the default foreground color.

highlight

VAX binding: **DWT$C_NHIGHLIGHT**
C binding: **DwtNhighlight**

The color used for highlighting gadget children. The default is the default foreground color.

highlight_pixmap

VAX binding: **DWT$C_NHIGHLIGHT_PIXMAP**
C binding: **DwtNhighlightPixmap**

The pattern and color used for highlighting gadget children. The default is null.

direction_r_to_l

VAX binding: **DWT$C_NDIRECTION_R_TO_L**
C binding: **DwtNdirectionRToL**

This attribute defines the predominant reading direction, but is not currently used by dialog box.

font

VAX binding: **DWT$C_NFONT**
C binding: **DwtNfont**

The font of the text used in gadget children. The default is the default XUI Toolkit font.

grab_key_syms
VAX binding: **DWT$C_NGRAB_KEY_SYMS**
C binding: **DwtNgrabKeySyms**

A null-terminated array of key symbols. The default array contains the Tab key symbol. The dialog box calls an Xlib routine GRAB KEY for each key symbol. GRAB KEY specifies Any Modifier for **modifiers**, Grab Mode Async for **pointer_mode**, and Grab Mode Sync for **keyboard_mode**. The GRAB KEY routine works in conjunction with the value of the **grab_merge_translations** attribute to implement moving the input focus among the dialog box children in a synchronous manner. See the *VMS DECwindows Xlib Routines Reference Manual* for more information about GRAB KEY.

This attribute cannot be changed after the widget is created.

grab_merge_translations
VAX binding: **DWT$C_NGRAB_MERGE_TRANSLATIONS**
C binding: **DwtNgrabMergeTranslations**

The parsed translation syntax to merge into the dialog box syntax to handle the key events. The syntax is merged when the dialog box is first realized. Any change made to this attribute after the dialog box is realized will not have any effect. See the *VMS DECwindows Guide to Application Programming* for information on translation tables.

The default syntax is as follows:

```
"~Shift<KeyPress>0xff09:      DWTDIMOVEFOCUSNEXT()\n\
  Shift<KeyPress>0xff09:      DWTDIMOVEFOCUSPREV()";
```

help_callback
VAX binding: **DWT$C_NHELP_CALLBACK**
C binding: **DwtNhelpCallback**

The callback routine or routines called when a user requests help. The default is null.

ATTRIBUTE EXCEPTIONS

None.

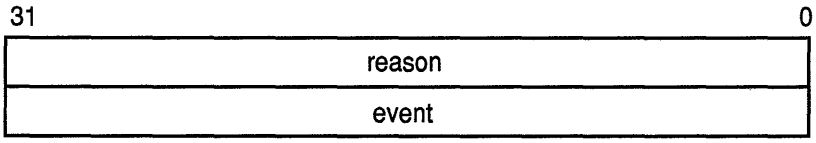
CONSTRAINT ATTRIBUTES

The following constraint attributes are passed on to any widget that is made a child of a dialog box widget. These constraint attributes are only used for dialog boxes that have the **units** attribute set to Font Units.

font_x
The placement of the left side of the widget window in font units. The default is the value of the core widget attribute **x**.

font_y
The placement of the top of the widget window in font units. The default is the value of the core widget attribute **y**.

CALLBACK DATA STRUCTURE



ZK-0091A-GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
    int     reason;
    XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Focus

VAX binding: **DWT$C_CRFOCUS**
C binding: **DwtCRFocus**

The dialog box has received input focus.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

DIALOG BOX CREATE creates a dialog box widget.

The dialog box widget is a composite widget that contains child widgets. Each child widget displays information or requests and handles input from the user. The dialog box widget functions as a container only. It

provides no input semantics over and above the expressions of the widgets it contains.

Child widgets can be positioned within the dialog box in two ways. By default, the child widgets are positioned in terms of font units (the default for **units**). The x font units are defined to be one-fourth the width of whatever font is supplied for the common attribute **font**. The y font units are defined to be one-eighth the height of whatever font is supplied for the common attribute **font**. Child widgets can also be positioned in terms of pixel units when **units** is defined as pixel units.

A dialog box widget can also be created with the high-level routine DIALOG BOX.

geometry management

The dialog box widget is a generic container that treats all of its children equally in terms of geometry management. When a child is first added to a dialog box, it is placed using the **x**, **y**, **width**, **height**, and **border_width** core attributes specified in the widget. The dialog box does not override any of the geometry of its children. The value of the **child_overlap** attribute affects how the geometry manager reacts to geometry requests from its children. If the value of **child_overlap** is true (the default), then the dialog box allows a request from a child widget, even if the request results in the child widget overlapping another child widget in the dialog box. If the value of **child_overlap** is false, the dialog box geometry manager disallows such requests.

resizing

The resizing behavior of the dialog box widget is controlled by the **resize** attribute. See the description for that attribute.

DIALOG BOX POPUP CREATE

Creates a pop-up dialog box widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
constraint
\
dialog

VAX FORMAT

widget = **DWT$DIALOG_BOX_POPUP_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
units	byte	uns byte	read	value
title	comp string	uns longword	read	reference
style	byte	uns byte	read	value
map_callback	callback	uns longword	read	reference
unmap_callback	callback	uns longword	read	reference
focus_callback	callback	uns longword	read	reference
text_merge_ translations	translations	uns longword	read	reference

Attribute	Usage	Data Type	Access	Mechanism
margin_width	dimension	uns word	read	value
margin_height	dimension	uns word	read	value
default_position	Boolean	uns byte	read	value
user_data	longword	uns longword	read	value
child_overlap	Boolean	uns byte	read	value
resize	byte	uns byte	read	value
take_focus	Boolean	uns byte	read	value
no_resize	Boolean	uns byte	read	value
auto_unmanage	Boolean	uns byte	read	value
default_button	identifier	uns longword	read	value
cancel_button	identifier	uns longword	read	value
foreground	pixel	uns longword	read	value
highlight	pixel	uns longword	read	value
highlight_pixmap	pixmap	uns longword	read	value
direction_r_to_l	Boolean	uns longword	read	value
font	font list	uns longword	read	reference
grab_key_syms	uns longword	uns longword	read	reference
grab_merge_ translations	translations	uns longword	read	reference
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtDialogBoxPopupCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtDialogBoxPopupCreate(parent_widget, name,
                               override_arglist,
                               override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_ arglist**:

```
unsigned char   units;
DwtCompString   title;
unsigned char   style;
DwtCallbackPtr  map_callback;
DwtCallbackPtr  unmap_callback;
DwtCallbackPtr  focus_callback;
XtTranslations  text_merge_translations;
Dimension       margin_width;
Dimension       margin_height;
Boolean         default_position;
Opaque          *user_data;
Boolean         child_overlap;
unsigned char   resize;
Boolean         take_focus;
Boolean         no_resize;
Boolean         auto_unmanage;
Widget          default_button;
Widget          cancel_button;
Pixel           foreground;
Pixel           highlight;
Pixmap          highlight_pixmap;
Boolean         direction_r_to_l;
DwtFontList     font;
KeySym          *grab_key_syms;
XtTranslations  grab_merge_translations;
DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

units
VAX binding: **DWT$C_NUNITS**
C binding: **DwtNunits**

The type of x and y units used when adding child widgets to the dialog box. This attribute cannot be changed after the widget is created. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_PIXEL_UNITS	DwtPixelUnits	Pixel Units
DWT$C_FONT_UNITS	DwtFontUnits	Font Units (default)

title
VAX binding: **DWT$C_NTITLE**
C binding: **DwtNtitle**

The text label for the title bar when **style** is Modeless. The default text label is the widget name. When **style** is Modal, no title bar appears.

style
VAX binding: **DWT$C_NSTYLE**
C binding: **DwtNstyle**

The style of the widget. This attribute cannot be changed after the widget is created. One of the following predefined values for this attribute must be specified in the argument list:

VAX	C	Description
DWT$C_MODAL	DwtModal	Modal type box
DWT$C_MODELESS	DwtModeless	Modeless type box (default)

map_callback
VAX binding: **DWT$C_NMAP_CALLBACK**
C binding: **DwtNmapCallback**

The callback routine or routines called when the window is about to be mapped. For this routine, the callback reason is **Map**. The default is null.

unmap_callback
VAX binding: **DWT$C_NUNMAP_CALLBACK**
C binding: **DwtNunmapCallback**

The callback routine or routines called when the window is unmapped. For this routine, the callback reason is **Unmap**. The default is null.

focus_callback
VAX binding: **DWT$C_NFOCUS_CALLBACK**
C binding: **DwtNfocusCallback**

The callback routine or routines when the dialog box has accepted the input focus. For this routine, the callback reason is **Focus**. The default is null.

text_merge_translations
VAX binding: **DWT$C_NTEXT_MERGE_TRANSLATIONS**
C binding: **DwtNtextMergeTranslations**

The translation manager syntax to be merged with each text widget. Note that when **text_merge_translations** is changed, existing widgets are unaffected. The new value for **text_merge_translations** acts only on widgets that are added after the pop-up dialog box is created. See the *VMS DECwindows Guide to Application Programming* for information on translation tables. The default is null.

margin_width
VAX binding: **DWT$C_NMARGIN_WIDTH**
C binding: **DwtNmarginWidth**

The number of pixels between the maximum right border of a child widget window and the dialog box. The default is 3 pixels.

margin_height
VAX binding: **DWT$C_NMARGIN_HEIGHT**
C binding: **DwtNmarginHeight**

The number of pixels between the maximum bottom border of a child widget window and the dialog box. The default is 3 pixels.

default_position
VAX binding: **DWT$C_NDEFAULT_POSITION**
C binding: **DwtNdefaultPosition**

A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. The default is false.

user_data
VAX binding: **DWT$C_NUSER_DATA**
C binding: **DwtNuserData**

A longword in which the application can store any value. To retrieve the value, use the intrinsic routine GET VALUES. The default is null.

child_overlap
VAX binding: **DWT$C_NCHILD_OVERLAP**
C binding: **DwtNchildOverlap**

Controls whether the dialog box allows its children to overlap on geometry requests. If true, the dialog box allows geometry requests from its children that result in one child overlapping other children. If false, the dialog box disallows these geometry requests. The default is true.

resize
VAX binding: **DWT$C_NRESIZE**
C binding: **DwtNresize**

Controls how the dialog box resizes when its children are managed and unmanaged and on geometry requests. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_RESIZE_FIXED	DwtResizeFixed	Dialog box does not change its size when children are added or deleted or on geometry requests from its children.

VAX	C	Description
DWT$C_RESIZE_GROW_ ONLY	DwtResizeGrowOnly	Dialog box always attempts to expand as necessary when children are added or deleted, or on geometry requests from its children. (default)
DWT$C_RESIZE_SHRINK_ WRAP	DwtResizeShrinkWrap	Dialog box always attempts to expand or shrink to fit its current set of managed children as children are added or deleted or on geometry requests from its children.

take_focus
VAX binding: **DWT$C_NTAKE_FOCUS**
C binding: **DwtNtakeFocus**

A Boolean attribute that specifies whether the dialog box takes the input focus when managed. If true, the dialog box takes the input focus when managed. The default is true for a modal dialog box and false for a modeless dialog box.

no_resize
VAX binding: **DWT$C_NNO_RESIZE**
C binding: **DwtNnoResize**

A Boolean attribute that controls whether or not a modeless dialog box has a window manager resize button. When true, the dialog box has no button. When false, the dialog box has a button. The default is true.

auto_unmanage
VAX binding: **DWT$C_NAUTO_UNMANAGE**
C binding: **DwtNautoUnmanage**

A Boolean attribute that specifies whether a dialog box unmanages itself when any push button is activated. This attribute applies only to modal dialog boxes. If true, the dialog box unmanages itself when any push button is activated. This attribute cannot be changed after the widget is created. The default is true.

default_button
VAX binding: **DWT$C_NDEFAULT_BUTTON**
C binding: **DwtNdefaultButton**

The identifier of the push button that is activated when the Return or Enter key is pressed. The default is null.

cancel_button
VAX binding: **DWT$C_NCANCEL_BUTTON**
C binding: **DwtNcancelButton**

The identifier of the push button that is activated when the Shift/Return keys are pressed. The default is null.

foreground
VAX binding: **DWT$C_NFOREGROUND**
C binding: **DwtNforeground**

The color of foreground objects in the widget window. The default is the default foreground color.

highlight
VAX binding: **DWT$C_NHIGHLIGHT**
C binding: **DwtNhighlight**

The color used for highlighting gadget children. The default is the default foreground color.

highlight_pixmap
VAX binding: **DWT$C_NHIGHLIGHT_PIXMAP**
C binding: **DwtNhighlightPixmap**

The pattern and color used for highlighting gadget children. The default is null.

direction_r_to_l
VAX binding: **DWT$C_NDIRECTION_R_TO_L**
C binding: **DwtNdirectionRToL**

This attribute defines the predominant reading direction, but it is not currently used by dialog box.

font
VAX binding: **DWT$C_NFONT**
C binding: **DwtNfont**

The font of the text used in gadget children. The default is the default XUI Toolkit font.

grab_key_syms
VAX binding: **DWT$C_NGRAB_KEY_SYMS**
C binding: **DwtNgrabKeySyms**

A null-terminated array of key symbols. The default array contains the Tab key symbol. The dialog box calls an Xlib routine GRAB KEY for each key symbol. GRAB KEY specifies Any Modifier for **modifiers**, Grab Mode Async for **pointer_mode**, and Grab Mode Sync for **keyboard_mode**. The GRAB KEY routine works in conjunction with the value of the **grab_merge_translations** attribute to implement moving the input focus among the dialog box children in a synchronous manner. See the *VMS DECwindows Xlib Routines Reference Manual* for more information about GRAB KEY.

This attribute cannot be changed after the widget is created.

grab_merge_translations
VAX binding: **DWT$C_NGRAB_MERGE_TRANSLATIONS**
C binding: **DwtNgrabMergeTranslations**

The parsed translation syntax to merge into the dialog box syntax to handle the key events. The syntax is merged when the dialog box is first realized. Any change made to this attribute after the dialog box is realized

will not have any effect. See the *VMS DECwindows Guide to Application Programming* for information on translation tables.

The default syntax is as follows:

```
"~Shift<KeyPress>0xff09:      DWTDIMOVEFOCUSNEXT()\n\
  Shift<KeyPress>0xff09:      DWTDIMOVEFOCUSPREV()";
```

help_callback

VAX binding: **DWT$C_NHELP_CALLBACK**
C binding: **DwtNhelpCallback**

The callback routine or routines called when a user requests help.

ATTRIBUTE EXCEPTIONS

None.

CONSTRAINT ATTRIBUTES

The following constraint attributes are passed on to any widget made a child of a dialog box widget. These constraint attributes are only used for dialog boxes that have the **units** attribute set to font units.

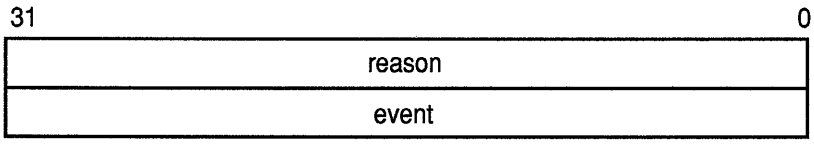
font_x

The placement of the left side of the widget window in font units. The default is the value of the core widget attribute **x**.

font_y

The placement of the top of the widget window in font units. The default is the value of the core widget attribute **y**.

CALLBACK DATA STRUCTURE



ZK-0091A-GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason
An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event
A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Map
VAX binding: **DWT$C_CRMAP**
C binding: **DwtCRMap**

The dialog box is about to be mapped.

Unmap
VAX binding: **DWT$C_CRUNMAP**
C binding: **DwtCRUnmap**

The dialog box was just unmapped.

Focus
VAX binding: **DWT$C_CRFOCUS**
C binding: **DwtCRFocus**

The dialog box has received input focus.

Help Requested
VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

DIALOG BOX POPUP CREATE creates a pop-up dialog box widget.

A pop-up dialog box widget can also be created with the high-level routine DIALOG BOX

geometry management

The pop-up dialog box widget follows the same rules for geometry management as the dialog widget, described in the low-level routine DIALOG BOX CREATE.

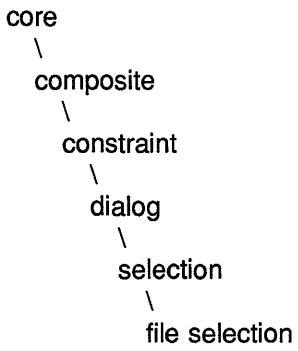
resizing

The pop-up dialog box widget follows the same rules for resizing as the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

FILE SELECTION CREATE

Creates a file selection widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$FILE_SELECTION_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
filter_label	comp string	uns longword	read	reference
apply_label	comp string	uns longword	read	reference
dir_mask	comp string	uns longword	read	reference

Attribute	Usage	Data Type	Access	Mechanism
dir_spec	comp string	uns longword	read	reference
file_search_proc	void proc	procedure entry mask	read	value
list_updated	Boolean	uns byte	read	value

MIT C FORMAT

widget = **DwtFileSelectionCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtFileSelectionCreate(parent_widget, name,
                              override_arglist,
                              override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
DwtCompString  filter_label;
DwtCompString  apply_label;
DwtCompString  dir_mask;
DwtCompString  dir_spec;
VoidProc       file_search_proc;
Boolean        list_updated;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

filter_label
VAX binding: **DWT$C_NFILTER_LABEL**
C binding: **DwtNfilterLabel**

The label for the search filter located above the text-entry field. The default is "File filter".

apply_label
VAX binding: **DWT$C_NAPPLY_LABEL**
C binding: **DwtNapplyLabel**

The label for the Apply push button, which activates new file searches. The default is "Filter".

dir_mask
VAX binding: **DWT$C_NDIR_MASK**
C binding: **DwtNdirMask**

The directory mask used in determining the files displayed in the file selection list box. The default is "*.*".

dir_spec
VAX binding: **DWT$C_NDIR_SPEC**
C binding: **DwtNdirSpec**

The full VMS file specification. This field is write only and cannot be modified using the intrinsic routine SET VALUES. The default is the null string.

file_search_proc
VAX binding: **DWT$C_NFILE_SEARCH_PROC**
C binding: **DwtNfileSearchProc**

A directory search procedure to replace the default file selection search procedure. The file selection widget's default file search procedure is written to fill the needs of most applications. However, since it is impossible to cover the requirements of all applications, the default search procedure can be replaced.

The **file_search_proc** is called with two arguments: one argument is the file selection widget; the other argument is the file selection callback structure. The callback structure contains all required information to conduct a directory search, including the current file search mask. Once called, it is up to the search routine to generate a new list of files and update the file selection widget by using the intrinsic routine SET VALUES. The following attributes must be set:

- **items**

  VAX binding: **DWT$C_NITEMS**
  C binding: **DwtNitems**

  Sets the **item** attribute described in the low-level routine SELECTION CREATE. Set the attribute to the new list of files. If there are no files, set the attribute to null.

- **items_count**

  VAX binding: **DWT$C_NITEMS_COUNT**
  C binding: **DwtNitemsCount**

Sets the **items_count** attribute described in the low-level routine SELECTION CREATE. If there are no files, set **items_count** to zero.

- **list_updated**

  Always set the **list_updated** attribute to true when updating the file list through a search procedure, even if there are no files. The default is false.

Setting the following field is optional, but recommended:

- **dir_spec**

  Set to the full file specification of the directory searched. The directory specification is displayed above the list box.

The default is the default file selection search procedure.

list_updated

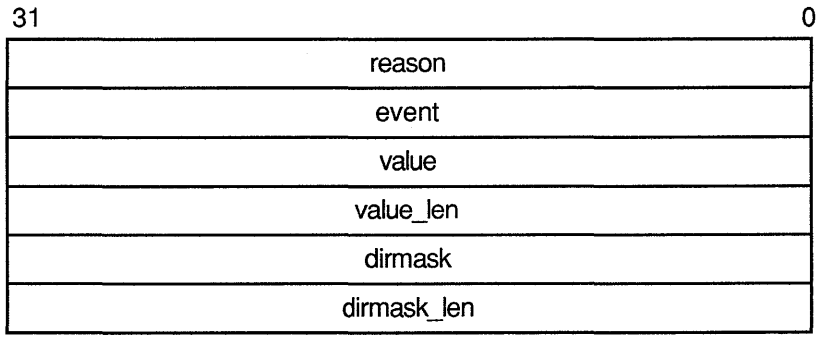
VAX binding: **DWT$C_NLIST_UPDATED**
C binding: **DwtNlistUpdated**

Set true, if the file list has been updated. This argument is set only by the file search procedure.

ATTRIBUTE EXCEPTIONS

The default for **selection_label** (located above the list box) is "Files in".

CALLBACK DATA STRUCTURE



ZK–0092A–GE

VAX field information

Structure name: DWT$FILSEL_CB_ST

Name	Usage	Data Type	Access	Mechanism
filsel_reason	callback reason	longword	read	value
filsel_event	event	uns longword	read	reference
filsel_value	comp string	uns longword	read	reference

Name	Usage	Data Type	Access	Mechanism
filsel_value_len	longword	longword	read	value
filsel_dirmask	comp string	uns longword	read	reference
filsel_dirmask_len	longword	longword	read	value

MIT C field information

```
typedef struct {
      int             reason;
      XEvent          *event;
      DwtCompString   value;
      int             value_len
      DwtCompString   dirmask;
      int             dirmask_len
} DwtFileSelectionCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason
An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event
A pointer to the X event structure describing the event that generated this callback.

value
The current selection when the callback occurred.

value_len
The length of the selection compound string.

dirmask
The current directory mask when the callback occurred.

dirmask_len
The length of the directory mask compound string.

CALLBACK REASONS

Activated
VAX binding: **DWT$C_CRACTIVATED**
C binding: **DwtCRActivated**

The user activated the OK push button.

Cancel
VAX binding: **DWT$C_CRCANCEL**
C binding: **DwtCRCancel**

The user activated the Cancel button.

Help Requested
VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help somewhere in the file selection box.

DESCRIPTION

This routine creates a file selection widget for the application to query the user for a file selection. This is a subclass of the selection widget, which is a subclass of the dialog widget. The file selection widget is a specialized pop-up dialog box, supporting either modal or modeless formats.

The file selection widget includes the following:

- A list box displaying the file names from which to choose
- A directory mask text entry field
- A selection text entry field
- An Apply push button to apply the directory mask, which generates a new list of files
- An OK push button to inform the application that a selection has been made
- A Cancel push button to inform the application that a selection has been canceled

Note that the callback data structure also includes the current values of the **value** and **dirmask** attributes. This allows user input text and directory information to be passed back to the application.

The file selection widget supports remote file searches between nodes on a network. Users can also perform remote file searches from VMS to ULTRIX systems, but currently not from ULTRIX to VMS systems.

geometry management

The file selection widget follows the same rules for geometry management as its superclass the selection widget, described in the low-level routine SELECTION CREATE.

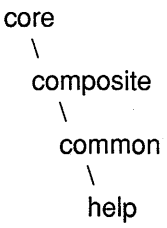
resizing

The file selection widget follows the same rules for resizing as its superclass the selection widget, described in the low-level routine SELECTION CREATE.

HELP CREATE

Creates a help widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$HELP_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Argument	Usage	Data Type	Access	Mechanism
about_label	comp string	uns longword	read	reference
add_topic_label	comp string	uns longword	read	reference
application_name	comp string	uns longword	read	reference
badframe_message	comp string	uns longword	read	reference
badlib_message	comp string	uns longword	read	reference
cols	uns longword	uns longword	read	value
copy_label	comp string	uns longword	read	reference
default_position	Boolean	uns byte	read	value

Argument	Usage	Data Type	Access	Mechanism
dismiss_label	comp string	uns longword	read	reference
edit_label	comp string	uns longword	read	reference
erroropen_ message	comp string	uns longword	read	reference
exit_label	comp string	uns longword	read	reference
file_label	comp string	uns longword	read	reference
first_topic	comp string	uns longword	read	reference
glossary_label	comp string	uns longword	read	reference
glossary_topic	comp string	uns longword	read	reference
goback_label	comp string	uns longword	read	reference
goover_label	comp string	uns longword	read	reference
goto_label	comp string	uns longword	read	reference
help_font	font list	uns longword	read	reference
help_label	comp string	uns longword	read	reference
helpmessage_title	comp string	uns longword	read	reference
helpmessage_title_ type	uns longword	uns longword	read	value
history_label	comp string	uns longword	read	reference
historybox_label	comp string	uns longword	read	reference
keyword_label	comp string	uns longword	read	reference
keywords_label	comp string	uns longword	read	reference
library_spec	comp string	uns longword	read	reference
library_type	uns longword	uns longword	read	value
nokeyword_ message	comp string	uns longword	read	reference
notitle_message	comp string	uns longword	read	reference
nulllib_message	comp string	uns longword	read	reference
nulltopic_message	comp string	uns longword	read	reference
overview_topic	comp string	uns longword	read	reference
rows	uns longword	uns longword	read	value
saveaslabel	comp string	uns longword	read	reference
searchapply_label	comp string	uns longword	read	reference
searchkeywordbox_ label	comp string	uns longword	read	reference
search_label	comp string	uns longword	read	reference
searchtitlebox_ label	comp string	uns longword	read	reference
selectall_label	comp string	uns longword	read	reference
title_label	comp string	uns longword	read	reference
titles_label	comp string	uns longword	read	reference

Argument	Usage	Data Type	Access	Mechanism
topictitles_label	comp string	uns longword	read	reference
view_label	comp string	uns longword	read	reference
visitglos_label	comp string	uns longword	read	reference
visit_label	comp string	uns longword	read	reference
unmap_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtHelpCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtHelpCreate(parent_widget, name, override_arglist,
                     override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
DwtCompString       about_label;
DwtCompString       add_topic_label;
DwtCompString       application_name;
DwtCompString       badframe_message;
DwtCompString       badlib_message;
int                 cols;
DwtCompString       copy_label;
Boolean             default_position;
DwtCompString       dismiss_label;
DwtCompString       edit_label;
DwtCompString       erroropen_message;
DwtCompString       exit_label;
DwtCompString       file_label;
DwtCompString       first_topic;
DwtCompString       glossary_label;
DwtCompString       glossary_topic;
DwtCompString       goback_label;
DwtCompString       goover_label;
DwtCompString       goto_label;
DwtFontList         help_font;
DwtCompString       help_label;
DwtCompString       helpmessage_title;
unsigned char       helpmessage_title_type;
DwtCompString       history_label;
DwtCompString       historybox_label;
DwtCompString       keyword_label;
DwtCompString       keywords_label;
DwtCompString       library_spec;
unsigned int        library_type;
DwtCompString       nokeyword_message;
DwtCompString       notitle_message;
```

```
DwtCompString       nulllib_message;
DwtCompString       nulltopic_message;
DwtCompString       overview_topic;
int                 rows;
DwtCompString       saveaslabel;
DwtCompString       searchapply_label;
DwtCompString       searchkeywordbox_label;
DwtCompString       search_label;
DwtCompString       searchtitlebox_label;
DwtCompString       selectall_label;
DwtCompString       title_label;
DwtCompString       titles_label;
DwtCompString       topictitles_label;
DwtCompString       view_label;
DwtCompString       visitglos_label;
DwtCompString       visit_label;
DwtCallbackPtr      unmap_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

about_label
VAX binding: **DWT$C_NABOUT_LABEL**
C binding: **DwtNaboutLabel**

The label for the About menu item in the Help menu. The default is "About".

add_topic_label
VAX binding: **DWT$C_NADD_TOPIC_LABEL**
C binding: **DwtNaddtopicLabel**

The text for the label indicating additional topics for help. The default is "Additional topics:".

application_name
VAX binding: **DWT$C_NAPPLICATION_NAME**
C binding: **DwtNapplicationName**

The application name to be used in the widget title bar. The default is null.

badframe_message
VAX binding: **DWT$C_NBADFRAME_MESSAGE**
C binding: **DwtNbadframeMessage**

The text for the message displayed when a frame could not be found. The default is "Couldn't find frame %s in %x library\n".

badlib_message
VAX binding: **DWT$C_NBADLIB_MESSAGE**
C binding: **DwtNbadlibMessage**

The text for the message displayed when a requested library could not be found. The default is "Couldn't open %s library\n".

cols
VAX binding: **DWT$C_NCOLS**
C binding: **DwtNcols**

The width, in characters, of the help text window. The default is language dependent; the American English default is 55.

copy_label
VAX binding: **DWT$C_NCOPY_LABEL**
C binding: **DwtNcopyLabel**

The text for the Copy menu item in the Edit menu. The default is "Copy".

default_position
VAX binding: **DWT$C_NDEFAULT_POSITION**
C binding: **DwtNdefaultPosition**

A Boolean attribute that, if true, causes the core attributes **x** and **y** to be ignored and forces the default widget position. The default widget position is centered in the parent window. If false, the specified **x** and **y** attributes are used to position the widget. The default is true.

dismiss_label
VAX binding: **DWT$C_NDISMISS_LABEL**
C binding: **DwtNdismissLabel**

The text for the push button label used to dismiss a help widget dialog box (for example, Search History, Search Title, Search Keyword boxes). The default is "Dismiss".

edit_label
VAX binding: **DWT$C_NEDIT_LABEL**
C binding: **DwtNeditLabel**

The label on the Edit pull-down menu. The default is "Edit".

erroropen_message
VAX binding: **DWT$C_NERROROPEN_MESSAGE**
C binding: **DwtNerroropenMessage**

The text for the error message displayed when a file cannot be opened. The default is "Error opening file %s\n".

exit_label
VAX binding: **DWT$C_NEXIT_LABEL**
C binding: **DwtNexitLabel**

The text for the push button and pull-down menu item that allows the user to exit from help. The default is "Exit".

file_label
VAX binding: **DWT$C_NFILE_LABEL**
C binding: **DwtNfileLabel**

The label on the File pull-down menu. The default is "File".

first_topic
VAX binding: **DWT$C_NFIRST_TOPIC**
C binding: **DwtNfirstTopic**

The first help topic to be displayed. If a null string is passed, the overview topic is displayed if provided. The overview topic is specified using the **overview_topic** attribute. If the value of **overview_topic** is null, an error message box is displayed. The default is null.

glossary_label
VAX binding: **DWT$C_NGLOSSARY_LABEL**
C binding: **DwtNglossaryLabel**

The text for the glossary menu item on the Help pull-down menu. The default is "Glossary".

glossary_topic
VAX binding: **DWT$C_NGLOSSARY_TOPIC**
C binding: **DwtNglossaryTopic**

The application glossary topic. If a null string is provided, the Visit Glossary menu item does not appear in the View pull-down menu. The default is null.

goback_label
VAX binding: **DWT$C_NGOBACK_LABEL**
C binding: **DwtNgobackLabel**

The text for the Go Back menu item on the View pull-down menu and on the push button in the help window. Clicking on this object returns the user to the previous topic displayed. The default is "Go Back".

goover_label
VAX binding: **DWT$C_NGOOVER_LABEL**
C binding: **DwtNgooverLabel**

The label for the Go to Overview item on the View pull-down menu. Clicking on this item causes the overview topic to appear in the help window. The default is "Go to Overview".

goto_label
VAX binding: **DWT$C_NGOTO_LABEL**
C binding: **DwtNgotoLabel**

The label for the Go To menu item on the View pull-down menu and on the push button in the help widget's dialog boxes. Clicking on this object after selecting a new topic displays help on the new topic in the same help window. The default is "Go To".

help_font

VAX binding: **DWT$C_NHELPFONT**
C binding: **DwtNhelpFont**

The font of the text displayed in the help widget. The default is the default help font.

help_label

VAX binding: **DWT$C_NHELP_LABEL**
C binding: **DwtNhelpLabel**

The label for the Help pull-down menu and for the Help menu item on the Help pull-down menu. The default is "Help".

help_title

VAX binding: **DWT$C_NHELPMESSAGE_TITLE**
C binding: **DwtNhelpmessageTitle**

The text for the title of the help message. The default is "Message".

helpmessage_title_type

VAX binding: **DWT$C_NHELPMESSAGE_TITLE_TYPE**
C binding: **DwtNhelpmessageTitleType**

The type of text used in the help message title. The predefined value for this attribute is as follows:

VAX	C	Description
DWT$C_CSTRNG	DwtCString	Compound string

history_label

VAX binding: **DWT$C_NHISTORY_LABEL**
C binding: **DwtNhistoryLabel**

The text for the History... menu item on the Search pull-down menu. The default is "History...".

historybox_label

VAX binding: **DWT$C_NHISTORYBOX_LABEL**
C binding: **DwtNhistoryboxLabel**

The label for the history dialog box. The default is "Help Topic History".

keyword_label

VAX binding: **DWT$C_NKEYWORD_LABEL**
C binding: **DwtNkeywordLabel**

The label for the Keyword... menu item in the Search pull-down menu. The default is "Keyword...".

keywords_label
VAX **DWT$C_NKEYWORDS_LABEL**
C binding: **DwtNkeywordsLabel**

The text for the label used in a Search Topic Keyword box to identify the text entry field. The default is "Keyword:".

library_spec
VAX binding: **DWT$C_NLIBRARY_SPEC**
C binding: **DwtNlibrarySpec**

A host system file specification that identifies the help topic library (for example, "SYS$HELP:DECW$CALENDAR" on VMS systems). The default is null.

library_type
VAX binding: **DWT$C_NLIBRARY_TYPE**
C binding: **DwtNlibraryType**

The type of help topic library specified by **library_spec**. The predefined value for this attribute follows:

VMS	C	Description
DWT$C_TEXT_LIBRARY	DwtTextLibrary	VMS help text in a VMS help library or ULTRIX help directory

nokeyword_message
VAX binding: **DWT$C_NNOKEYWORD_MESSAGE**
C binding: **DwtNnokeywordMessage**

The text for the message displayed when a requested keyword cannot be found. The default is "Couldn't find keyword %s\n".

notitle_message
VAX binding: **DWT$C_NNOTITLE_MESSAGE**
C binding: **DwtNnotitleMessage**

The text for the message displayed when a requested title cannot be found. The default is "No title to match string %s\n".

nulllib_message
VAX binding: **DWT$C_NNULLLIB_MESSAGE**
C binding: **DwtNnulllibMessage**

The text for the message displayed when no library has been specified. The default is "No library specified\n".

nulltopic_message
VAX binding: **DWT$C_NNULLTOPIC_MESSAGE**
C binding: **DwtNnulltopicMessage**

The text for the message displayed when neither a **first_topic** nor an **overview_topic** has been specified. The default is "No first topic and overview topic specified\n".

overview_topic
VAX binding: **DWT$C_NOVERVIEW_TOPIC**
C binding: **DwtNoverviewTopic**

The application overview topic. The default is null.

rows
VAX binding: **DWT$C_NROWS**
C binding: **DwtNrows**

Height, in characters, of the help text window. The default is language-dependent; the American English default is 20.

saveas_label
VAX binding: **DWT$C_NSAVEAS_LABEL**
C binding: **DwtNsaveasLabel**

The label for the Save As... item on a File pull-down menu. Clicking on this item allows a user to save the current help text in a file. A file selection dialog box is displayed. The default is "Save As...".

searchapply_label
VAX binding: **DWT$C_NSEARCHAPPLY_LABEL**
C binding: **DwtNsearchapplyLabel**

The text for the push button label used to initiate a search action in a Search dialog box. The default is "Apply".

searchkeywordbox_label
VAX binding: **DWT$C_NSEARCHKEYWORDBOX_LABEL**
C binding: **DwtNsearchkeywordboxLabel**

The text for the label used in a Search Topic Keywords dialog box. The default is "Search Topic Keywords".

search_label
VAX binding: **DWT$C_NSEARCH_LABEL**
C binding: **DwtNsearchLabel**

The label for the Search pull-down menu. The default is "Search".

searchtitlebox_label
VAX binding: **DWT$C_NSEARCHTITLEBOX_LABEL**
C binding: **DwtNsearchtitleboxLabel**

The text for the title of a Search Topic Titles box. The default is "Search Topic Titles".

selectall_label
VAX binding: **DWT$C_NSELECTALL_LABEL**
C binding: **DwtNselectallLabel**

The label for the Select All item on the Edit pull-down menu. Clicking on this item selects all the text in the work area (text widget only). The default is "Select All".

title_label
VAX binding: **DWT$C_NTITLE_LABEL**
C binding: **DwtNtitleLabel**

The label for the Title... item on the Search pull-down menu. Clicking on this entry allows a user to search for a topic by title. The default is "Title...".

titles_label
VAX binding: **DWT$C_NTITLES_LABEL**
C binding: **DwtNtitlesLabel**

The text for the label identifying the text entry field on the Search Topic Titles box. The default is "Title:".

topictitles_label
VAX binding: **DWT$C_NTOPICTITLES_LABEL**
C binding: **DwtNtopictitlesLabel**

The text for the label identifying topics found as a result of a title search in a Search Topic Titles box. The default is "Topic Titles:".

view_label
VAX binding: **DWT$C_NVIEW_LABEL**
C binding: **DwtNviewLabel**

The label for the View menu. The default is "View".

visitglos_label
VAX binding: **DWT$C_NVISITGLOS_LABEL**
C binding: **DwtNvisitglosLabel**

The label for the Visit Glossary item on the View pull-down menu. Clicking on this item causes the glossary to be displayed in a new Help window. The default is "Visit Glossary".

visit_label
VAX binding: **DWT$C_NVISIT_LABEL**
C binding: **DwtNvisitLabel**

The label for the Visit item on the View pull-down menu and the Visit push button in a dialog box. Clicking on this object causes information on a selected topic to be displayed in a new window. The default is "Visit".

unmap_callback
VAX binding: **DWT$C_NUNMAP_CALLBACK**
C binding: **DwtNunmapCallback**

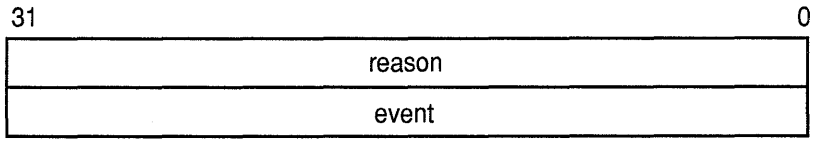
The callback routine or routines called when the help widget is unmapped. For this callback routine, the reason is **Unmap**. The default is null.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by HELP CREATE:

- The attributes **width** and **height** cannot be set by the caller; these values are calculated by the widget, based on the size of the text window (**cols** and **rows**).

CALLBACK DATA STRUCTURE



ZK–0091A–GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Unmap

VAX binding: **DWT$C_CRUNMAP**
C binding: **DwtCRUnmap**

The help window was just unmapped.

DESCRIPTION

HELP CREATE creates a help widget. A help widget is a modeless widget that enables the application to display appropriate user assistance information in response to a user request. When the user requests help, the help widget displays an initial help topic, then gives the user the ability to view additional help topics.

The **first_topic** argument allows the application to provide context-sensitive help by selecting a specific topic based on implicit or explicit cues from the user.

After the widget has been created, you can change the help topic by specifying a new **first_topic** (using the intrinsic routine SET VALUES), and then managing the widget (using the intrinsic routine MANAGE CHILD) to cause the help window to appear.

When the user exits from a help session the widget is automatically unmanaged. A Help widget can also be created with the high-level routine HELP.

geometry management

The help widget does not support children.

resizing

The help widget sizes itself at creation, based on **rows** and **cols**.

LABEL CREATE

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

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
label

VAX FORMAT

widget = **DWT$LABEL_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
label_type	uns byte	uns byte	read	value
label	comp string	uns longword	read	reference
margin_width	dimension	uns word	read	value
margin_height	dimension	uns word	read	value
alignment	uns byte	uns byte	read	value

Attribute	Usage	Data Type	Access	Mechanism
pixmap	pixmap	uns longword	read	value
margin_left	dimension	uns word	read	value
margin_right	dimension	uns word	read	value
margin_top	dimension	uns word	read	value
margin_bottom	dimension	uns word	read	value
conform_to_text	Boolean	uns byte	read	value

MIT C FORMAT

widget = **DwtLabelCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtLabelCreate(parent_widget, name, override_arglist,
                      override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
unsigned char  label_type;
DwtCompString  label;
Dimension      margin_width;
Dimension      margin_height;
unsigned char  alignment;
Pixmap         pixmap;
Dimension      margin_left;
Dimension      margin_right;
Dimension      margin_top;
Dimension      margin_bottom;
Boolean        conform_to_text;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

label_type
VAX binding: **DWT$C_NLABEL_TYPE**
C binding: **DwtNlabelType**

The label type. The predefined values for this attribute are as follows:

VMS	C	Description
DWT$C_CSTRNG	DwtCString	Compound string (default)
DWT$C_PIXMAP	DwtPixmap	Icon data in pixmap

label
VAX binding: **DWT$C_NLABEL**
C binding: **DwtNlabel**

The label for the label widget. The default is the widget name.

margin_width
VAX binding: **DWT$C_NMARGIN_WIDTH**
C binding: **DwtNmarginWidth**

The number of pixels between the border of the widget window and the label. The default is 2 pixels for text, 0 pixels for pixmap.

margin_height
VAX binding: **DWT$C_NMARGIN_HEIGHT**
C binding: **DwtNmarginHeight**

The number of pixels between the border of the widget window and the label. The default is 2 pixels for text, 0 pixels for pixmap.

alignment
VAX binding: **DWT$C_NALIGNMENT**
C binding: **DwtNalignment**

The label alignment for text style. The predefined values for this attribute are as follows:

VMS	C	Description
DWT$C_ALIGNMENT_CENTER	DwtAlignmentCenter	Center alignment (default)
DWT$C_ALIGNMENT_ BEGINNING	DwtAlignmentBeginning	Alignment at the beginning
DWT$C_ALIGNMENT_END	DwtAlignmentEnd	Alignment at the end

pixmap
VAX binding: **DWT$C_NPIXMAP**
C binding: **DwtNpixmap**

Icon data for the label. A pixmap is used when **label_type** is defined as pixmap. The default is null.

margin_left

VAX binding: **DWT$C_NMARGIN_LEFT**
C binding: **DwtNmarginLeft**

The number of pixels to remain inside the left margin (**margin_width**) of the widget before the label is drawn. The default is zero.

margin_right

VAX binding: **DWT$C_NMARGIN_RIGHT**
C binding: **DwtNmarginRight**

The number of pixels to remain inside the right margin (**margin_width**) of the widget before the label is drawn. The default is zero.

margin_top

VAX binding: **DWT$C_NMARGIN_TOP**
C binding: **DwtNmarginTop**

The number of pixels to remain inside the top margin (**margin_height**) of the widget before the label is drawn. The default is zero.

margin_bottom

VAX binding: **DWT$C_NMARGIN_BOTTOM**
C binding: **DwtNmarginBottom**

The number of pixels to remain inside the bottom margin (**margin_height**) of the widget before the label is drawn. The default is zero.

conform_to_text

VAX binding: **DWT$C_NCONFORM_TO_TEXT**
C binding: **DwtNconformToText**

A Boolean attribute that specifies whether the widget resizes to contain the label. When true, an intrinsic routine SET VALUES with a new label string causes the widget to shrink or expand to fit exactly (accounting for margins) the new label string. Note that the results of the attempted resize are up to the geometry manager. When false, the widget never resizes on its own.

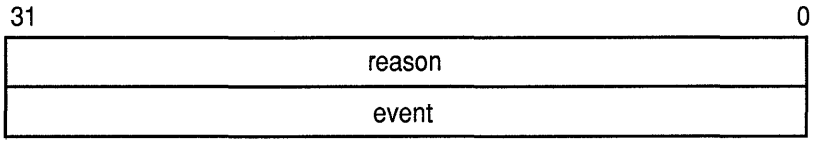
The default is true if the widget is created with width and height of zero; the default is false if the widget is created with nonzero width and height.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by LABEL CREATE:

- The default for **width** is the width of the label or pixmap, plus two times **margin_width**.
- The default for **height** is the height of the label or pixmap, plus two times **margin_height**.
- The default for **border_width** is zero.

CALLBACK DATA STRUCTURE



ZK-0091A-GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

LABEL CREATE creates a label widget for the application to display read-only information anywhere within the parent widget window. A label widget can also be created with the high-level routine LABEL.

geometry management

Because a label widget does not support children, it always refuses geometry requests.

resizing

By default the label widget sizes itself to be just large enough to display the label.

LIST BOX CREATE

Creates a list box widget.

WIDGET CLASS HIERARCHY

core
  \
  composite
    \
    common
      \
      scroll window
        \
        list box

VAX FORMAT

widget = **DWT$LIST_BOX_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
margin_width	dimension	uns word	read	value
margin_height	dimension	uns word	read	value
spacing	dimension	uns word	read	value
items	array	uns longword	read	reference
item_count	longword	uns longword	read	value
selected_items	array	uns longword	read	reference

Attribute	Usage	Data Type	Access	Mechanism
selected_items_ count	uns longword	uns longword	read	value
visible_item_count	longword	uns longword	read	value
single_selection	Boolean	uns byte	read	value
resize	byte	uns byte	read	value
horiz	Boolean	uns byte	read	value
single_callback	callback	uns longword	read	reference
single_confirm_ callback	callback	uns longword	read	reference
extend_callback	callback	uns longword	read	reference
extend_confirm_ callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtListBoxCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtListBoxCreate(parent_widget, name, override_arglist,
                        override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_ arglist**:

```
Dimension        margin_width;
Dimension        margin_height;
Dimension        spacing;
DwtCompString    *items;
int              item_count;
DwtCompString    *selected_items;
int              selected_items_count;
int              visible_item_count;
Boolean          single_selection;
unsigned char    resize;
Boolean          horiz;
DwtCallbackPtr   single_callback;
DwtCallbackPtr   single_confirm_callback;
DwtCallbackPtr   extend_callback;
DwtCallbackPtr   extend_confirm_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

margin_width
VAX binding: **DWT$C_NMARGIN_WIDTH**
C binding: **DwtNmarginWidth**

The number of pixels between the border of the widget window and the items. This sets the list box menu margin width. The default is 10 pixels.

margin_height
VAX binding: **DWT$C_NMARGIN_HEIGHT**
C binding: **DwtNmarginHeight**

The number of pixels between characters of each pair of consecutive items. This sets the list box menu margin height. The default is 4 pixels.

spacing
VAX binding: **DWT$C_NSPACING**
C binding: **DwtNspacing**

The spacing, in pixels, between list box entries. The default is 1 pixel.

items
VAX binding: **DWT$C_NITEMS**
C binding: **DwtNitems**

The list of items to be displayed by the list box widget. Each item in the list must be unique. No multiline items are allowed. When modifying **items**, always update **item_count** and **selected_items_count**. When **items** is null, **item_count** and **selected_items_count** must be zero. The default is null.

item_count
VAX binding: **DWT$C_NITEMS_COUNT**
C binding: **DwtNitemsCount**

The number of items in **items**. When **item_count** is zero, **items** does not have to be null. The list box widget uses **item_count** and **selected_items_count**, not **items**, to determine if the list contains any items. Therefore, **item_count** must be specified whenever **items** is modified. The default is zero.

selected_items
VAX binding: **DWT$C_NSELECTED_ITEMS**
C binding: **DwtNselectedItems**

The items that are selected in the list box. The last selected item is visible in the list box. The default is null.

selected_items_count
VAX binding: **DWT$C_NSELECTED_ITEMS_COUNT**
C binding: **DwtNselectedItemsCount**

The number of items selected in the list box. When **selected_items_count** is zero, **selected_items** does not have to be null. The list box uses **selected_items_count** not **selected_items** to determine if the list contains any selected items. Therefore, **selected_items_count** must be specified whenever **selected_items** is modified. The default is zero.

visible_item_count
VAX binding: **DWT$C_NVISIBLE_ITEMS_COUNT**
C binding: **DwtNvisibleItemsCount**

The number of visible items. The default is as many items as can fit in the core attribute **height**. The minimum is 1 item.

The list box widget is designed so that its height is based on the **visible_item_count** attribute. Therefore, it is preferable to control the list box height by using **visible_item_count** rather than **height**.

Applications that control list box height through **height** are responsible for handling font changes.

single_selection
VAX binding: **DWT$C_NSINGLE_SELECTION**
C binding: **DwtNsingleSelection**

A Boolean attribute that is true if only one item can be selected at a time. The default is true.

resize
VAX binding: **DWT$C_NRESIZE**
C binding: **DwtNresize**

Controls how the list box resizes when its children are managed and unmanaged and on geometry requests. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_RESIZE_FIXED	DwtResizeFixed	List box does not change its size when items are added or deleted.
DWT$C_RESIZE_GROW_ONLY	DwtResizeGrowOnly	List box attempts to expand as necessary when items are added. (default)

If **resize** is set to Fixed, DIGITAL recommends that **horiz** be set true.

horiz

VAX binding: **DWT$C_NHORIZONTAL**
C binding: **DwtNhorizontal**

A Boolean attribute set true if the list box contains a horizontal scroll bar. This field cannot be changed after widget creation. The default is false.

single_callback

VAX binding: **DWT$C_NSINGLE_CALLBACK**
C binding: **DwtNsingleCallback**

The callback routine or routines called by the application when the user has clicked MB1 on a single item. The default is null.

single_confirm_callback

VAX binding: **DWT$C_NSINGLE_CONFIRM_CALLBACK**
C binding: **DwtNsingleConfirmCallback**

The callback routine or routines called by the application when the user has double clicked MB1 on a single item. The default is null.

extend_callback

VAX binding: **DWT$C_NEXTEND_CALLBACK**
C binding: **DwtNextendCallback**

The callback routine or routines called by the application when the user has clicked MB1 while pressing the Shift key when more than one item is selected (multiple selection). See the description for the **single_selection** attribute. The default is null.

extend_confirm_callback

VAX binding: **DWT$C_NEXTEND_CONFIRM_CALLBACK**
C binding: **DwtNextendConfirmCallback**

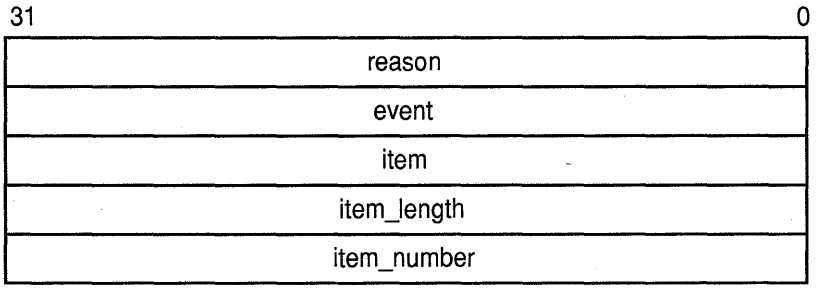
The callback routine or routines called by the application when the user double clicks MB1 while pressing the Shift key when more than one item is selected (multiple selection). See the description for the **single_selection** attribute. The default is null.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by LIST BOX CREATE:

- The default for **width** is as large as necessary to hold the longest item without exceeding the size of its parent.
- The default for **height** is as large as necessary to hold the number of items specified by **visible_item_count**, without exceeding the size of the parent widget.
- The default for the SCROLL WINDOW CREATE attribute **shown_value_automatic_vert** is set to false.

CALLBACK DATA STRUCTURE



ZK-0093A-GE

VAX field information

Structure name: DWT$LISTBOX_CB_ST

Name	Usage	Data Type	Access	Mechanism
listbox_reason	callback reason	longword	read	value
listbox_event	event	uns longword	read	reference
listbox_item	comp string	uns longword	read	reference
listbox_item_length	longword	longword	read	value
listbox_item_number	longword	longword	read	value

MIT C field information

```
typedef struct {
      int             reason;
      XEvent          *event;
      DwtCompString   item;
      int             item_length;
      int             item_number;
} DwtListBoxCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

item

The last item selected when the callback occurred. Note that only the last selected item, not all selected items, is returned.

item_length

The length of the selected item when the callback occurred.

item_number

The position of the item in the list box when the callback occurred. The first position is 1.

CALLBACK REASONS

Single

VAX binding: **DWT$C_CRSINGLE**
C binding: **DwtCRSingle**

The user selected a single item in the list by clicking MB1 on the item.

Single Confirm

VAX binding: **DWT$C_CRSINGLE_CONFIRM**
C binding: **DwtCRSingleConfirm**

The user selected a single item in the list and confirmed another action to be taken (by a callback) by double clicking on an item. For example, a double click on a file in the file selection box selects that file and confirms another action to be taken.

Extend

VAX binding: **DWT$C_CREXTEND**
C binding: **DwtCRExtend**

The user selected an item (by clicking MB1 on a single item while pressing the Shift key) while there is at least one other selected item.

Extend Confirm

VAX binding: **DWT$C_CREXTEND_CONFIRM**
C binding: **DwtCRExtendConfirm**

The user selected an item and confirmed another action to be taken (by double clicking MB1 on a single item while pressing the Shift key) while there is at least one other selected item. This reason applies only if **single_selection** is false. The default is true.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

LIST BOX CREATE creates a list box widget. The list box widget is a composite widget comprising a list box, a menu with gadgets, and scroll bars.

A list box widget can also be created with the high-level routine LIST BOX.

geometry management

The list box widget does not support children.

resizing

The size of the list box is determined by the following attributes in descending precedence:

- **height** and **width**
- **visible_item_count**
- **resize**

Setting the common attributes **height** and **width** overrides the widget-specific default settings. The following paragraphs describe the sizing option.

The default list box height is determined by **visible_item_count**. Once set, the list box height does not change, regardless of the number of items the list box actually contains, unless the common attribute **height** or **visible_item_count** is modified. It is recommended that you control list box height by setting **visible_item_count** rather than **height**.

The default list box width is controlled by the **resize** attribute. By default **resize** is true, and the list box increases its width to accommodate items wider than its current width. However, the list box does not shrink if wider items are removed.

To keep the width of the list box constant, set **width** to the desired width and set **resize** to false. Also set **horiz** to true so that users can scroll the item list horizontally to see items wider than the list box.

MAIN WINDOW CREATE

Creates a main window widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
main window

VAX FORMAT

widget = **DWT$MAIN_WINDOW_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attributes	Usage	Data Type	Access	Mechanism
command_window	widget	uns longword	read	value
work_window	widget	uns longword	read	value
menu_bar	widget	uns longword	read	value
horizontal_scroll_bar	widget	uns longword	read	value
vertical_scroll_bar	widget	uns longword	read	value
accept_focus	Boolean	uns byte	read	value
focus_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtMainWindowCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtMainWindowCreate(parent_widget, name, override_arglist,
                           override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
Widget          command_window;
Widget          work_window;
Widget          menu_bar;
Widget          horizontal_scroll_bar;
Widget          vertical_scroll_bar;
Boolean         accept_focus;
DwtCallbackPtr  focus_callback;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget. For some applications, the parent widget identifier for the main window widget is the identifier returned by the intrinsic routine INITIALIZE. However, the main window widget is not restricted to this type of parent.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

command_window

VAX binding: **DWT$C_NCOMMAND_WINDOW**
C binding: **DwtNcommandWindow**

The widget identifier for the command window to be associated with the main window widget. This cannot be supplied at main window creation time, but can be set or specified after creation. The default is null.

work_window
VAX binding: **DWT$C_NWORK_WINDOW**
C binding: **DwtNworkWindow**

The widget identifier for the work window to be associated with the main window widget. This cannot be supplied at main window creation time, but can be set or specified after creation. The default is null.

menu_bar
VAX binding: **DWT$C_NMENU_BAR**
C binding: **DwtNmenuBar**

The widget identifier for the menu bar to be associated with the main window widget. This cannot be supplied at main window creation time, but can be set or specified after creation. The default is null.

horizontal_scroll_bar
VAX binding: **DWT$C_NHORIZONTAL_SCROLL_BAR**
C binding: **DwtNhorizontalScrollBar**

The widget identifier for the scroll bar to be used as the horizontal scroll bar in the main window widget. This cannot be supplied at main window creation time, but can be set or specified after creation. The default is null.

vertical_scroll_bar
VAX binding: **DWT$C_NVERTICAL_SCROLL_BAR**
C binding: **DwtNverticalScrollBar**

The widget identifier for the scroll bar widget to be used as the vertical scroll bar in the main window widget. This cannot be supplied at main window creation time, but can be set or specified after creation. The default is null.

accept_focus
VAX binding: **DWT$C_NACCEPT_FOCUS**
C binding: **DwtNacceptFocus**

A Boolean attribute that specifies whether the main window widget accepts the input focus. When the main window widget is asked to accept the input focus, it attempts to give the input focus first to the work window, and then to the command window. If neither the work window nor the command window accepts the input focus and **accept_focus** is true, the main window widget accepts the input focus itself. If false, the main window widget does not accept the input focus. The default is false.

focus_callback
VAX binding: **DWT$C_NFOCUS_CALLBACK**
C binding: **DwtNfocusCallback**

The callback routine or routines called when the main window has accepted the input focus. For this routine, the callback reason is **Focus**. The default is null.

ATTRIBUTE EXCEPTIONS

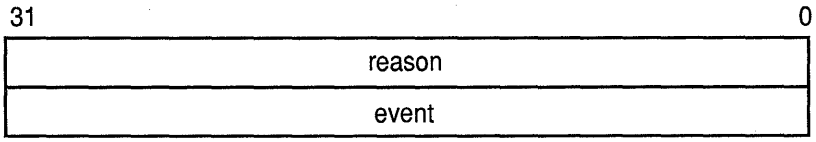
The following common attributes are not supported by MAIN WINDOW CREATE:

- **font**
- **highlight**
- **highlight_pixmap**

The following common attributes are supported differently by MAIN WINDOW CREATE:

- The default value of the common attributes **width** and **height** is 5 pixels.

CALLBACK DATA STRUCTURE



ZK–0091A–GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Focus

VAX binding: **DWT$C_CRFOCUS**
C binding: **DwtCRFocus**

The main window widget has received the input focus.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

MAIN WINDOW CREATE creates a main window widget. It can contain a menu bar region, a work area with optional scroll bars, and a command area.

A main window widget can also be created with the high-level routine MAIN WINDOW.

geometry management

The main window widget tiles the insides of its window with up to five children, as follows:

- The child widget designated as the menu bar widget is placed at the top and extends all the way across the main window. The height of the menu bar widget is set to whatever the menu requests.
- The child widget designated as the command widget is placed at the bottom and extends all the way across the main window. The height of the command widget is not altered.
- The child widget designated as the horizontal scroll bar widget is placed just above the command widget (if there is no command widget, it is placed at the bottom). The width of the horizontal scroll bar is the width of the main window minus the width of the vertical scroll bar. The height is not altered.
- The child widget designated as the vertical scroll bar widget is placed on the right edge below the menu bar. The height is the distance between the bottom of the menu bar and the top of the horizontal scroll bar. The width is not altered.
- The child widget designated as the work area fills the area under the menu bar, to the left of the vertical scroll bar and above the horizontal scroll bar. Both dimensions are altered.

Note that designating which child fills which role can be accomplished in three ways:

- Use the high-level routine MAIN WINDOW SET AREAS to explictly designate which child is which.
- Call the intrinsic routine SET VALUES and set the area attributes with the appropriate widget identifiers.

- Let the main window widget determine which child is which, by using the following algorithm. Only currently managed children are eligible to be designated for a role, as follows:
  - — A child of menu widget class (or subclass) is assumed to be the menu bar.
  - — A child of command widget class (or subclass) is assumed to be the command widget.
  - — A child of scroll widget class (or subclass) is either the horizontal or vertical scroll bar, which is determined by looking at the **orientation** attribute of the child.
  - — A child of any other class is assumed to be the work area.

For many applications, using the intrinsic routines SET AREAS or SET VALUES is redundant. A single main widget might have a number of menu bars as children. By managing and unmanaging the menu bar children appropriately, the application can switch between menu bars without using SET AREAS or SET VALUES.

The size of the main window widget can be specified in two ways:

- Specifying a nonzero height and width at widget creation time. In this case, the main window widget does not change its size on a geometry request from one of its children.
- Specifying zero for both height and width at widget creation time. In this case, the main window widget uses the width and height of the widget designated as the work area widget in determining its width and height. The main window widget does not alter the width and height of the work area widget and places the remaining widgets based upon the size of the work area. The work area widget can later request a size change. The main window widget will honor the request and reconfigure its size.

As a geometry manager, the main window widget allows the following requests:

- The menu bar, command window, and horizontal scroll bar can change height.
- The vertical scroll bar can change width.
- The work area can change width and height, if the main menu widget was created with a width and height of zero.

---

resizing

When resized, the main window widget reformats itself as described previously.

MENU BAR CREATE

Creates a menu bar widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
menu bar

VAX FORMAT

widget = **DWT$MENU_BAR_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The widget-specific attributes described in the low-level routine MENU CREATE can be set in the **override_arglist**.

MIT C FORMAT

widget = **DwtMenuBarCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtMenuBarCreate(parent_widget, name, override_arglist,
                        overide_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

The widget-specific attributes described in the low-level routine MENU CREATE can be set in the **override_arglist**.

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

See the widget-specific attributes for the low-level routine MENU CREATE.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by MENU BAR CREATE:

- Setting the sensitivity of the menu bar causes all widgets contained in that menu bar to be set to the same sensitivity as the menu bar.
- The default of **width** is 16 pixels.
- The default of **height** is the number of lines needed to display all entries.
- The **Font** attribute is used only by gadget children.

The following attribute from MENU CREATE is supported differently:

- The default for **margin_height** is 0 pixels.

CALLBACK DATA STRUCTURE

See the low-level routine MENU CREATE.

DESCRIPTION

MENU BAR CREATE creates a menu bar widget. A menu bar widget is a composite widget that contains pull-down menu entry subwidgets. The subwidgets handle most of the I/O activity that displays information and queries the user for input. The menu bar provides no input semantics over and above the semantics of its subwidgets.

If the menu bar does not have enough room to fit all its subwidgets on a single line, the menu bar attempts to wrap the remaining entries onto additional lines (if allowed by the geometry manager of the parent widget).

The menu bar widget works with widget classes: pull-down menu entries, labels, and separators. If the menu attribute **entry_callback** is not null when it is activated, all subwidgets call back to this callback. Otherwise, activation callbacks are handled by the individual subwidgets.

A menu bar widget can also be created with the high-level routine MENU BAR.

geometry management

See the low-level routine MENU CREATE.

resizing

See the low-level routine MENU CREATE.

MENU CREATE

Creates a menu widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
menu

VAX FORMAT

widget = **DWT$MENU_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Argument	Usage	Data Type	Access	Mechanism
orientation	uns byte	uns longword	read	value
spacing	dimension	uns word	read	value
adjust_margin	Boolean	uns byte	read	value
margin_width	dimension	uns word	read	value
margin_height	dimension	uns word	read	value
entry_border	dimension	uns word	read	value
menu_alignment	Boolean	uns byte	read	value
entry_alignment	alignment	uns longword	read	value

Argument	Usage	Data Type	Access	Mechanism
menu_packing	uns byte	uns longword	read	value
menu_num_ columns	uns longword	uns longword	read	value
menu_radio	Boolean	uns byte	read	value
radio_always_one	Boolean	uns byte	read	value
menu_is_ homogeneous	Boolean	uns byte	read	value
menu_entry_class	widget class	identifier	read	value
menu_history	widget	uns longword	read	value
help_callback	callback	uns longword	read	reference
entry_callback	callback	uns longword	read	reference
map_callback	callback	uns longword	read	reference
unmap_callback	callback	uns longword	read	reference
menu_help_widget	widget	uns longword	read	value

MIT C FORMAT

widget = *DwtMenuCreate (parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtMenuCreate(parent_widget, name, override_arglist,
                     override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_ arglist**:

```
unsigned char   orientation;
Dimension       spacing;
Boolean         adjust_margin;
Dimension       margin_width;
Dimension       margin_height;
Dimension       entry_border;
```

```
Boolean          menu_alignment;
unsigned char    entry_alignment;
unsigned char    menu_packing;
int              menu_num_columns;
Boolean          menu_radio;
Boolean          radio_always_one;
Boolean          menu_is_homogeneous;
WidgetClass      menu_entry_class;
Widget           menu_history
DwtCallbackPtr   help_callback;
DwtCallbackPtr   entry_callback;
DwtCallbackPtr   map_callback;
DwtCallbackPtr   unmap_callback;
Widget           menu_help_widget;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

orientation
VAX binding: **DWT$C_NORIENTATION**
C binding: **DwtNorientation**

The menu orientation. The predefined values for this attribute are as follows:

VMS	C	Description
DWT$C_ORIENTATION_HORIZONTAL	DwtOrientationHorizontal	Horizontal menu
DWT$C_ORIENTATION_VERTICAL	DwtOrientationVertical	Vertical menu

spacing
VAX binding: **DWT$C_NSPACING**
C binding: **DwtNspacing**

The spacing, in pixels, between menu bar entry windows. The default is 0 pixels.

adjust_margin

VAX binding: **DWT$C_NADJUST_MARGIN**
C binding: **DwtNadjustMargin**

A Boolean attribute that indicates whether the inner minor dimension margins of all entries should be set to the same value. The default is true.

All label subclass widgets have two types of margins. The two outer margins (**margin_width** and **margin_height**) are symmetrical about the center of the widget. The attribute **margin_width** specifies the number of blank pixels to the right and to the left of the widget. The four inner margins (**margin_left**, **margin_right**, **margin_top**, and **margin_bottom**) specify the number of blank pixels to leave inside the outer margins.

The outer margins are used to accommodate features like border highlighting of widgets. The inner margins are used to accommodate features like pull-down widget hot spots and toggle button indicators.

If this attribute is true, all entries in a given column or row have exactly the same minor dimension margins. (If **orientation** is horizontal, the minor dimension is vertical; if **orientation** is vertical, the minor dimension is horizontal.) All margins have the value of the largest individual margin in the group. This aligns the left edge of text regardless of whether some entries have toggle indicators.

The default is true.

margin_width

VAX binding: **DWT$C_NMARGIN_WIDTH**
C binding: **DwtNmarginWidth**

The number of blank pixels remaining around the menu entries. The width is the number of blank pixels between the left and right edges of the menu and the border of the entries. The default is 3.

margin_height

VAX binding: **DWT$C_NMARGIN_HEIGHT**
C binding: **DwtNmarginHeight**

The number of blank pixels remaining around the menu entries. The height is the number of blank pixels above the first entry and below the last entry (for vertical menus). The default is 3.

entry_border

VAX binding: **DWT$C_NENTRY_BORDER**
C binding: **DwtNentryBorder**

The border width, in pixels, of menu bar entry windows. The default is 0 pixels.

menu_alignment

VAX binding: **DWT$C_NMENU_ALIGNMENT**
C binding: **DwtNmenuAlignment**

A Boolean attribute that specifies whether the menu entries should all have the same text alignment. If true, all entries are aligned. If false, entry alignment is unchanged. This applies only to subclasses of labelwidgetclass. The default is true.

entry_alignment

VAX binding: **DWT$C_NENTRY_ALIGNMENT**
C binding: **DwtNentryAlignment**

The type of label alignment that is enforced for all entries when **menu_alignment** is true. The predefined values for this attribute are as follows:

VMS	C	Description
DWT$C_ALIGNMENT_CENTER	DwtAlignmentCenter	Center alignment
DWT$C_ALIGNMENT_BEGINNING	DwtAlignmentBeginning	Alignment at the beginning (default)
DWT$C_ALIGNMENT_END	DwtAlignmentEnd	Alignment at the end

menu_packing

VAX binding: **DWT$C_NMENU_PACKING**
C binding: **DwtNmenuPacking**

The placement of menu entries into the menu. The value of **orientation** determines the major dimension. The predefined values for this attribute are as follows:

VMS	C	Description
DWT$C_MENU_PACKING_TIGHT	DwtNmenuPackingTight	Given the current major dimension of the menu, entries are placed one after the other until the menu must wrap. When the menu wraps, it extends in the minor dimension as many times as required. Each entry's major dimension is unchanged; its minor dimension is set to the same value as the longest entry in that particular row or column. Note that the minor dimension of any particular row or column is independent of the minor dimension of other rows or columns.
DWT$C_MENU_PACKING_COLUMN	DwtNmenuPackingColumn	All entries are placed in identically sized boxes based on the size of the largest entry. The value of **menu_num_columns** determines how many boxes are placed in the major dimension before extending in the minor dimension.
DWT$C_MENU_PACKING_NONE	DwtNmenuPackingNone	No packing is performed. The **x** and **y** attributes of each entry are left alone. The menu attempts to become large enough to enclose all entries.

The default is Menu Packing Tight for all menu types except for radio boxes. Radio box menus default to Menu Packing Column.

menu_num_columns

VAX binding: **DWT$C_NMENU_NUM_COLUMNS**
C binding: **DwtNmenu_num_columns**

The number of minor dimension extensions that can be made to accommodate the entries. This attribute is only used if the **menu_packing** attribute has the value Menu Packing Column. The default is 1.

For menus with vertical orientation, this attribute indicates how many columns are built. The number of entries per column is adjusted to maintain this number of columns (if possible). For menus with horizontal orientation, this attribute indicates how many rows are built.

menu_radio

VAX binding: **DWT$C_NMENU_RADIO**
C binding: **DwtNmenuRadio**

Indicates whether or not radio button exclusivity is enforced. If true, when one button is on and another button is turned on, the first one is turned off automatically.

The default is false except for radio box menus, which default to true.

radio_always_one

VAX binding: **DWT$C_NRADIO_ALWAYS_ONE**
C binding: **DwtNradioAlwaysOne**

Indicates if the radio button exclusivity also ensures that one button must always be on. If true, when the only radio button on is turned off, it is automatically turned back on.

The default is true. Note that this attribute has no effect unless **menu_radio** is true.

menu_is_homogeneous

VAX binding: **DWT$C_NMENU_IS_HOMOGENEOUS**
C binding: **DwtNmenuIsHomogeneous**

A Boolean attribute that indicates if the menu enforces exact homogeneity among the children of this menu. If true, then only widgets from the widget class specified in **menu_entry_class** (not subclass but exact class) are allowed as children of the menu.

The default is false, except for radio boxes which default to true.

menu_entry_class

VAX binding: **DWT$C_NMENU_ENTRY_CLASS**
C binding: **DwtNmenuEntryClass**

The class of child widgets that can be added to the menu. If **menu_is_homogeneous** is true, only the class of widgets specified by this attribute widgets can be added to the menu. All other classes of widgets produce a warning message.

The default is null, except for radio box menus which default to the *togglebuttonwidgetclass*.

menu_history

VAX binding: **DWT$C_NMENU_HISTORY**
C binding: **DwtNmenuHistory**

The widget identifier of the last menu entry which was activated. If **menu_radio** is true, **menu_history** holds the widget identifier of the last toggle button to change from off to on. This attribute can be set to pre-condition option menus and pop-up menu mouse memory. The default is null.

help_callback
VAX binding: **DWT$C_NHELP_CALLBACK**
C binding: **DwtNhelpCallback**

The callback routine or routines to be called back on a help request. The default is null.

entry_callback
VAX binding: **DWT$C_NENTRY_CALLBACK**
C binding: **DwtNentryCallback**

If this callback is defined, all menu entry activation callbacks are revectored to call back through this callback. If this callback is null, then individual menu entry callbacks work as usual. The reason for this callback is **Activate**. The default is null.

map_callback
VAX binding: **DWT$C_NMAP_CALLBACK**
C binding: **DwtNmapCallback**

The callback routine or routines called when the window is about to be mapped. For this routine, the reason is **Map**. The default is null.

unmap_callback
VAX binding: **DWT$C_NUNMAP_CALLBACK**
C binding: **DwtNunmapCallback**

The callback routine or routines called when the window is unmapped. For this routine, the reason is **Unmap**. The default is null.

menu_help_widget
VAX binding: **DWT$C_NMENU_HELP_WIDGET**
C binding: **DwtNmenuHelpWidget**

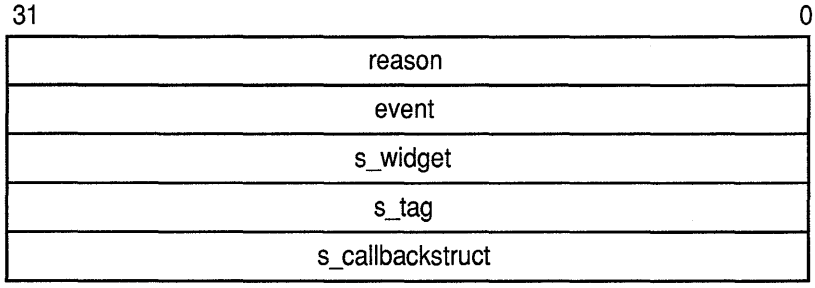
If not null, the help widget points to the menu item to be placed in the bottom right corner of the menu bar. The default is null.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by MENU CREATE:

- The **height** and **width** attributes depend on the **orientation**, the number of managed children, and the value of **menu_packing**.
- The default for **border_width** is 1 pixel.

CALLBACK DATA STRUCTURE



ZK-0094A-GE

VAX field information

Structure name: DWT$MENU_CB_ST

Name	Usage	Data Type	Access	Mechanism
menu_reason	callback reason	longword	read	value
menu_event	event	uns longword	read	reference
menu_s_widget	widget	longword	read	reference
menu_s_tag	longword	uns longword	read	value
menu_s_callbackstruct	callback	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
      Widget  s_widget;
      char    *s_tag;
      char    *s_callbackstruct;
} DwtMenuCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

s_widget

The identifier of the activating subwidget.

s_tag
The tag supplied by the application programmer when the subwidget callback routine was specified.

s_callbackstruct
The callback structure of the subwidget.

CALLBACK REASONS

Activate
VAX binding: **DWT$C_CRACTIVATE**
C binding: **DwtCRActivate**

The user selected a menu entry.

Map
VAX binding: **DWT$C_CRMAP**
C binding: **DwtCRMap**

The menu window is about to be mapped.

Unmap
VAX binding: **DWT$C_CRUNMAP**
C binding: **DwtCRUnmap**

The menu window was just unmapped.

Help Requested
VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

MENU CREATE creates a menu widget.

The menu widget is a composite widget that contains other widgets (push buttons, pull-down menu widgets, toggle buttons, labels, and separators). The subwidgets handle most I/O that display information and query the user for input. The menu widget provides no input semantics over and above the semantics of its subwidgets.

The menu widget works with widget subclasses: push buttons, toggle buttons, pull-down menu entries, labels, and separators.

A menu widget can also be created with the high-level routine MENU.

geometry management

In general, a menu enforces positions, dimensions, and border widths for all children. In a vertical menu, entries have uniform widths—the width of the widest item in the current column. The height of each entry is unaffected and is the responsibility of the item itself. In a horizontal menu, items have uniform height; the width is unaffected.

In all menu packing modes except Menu Packing None, position of an item is completely determined by the menu; the child widget has no control of its position. In the None packing mode, the menu does not position items.

A menu complies with all geometry requests made by its children. The menu determines the size needed to resize around its managed children and then makes the request of its geometry manager. Even if the menu's parent does not allow the request, the menu resizes. The child may then be clipped.

Height and width of a menu child are jointly controlled by the menu and the child. If a child requests a larger size, the menu accepts the request and then resizes all the other children to match. If a child requests a smaller size, the menu accepts the request; however, the menu might make the child bigger again as the menu resizes the other children.

If **menu_uniform_border** is true, all entries have exactly the same border width. If **menu_uniform_border** is false, the menu does not change any of the children's border widths.

resizing

When a menu widget is resized, it places all its managed children in exactly the same manner as described in the geometry management section.

MENU POPUP CREATE

Creates a pop-up menu widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
menu popup

FORMAT

widget = **DWT$MENU_POPUP_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

The widget-specific attributes described in the low-level routine MENU CREATE can be set in the **override_arglist**.

MIT C FORMAT

widget = **DwtMenuPopupCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtMenuPopupCreate(parent_widget, name, override_arglist,
                          override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

The widget-specific attributes described in the low-level routine MENU CREATE can be set in the **override_arglist**.

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

See the widget-specific attributes for the low-level routine MENU CREATE.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by MENU POPUP CREATE:

- The defaults for **width** and **height** are set as large as necessary to hold all child widgets.

CALLBACK DATA STRUCTURE

See the low-level routine MENU CREATE.

DESCRIPTION

MENU POPUP CREATE creates a pop-up menu widget.

geometry management

See the low-level routine MENU CREATE.

resizing

See the low-level routine MENU CREATE.

MENU PULLDOWN CREATE

Creates a pull-down menu.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
menu pulldown

FORMAT

widget = **DWT$MENU_PULLDOWN_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

The widget-specific attributes described in the low-level routine MENU CREATE can be set in the **override_arglist**.

MIT C FORMAT

widget = **DwtMenuPulldownCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtMenuPulldownCreate(parent_widget, name,
                             override_arglist,
                             override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

The widget-specific attributes described in the low-level routine MENU CREATE can be set in the **override_arglist**.

RETURNS

widget
Identifier of the created widget.

ARGUMENTS

parent_widget
identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

See the widget-specific attributes for the low-level routine MENU CREATE.

ATTRIBUTE EXCEPTIONS

The following core attributes are not supported by MENU PULLDOWN CREATE:

- **x**
- **y**

The following core attributes are supported differently by MENU PULLDOWN CREATE:

- The default for **width** is set as large as necessary to hold all child widgets.
- The default for **height** is set as large as necessary to hold all child widgets.

CALLBACK DATA STRUCTURE

See the low-level routine MENU CREATE.

DESCRIPTION

MENU PULLDOWN CREATE creates a pull-down menu widget.

geometry management

See the low-level widget routine MENU CREATE.

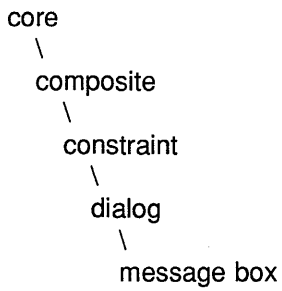
resizing

See the low-level widget routine MENU CREATE.

MESSAGE BOX CREATE

Creates a message box widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$MESSAGE_BOX_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
label	comp string	uns longword	read	reference
ok_label	comp string	uns longword	read	reference
yes_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtMessageBoxCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtMessageBoxCreate(parent_widget, name, override_arglist,
                           override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
DwtCompString   label;
DwtCompString   ok_label;
DwtCallbackPtr  yes_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

label
VAX binding: **DWT$C_NLABEL**
C binding: **DwtNlabel**

The text in the message line or lines. This argument defaults to the widget name.

ok_label
VAX binding: **DWT$C_NOk_LABEL**
C binding: **DwtNokLabel**

The label for the OK push button. If the label is a null string, the button is not displayed. The default is "Acknowledged".

yes_callback
VAX binding: **DWT$C_NYES_CALLBACK**
C binding: **DwtNyesCallback**

The callback routine or routines called when the OK button is activated. The default is null.

ATTRIBUTE EXCEPTIONS

The following common attribute is not supported by MESSAGE BOX CREATE:

- **direction_r_to_l**

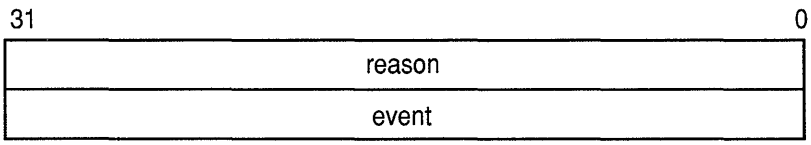
The following attributes inherited from DIALOG BOX POPUP CREATE are not supported:

- **cancel_button**
- **child_overlap**
- **default_button**
- **text_merge_translation**
- **title_type**
- **units**

The following attributes inherited from DIALOG BOX POPUP CREATE are supported differently:

- The **resize** attribute is set to Shrink Wrap.
- The default for **margin_width** is 12 pixels.
- The default for **margin_height** is 10 pixels.
- The default for **style** is Modal.

CALLBACK DATA STRUCTURE



ZK-0091A-GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason
An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event
A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Yes
VAX binding: **DWT$C_CRYES**
C binding: **DwtCRYes**

The user activated the OK push button.

Help Requested
VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help somewhere in the message box (reason in callback data).

DESCRIPTION

The message box widget is a dialog box that allows the application to display informational messages to the user. Call this routine to create a message box when the user does something unexpected, or when your application needs to display information to the user.

The box can contain an OK push button (whose label is Acknowledged by default). When style is Modal, the message box freezes the application and requires the user to explicitly dismiss the message box before the application proceeds. If the style is Modal when the user selects the OK push button, the widget is cleared from screen but not destroyed. The widget can be displayed again by using the intrinsic routine MANAGE CHILD.

A message box widget can also be created with the high-level routine MESSAGE BOX.

geometry management

The message box widget follows the same rules for geometry management as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

resizing

The message box widget follows the same rules for resizing as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

OPTION MENU CREATE

Creates an option menu widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
option menu

VAX FORMAT

widget = **DWT$OPTION_MENU_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
label	comp string	uns longword	read	reference
sub_menu_id	widget	longword	read	reference

MIT C FORMAT

widget = **DwtOptionMenuCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtOptionMenuCreate(parent_widget, name, override_arglist,
                           override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

The following widget-specific attributes can be set in the **override_arglist**:

```
DwtCompString  label;
Widget         sub_menu_id;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

label
VAX binding: **DWT$C_NLABEL**
C binding: **DwtNlabel**

The label to be placed to the left of the current value. The default is the widget name.

sub_menu_id
VAX binding: **DWT$C_NSUB_MENU_ID**
C binding: **DwtNsubMenuId**

The widget identifier of the pull-down menu associated with the option menu during widget creation. The default is zero.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by OPTION MENU CREATE:

- The defaults for **width** and **height** are set as large as necessary to hold all child widgets.
- The **font** argument is used only for gadget children.

CALLBACK DATA STRUCTURE

See the low-level routine MENU CREATE.

DESCRIPTION

OPTION MENU CREATE creates an option menu widget. The option menu widget is a composite widget containing a label identifying the menu and toggle buttons. The option menu allows the user to select only one option from the menu. It displays the current option selected and, on request, generates a pop-up menu with specific options available. The pop-up menu covers the current selection, but it displays the current selection as one of the options.

If **entry_callback** is not null, then all the toggle button callbacks will execute the entry_callback routine, rather than the procedure specified in the toggle button. If **entry_callback** is null, then individual callbacks work as usual.

An option menu widget can also be created with the high-level routine OPTION MENU.

geometry management

See the low-level routine MENU CREATE.

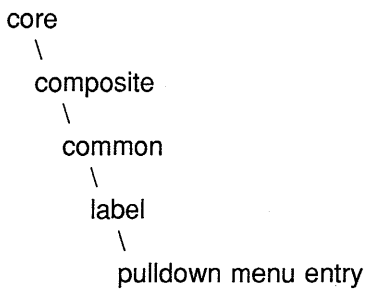
resizing

See the low-level routine MENU CREATE.

PULL DOWN MENU ENTRY CREATE

Creates a pull-down menu entry widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$PULL_DOWN_MENU_ENTRY_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Argument	Usage	Data Type	Access	Mechanism
sub_menu_id	widget	uns longword	read	reference
activate_callback	callback	uns longword	read	reference
pulling_callback	callback	uns longword	read	reference
hot_spot_pixmap	pixmap	uns longword	read	value

MIT C FORMAT

widget = **DwtPullDownMenuEntryCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtPullDownMenuEntryCreate(parent_widget, name,
                                  override_arglist,
                                  override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
Widget          sub_menu_id;
DwtCallbackPtr  activate_callback;
DwtCallbackPtr  pulling_callback;
Pixmap          hot_spot_pixmap;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

sub_menu_id

VAX binding: **DWT$C_NSUB_MENU_ID**
C binding: **DwtNsubMenuId**

The identifier of the submenu to be displayed when the pull-down menu is activated. The default is null.

activate_callback

VAX binding: **DWT$C_NACTIVATE_CALLBACK**
C binding: **DwtNactivateCallback**

A callback routine or routines called when the user releases a button inside the pull-down menu entry widget. The reason for this callback is **Activated**. The default is null.

pulling_callback

VAX binding: **DWT$C_NPULLING_CALLBACK**
C binding: **DwtNpullingCallback**

A callback routine or routines called just prior to pulling down the submenu. This callback occurs just before the submenu's map callback. This callback can be used for deferred creation of the submenu. The reason for this callback is **Activated**. The default is null.

hot_spot_pixmap

VAX binding: **DWT$C_NHOT_SPOT_PIXMAP**
C binding: **DwtNhotSpotPixmap**

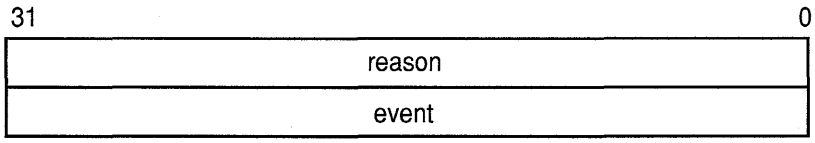
The pixmap to use for the hotspot icon. The default is null.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by PULL DOWN MENU ENTRY CREATE:

- The default for **width** equals the label width, plus the hotspot width, plus 2 times **margin_width**.
- The default for **height** equals the text label or pixmap label height plus 2 times **margin_height**.
- The default for **border_width** equals 0 pixels.

CALLBACK DATA STRUCTURE



ZK–0091A–GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason
An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event
A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Activated
VAX binding: **DWT$C_CRACTIVATED**
C binding: **DwtCRActivated**

The user selected the pull-down menu entry.

Help Requested
VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

PULL DOWN MENU ENTRY CREATE creates a pull-down menu entry widget. A pull-down menu entry widget is made up of two parts: a label (within the parent menu) and a select area or hotspot. In menu bars and work menus the hotspot is the full widget window. Otherwise, the hotspot is a separate rectangle on the right side of the entry label. A pull down menu entry widget can also be created with the high-level routine PULL DOWN MENU ENTRY.

geometry management

The pull-down menu entry widget can have a push button child, the hotspot widget. This is the only child it allows. The pull-down menu entry widget does not accept any geometry management requests.

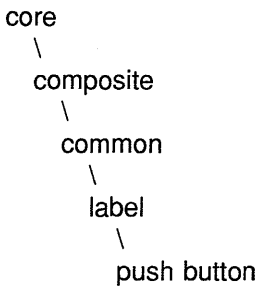
resizing

When the pull-down menu entry widget is resized by its parent, it lays out its label and then repositions and resizes the hotspot widget.

PUSH BUTTON CREATE

Creates a push button widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$PUSH_BUTTON_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
border_highlight	Boolean	uns byte	read	value
fill_highlight	Boolean	uns byte	read	value
activate_callback	callback	uns longword	read	reference
arm_callback	callback	uns longword	read	reference

Attribute	Usage	Data Type	Access	Mechanism
disarm_callback	callback	uns longword	read	reference
accelerator_text	comp string	uns longword	read	reference
button_accelerator	char string	uns longword	read	reference
shadow	Boolean	uns byte	read	value

MIT C FORMAT

widget = **DwtPushButtonCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtPushButtonCreate(parent_widget, name, override_arglist,
                           override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
Boolean          border_highlight;
Boolean          fill highlight;
DwtCallbackPtr   activate callback;
DwtCallbackPtr   arm callback;
DwtCallbackPtr   disarm callback;
DwtCompString    accelerator_text;
char             *button_accelerator;
Boolean          shadow;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

border_highlight
VAX binding: **DWT$C_NBORD_HIGHLIGHT**
C binding: **DwtNbordHighlight**

A Boolean attribute that specifies whether the border is highlighted. The highlighting is a 1-pixel line inside the border window. If true, the border is highlighted. The default is false.

fill_highlight
VAX binding: **DWT$C_NFILL_HIGHLIGHT**
C binding: **DwtNfillHighlight**

A Boolean attribute that specifies whether the border of the button is filled. If true, displays the border of the button in reverse video. The default is false.

activate_callback
VAX binding: **DWT$C_NACTIVATE_CALLBACK**
C binding: **DwtNactivateCallback**

The callback routine or routines called when the push button is activated. The button is activated when the user presses and releases MB1 while the pointer is inside the push button widget. Activating the push button also disarms the push button. See **disarm_callback**. The reason for this callback is **Activated**. The default is null.

arm_callback
VAX binding: **DWT$C_NARM_CALLBACK**
C binding: **DwtNarmCallback**

The callback routine or routines called when the push button is armed. The button is armed when the user presses MB1 while the pointer is inside the push button widget. The reason for this callback is **Arm**.

disarm_callback
VAX binding: **DWT$C_NDISARM_CALLBACK**
C binding: **DwtNdisarmCallback**

The callback routine or routines called when the push button is disarmed. The button is disarmed in two ways. After the user activates the button (presses and releases MB1 while the pointer is inside the push button widget), the button is disarmed. When the user presses MB1 while the pointer is inside the push button widget but moves the pointer outside the push button before releasing MB1, the button is disarmed. The reason for this callback is **Disarm**. The default is null.

accelerator_text
VAX binding: **DWT$C_NACCELERATOR_TEXT**
C binding: **DwtNacceleratorText**

The text to be displayed for the accelerator. The default is null.

button_accelerator
VAX binding: **DWT$C_NBUTTON_ACCELERATOR**
C binding: **DwtNbuttonAccelerator**

An accelerator on a push button widget. The same as the common argument **translations** except that only the left side of the table is passed as a character string, not compiled. The application is responsible for calling the intrinsic routine INSTALL ALL ACCELERATORS to install the accelerator where the application needs it. See the *VMS DECwindows Guide to Application Programming* for information on translation tables. The default is null.

shadow

VAX binding: **DWT$C_NSHADOW**
C binding: **DwtNshadow**

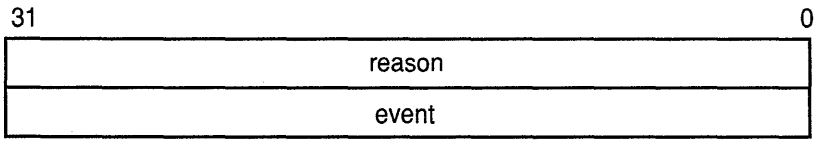
Determines whether the shadow of the push button is displayed. The default is true.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by PUSH BUTTON CREATE:

- The default for **width** equals the width of the label or pixmap plus 2 times **margin_width**.
- The default for **height** equals the height of the label or pixmap plus 2 times **margin_height**.
- The default for **border_width** equals 1 pixel.

CALLBACK DATA STRUCTURE



ZK-0091A-GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason
An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event
A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Activated
VAX binding: **DWT$C_CRACTIVATED**
C binding: **DwtCRActivated**

The user activated the push button by pressing and releasing MB1 while the pointer was inside the push button widget.

Arm
VAX binding: **DWT$C_CRARM**
C binding: **DwtCRArm**

The user armed the push button by pressing MB1 while the pointer was inside the push button widget.

Disarm
VAX binding: **DWT$C_CRDISARM**
C binding: **DwtCRDisarm**

The user disarmed the push button in one of two ways. The user pressed MB1 while the pointer was inside the push button widget, but did not release it until after moving the pointer outside the push button widget. Alternately, the user activated the push button which simultaneously disarms it.

Help Requested
VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtHelpRequested**

The user selected help.

DESCRIPTION

PUSH BUTTON CREATE creates a push button widget. The push button is a primitive widget that displays a rectangular border around a label. The label defines the immediate action of the button function (for example, OK or Cancel in a dialog box).

The sizing is affected by the spacing, the font, and the label. See the LABEL WIDGET routine for more information. A push button widget can also be created with the high-level routine PUSH BUTTON.

geometry management

The push button widget follows the same rules for geometry management as its superclass the label widget, described in the low-level routine LABEL CREATE.

resizing

The push button widget follows the same rules for resizing as its superclass the label widget, described in the low-level routine LABEL CREATE.

RADIO BOX CREATE

Creates a radio box widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
radio box

VAX FORMAT

widget = **DWT$RADIO_BOX_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

Attributes described in the low-level routine MENU CREATE can be set in the **override_arglist**.

MIT C FORMAT

widget = **DwtRadioBoxCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtRadioBoxCreate(parent_widget, name, override_arglist,
                         override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

Attributes described in the low-level routine MENU CREATE can be set in the **override_arglist**.

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

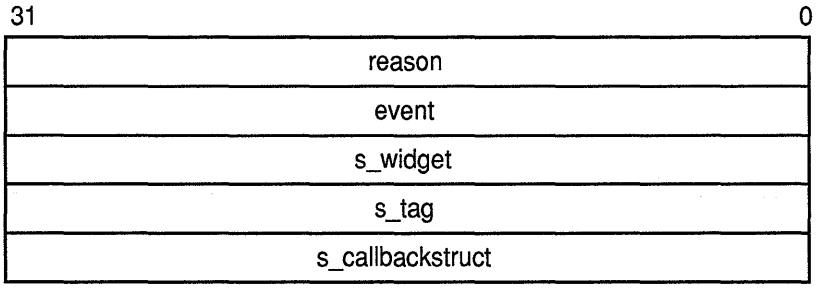
See the widget-specific attributes for the low-level routine MENU CREATE.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by RADIO BOX CREATE:

- The defaults for **width** and **height** are set as large as necessary to hold all widgets.
- Setting the sensitivity causes all widgets contained in the radio box to be set to the same sensitivity.

CALLBACK DATA STRUCTURE



ZK-0094A-GE

VAX field information

Structure name: DWT$RADIOBOX_CB_ST

Name	Usage	Data Type	Access	Mechanism
radiobox_reason	callback reason	longword	read	value
radiobox_event	event	uns longword	read	reference
radiobox_s_widget	widget	longword	read	reference
radiobox_s_tag	longword	uns longword	read	value
radiobox_s_callbackstruct	callback	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
      int     s_widget;
      char    *s_tag;
      char    *s_callbackstruct
} DwtRadioBoxCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason
An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event
A pointer to the X event structure describing the event that generated this callback.

s_widget
The identifier of the calling subwidget.

s_tag
The tag supplied by the application programmer when the callback routine was specified.

s_callbackstruct
The subwidget's callback structure.

CALLBACK REASONS

Value Changed
VAX binding: **DWT$C_CRVALUE_CHANGED**
C binding: **DwtCRValueChanged**

The user activated the toggle button or push button.

Map
VAX binding: **DWT$C_CRMAP**
C binding: **DwtCRMap**

The radio box is about to be mapped.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

RADIO BOX CREATE creates a radio box widget. The radio box is a composite widget that contains multiple toggle button widgets. The radio box ensures that only one toggle button is on at any given time.

A radio box widget can also be created with the high-level routine RADIO BOX.

geometry management

See the low-level routine MENU CREATE.

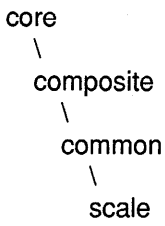
resizing

See the low-level routine MENU CREATE.

SCALE CREATE

Creates a scale widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$SCALE_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
overrride_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Argument	Usage	Data Type	Access	Mechanism
value	longword	longword	read	value
title	comp string	uns longword	read	reference
orientation	uns byte	uns byte	read	value
scale_width	dimension	uns word	read	value
scale_height	dimension	uns word	read	value
min_value	longword	longword	read	value
max_value	longword	longword	read	value
decimal_points	uns word	uns word	read	value

Argument	Usage	Data Type	Access	Mechanism
show_value	Boolean	uns byte	read	value
slider_pixmap	pixmap	uns longword	read	value
drag_callback	callback	uns longword	read	reference
value_changed_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtScaleCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtScaleCreate(parent_widget, name, override_arglist,
                      override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
int             value;
DwtCompString   title;
unsigned char   orientation;
Dimension       scale_width;
Dimension       scale_height;
int             min_value;
int             max_value;
short           decimal_points;
Boolean         show_value;
Pixmap          slider_pixmap;
DwtCallbackPtr  drag_callback;
DwtCallbackPtr  value_changed_callback;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

value
VAX binding: **DWT$C_NVALUE**
C binding: **DwtNvalue**

The value represented by the slider's current position. The default is zero.

title
VAX binding: **DWT$C_NTITLE**
C binding: **DwtNtitle**

The title text string to appear in the scale window widget. The default is the scale name.

orientation
VAX binding: **DWT$C_NORIENTATION**
C binding: **DwtNorientation**

The orientation of the scale. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ORIENTATION_HORIZONTAL	DwtOrientationHorizontal	Horizontal scale (default)
DWT$C_ORIENTATION_VERTICAL	DwtOrientationVertical	Vertical scale

scale_width
VAX binding: **DWT$C_NSCALE_WIDTH**
C binding: **DwtNscaleWidth**

The width of the scale (excluding labels). The default is 20 pixels for vertical scales and is calculated for horizontal scales.

scale_height
VAX binding: **DWT$C_NSCALE_HEIGHT**
C binding: **DwtNscaleHeight**

The height of the scale (excluding labels). The default is 20 pixels for horizontal scales and is calculated for vertical scales.

min_value
VAX binding: **DWT$C_NMIN_VALUE**
C binding: **DwtNminValue**

The value represented by the top or left end of the scale. The default is zero.

max_value
VAX binding: **DWT$C_NMAX_VALUE**
C binding: **DwtNmaxValue**

The value represented by the bottom or right end of the scale. The default is 100.

decimal_points
VAX binding: **DWT$C_NDECIMAL_POINTS**
C binding: **DwtNdecimalPoints**

The number of decimal points to shift the current slider value for display next to the slider. The default is zero.

show_value
VAX binding: **DWT$C_NSHOW_VALUE**
C binding: **DwtNshowValue**

A Boolean attribute that, when true, states that the current value of the slider label string will be displayed next to the slider. The default is true.

slider_pixmap
VAX binding: **DWT$C_NSLIDER_PIXMAP**
C binding: **DwtNsliderPixmap**

The pixmap for the slider. The default is null, with an arrow being displayed.

drag_callback
VAX binding: **DWT$C_NDRAG_CALLBACK**
C binding: **DwtNdragCallback**

The callback routine or routines called because the user is dragging the slider. For this routine, the callback reason is **Drag**.

value_changed_callback
VAX binding: **DWT$C_NVALUE_CHANGED_CALLBACK**
C binding: **DwtNvalueChangedCallback**

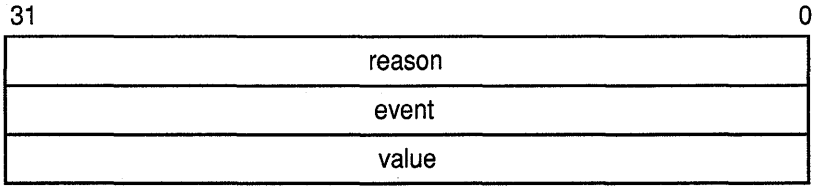
The callback routine or routines called when the value of the slider changes. For this routine, the callback reason is **Value Changed**.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by SCALE CREATE:

- The defaults for **width** and **height** are calculated based on the scale width or height, the label widths, and the orientation.
- The default for **border_width** is 0 pixels.

CALLBACK DATA STRUCTURE



ZK-0096A-GE

VAX field information

Structure name: DWT$SCALE_CB_ST

Name	Usage	Data Type	Access	Mechanism
sc_reason	callback reason	longword	read	value
sc_event	event	uns longword	read	reference
sc_value	longword	longword	read	value

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
      int     value;
} DwtScaleCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

value

The current value of the scale.

CALLBACK REASONS

Value Changed

VAX binding: **DWT$C_CRVALUE_CHANGED**
C binding: **DwtCRValueChanged**

The user moved the slider in the scale by dragging it or clicking on it, or the intrinsic routine SET VALUES changed the value of the slider.

Drag

VAX binding: **DWT$C_CRDRAG**
C binding: **DwtCRDrag**

The user is dragging the slider.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

SCALE CREATE creates a scale widget. The scale widget allows the application to display a scale for vernier control of a specific parameter by the user. The user places the slider at a position representing the desired value. The scale may have labeled text at any number of points identifying the values corresponding to the points. The scale can be made insensitive and used as an output value indicator only (for example, a thermometer or percent completion indicator).

The application passes lower and upper values for the scale as integers and can (optionally) indicate a decimal point position. For example, a **min_value** of 100, a **max_value** of 10000, and a **decimal_points** of 2 would produce a scale from 1.00 to 100.00. Possible values returned from this example could be 230 or 5783 (representing decimal values 2.30 and 57.83).

Tick mark labels along the scale are provided by the scale's children. The labels can be any type of widget created with the scale widget as its parent. Normally a label widget is used. The order of creation of the widgets determines their placement along the scale.

A scale widget can also be created with the high-level routine SCALE.

geometry management

The scale widget moves its children so they will be within the scale widget's calculated size. DIGITAL recommends that you create the scale widget with **width** and **height** attributes set to zero. This allows the scale widget to make the best decisions for its layout.

resizing

If resized, the scale widget positions its children so they all fit within the scale widget.

SCROLL BAR CREATE

Creates a scroll bar widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
scroll

VAX FORMAT

widget = **DWT$SCROLL_BAR_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
int_value	longword	longword	read	value
min_value	longword	longword	read	value
max_value	longword	longword	read	value
orientation	uns byte	uns longword	read	value
translations1	translations	uns longword	read	reference
translations2	translations	uns longword	read	reference
shown	longword	longword	read	value
inc	longword	longword	read	value

Attribute	Usage	Data Type	Access	Mechanism
page_inc	longword	longword	read	value
unit_inc_callback	callback	uns longword	read	reference
unit_dec_callback	callback	uns longword	read	reference
page_inc_callback	callback	uns longword	read	reference
page_dec_callback	callback	uns longword	read	reference
to_top_callback	callback	uns longword	read	reference
to_bottom_callback	callback	uns longword	read	reference
drag_callback	callback	uns longword	read	reference
value_changed_ callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtScrollBarCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtScrollBarCreate(parent_widget, name, override_arglist,
                          override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
int             int_value;
int             min_value;
int             max_value;
unsigned char   orientation;
XtTranslations  translations1;
XtTranslations  translations2;
int             shown;
int             inc;
int             page_inc;
DwtCallbackPtr  unit_inc_callback;
DwtCallbackPtr  unit_dec_callback;
DwtCallbackPtr  page_inc_callback;
DwtCallbackPtr  page_dec_callback;
DwtCallbackPtr  to_top_callback;
DwtCallbackPtr  to_bottom_callback;
DwtCallbackPtr  drag_callback;
DwtCallbackPtr  value_changed_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

int_value
VAX binding: **DWT$C_NVALUE**
C binding: **DwtNvalue**

The position of the top or left end of the scroll bar's slider. The default is zero.

min_value
VAX binding: **DWT$C_NMIN_VALUE**
C binding: **DwtNminValue**

The scroll bar's minimum value. The default is zero.

max_value
VAX binding: **DWT$C_NMAX_VALUE**
C binding: **DwtNmaxValue**

The scroll bar's maximum value. The default is 100.

orientation
VAX binding: **DWT$C_NORIENTATION**
C binding: **DwtNorientation**

The orientation of the scroll bar. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ORIENTATION_HORIZONTAL	DwtOrientationHorizontal	Horizontal scroll bar
DWT$C_ORIENTATION_VERTICAL	DwtOrientationVertical	Vertical scroll bar (default)

translations1
VAX binding: **DWT$C_NTRANSLATIONS1**
C binding: **DwtNtranslations1**

The translation table for events after being parsed by the intrinsic routine PARSE TRANSLATION TABLE for the decrement stepping arrow. See the *VMS DECwindows Guide to Application Programming* for information on translation tables. The default is null.

translations2

VAX binding: **DWT$C_NTRANSLATIONS2**
C binding: **DwtNtranslations2**

The translation table for events after being parsed by PARSE TRANSLATION TABLE for the increment stepping arrow. See the *VMS DECwindows Guide to Application Programming* for information on translation tables. The default is null.

shown

VAX binding: **DWT$C_NSHOWN**
C binding: **DwtNshown**

The size of the slider as value between zero and the absolute value of **max_value** minus **min_value**. The size of the slider varies, depending on how much of the slider scroll area it represents. The default is 10 units.

inc

VAX binding: **DWT$C_NINC**
C binding: **DwtNinc**

The amount of unit increment and decrement. The default is 10 units. If this argument is not zero, the widget automatically adjusts the slider when a increment/decrement action occurs.

page_inc

VAX binding: **DWT$C_NPAGE_INC**
C binding: **DwtNpageInc**

The amount of page increment and decrement. The default is 10 units. If this argument is not zero, the widget automatically adjusts the slider when an increment/decrement action occurs.

unit_inc_callback

VAX binding: **DWT$C_NUNIT_INC_CALLBACK**
C binding: **DwtNunitIncCallback**

The callback routine or routines called because the user selected the unit increment function. For this routine, the callback reason is **Unit Inc**. The default is null.

unit_dec_callback

VAX binding: **DWT$C_NUNIT_DEC_CALLBACK**
C binding: **DwtNunitDecCallback**

The callback routine or routines called because the user selected the unit decrement function. For this routine, the callback reason is **Unit Dec**. The default is null.

page_inc_callback

VAX binding: **DWT$C_NPAGE_INC_CALLBACK**
C binding: **DwtNpageIncCallback**

The callback routine or routines called because the user selected the page increment function. For this routine, the callback reason is **Page Inc**. The default is null.

page_dec_callback

VAX binding: **DWT$C_NPAGE_DEC_CALLBACK**
C binding: **DwtNpageDecCallback**

The callback routine or routines called because the user selected the page decrement function. For this routine, the callback reason is **Page Dec**. The default is null.

to_top_callback

VAX binding: **DWT$C_NTO_TOP_CALLBACK**
C binding: **DwtNtoTopCallback**

The callback routine or routines called because the user selected the to-top scroll function. For this routine, the callback reason is **To Top**.

The scroll bar does not automatically change the scroll bar's value for this callback. The default is null.

to_bottom_callback

VAX binding: **DWT$C_NTO_BOTTOM_CALLBACK**
C binding: **DwtNtoBottomCallback**

The callback routine or routines called because the user selected the to-bottom scroll function. For this routine, the callback reason is **To Bottom**.

The scroll bar does not automatically change the scroll bar's value for this callback. The default is null.

drag_callback

VAX binding: **DWT$C_NDRAG_CALLBACK**
C binding: **DwtNdragCallback**

The callback routine or routines called because the user is dragging the scroll bar slider. For this routine, the callback reason is **Drag**. The default is null.

value_changed_callback

VAX binding: **DWT$C_NVALUE_CHANGED_CALLBACK**
C binding: **DwtNvalueChangedCallback**

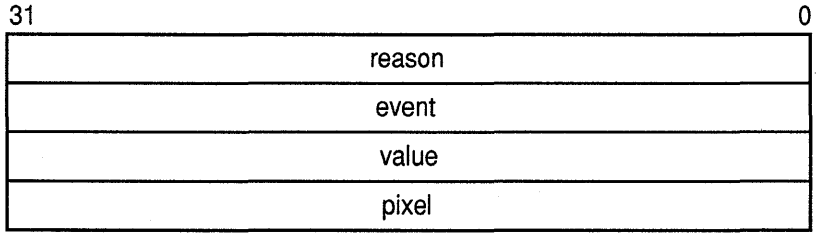
The callback routine or routines called when the value of the scroll has changed. For this routine, the callback reason is **Value Changed**. The default is null.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by SCROLL BAR CREATE:

- The default for **width** is 17 pixels for a vertical orientation, and the width of the parent minus 17 pixels for a horizontal orientation.
- The default for **height** is 17 pixels for a horizontal orientation, and the height of the parent minus 17 pixels for a vertical orientation.
- The **font** attribute is not supported.

CALLBACK DATA STRUCTURE



ZK-0253A-GE

VAX field information

Structure name: DWT$SCRL_BAR_CB_ST

Name	Usage	Data Type	Access	Mechanism
scroll_reason	callback reason	longword	read	value
scroll_event	event	uns longword	read	reference
scroll_value	longword	longword	read	value
scroll_pixel	longword	longword	read	value

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
      int     value;
      int     pixel;
} DwtScrollBarCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

value

The current scroll bar value.

pixel

The pixel distance from the top right of the scroll bar (for vertical scroll bars) or from the left of the scroll bar (for horizontal scroll bars) where the event occurred. This value is used for the **to_top_callback** and **to_bottom_callback** attributes.

CALLBACK REASONS

Value Changed

VAX binding: **DWT$C_CRVALUE_CHANGED**
C binding: **DwtCRValueChanged**

The user changed the value of the scroll bar slider, or the intrinsic routine SET VALUES changed the value of the scroll bar slider.

Unit Inc

VAX binding: **DWT$C_CRUNIT_INC**
C binding: **DwtCRUnitInc**

The user selected the unit increment function.

Unit Dec

VAX binding: **DWT$C_CRUNIT_DEC**
C binding: **DwtCRUnitDec**

The user selected the decrement function.

Page Inc

VAX binding: **DWT$C_CRPAGE_INC**
C binding: **DwtCRPageInc**

The user selected the page increment function.

Page Dec

VAX binding: **DWT$C_CRUNIT_DEC**
C binding: **DwtCRPageDec**

The user selected the page decrement function.

To Top

VAX binding: **DWT$C_CRTO_TOP**
C binding: **DwtCRToTop**

The user selected the to-top scroll function.

To Bottom

VAX binding: **DWT$C_CRTO_BOTTOM**
C binding: **DwtCRToBottom**

The user selected the to-bottom scroll function.

Drag

VAX binding: **DWT$C_CRDRAG**
C binding: **DwtCRDrag**

The user is dragging the scroll bar slider.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

SCROLL BAR CREATE creates a scroll bar widget. The scroll bar widget is a screen object that allows the user to scroll through display data too large for the screen. The scroll bar widget consists of two stepping arrows at either end of an elongated rectangle called the scroll region. The scroll region is overlaid with a slider bar that is adjusted in size and position as scrolling occurs.

Users select the scrolling functions as follows:

- Clicking MB1 on the right or bottom stepping arrow selects the unit increment function.
- Clicking MB1 on the left or top stepping arrow selects the unit decrement function.
- Clicking MB1 on the scroll region between the slider and the right or bottom stepping arrow selects the page increment function.
- Clicking MB1 on the scroll region between the slider and the left or top stepping arrow selects the page decrement function.
- Clicking MB2 anywhere within the scroll bar selects the to-top function.
- Clicking MB3 anywhere within the scroll bar selects the to-bottom function.

The **to_top_callback** and **to_bottom_callback** attributes do not automatically set the slider as the other callbacks do.

A scroll bar widget can also be created with the high-level routine SCROLL BAR.

geometry management

The scroll bar widget does not support children.

resizing

If the core widget attribute **width** or **height** is set to zero, a default value is used. The default value varies with the orientation of the scroll bar and the attribute set to zero.

The following are the defaults for vertical scroll bars:

- If **x** is zero, **x** becomes the width of the parent widget minus 17 pixels.
- If **height** is zero, **height** becomes the height of the parent widget minus 17 pixels.
- If **width** is zero, **width** becomes 17 pixels.

The following are the defaults for horizontal scroll bars:

- If **y** is zero, **y** becomes the height of the parent widget minus 17 pixels.
- If **width** is zero, **width** becomes the width of the parent widget minus 17 pixels.
- If **height** is zero, the height becomes 17 pixels.

SCROLL WINDOW CREATE

Creates a scroll window widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
scroll window

VAX FORMAT

widget = **DWT$SCROLL_WINDOW_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
h_scroll	widget	uns longword	read	value
v_scroll	widget	uns longword	read	value
work_window	widget	uns longword	read	value
shown_value_automatic_horiz	Boolean	uns byte	read	value
shown_value_automatic_vert	Boolean	uns byte	read	value

MIT C FORMAT

widget = **DwtScrollWindowCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtScrollWindowCreate(parent_widget, name,
                             override_arglist, override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
Widget   h_scroll;
Widget   v_scroll;
Widget   work_window;
Boolean  shown_value_automatic_horiz;
Boolean  shown_value_automatic_vert;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

h_scroll

VAX binding: **DWT$C_NHORIZONTAL_SCROLL_BAR**
C binding: **DwtNhorizontalScrollBar**

The widget identifier for the horizontal scroll bar in the scroll window widget. This cannot be supplied at main window creation time, but it can be set or specified after creation. The default is null.

v_scroll

VAX binding: **DWT$C_NVERTICAL_SCROLL_BAR**
C binding: **DwtNverticalScrollBar**

The widget identifier for the vertical scroll bar in the scroll window widget. This cannot be supplied at main window creation time, but it can be set or specified after creation. The default is null.

work_window

VAX binding: **DWT$C_NWORK_WINDOW**
C binding: **DwtNworkWindow**

The widget identifier for the window to be associated with the scroll window work area. This cannot be supplied at main window creation time, but it can be set or specified after creation. The default is null.

shown_value_automatic_horiz

VAX binding: **DWT$C_NSHOWN_VALUE_AUTOMATIC_HORIZ**
C binding: **DwtNshownValueAutomaticHoriz**

A Boolean attribute that specifies whether the horizontal scroll bar's shown value is automatically set. If true, the horizontal scroll bar's shown value is automatically set. The default is true.

shown_value_automatic_vert

VAX binding: **DWT$C_NSHOWN_VALUE_AUTOMATIC_VERT**
C binding: **DwtNshownValueAutomaticVert**

A Boolean attribute that specifies whether the vertical scroll bar's shown value is automatically set. If true, the vertical scroll bar's shown value is automatically set. The default is true.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by SCROLL WINDOW CREATE:

- Setting the sensitivity of the scroll window causes all widgets contained in that window to be set to the same sensitivity as the scroll window.
- The **font**, and **help_callback** common attributes are not supported.

DESCRIPTION

SCROLL WINDOW CREATE creates the scroll window widget that provides a more direct interface for applications with scroll bars. It is a composite widget that can contain both vertical and horizontal scroll bar widgets and any widget as the window region. Scroll bar positioning and scroll bar slider sizes are automatically maintained. The scroll window widget simplifies programming by allowing programmers to create an application with scroll bars directly in the scroll window widget work area.

A scroll window widget can also be created using the high-level routine SCROLL WINDOW.

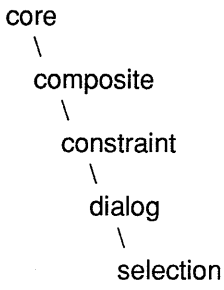
See also the high-level routine SCROLL WINDOW SET AREAS.

geometry management	The scroll window widget does not size or position the work region widget. The scroll window widget does size and position the scroll bars.
resizing	The scroll window widget automatically resizes the scroll bars, but not the work region widget.

SELECTION CREATE

Creates a selection widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$SELECTION_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
label	comp string	uns longword	read	reference
value	comp string	uns longword	read	reference
selection_label	comp string	uns longword	read	reference
ok_label	comp string	uns longword	read	reference
cancel_label	comp string	uns longword	read	reference
activate_callback	callback	uns longword	read	reference
cancel_callback	callback	uns longword	read	reference

Attribute	Usage	Data Type	Access	Mechanism
no_match_callback	callback	uns longword	read	reference
visible_item_count	longword	longword	read	value
items	array	uns longword	read	reference
item_count	longword	longword	read	value
must_match	Boolean	uns byte	read	value

MIT C FORMAT

widget = **DwtSelectionCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtSelectionCreate(parent_widget, name, override_arglist,
                          override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
DwtCompString   label;
DwtCompString   value;
DwtCompString   selection_label;
DwtCompString   ok_label;
DwtCompString   cancel_label;
DwtCallbackPtr  activate_callback;
DwtCallbackPtr  cancel_callback;
DwtCallbackPtr  no_match_callback;
int             visible_item_count;
DwtCompString   *items;
int             item_count;
Boolean         must_match;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

label
VAX binding: **DWT$C_NLABEL**
C binding: **DwtNlabel**

The label to appear above the list box containing the items. The default is "Items".

value
VAX binding: **DWT$C_NVALUE**
C binding: **DwtNvalue**

The current value of the selection widget displayed in the text edit field. The default is the null string.

selection_label
VAX binding: **DWT$C_NSELECTION_LABEL**
C binding: **DwtNselectionLabel**

The label above the selection text entry field. The default is "Selection".

ok_label
VAX binding: **DWT$C_NOK_LABEL**
C binding: **DwtNokLabel**

The label for the OK push button. If the label is a null string, the button is not displayed. The default is "OK".

cancel_label
VAX binding: **DWT$C_NCANCEL_LABEL**
C binding: **DwtNcancelLabel**

The label for the Cancel push button. If the label is a null string, the button is not displayed. The default is "Cancel".

activate_callback
VAX binding: **DWT$C_NACTIVATE_CALLBACK**
C binding: **DwtNactivateCallback**

The callback routine or routines called when the user makes a selection. The reason for this callback is **Activated**. The default is null.

cancel_callback
VAX binding: **DWT$C_NCANCEL_CALLBACK**
C binding: **DwtNcancelCallback**

The callback routine or routines called when the user cancels a selection. The reason for the callback is **Cancel**. The default is null.

no_match_callback
VAX binding: **DWT$C_NNO_MATCH_CALLBACK**
C binding: **DwtNnoMatchCallback**

The callback routine or routines called when **must_match** is true, and a user's selection does not exactly match any items in the list box. The reason for this callback is **No Match**. The default is null.

visible_item_count

VAX binding: **DWT$C_NVISIBLE_ITEMS_COUNT**
C binding: **DwtNvisibleItemsCount**

The number of items displayed in the selection widget's list box. The default is 8 items.

items

VAX binding: **DWT$C_NITEMS**
C binding: **DwtNitems**

The items in the selection widget's list box. The default is null.

item_count

VAX binding: **DWT$C_NITEMS_COUNT**
C binding: **DwtNitemsCount**

The number of items in the selection widget's list box. The default is zero.

must_match

VAX binding: **DWT$C_NMUST_MATCH**
C binding: **DwtNmustMatch**

A Boolean attribute that specifies whether the selection widget checks for an exact match. If true, the selection widget checks whether the user's selection exactly matches an item in the list box. If there is not an exact match, the **no_match_callback** is activated. If there is an exact match, the **activate_callback** is activated. The default is false.

ATTRIBUTE EXCEPTIONS

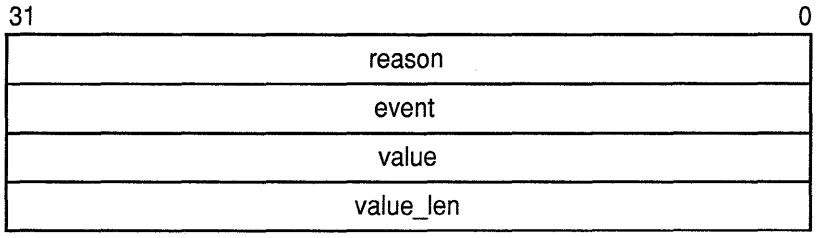
The following common attributes are supported differently by SELECTION CREATE:

- The default for **width** is the width of the list box, plus the width of the push buttons, plus three times **margin_width**. The list box expands to accommodate items wider than the title.
- The default for **height** is the height of the list box, plus the height of the text edit field, plus the height of the label, plus three times **margin_height**.
- The default for **x** and **y** is centered in the parent window.

The following attributes inherited from DIALOG BOX POPUP are supported differently:

- The default for **margin_width** is 5 pixels.
- The default for **margin_height** is 5 pixels.
- The default for **title** is "Open".
- The default for **style** is Modal.

CALLBACK DATA STRUCTURE



ZK-0095A-GE

VAX field information

Structure name: DWT$SEL_CB_ST

Name	Usage	Data Type	Access	Mechanism
sel_reason	callback reason	longword	read	value
sel_event	event	uns longword	read	reference
sel_value	comp string	uns longword	read	reference
sel_value_len	longword	longword	read	value

MIT C field information

```
typedef struct {
      int             reason;
      XEvent          *event;
      DwtCompString   value;
      int             value_len;
} DwtSelectionCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

value

The current selection when the callback occurred.

value length

The length of the selection compound string.

CALLBACK REASONS

Activated

VAX binding: **DWT$C_CRACTIVATED**
C binding: **DwtCRActivated**

The user activated the OK push button or double clicked on an item that has an exact match in the list box.

No Match

VAX binding: **DWT$C_CRNO_MATCH**
C binding: **DwtCRNoMatch**

The user activated the OK push button or double clicked on an item that does not have an exact match in the list box.

Cancel

VAX binding: **DWT$C_CRCANCEL**
C binding: **DwtCRCancel**

The user activated the Cancel button.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help somewhere in the file selection box.

DESCRIPTION

SELECTION CREATE creates a selection widget. The selection widget is a pop-up dialog box containing a label widget, a text entry widget holding the current value, a list box displaying the current item list, and OK and Cancel push buttons.

When realized, the selection widget displays the item list passed by the caller. The current value is displayed in the text entry field. Users make selections by clicking the mouse in the list box or by typing item names in the text entry field. The selection widget does not do file searches. Use FILE SELECTION to perform file searches.

geometry management

The selection widget follows the same rules for geometry management as its superclass the dialog box widget, described in the low-level routine DIALOG BOX POPUP CREATE. However, the selection widget allows only one child and places the child between the list box and the push buttons. That child cannot be a gadget.

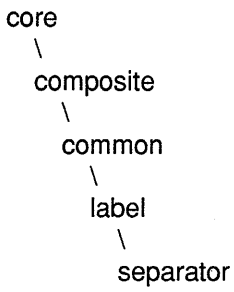
resizing

The selection widget follows the same rules for resizing as its superclass the dialog box widget, described in the low-level routine DIALOG BOX POPUP CREATE.

SEPARATOR CREATE

Creates a separator widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$SEPARATOR_CREATE** *(parent_name, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
orientation	uns byte	uns longword	read	value

MIT C FORMAT

widget = **DwtSeparatorCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtSeparatorCreate(parent_widget, name, override_arglist,
                          override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
unsigned char  orientation;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

orientation

VAX binding: **DWT$C_NORIENTATION**
C binding: **DwtNorientation**

The orientation of the separator widget. The predefined values for this attribute are as follows:

VMS	C	Description
DWT$C_ORIENTATION_HORIZONTAL	DwtOrientationHorizontal	Horizontal separator (default)
DWT$C_ORIENTATION_VERTICAL	DwtOrientationVertical	Vertical separator

A separator widget draws a centered, single-pixel line between the appropriate margins. For example, a horizontal separator draws a horizontal line from the left margin to the right margin. It is placed vertically in the middle of the widget.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by SEPARATOR CREATE:

- The default **width** is 3 pixels.
- The default **height** is 3 pixels.
- The default for **border_width** is 0 pixels.
- The **translations**, **help_callback**, and **font** attributes are not supported.

DESCRIPTION

SEPARATOR CREATE creates a separator widget. The separator widget is a screen object that allows the application to draw horizontal or vertical lines between items in a display.

A separator widget can also be created with the high-level routine SEPARATOR.

geometry management

Because a separator widget does not support children, it always refuses geometry requests.

resizing

The separator widget does nothing on a resize by its parents.

S TEXT CREATE

Creates a simple text widget.

WIDGET CLASS HIERARCHY

core
\
text
\
s text

VAX FORMAT

widget = **DWT$S_TEXT_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
margin_width	dimension	uns word	read	value
margin_height	dimension	uns word	read	value
cols	uns longword	uns longword	read	value
rows	uns longword	uns longword	read	value
top_position	uns longword	uns longword	read	value
word_wrap	Boolean	uns byte	read	value
scroll_vertical	Boolean	uns byte	read	value
resize_height	Boolean	uns byte	read	value
resize_width	Boolean	uns byte	read	value
value	char string	char string	read	reference

Attribute	Usage	Data Type	Access	Mechanism
editable	Boolean	uns byte	read	value
max_len	uns longword	uns longword	read	value
focus_callback	callback	uns longword	read	reference
help_callback	callback	uns longword	read	reference
lost_focus_ callback	callback	uns longword	read	reference
value_changed_ callback	callback	uns longword	read	reference
auto_show_insert_ point	Boolean	uns byte	read	value
insertion_position	uns longword	uns longword	read	value
foreground	pixel	uns longword	read	value
font	font list	uns longword	read	reference
blink_rate	uns longword	uns longword	read	value
scroll_left_side	Boolean	uns byte	read	value
insertion_point_ visible	Boolean	uns byte	read	value
half_border	Boolean	uns byte	read	value
pending_delete	Boolean	uns byte	read	value

MIT C FORMAT

widget = **DwtSTextCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtSTextCreate(parent_widget, name, override_arglist,
                      override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_ arglist**:

```
Dimension       margin_width;
Dimension       margin_height;
int             cols;
int             rows;
int             top_position;
Boolean         word_wrap;
Boolean         scroll_vertical;
Boolean         resize_height;
Boolean         resize_width;
char            *value;
Boolean         editable;
int             max_len;
DwtCallbackPtr  focus_callback;
DwtCallbackPtr  help_callback;
DwtCallbackPtr  lost_focus_callback;
DwtCallbackPtr  value_changed_callback;
Boolean         insertion_point_visible;
Boolean         auto_show_insert_point;
int             insertion_position;
Pixel           foreground;
FontList        font;
int             blink_rate;
Boolean         scroll_left_side;
Boolean         half_border;
Boolean         pending_delete;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

margin_width
VAX binding: **DWT$C_NMARGIN_WIDTH**
C binding: **DwtNmarginWidth**

The number of pixels between the left or right edge of the window and the text. The default is 2 pixels.

margin_height
VAX binding: **DWT$C_NMARGIN_HEIGHT**
C binding: **DwtNmarginHeight**

The number of pixels between the top or bottom edge of the window and the text. The default is 2 pixels.

cols
VAX binding: **DWT$C_NCOLS**
C binding: **DwtNcols**

The width, in characters, of the window. The default is 20 characters.

rows
VAX binding: **DWT$C_NROWS**
C binding: **DwtNrows**

The height, in characters, of the window. The default is 1 character.

top_position
VAX binding: **DWT$C_NTOP_POSITION**
C binding: **DwtNtopPosition**

A position to display at the top of the window. The default is zero.

word_wrap
VAX binding: **DWT$C_NWORD_WRAP**
C binding: **DwtNwordWrap**

A Boolean attribute that specifies whether word wrap is set. If true, lines are broken at word breaks and text does not run off the right edge of the window. The default is false.

scroll_vertical
VAX binding: **DWT$C_NSCROLL_VERTICAL**
C binding: **DwtNscrollVertical**

A Boolean attribute that specifies whether a scroll bar is present. If true, adds a scroll bar that allows the user to scroll vertically through the text. The default is false.

resize_height
VAX binding: **DWT$C_NRESIZE_HEIGHT**
C binding: **DwtNresizeHeight**

A Boolean attribute that specifies whether the simple text widget resizes its height to accommodate all the text contained in the widget. If true, the simple text widget resizes its height and the text always displays starting from the first position in the source, even if instructed otherwise. This attribute is ignored if **scroll_vertical** is true. The default is true.

resize_width
VAX binding: **DWT$C_NRESIZE_WIDTH**
C binding: **DwtNresizeWidth**

A Boolean attribute that specifies whether the simple text widget resizes its width to accommodate all the text contained in the widget. If true, the simple text widget resizes its width. This argument is ignored if **word_wrap** is true. The default is true.

value
VAX binding: **DWT$C_NVALUE**
C binding: **DwtNvalue**

The text contents of the simple text widget. The default is the null string.

editable
VAX binding: **DWT$C_NEDITABLE**
C binding: **DwtNeditable**

A Boolean attribute that, if true, specifies that the user is allowed to edit the text. The default is true.

max_len
VAX binding: **DWT$C_NMAX_LENGTH**
C binding: **DwtNmaxLength**

The maximum length of the text string in the simple text widget. The default is $2^{31}-1$

focus_callback
VAX binding: **DWT$C_NFOCUS_CALLBACK**
C binding: **DwtNfocusCallback**

A callback routine or routines called when the simple text widget has accepted the input focus. The reason for this callback is **Focus**. The default is null.

help_callback
VAX binding: **DWT$C_NHELP_CALLBACK**
C binding: **DwtNhelpCallback**

The callback routine or routines called on a help request. The reason for this callback is **Help Requested**. The default is null.

lost_focus_callback
VAX binding: **DWT$C_NLOST_FOCUS_CALLBACK**
C binding: **DwtNlostFocusCallback**

The callback routine or routines called when the simple text widget loses input focus. The reason for this callback is **Focus**. The default is null.

value_changed_callback
VAX binding: **DWT$C_NVALUE_CHANGED_CALLBACK**
C binding: **DwtNvalueChangedCallback**

The callback routine or routines called when the value of the simple text widget changes. The reason for this callback is **Value Changed**. The default is null.

insertion_point_visible
VAX binding: **DWT$C_NINSERTION_POINT_VISIBLE**
C binding: **DwtNinsertionPointVisible**

A Boolean attribute that specifies whether the insertion point is marked by a blinking text cursor. If true, the insertion point is marked by a blinking text cursor. The default is true.

auto_show_insert_point
VAX binding: **DWT$C_NAUTO_SHOW_INSERT_POINT**
C binding: **DwtNautoShowInsertPoint**

A Boolean attribute that, if true, ensures that the text visible in the simple text widget window contains the insertion point. This means that if the insertion point changes, the contents of the simple text widget window might scroll in order to bring the insertion point into the window. The default is true.

insertion_position

VAX binding: **DWT$C_NINSERTION_POSITION**
C binding: **DwtNinsertionPosition**

An integer indicating the current location of the insertion point. The default is zero.

foreground

VAX binding: **DWT$C_NFOREGROUND**
C binding: **DwtNforeground**

The pixel for the foreground of the simple text widget. The default is the current server default foreground.

font

VAX binding: **DWT$C_NFONT**
C binding: **DwtNfont**

The font list to be used for the simple text widget. The default is the current server font list.

blink_rate

VAX binding: **DWT$C_NBLINK_RATE**
C binding: **DwtNblinkRate**

An integer indicating the blink rate of the text cursor in milliseconds. The default is 500 milliseconds.

scroll_left_side

VAX binding: **DWT$C_NSCROLL_LEFT_SIDE**
C binding: **DwtNscrollLeftSide**

A Boolean attribute that, if true, indicates that the vertical scroll bar should be placed on the left side of the simple text window. This argument is ignored if the **scroll_vertical** attribute is false. The default is false.

half_border

VAX binding: **DWT$C_NHALF_BORDER**
C binding: **DwtNhalfBorder**

A Boolean attribute that, if true, specifies that a border is displayed only on the left and bottom edge of the simple text widget. The default is true.

pending_delete

VAX binding: **DWT$C_NPENDING_DELETE**
C binding: **DwtNpendingDelete**

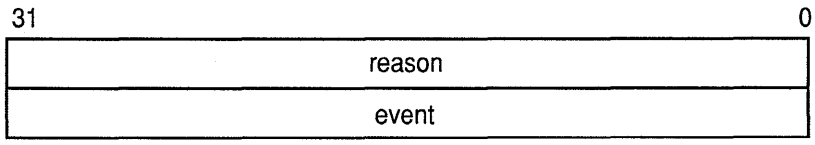
A Boolean attribute that, if true, specifies that selected text containing the insertion point is deleted when new text is entered. The default is true.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by S TEXT CREATE:

- The defaults for **width** and **height** are set as large as necessary to display the rows and columns with the given margin width and margin height.

CALLBACK DATA STRUCTURE



ZK-0091A-GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Focus

VAX binding: **DWT$C_CRFOCUS**
C binding: **DwtCRFocus**

The simple text widget has received the input focus.

Lost Focus

VAX binding: **DWT$C_CRLOST_FOCUS**
C binding: **DwtCRLostFocus**

The simple text widget has lost the input focus.

Value Changed

VAX binding: **DWT$C_CRVALUE_CHANGED**
C binding: **DwtCRValueChanged**

The user changed the value of the text string in the simple text widget.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

S TEXT CREATE creates a simple text widget. This widget enables the application to display a single or multiline field of text for input and editing by the user. By default the text window expands or shrinks as the user enters or deletes text characters. Note that the text window does not shrink below the initial size set at creation time.

A simple text widget can also be created with the high-level routine S TEXT.

geometry management

The simple text widget does not support children.

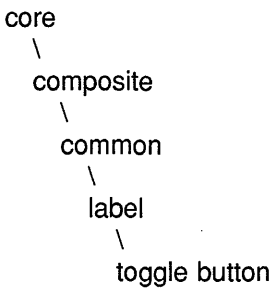
resizing

The simple text widget does not support children.

TOGGLE BUTTON CREATE

Creates a toggle button widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$TOGGLE_BUTTON_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
shape	uns byte	uns byte	read	value
visible_when_off	Boolean	uns byte	read	value
spacing	uns longword	uns longword	read	value
pixmap_on	uns longword	uns longword	read	value
pixmap_off	pixmap	uns longword	read	value
value	Boolean	uns byte	read	value
arm_callback	callback	uns longword	read	reference

Attribute	Usage	Data Type	Access	Mechanism
disarm_callback	callback	uns longword	read	reference
value_changed_ callback	callback	uns longword	read	reference
indicator	Boolean	uns byte	read	value
accelerator_text	comp string	uns longword	read	reference
button_accelerator	char string	char string	read	reference

MIT C FORMAT

widget = **DwtToggleButtonCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtToggleButtonCreate(parent_widget, name,
                             override_arglist, override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_ arglist**:

```
unsigned char   shape;
Boolean         visible_when_off;
short           spacing;
Pixmap          pixmap_on;
Pixmap          pixmap_off;
Boolean         value;
DwtCallbackPtr  arm_callback;
DwtCallbackPtr  disarm_callback;
DwtCallbackPtr  value_changed_callback;
Boolean         indicator;
DwtCompString   *accelerator_text;
char            *button_accelerator;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

shape
VAX binding: **DWT$C_NSHAPE**
C binding: **DwtNshape**

The toggle button indicator shape, which can be either rectangular or oval. The shape is used only when the **label_type** attribute is defined as text. The predefined values for this attribute are the following:

VMS	C	Description
DWT$C_RECTANGULAR	DwtRectangular	A rectangular toggle button indicator (default)
DWT$C_OVAL	DwtOval	An oval toggle button indicator (used for radio buttons)

Programmers cannot replace the indicator with their own pixmap.

visible_when_off
VAX binding: **DWT$C_NVISIBLE_WHEN_OFF**
C binding: **DwtNvisibleWhenOff**

A Boolean attribute that, if true, specifies that the button is visible when in the off state. The default is true.

spacing
VAX binding: **DWT$C_NSPACING**
C binding: **DwtNspacing**

The number of pixels between the label and the button if **label_type** is not pixmap. The default is 4 pixels.

pixmap_on
VAX binding: **DWT$C_NPIXMAP_ON**
C binding: **DwtNpixmapOn**

The pixmap to be used as the button label if **label_type** is pixmap and the toggle button is in the on state. The default is null.

pixmap_off
VAX binding: **DWT$C_NPIXMAP_OFF**
C binding: **DwtNpixmapOff**

The pixmap to be used as the button label if **label_type** is pixmap and the toggle button is in the off state. The default is null.

value
VAX binding: **DWT$C_NVALUE**
C binding: **DwtNvalue**

The value of the toggle button, which can be either true or false. The default is false.

arm_callback
VAX binding: **DWT$C_NARM_CALLBACK**
C binding: **DwtNarmCallback**

The callback routine or routines called when the toggle button is armed. The toggle button is armed when the user presses and releases MB1 while the pointer is inside the toggle button widget. For this routine, the callback reason is **Arm**. The default is null.

disarm_callback
VAX binding: **DWT$C_NDISARM_CALLBACK**
C binding: **DwtNdisarmCallback**

The callback routine or routines called when the toggle button is disarmed. The button is disarmed when the user presses MB1 while the pointer is inside the toggle button widget, but moves the pointer outside the toggle button before releasing MB1. For this routine, the callback reason is **Disarm**. The default is null.

value_changed_callback
VAX binding: **DWT$C_NVALUE_CHANGED_CALLBACK**
C binding: **DwtNvalueChangedCallback**

The callback routine or routines called because the value has changed. For this routine, the callback reason is **Value Changed**. The default is null.

indicator
VAX binding: **DWT$C_NINDICATOR**
C binding: **DwtNindicator**

A Boolean attribute that, if true, specifies that the indicator is present in the toggle button. If false, the indicator is not present. The default is true if the label is text, and false if the label is pixmap.

accelerator_text
VAX binding: **DWT$C_NACCELERATOR_TEXT**
C binding: **DwtNacceratorText**

The text displayed with the accelerator. The default is null.

button_accelerator
VAX binding: **DWT$C_NBUTTON_ACCELERATOR**
C binding: **DwtNbuttonAccelerator**

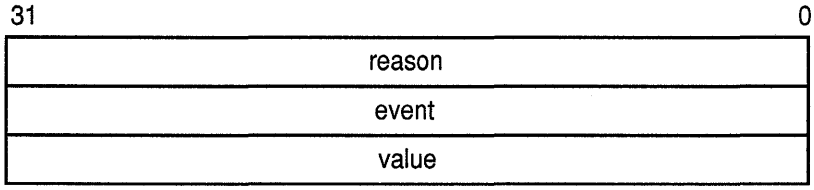
The accelerator on a push button widget. This attribute is the same as the common argument **translations** except that only the left side of the table is passed as a character string, not compiled. The application is responsible for calling the intrinsic routine INSTALL ALL ACCELERATORS to install the accelerator where the application needs it. See the *VMS DECwindows Guide to Application Programming* for information on translation tables. The default is null.

ATTRIBUTE EXCEPTIONS

The following common attributes are supported differently by TOGGLE BUTTON CREATE:

- The default for **width** equals the width of the label or pixmap plus three times **margin_width**, plus the width of the indicator.
- The default for **height** equals the height of the label or pixmap plus two times **margin_height**.
- The default for **border_width** is zero.

CALLBACK DATA STRUCTURE



ZK-0096A-GE

VAX field information

Structure name: DWT$TOGGLE_CB_ST

Name	Usage	Data Type	Access	Mechanism
toggle_reason	callback reason	longword	read	value
toggle_event	event	uns longword	read	reference
toggle_value	longword	longword	read	value

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
      int     value;
} DwtToggleButtonCallbackStruct
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

value

The state of the toggle button when the callback occurred.

CALLBACK REASONS

Value Changed

VAX binding: **DWT$C_CRVALUE_CHANGED**
C binding: **DwtCRValueChanged**

The user activated the toggle button to change the state.

Arm

VAX binding: **DWT$C_CRARM**
C binding: **DwtCRArm**

The user armed the toggle button by pressing MB1 while the pointer was inside the toggle button widget.

Disarm

VAX binding: **DWT$C_CRDISARM**
C binding: **DwtCRDisarm**

The user disarmed the toggle button by pressing MB1 while the pointer was inside the toggle button widget, and not releasing it until after moving the pointer outside the toggle button widget.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

TOGGLE BUTTON CREATE creates a toggle button widget. The toggle button widget consists of either a label and indicator button combination or simply a pixmap (icon). Toggle buttons imply a bi-state (true/false). Note that the callback data structure also includes **value**. This allows the callback procedure to pass the status of the toggle switch back to the application.

If **label_type** is pixmap, that icon is used for both the on and off states of the toggle button.

A toggle button widget can also be created with the high-level routine TOGGLE BUTTON.

geometry management

The toggle button widget follows the same rules for geometry management as its superclass the label widget, described in the low-level routine LABEL CREATE.

resizing

The sizing is affected by spacing, the font, and the label. See the low-level routine LABEL CREATE for more information.

Indicator size is based on the height of the toggle button minus twice the margin height. The indicator width is equal to the indicator height.

The default margin height is 4 pixels. The default margin width is 5 pixels.

WINDOW CREATE

Creates a window widget.

WIDGET CLASS HIERARCHY

core
\
composite
\
common
\
window

VAX FORMAT

widget = **DWT$WINDOW_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attribute can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
expose_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtWindowCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtWindowCreate(parent_widget, name, override_arglist,
                       override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attribute can be set in the **override_arglist**:

```
DwtCallbackPtr  expose_callback;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

expose_callback

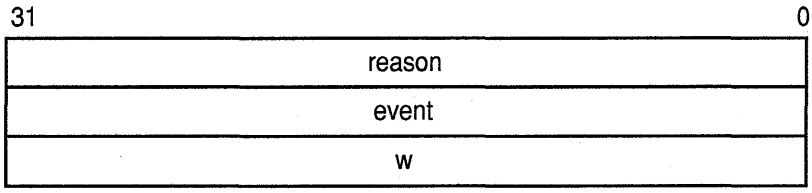
The callback routine or routines called when the window had an expose event. The reason for this callback is **Expose**. The default is null.

ATTRIBUTE EXCEPTIONS

The following common attributes are not supported by WINDOW CREATE:

- **font**
- **help_callback**

CALLBACK DATA STRUCTURE



ZK-0097A-GE

VAX field information

Structure name: DWT$WINDOW_CB_ST

Name	Usage	Data Type	Access	Mechanism
window_reason	callback reason	longword	read	value
window_event	event	uns longword	read	reference
window_w	window	uns longword	read	value

MIT C field information

```
typedef struct {
      int           reason;
      XExposeEvent  *event;
      Window        w;
} DwtWindowCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

w

The X window identifier where the expose event occurred.

CALLBACK REASONS

Expose

VAX binding: **DWT$C_CREXPOSE**
C binding: **DwtCRExpose**

An expose event occurred.

DESCRIPTION

WINDOW CREATE creates a window widget.

This routine creates a window in which applications can display graphics.

A window widget can also be created with the high-level routine WINDOW.

geometry management

Because a window widget does not support children, it does not accept geometry requests.

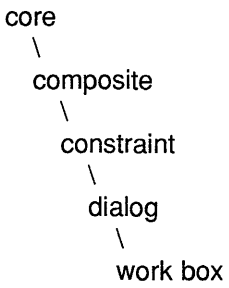
resizing

The window widget does not support children.

WORK BOX CREATE

Creates a work box widget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$WORK_BOX_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	widget	uns longword	write	value
parent_widget	widget	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

The following widget-specific attributes can be set in the **override_arglist**:

Attributes	Usage	Data Type	Access	Mechanism
label	comp string	uns longword	read	reference
cancel_label	comp string	uns longword	read	reference
cancel_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtWorkBoxCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtWorkBoxCreate(parent_widget, name, override_arglist,
                        override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

Attributes described in the low-level routine CAUTION BOX CREATE (with exceptions) and the following widget-specific attributes can be set in the **override_arglist**:

```
DwtCompString   label;
DwtCompString   cancel_label;
DwtCallbackPtr  cancel_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

label
VAX binding: **DWT$C_NLABEL**
C binding: **DwtNlabel**

The text for the message line or lines. The default is the widget name.

cancel_label
VAX binding: **DWT$C_NANCEL_LABEL**
C binding: **DwtNcancelLabel**

The label for the Cancel push button. If the label is a null string, the button is not displayed. The default is "Cancel".

cancel_callback
VAX binding: **DWT$C_NCANCEL_CALLBACK**
C binding: **DwtNcancelCallback**

The callback routine or routines called when the Cancel button is displayed. The reason for this callback is **Cancel**. The default is null.

ATTRIBUTE EXCEPTIONS

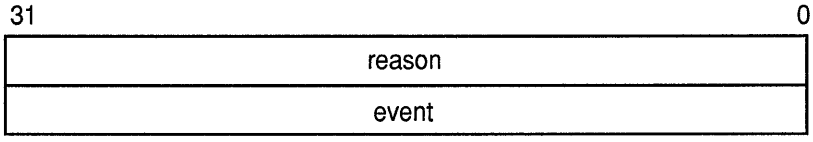
The following attributes of the low-level routine DIALOG BOX POPUP CREATE are not supported:

- **cancel_button**
- **child_overlap**
- **default_button**
- **text_merge_translations**
- **units**
- **direction_r_to_l**

The following attributes of the low-level routine DIALOG BOX CREATE are supported differently:

- The default for **margin_width** is 12.
- The default for **margin_height** is 10.
- The default for **resize** is Shrink Wrap.
- The default for **style** is Modal.

CALLBACK DATA STRUCTURE



ZK–0091A–GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	reference

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Cancel

VAX binding: **DWT$C_NCANCEL**
C binding: **DwtNcancel**

The user activated the Cancel push button.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help somewhere in the work box.

DESCRIPTION

WORK BOX CREATE creates a work box widget. The work box widget is a dialog box that allows the application to display work-in-progress messages to the user. When the application determines that an operation will take longer than five seconds, it is recommended that the application call this routine to display a work box with a message such as *Work in Progress/Please Wait.*

The work box can contain a push button labeled Cancel Operation. Do not include the push button if the operation cannot be canceled. If the style is Modal when the user selects the Cancel push button, the widget is cleared from screen but not destroyed. The widget can be displayed again by using the intrinsic routine MANAGE CHILD.

A work box widget can also be created with the high-level routine WORK BOX.

geometry management

The work box widget follows the same rules for geometry management as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

resizing

The work box widget follows the same rules for resizing as its superclass the dialog box widget, described in the low-level routine DIALOG BOX CREATE.

9 Gadget Creation Routines

Gadget creation routines allow application programmers to create gadgets for use in their applications. Gadgets are simple, windowless widgets. These low-level gadget creation routines have corresponding low-level widget creation routines.

For concepts related to gadget routines and information about how to use them, see the *VMS DECwindows Guide to Application Programming*.

Table 9–1 lists the supported gadget creation routines.

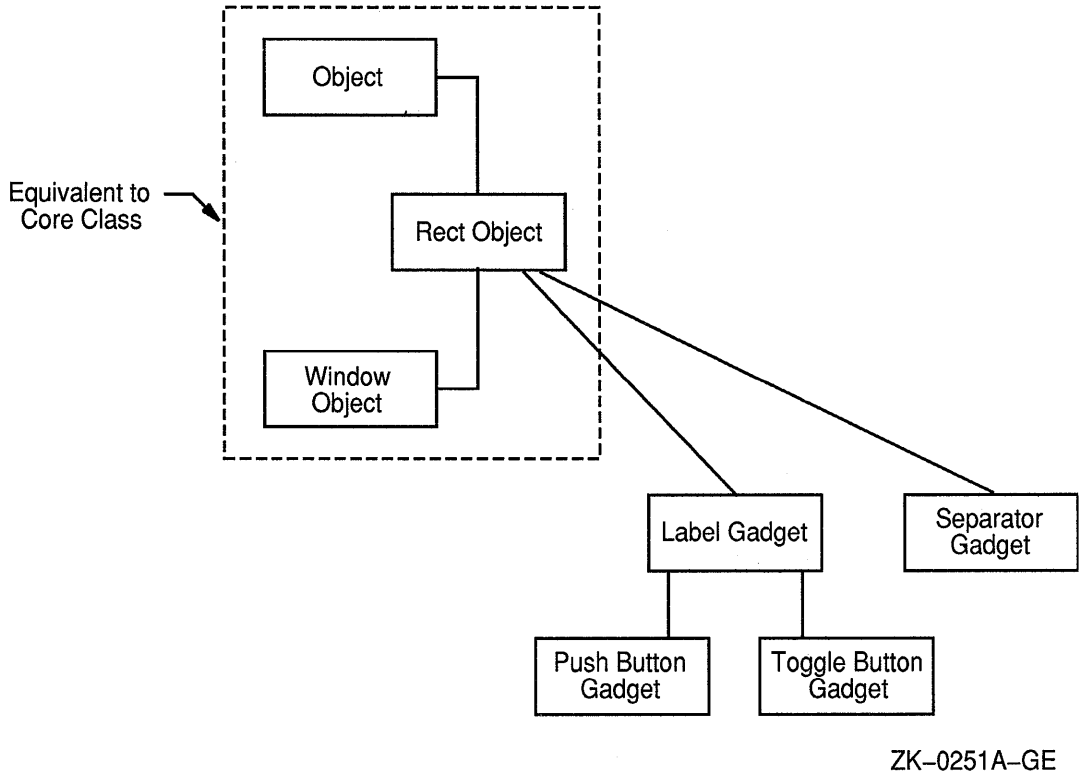
Table 9–1 Gadget Creation Routines

Routine Name	Description
LABEL GADGET CREATE	Creates a label gadget.
PUSH BUTTON GADGET CREATE	Creates a push button gadget.
SEPARATOR GADGET CREATE	Creates a separator gadget.
TOGGLE BUTTON GADGET CREATE	Creates a toggle button gadget.

9.1 Gadget Hierarchy

Like their low-level widget counterparts, gadgets are arranged in a hierarchy. The core widget class shown in Figure 9–1 is actually composed of three parts: object, rect object, and window object. All gadgets are subclasses of rect object. Figure 9–1 shows the gadget hierarchy.

Figure 9–1 Gadget Hierarchy



Gadgets support only those core attributes that belong to rect object, not the entire set of core attributes as their low-level widget counterparts. The core attributes supported by gadgets are the following:

- **x**
- **y**
- **width**
- **height**
- **border_width**
- **sensitive**
- **ancestor_sensitive**
- **destroy_callback**

For information on these attributes, see Section 8.2.

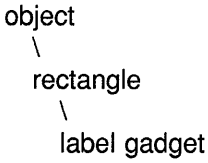
9.2 Gadget Creation Routines

The following pages describe the XUI Toolkit gadget creation routines.

LABEL GADGET CREATE

Creates a label gadget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$LABEL_GADGET_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	longword	longword	write	value
parent_widget	uns longword	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
label	comp string	uns longword	read	reference
alignment	uns longword	uns longword	read	value
direction_r_to_l	Boolean	uns longword	read	value
help_callback	callback	uns longword	read	reference

MIT C FORMAT

widget = **DwtLabelGadgetCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtLabelGadgetCreate(parent_widget, name,
                            override_arglist,
                            override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in **override_arglist:**

```
DwtCompString   label;
uns char        alignment;
Boolean         direction_r_to_l;
DwtCallbackPtr  help_callback;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

label
VAX binding: **DWT$C_NLABEL**
C binding: **DwtNlabel**

The label for the label widget. The default is the widget name.

alignment
VAX binding: **DWT$C_NALIGNMENT**
C binding: **DwtNalignment**

The label alignment for the text style. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ALIGNMENT_CENTER	DwtAlignmentCenter	Center alignment (default)
DWT$C_ALIGNMENT_BEGINNING	DwtAlignmentBeginning	Alignment at the beginning
DWT$C_ALIGNMENT_END	DwtAlignmentEnd	Alignment at the end

direction_r_to_l

VAX binding: **DWT$C_NDIRECTION_R_TO_L**
C binding: **DwtNdirectionRToL**

The direction in which the text is drawn. If false, text is drawn from left to right. If true, text is drawn from right to left. The default is false.

help_callback

VAX binding: **DWT$C_NHELP_CALLBACK**
C binding: **DwtNhelpCallback**

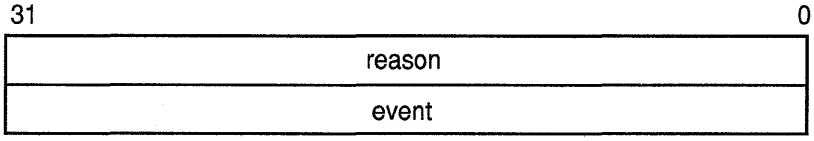
The callback descriptor data structure. This data structure defines the application routines to be called back on a help request. The default is no callback.

ATTRIBUTE EXCEPTIONS

The following core attributes are supported differently by LABEL GADGET CREATE:

- The default for **width** is the width of the label plus margins.
- The default for **height** is the height of the label plus margins.
- The default for **border_width** is 0 pixels.

CALLBACK DATA STRUCTURE



ZK-0091A-GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
reason	callback reason	longword	read	value
event	event	uns longword	read	value

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

LABEL GADGET CREATE creates a label gadget.

A label gadget is similar in appearance and semantics to a label widget. Like all gadgets, LABEL GADGET CREATE does not have a window but uses the window of the closest antecedent widget. Consequently, the antecedent widget provides all event dispatching for the gadget. This currently restricts gadgets to being descendants of menu or dialog class (or subclass) widgets. Drawing information, such as font and color, is also controlled by the closest antecedent widget.

geometry management

Because a label gadget is not a subclass of composite, children are not supported.

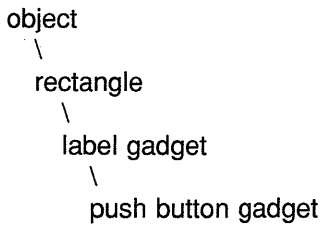
resizing

By default, the label gadget is as large as necessary to display the label.

PUSH BUTTON GADGET CREATE

Creates a push button gadget.

WIDGET CLASS HIERARCHY



VAX FORMAT

widget = **DWT$PUSH_BUTTON_GADGET_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	longword	longword	write	value
parent_widget	uns longword	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
activate_callback	callback	uns longword	read	reference
label	comp string	uns longword	read	reference
accelerator_text	comp string	uns longword	read	reference
button_accelerator	char string	char string	read	reference

MIT C FORMAT

widget = **DwtPushButtonGadgetCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtPushButtonGadgetCreate(parent_widget, name,
                                 override_arglist,
                                 override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
DwtCallbackPtr  activate_callback;
DwtCompString   label;
DwtCompString   accelerator_text;
char            *button_accelerator;
```

RETURNS

widget
The identifier of the created widget.

ARGUMENTS

parent_widget
The identifier of the parent widget.

name
The name of the created widget.

override_arglist
The application override argument list.

override_argcount
The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

activate_callback
VAX binding: **DWT$C_NACTIVATE_CALLBACK**
C binding: **DwtNactivateCallback**

A widget-specific callback routine called when the push button is activated. The button is activated when the user presses and releases MB1 while the pointer is inside the gadget.

label
VAX binding: **DWT$C_NLABEL**
C binding: **DwtNlabel**

The push button label. The default is the widget name.

accelerator_text
VAX binding: **DWT$C_NACCELERATOR_TEXT**
C binding: **DwtNacceleratorText**

Compound string text to be displayed for the accelerator. The default is no accelerator.

button_accelerator

VAX binding: **DWT$C_NBUTTON_ACCELERATOR**
C binding: **DwtNbuttonAccelerator**

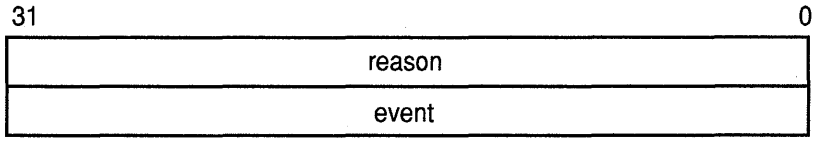
Sets an accelerator on a push button widget. Same as the common argument **translations** except that only the left side of the table is to be passed as a character string, not compiled. The application is responsible for calling the intrinsic routine INSTALL ACCELERATOR to install the accelerator where the application needs it. See the *VMS DECwindows Guide to Application Programming* for information on translation tables. The default is null.

ATTRIBUTE EXCEPTIONS

The following core attributes are supported differently by PUSH BUTTON GADGET CREATE:

- The default for **width** is the width of the label plus margins.
- The default for **height** is the height of the label plus margins.
- The default for **border_width** is 1 pixel.

CALLBACK DATA STRUCTURE



ZK-0091A-GE

VAX field information

Structure name: DWT$ANY_CB_ST

Name	Usage	Data Type	Access	Mechanism
any_reason	callback reason	longword	read	value
any_event	event	uns longword	read	value

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
} DwtAnyCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason.

event

A pointer to the X event structure describing the event that generated this callback.

CALLBACK REASONS

Activated

VAX binding: **DWT$C_CRACTIVATED**
C binding: **DwtCRActivated**

The user activated the push button.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtHelpRequested**

The user selected help.

DESCRIPTION

PUSH BUTTON GADGET CREATE creates a push button gadget.

The sizing is affected by the font and the label. See the low-level widget routine LABEL GADGET CREATE for more information.

geometry management

Because a push button gadget is not a subclass of composite, children are not supported.

resizing

The push button gadget does nothing on a Resize command by its parents.

SEPARATOR GADGET CREATE

Creates a separator gadget.

WIDGET CLASS HIERARCHY

object
\
rectangle
\
separator gadget

VAX FORMAT

widget = **DWT$SEPARATOR_GADGET_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	longword	longword	write	value
parent_widget	uns longword	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
orientation	uns longword	uns longword	read	value

MIT C FORMAT

widget = **DwtSeparatorGadgetCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtSeparatorGadgetCreate(parent_widget, name,
                                override_arglist,
                                override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
int  orientation;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

orientation

VAX binding: **DWT$C_NORIENTATION**
C binding: **DwtNorientation**

The orientation of the separator gadget. The predefined values for this attribute are as follows:

VAX	C	Description
DWT$C_ORIENTATION_HORIZONTAL	DwtOrientationHorizontal	Horizontal separator (default)
DWT$C_ORIENTATION_VERTICAL	DwtOrientationVertical	Vertical separator

A separator gadget draws a centered single-pixel line between the appropriate margins. For example, a horizontal separator draws a horizontal line from the left margin to the right margin. It is placed vertically in the center of the gadget.

ATTRIBUTE EXCEPTIONS

The following core attributes are supported differently by SEPARATOR GADGET CREATE:

- The default **width** is 3 pixels.
- The default **height** is 3 pixels.
- The default for **border_width** is 0 pixels.

DESCRIPTION

SEPARATOR GADGET CREATE creates a separator gadget.

A separator gadget is similar in appearance and semantics to a separator widget. Like all gadgets, SEPARATOR GADGET CREATE does not have a window but uses the window of the closest antecedent widget. Consequently, the antecedent widget provides all event dispatching for the gadget. This currently restricts gadgets to being descendants of menu or dialog class (or subclass) widgets.

geometry management

Because a separator gadget is not a subclass of composite, children are not supported.

resizing

The separator gadget does nothing on a resize by its parents.

TOGGLE BUTTON GADGET CREATE

Creates a toggle button gadget.

WIDGET CLASS HIERARCHY

```
object
  \
  rectangle
    \
    label gadget
      \
      toggle button gadget
```

VAX FORMAT

widget = **DWT$TOGGLE_BUTTON_GADGET_CREATE** *(parent_widget, name, override_arglist, override_argcount)*

argument information

Argument	Usage	Data Type	Access	Mechanism
widget	longword	longword	write	value
parent_widget	uns longword	uns longword	read	reference
name	char string	char string	read	descriptor
override_arglist	arglist	uns longword	read	reference
override_argcount	uns longword	uns longword	read	reference

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

Attribute	Usage	Data Type	Access	Mechanism
shape	uns longword	uns longword	read	value
value	Boolean	uns longword	read	value
value_changed_ callback	callback	uns longword	read	reference
accelerator_text	comp string	uns longword	read	reference
button_accelerator	char string	char string	read	reference

MIT C FORMAT

widget = **DwtToggleButtonGadgetCreate** *(parent_widget, name, override_arglist, override_argcount)*

argument information

```
Widget DwtToggleButtonGadgetCreate(parent_widget, name,
                                   override_arglist,
                                   override_argcount)
      Widget   parent_widget;
      char     *name;
      ArgList  override_arglist;
      int      override_argcount;
```

attribute information

The following widget-specific attributes can be set in the **override_arglist**:

```
int             shape;
Boolean         value;
DwtCallbackPtr  value_changed_callback;
DwtCompString   accelerator_text;
char            *button_accelerator;
```

RETURNS

widget

The identifier of the created widget.

ARGUMENTS

parent_widget

The identifier of the parent widget.

name

The name of the created widget.

override_arglist

The application override argument list.

override_argcount

The number of arguments in the application override argument list.

WIDGET-SPECIFIC ATTRIBUTES

shape

VAX binding: **DWT$C_NSHAPE**
C binding: **DwtNshape**

The toggle button indicator shape. The predefined values for this attribute are as follows:

VMS	C	Description
DWT$C_RECTANGULAR	DwtRectangular	A rectangular toggle button indicator (default)
DWT$C_OVAL	DwtOval	An oval toggle button indicator (used for radio buttons)

Programmers cannot replace the indicator with their own pixmap.

value

VAX binding: **DWT$C_NVALUE**
C binding: **DwtNvalue**

The value of the toggle button, which can be either true or false. The default is false.

value_changed_callback

VAX binding: **DWT$C_NVALUE_CHANGED_CALLBACK**
C binding: **DwtNvalueChangedCallback**

A widget-specific callback routine that enables the application to request a callback because the value has changed. For this routine the callback reason is **Value Changed**. The default is null.

accelerator_text

VAX binding: **DWT$C_NACCELERATOR_TEXT**
C binding: **DwtNacceleratorText**

Text displayed with the accelerator. The default is no accelerator.

button_accelerator

VAX binding: **DWT$C_NBUTTON_ACCELERATOR**
C binding: **DwtNbuttonAccelerator**

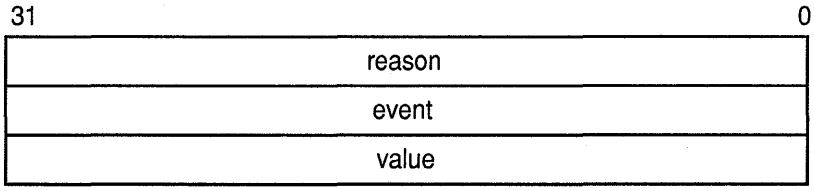
Sets an accelerator on a toggle button gadget. This argument is the same as the common attribute **translations** except that only the left side of the table is to be passed as a character string, not compiled. The application is responsible for calling the intrinsic routine INSTALL ACCELERATOR to install the accelerator where the application needs it. See the *VMS DECwindows Guide to Application Programming* for information on translation tables. The default is null.

ATTRIBUTE EXCEPTIONS

The following core attributes are supported differently by TOGGLE BUTTON GADGET CREATE:

- The default for **width** is the width of the label plus margins.
- The default for **height** is the height of the label plus margins.
- The default for **border_width** is 0 pixels.

CALLBACK DATA STRUCTURE



ZK-0096A-GE

VAX field information

Structure name: DWT$TOGGLE_CB_ST

Name	Usage	Data Type	Access	Mechanism
toggle_reason	callback reason	longword	read	value
toggle_event	event	uns longword	read	value
toggle_value	Boolean	longword	read	value

MIT C field information

```
typedef struct {
      int     reason;
      XEvent  *event;
      int     value;
} DwtToggleButtonCallbackStruct;
```

CALLBACK FIELD DESCRIPTIONS

reason

An integer set to the callback reason. See the Callback Reasons section for the values applicable to this widget.

event

A pointer to the X event structure describing the event that generated this callback.

value

The state of the toggle button when the callback occurred.

CALLBACK REASONS

Value Changed

VAX binding: **DWT$C_CRVALUE_CHANGED**
C binding: **DwtCRValueChanged**

The user changed the state of the toggle button.

Help Requested

VAX binding: **DWT$C_CRHELP_REQUESTED**
C binding: **DwtCRHelpRequested**

The user selected help.

DESCRIPTION

TOGGLE BUTTON GADGET CREATE creates a toggle button gadget.

A toggle button gadget is similar in appearance and semantics to a toggle button widget. Like all gadgets, TOGGLE BUTTON GADGET CREATE does not have a window but uses the window of the closest antecedent widget. Consequently, the antecedent widget provides all even dispatching for the gadget. This currently restricts gadgets to being descendants of menu or dialog class (or subclass) widgets.

resizing

The sizing is affected by the font and the label. See the low-level widget routine TOGGLE BUTTON CREATE for more information.

The indicator size is based on the height of the toggle button. The indicator width is equal to the indicator height.

A Summary of Widget Attributes (VAX Binding)

This appendix contains tables listing all attributes for each widget. The attributes are listed according to the widget class hierarchy beginning with the superclass widget. Any widget-specific exceptions to the inherited defaults are listed in the defaults column.

Each table contains the following information:

- Attribute name (same for VAX and C binding)
- VAX name for the attribute
- VAX data type
- Default value

A.1 Attached Dialog Box

See Table A–1 for a summary of attached dialog box widget attributes.

Table A–1 Attached Dialog Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	5 pixels
height	DWT$C_NHEIGHT	uns word	5 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null

(continued on next page)

Summary of Widget Attributes (VAX Binding)

A.1 Attached Dialog Box

Table A–1 (Cont.) Attached Dialog Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Dialog Box Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
units	DWT$C_NUNITS	uns byte	DWT$C_FONT_UNITS
style	DWT$C_NSTYLE	uns byte	DWT$C_WORK_AREA
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Null
margin_width	DWT$C_NMARGIN_WIDTH	uns word	1 pixel
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	1 pixel
default_position	DWT$C_NDEFAULT_POSITION	uns byte	False
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	True
resize	DWT$C_NRESIZE	uns byte	DWT$C_RESIZE_GROW_ ONLY

(continued on next page)

Table A–1 (Cont.) Attached Dialog Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
default_horizontal_ offset	DWT$C_NDEFAULT_ HORIZONTAL_OFFSET	longword	0 pixels
default_vertical_offset	DWT$C_NDEFAULT_VERTICAL_ OFFSET	longword	0 pixels
rubber_positioning	DWT$C_NRUBBER_POSITIONING	uns byte	uns byte
fraction_base	DWT$C_NFRACTION_BASE	longword	100

A.2 Attached Dialog Box Popup

See Table A–2 for a summary of attached dialog box pop-up widget attributes:

Table A–2 Attached Dialog Box Pop-Up Attributes

Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	0 pixels
height	DWT$C_NHEIGHT	uns word	0 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window

(continued on next page)

Table A–2 (Cont.) Attached Dialog Box Pop-Up Attributes

Core Widget Attributes			
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Dialog Box Pop-Up Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
units	DWT$C_NUNITS	uns byte	DWT$C_FONT_UNITS
style	DWT$C_NSTYLE	uns byte	DWT$C_MODELESS
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Null
margin_width	DWT$C_NMARGIN_WIDTH	uns word	3 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	3 pixels
default_position	DWT$C_NDEFAULT_POSITION	uns byte	False
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	True
resize	DWT$C_NRESIZE	uns byte	DWT$C_RESIZE_GROW_ ONLY
no_resize	DWT$C_NNO_RESIZE	uns byte	True
title	DWT$C_NTITLE	uns longword	Widget name
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null

(continued on next page)

Table A–2 (Cont.) Attached Dialog Box Pop-Up Attributes

Dialog Box Pop-Up Attributes			
take_focus	DWT$C_NTAKE_FOCUS	uns byte	True for modal dialog box; false for modeless dialog box
auto_unmanage	DWT$C_NAUTO_UNMANAGE	uns byte	True
default_button	DWT$C_NDEFAULT_BUTTON	uns longword	Null
cancel_button	DWT$C_NCANCEL_BUTTON	uns longword	Null
Widget-Specific Attributes			
default_horizontal_offset	DWT$C_NDEFAULT_HORIZONTAL_OFFSET	longword	0 pixels
default_vertical_offset	DWT$C_NDEFAULT_VERTICAL_OFFSET	longword	0 pixels
rubber_positioning	DWT$C_NRUBBER_POSITIONING	uns byte	uns byte
fraction_base	DWT$C_NFRACTION_BASE	longword	100

A.3 Caution Box

See Table A–3 for a summary of caution box widget attributes.

Table A–3 Caution Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	5 pixels
height	DWT$C_NHEIGHT	uns word	5 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True

(continued on next page)

Table A–3 (Cont.) Caution Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Dialog Box Pop-Up Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	Not supported
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
units	DWT$C_NUNITS	uns byte	Not supported
title	DWT$C_NTITLE	uns longword	Widget name
style	DWT$C_NSTYLE	uns byte	DWT$C_MODAL
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Not supported
margin_width	DWT$C_NMARGIN_WIDTH	uns word	12 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	10 pixels
default_position	DWT$C_NDEFAULT_POSITION	uns byte	False
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	Not supported

(continued on next page)

Table A–3 (Cont.) Caution Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Dialog Box Pop-Up Attributes			
resize	DWT$C_NRESIZE	uns byte	DWT$C_RESIZE_SHRINK_WRAP
take_focus	DWT$C_NTAKE_FOCUS	uns byte	True for modal dialog box; false for modeless dialog box
no_resize	DWT$C_NNO_RESIZE	uns byte	True
auto_unmanage	DWT$C_NAUTO_UNMANAGE	uns byte	True
default_button	DWT$C_NDEFAULT_BUTTON	uns longword	Not supported
cancel_button	DWT$C_NCANCEL_BUTTON	uns longword	Not supported
Widget-Specific Attributes			
label	DWT$C_NLABEL	uns longword	Widget name
yes_label	DWT$C_NYES_LABEL	uns longword	"Yes"
no_label	DWT$C_NNO_LABEL	uns longword	"No"
cancel_label	DWT$C_NCANCEL_LABEL	uns longword	"Cancel"
default_push_button	DWT$C_NDEFAULT_PUSH_BUTTON	uns longword	Yes button
yes_callback	DWT$C_NYES_CALLBACK	uns longword	Null
no_callback	DWT$C_NNO_CALLBACK	uns longword	Null
cancel_callback	DWT$C_NCANCEL_CALLBACK	uns longword	Null

A.4 Command Window

See Table A–4 for a summary of command window widget attributes.

Table A–4 Command Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	Zero
height	DWT$C_NHEIGHT	uns word	Zero
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel

(continued on next page)

Table A–4 (Cont.) Command Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Dialog Box Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	Not supported
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
units	DWT$C_NUNITS	uns byte	Not supported
style	DWT$C_NSTYLE	uns byte	DWT$C_MODAL
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Not supported
margin_width	DWT$C_NMARGIN_WIDTH	uns word	12 pixels

(continued on next page)

Table A–4 (Cont.) Command Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Dialog Box Attributes			
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	10 pixels
default_position	DWT$C_NDEFAULT_POSITION	uns byte	True
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	Not supported
resize	DWT$C_NRESIZE	uns byte	Not supported
Widget-Specific Attributes			
value	DWT$C_NVALUE	char string	Null
prompt	DWT$C_NPROMPT	uns longword	">"
lines	DWT$C_NLINES	uns longword	2 lines
history	DWT$C_NHISTORY	char string	Null string
command_entered_ callback	DWT$C_NCOMMAND_ENTERED_ CALLBACK	uns longword	Null
value_changed_ callback	DWT$C_NVALUE_CHANGED_ CALLBACK	uns longword	Null
t_translation	DWT$C_NT_TRANSLATION	uns longword	Null

A.5 Dialog Box

See Table A–5 for a summary of dialog box widget attributes.

Table A–5 Dialog Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	5 pixels
height	DWT$C_NHEIGHT	uns word	5 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map

(continued on next page)

Table A–5 (Cont.) Dialog Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Widget-Specific Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	Not used
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
units	DWT$C_NUNITS	uns byte	DWT$C_FONT_UNITS
style	DWT$C_NSTYLE	uns byte	DWT$C_WORKAREA
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Null
margin_width	DWT$C_NMARGIN_WIDTH	uns word	1 pixel
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	1 pixel
default_position	DWT$C_NDEFAULT_POSITION	uns byte	False
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	True
resize	DWT$C_NRESIZE	uns byte	DWT$C_RESIZE_GROW_ ONLY

A.6 Dialog Box Popup

See Table A–6 for a summary of dialog box pop-up widget attributes.

Table A–6 Dialog Box Pop-Up Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	0 pixels
height	DWT$C_NHEIGHT	uns word	0 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Widget-Specific Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	Not used
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key

(continued on next page)

Table A–6 (Cont.) Dialog Box Pop-Up Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
units	DWT$C_NUNITS	uns byte	DWT$C_FONT_UNITS
style	DWT$C_NSTYLE	uns byte	DWT$C_MODELESS
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Null
margin_width	DWT$C_NMARGIN_WIDTH	uns word	5 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	5 pixels
default_position	DWT$C_NDEFAULT_POSITION	uns byte	False
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	True
resize	DWT$C_NRESIZE	uns byte	DWT$C_RESIZE_GROW_ ONLY
no_resize	DWT$C_NNO_RESIZE	uns byte	True
title	DWT$C_NTITLE	uns longword	Widget name
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
take_focus	DWT$C_NTAKE_FOCUS	uns byte	True for modal dialog box; false for modeless dialog box
auto_unmanage	DWT$C_NAUTO_UNMANAGE	uns byte	True
default_button	DWT$C_NDEFAULT_BUTTON	uns longword	Null
cancel_button	DWT$C_NCANCEL_BUTTON	uns longword	Null

A.7 File Selection

See Table A–7 for a summary of file selection widget attributes.

Table A–7 File Selection Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Centered in the parent window
y	DWT$C_NY	longword	Centered in the parent window

(continued on next page)

Table A–7 (Cont.) File Selection Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
width	DWT$C_NWIDTH	uns word	Width of the list box, plus the width of the push buttons, plus three times **margin_ width**
height	DWT$C_NHEIGHT	uns word	Height of the list box, plus the height of the text edit field, plus the height of the label, plus three times **margin_height**
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Dialog Box Pop-Up Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.

(continued on next page)

Table A–7 (Cont.) File Selection Attributes

Attribute	VAX Name	VAX Data Type	Default
Dialog Box Pop-Up Attributes			
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
units	DWT$C_NUNITS	uns byte	DWT$C_FONT_UNITS
style	DWT$C_NSTYLE	uns byte	DWT$C_MODELESS
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Null
margin_width	DWT$C_NMARGIN_WIDTH	uns word	5 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	5 pixels
default_position	DWT$C_NDEFAULT_POSITION	uns byte	False
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	True
resize	DWT$C_NRESIZE	uns byte	DWT$C_RESIZE_GROW_ ONLY
no_resize	DWT$C_NNO_RESIZE	uns byte	True
title	DWT$C_NTITLE	uns longword	"Open"
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
take_focus	DWT$C_NTAKE_FOCUS	uns byte	True for modal dialog box; false for modeless dialog box
auto_unmanage	DWT$C_NAUTO_UNMANAGE	uns byte	True
default_button	DWT$C_NDEFAULT_BUTTON	uns longword	Null
cancel_button	DWT$C_NCANCEL_BUTTON	uns longword	Null
Selection Attributes			
label	DWT$C_NLABEL	uns longword	"Items"
value	DWT$C_NVALUE	uns longword	Null string
selection_label	DWT$C_NSELECTION_LABEL	uns longword	"Files in"
ok_label	DWT$C_NOK_LABEL	uns longword	"OK"
cancel_label	DWT$C_NCANCEL_LABEL	uns longword	"Cancel"
activate_callback	DWT$C_NACTIVATE_CALLBACK	uns longword	Null
cancel_callback	DWT$C_NCANCEL_CALLBACK	uns longword	Null
no_match_callback	DWT$C_NNO_MATCH_CALLBACK	uns longword	Null

(continued on next page)

Table A–7 (Cont.) File Selection Attributes

Attribute	VAX Name	VAX Data Type	Default
Selection Attributes			
visible_item_count	DWT$C_NVISIBLE_ITEMS_COUNT	uns longword	8 items
items	DWT$C_NITEMS	uns longword	Null
item_count	DWT$C_NITEMS_COUNT	uns longword	Zero
must_match	DWT$C_NMUST_MATCH	uns byte	False
Widget-Specific Attributes			
filter_label	DWT$C_NFILTER_LABEL	uns longword	"File filter"
apply_label	DWT$C_NAPPLY_LABEL	uns longword	"Filter"
dir_mask	DWT$C_NDIR_MASK	uns longword	"*.*"
dir_spec	DWT$C_NDIR_SPEC	uns longword	Null string
file_search_proc	DWT$C_NFILE_SEARCH_PROC	Proc entry mask	Default file search procedure
list_updated	DWT$C_NLIST_UPDATED	uns byte	False

A.8 Help

See Table A–8 for a summary of help widget attributes.

Table A–8 Help Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	Cannot be set by the caller
height	DWT$C_NHEIGHT	uns word	Cannot be set by the caller
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True

(continued on next page)

Table A–8 (Cont.) Help Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Widget-Specific Attributes			
about_label	DWT$C_NABOUT_LABEL	uns longword	"About"
add_topic_label	DWT$C_NADD_TOPIC_LABEL	uns longword	"Additional topics:"
application_name	DWT$C_NAPPLICATION_NAME	uns longword	Null
badframe_message	DWT$C_NBADFRAME_MESSAGE	uns longword	"Couldn't find frame %s in %x library\n"
badlib_message	DWT$C_NBADLIB_MESSAGE	uns longword	"Couldn't open %s library \n"
cols	DWT$C_NCOLS	uns longword	55 character cells
copy_label	DWT$C_NCOPY_LABEL	uns longword	"Copy"
default_position	DWT$C_NDEFAULT_POSITION	uns byte	True
dismiss_label	DWT$C_NDISMISS_LABEL	uns longword	"Dismiss"

(continued on next page)

Table A–8 (Cont.) Help Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
edit_label	DWT$C_NEDIT_LABEL	uns longword	"Edit"
erroropen_message	DWT$C_NERROROPEN_ MESSAGE	uns longword	"Error opening file %s\n"
exit_label	DWT$C_NEXIT_LABEL	uns longword	"Exit"
file_label	DWT$C_NFILE_LABEL	uns longword	"File"
first_topic	DWT$C_NFIRST_TOPIC	uns longword	Null
glossary_label	DWT$C_NGLOSSARY_LABEL	uns longword	"Glossary"
glossary_topic	DWT$C_NGLOSSARY_TOPIC	uns longword	Null
goback_label	DWT$C_NGOBACK_LABEL	uns longword	"Go Back"
goover_label	DWT$C_NGOOVER_LABEL	uns longword	"Go to Overview"
goto_label	DWT$C_NGOTO_LABEL	uns longword	"Go To"
help_font	DWT$C_NHELP_FONT	uns longword	The default help font
help_label	DWT$C_NHELP_LABEL	uns longword	"Help"
helpmessage_title	DWT$C_NHELPMESSAGE_TITLE	uns longword	"Message"
helpmessage_title_ type	DWT$C_NHELPMESSAGE_TITLE_ TYPE	unsigned char	DWT$C_CSTRNG
history_label	DWT$C_NHISTORY_LABEL	uns longword	"History..."
historybox_label	DWT$C_NHISTORYBOX_LABEL	uns longword	"Help Topic History"
keyword_label	DWT$C_NKEYWORD_LABEL	uns longword	"Keyword..."
keywords_label	DWT$C_NKEYWORDS_LABEL	uns longword	"Keyword: "
library_spec	DWT$C_NLIBRARY_SPEC	uns longword	Null
library_type	DWT$C_NLIBRARY_TYPE	uns longword	DWT$C_TEXT_LIBRARY
nokeyword_message	DWT$C_NNOKEYWORD_ MESSAGE	uns longword	"Couldn't find keyword %s\n"
notitle_message	DWT$C_NNOTITLE_MESSAGE	uns longword	"No title to match string%s\n"
nulllib_message	DWT$C_NNULLLIB_MESSAGE	uns longword	"No library specified\n"
nulltopic_message	DWT$C_NNULLTOPIC_MESSAGE	uns longword	"No first topic and overview topic specified\n"
overview_topic	DWT$C_NOVERVIEW_TOPIC	uns longword	Null
rows	DWT$C_NROWS	uns longword	20 lines
saveas_label	DWT$C_NSAVEAS_LABEL	uns longword	"Save As..."
searchapply_label	DWT$C_NSEARCHAPPLY_LABEL	uns longword	"Apply"
searchkeywordbox_ label	DWT$C_ NSEARCHKEYWORDBOX_LABEL	uns longword	"Search Topic Keywords"
search_label	DWT$C_NSEARCH_LABEL	uns longword	"Search"

(continued on next page)

Table A–8 (Cont.) Help Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
searchtitlebox_label	DWT$C_NSEARCHTITLEBOX_ LABEL	uns longword	"Search Topic Titles"
selectall_label	DWT$C_NSELECTALL_LABEL	uns longword	"Select All"
title_label	DWT$C_NTITLE_LABEL	uns longword	"Title..."
titles_label	DWT$C_NTITLES_LABEL	uns longword	"Title:"
topictitles_label	DWT$C_NTOPICTITLES_LABEL	uns longword	"Topic Titles:"
view_label	DWT$C_NVIEW_LABEL	uns longword	"View"
visitglos_label	DWT$C_NVISITGLOS_LABEL	uns longword	"Visit Glossary"
visit_label	DWT$C_NVISIT_LABEL	uns longword	"Visit"
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null

A.9 Label

See Table A–9 for a summary of label widget attributes.

Table A–9 Label Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	Width of the label or pixmap, plus two times **margin_ width**
height	DWT$C_NHEIGHT	uns word	Height of the label or pixmap, plus two times **margin_height**
border_width	DWT$C_NBORDER_WIDTH	uns word	0 pixels
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True

(continued on next page)

Table A–9 (Cont.) Label Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Widget-Specific Attributes			
label_type	DWT$C_NLABEL_TYPE	uns byte	DWT$C_CSTRNG
label	DWT$C_NLABEL	uns longword	Widget name
margin_width	DWT$C_NMARGIN_WIDTH	uns word	2 pixels for text; 0 pixels for pixmap
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	2 pixels for text; 0 pixels for pixmap
alignment	DWT$C_NALIGNMENT	uns byte	DWT$C_ALIGNMENT_CENTER
pixmap	DWT$C_NPIXMAP	uns longword	Null

(continued on next page)

Table A–9 (Cont.) Label Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
margin_left	DWT$C_NMARGIN_LEFT	uns word	0 pixels
margin_right	DWT$C_NMARGIN_RIGHT	uns word	0 pixels
margin_top	DWT$C_NMARGIN_TOP	uns word	0 pixels
margin_bottom	DWT$C_NMARGIN_BOTTOM	uns word	0 pixels
conform_to_text	DWT$C_NCONFORM_TO_TEXT	uns byte	True if width and height are zero; false if width and height are not zero

A.10 List Box

See Table A–10 for a summary of list box widget attributes.

Table A–10 List Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	Dimension	As large as necessary to hold the longest item without exceeding the size of its parent
height	DWT$C_NHEIGHT	uns word	As large as necessary to hold the number of items specified by **visible_item_count** without exceeding the size of its parent
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True

(continued on next page)

Table A–10 (Cont.) List Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	Not supported
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Not supported
Scroll Window Attributes			
h_scroll	DWT$C_NHORIZONTAL_SCROLL_BAR	uns longword	Null
v_scroll	DWT$C_NVERTICAL_SCROLL_BAR	uns longword	Null
work_window	DWT$C_NWORK_WINDOW	uns longword	Null
shown_value_automatic_horiz	DWT$C_NSHOWN_VALUE_AUTOMATIC_HORIZ	uns byte	True
shown_value_automatic_vert	DWT$C_NSHOWN_VALUE_AUTOMATIC_VERT	uns byte	False

(continued on next page)

Table A–10 (Cont.) List Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
margin_width	DWT$C_NMARGIN_WIDTH	uns word	10 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	4 pixels
spacing	DWT$C_NSPACING	uns word	1 pixel
items	DWT$C_NITEMS	comp string	Null
item_count	DWT$C_NITEMS_COUNT	uns longword	Zero
selected_items	DWT$C_NSELECTED_ITEMS	uns longword	Null
selected_items_count	DWT$C_NSELECTED_ITEMS_ COUNT	uns longword	Zero
visible_item_count	DWT$C_NVISIBLE_ITEMS_COUNT	uns longword	As many items as can fit in the core attribute **height**
single_selection	DWT$C_NSINGLE_SELECTION	uns byte	True
resize	DWT$C_NRESIZE	uns byte	DWT$C_GROW_ONLY
horiz	DWT$C_NHORIZONTAL	uns byte	False
single_callback	DWT$C_NSINGLE_CALLBACK	uns longword	Null
single_confirm_ callback	DWT$C_NSINGLE_CONFIRM_ CALLBACK	uns longword	Null
extend_callback	DWT$C_NEXTEND_CALLBACK	uns longword	Null
extend_confirm_ callback	DWT$C_NEXTEND_CONFIRM_ CALLBACK	uns longword	Null

A.11 Main Window

See Table A–11 for a summary of main window widget attributes.

Table A–11 Main Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	5 pixels
height	DWT$C_NHEIGHT	uns word	5 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null

(continued on next page)

Table A–11 (Cont.) Main Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Not supported
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Not supported
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	Not supported
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Widget-Specific Attributes			
command_window	DWT$C_NCOMMAND_WINDOW	uns longword	Null
work_window	DWT$C_NWORK_WINDOW	uns longword	Null
menu_bar	DWT$C_NMENU_BAR	uns longword	Null

(continued on next page)

Table A–11 (Cont.) Main Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
horizontal_scroll_bar	DWT$C_NHORIZONTAL_SCROLL_BAR	uns longword	Null
vertical_scroll_bar	DWT$C_NVERTICAL_SCROLL_BAR	uns longword	Null
accept_focus	DWT$C_NACCEPT_FOCUS	uns byte	False
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null

A.12 Menu Bar

See Table A–12 for a summary of menu bar widget attributes.

Table A–12 Menu Bar Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	16 pixels
height	DWT$C_NHEIGHT	uns word	Number of lines needed to display all entries
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null

(continued on next page)

Table A–12 (Cont.) Menu Bar Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	The default XUI Toolkit font; used only by gadget children
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Menu Attributes			
orientation	DWT$C_NORIENTATION	uns byte	DWT$C_ORIENTATION_ VERTICAL
spacing	DWT$C_NSPACING	uns word	0 pixels
adjust_margin	DWT$C_NADJUST_MARGIN	uns byte	True
margin_width	DWT$C_NMARGIN_WIDTH	uns word	3 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	0 pixels
entry_border	DWT$C_NENTRY_BORDER	uns word	0 pixels
menu_alignment	DWT$C_NMENU_ALIGNMENT	uns byte	True
entry_alignment	DWT$C_NENTRY_ALIGNMENT	uns byte	DWT$C_ALIGNMENT_ BEGINNING
menu_packing	DWT$C_NMENU_PACKING	uns byte	DWT$C_MENU_PACKING_ TIGHT
menu_num_columns	DWT$C_NMENU_NUM_COLUMNS	uns longword	1 row or column
menu_radio	DWT$C_NMENU_RADIO	uns byte	False except for radio boxes which default to true

(continued on next page)

Table A–12 (Cont.) Menu Bar Attributes

Attribute	VAX Name	VAX Data Type	Default
Menu Attributes			
radio_always_one	DWT$C_NRADIO_ALWAYS_ONE	uns byte	True
menu_is_ homogeneous	DWT$C_NMENU_IS_ HOMOGENEOUS	uns byte	False except for radio boxes, which default to true
menu_entry_class	DWT$C_NMENU_ENTRY_CLASS	identifier	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DWT$C_NMENU_HISTORY	uns longword	Zero
entry_callback	DWT$C_NENTRY_CALLBACK	uns longword	Null
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
menu_help_widget	DWT$C_NMENU_HELP_WIDGET	uns longword	Null

A.13 Menu Attributes

See Table A–13 for a summary of menu widget attributes.

Table A–13 Menu Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	If **orientation** is vertical, **width** is the maximum entry **width** or 16 pixels. If **orientation** is horizontal, **width** is the sum of **width** and **spacing**.
height	DWT$C_NHEIGHT	uns word	If **orientation** is vertical, **height** is the sum of **height** and **spacing** or 16 pixels. If **orientation** is horizontal, **height** is the maximum entry **height** or 16 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null

(continued on next page)

Table A–13 (Cont.) Menu Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Widget-Specific Attributes			
orientation	DWT$C_NORIENTATION	uns byte	DWT$C_ORIENTATION_VERTICAL
spacing	DWT$C_NSPACING	uns word	0 pixels
adjust_margin	DWT$C_NADJUST_MARGIN	uns byte	True
margin_width	DWT$C_NMARGIN_WIDTH	uns word	3 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	3 pixels
entry_border	DWT$C_NENTRY_BORDER	uns word	0 pixels

(continued on next page)

A.13 Menu Attributes

Table A–13 (Cont.) Menu Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
menu_alignment	DWT$C_NMENU_ALIGNMENT	uns byte	True
entry_alignment	DWT$C_NENTRY_ALIGNMENT	uns byte	DWT$C_ALIGNMENT_ BEGINNING
menu_packing	DWT$C_NMENU_PACKING	uns byte	DWT$C_MENU_PACKING_ TIGHT for all menu types except for radio boxes. Radio boxes default to DWT$C_MENU_PACKING_ COLUMN.
menu_num_columns	DWT$C_NMENU_NUM_COLUMNS	uns longword	1 column
menu_radio	DWT$C_NMENU_RADIO	uns byte	False except for radio boxes, which default to true
radio_always_one	DWT$C_NRADIO_ALWAYS_ONE	uns byte	True
menu_is_ homogeneous	DWT$C_NMENU_IS_ HOMOGENEOUS	uns byte	False except for radio boxes, which default to true
menu_entry_class	DWT$C_NMENU_ENTRY_CLASS	identifier	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DWT$C_NMENU_HISTORY	uns longword	Null
entry_callback	DWT$C_NENTRY_CALLBACK	uns longword	Null
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
help_widget	DWT$C_NHELP_WIDGET	uns longword	Null

A.14 Menu Popup

See Table A–14 for a summary of menu pop-up widget attributes.

Table A–14 Menu Pop-Up Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	As large as necessary to hold all child widgets

(continued on next page)

Table A–14 (Cont.) Menu Pop-Up Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
height	DWT$C_NHEIGHT	uns word	As large as necessary to hold all child widgets
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null

(continued on next page)

A.14 Menu Popup

Table A–14 (Cont.) Menu Pop-Up Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
orientation	DWT$C_NORIENTATION	uns byte	DWT$C_ORIENTATION_ VERTICAL
spacing	DWT$C_NSPACING	uns word	0 pixels
adjust_margin	DWT$C_NADJUST_MARGIN	uns byte	True
margin_width	DWT$C_NMARGIN_WIDTH	uns word	3 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	3 pixels
entry_border	DWT$C_NENTRY_BORDER	uns word	0 pixels
menu_alignment	DWT$C_NMENU_ALIGNMENT	uns byte	True
entry_alignment	DWT$C_NENTRY_ALIGNMENT	uns byte	DWT$C_ALIGNMENT_ BEGINNING
menu_packing	DWT$C_NMENU_PACKING	uns byte	DWT$C_MENU_PACKING_ TIGHT for all menu types except for radio boxes. Radio boxes default to DWT$C_MENU_PACKING_ COLUMN.
menu_num_columns	DWT$C_NMENU_NUM_COLUMNS	uns longword	1 column
menu_radio	DWT$C_NMENU_RADIO	uns byte	False except for radio boxes, which default to true
radio_always_one	DWT$C_NRADIO_ALWAYS_ONE	uns byte	True
menu_is_ homogeneous	DWT$C_NMENU_IS_ HOMOGENEOUS	uns byte	False except for radio boxes, which default to true
menu_entry_class	DWT$C_NMENU_ENTRY_CLASS	identifier	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DWT$C_NMENU_HISTORY	uns longword	Null
entry_callback	DWT$C_NENTRY_CALLBACK	uns longword	Null
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
help_widget	DWT$C_NHELP_WIDGET	uns longword	Null

A.15 Menu Pulldown

See Table A–15 for a summary of menu pull-down widget attributes.

Table A–15 Menu Pull-Down Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Not supported
y	DWT$C_NY	longword	Not supported
width	DWT$C_NWIDTH	uns word	As large as necessary to hold all child widgets
height	DWT$C_NHEIGHT	uns word	As large as necessary to hold all child widgets
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null

(continued on next page)

Table A–15 (Cont.) Menu Pull-Down Attributes

Attribute	VAX Name	VAX Data Type	Default
Menu Attributes			
orientation	DWT$C_NORIENTATION	uns byte	DWT$C_ORIENTATION_ VERTICAL
spacing	DWT$C_NSPACING	uns word	0 pixels
adjust_margin	DWT$C_NADJUST_MARGIN	uns byte	True
margin_width	DWT$C_NMARGIN_WIDTH	uns word	3 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	3 pixels
entry_border	DWT$C_NENTRY_BORDER	uns word	0 pixels
menu_alignment	DWT$C_NMENU_ALIGNMENT	uns byte	True
entry_alignment	DWT$C_NENTRY_ALIGNMENT	uns byte	DWT$C_ALIGNMENT_ BEGINNING
menu_packing	DWT$C_NMENU_PACKING	uns byte	DWT$C_MENU_PACKING_ TIGHT for all menu types except for radio boxes. Radio boxes default to DWT$C_MENU_PACKING_ COLUMN.
menu_num_columns	DWT$C_NMENU_NUM_COLUMNS	uns longword	1
menu_radio	DWT$C_NMENU_RADIO	uns byte	False except for radio boxes, which default to true
radio_always_one	DWT$C_NRADIO_ALWAYS_ONE	uns byte	True
menu_is_ homogeneous	DWT$C_NMENU_IS_ HOMOGENEOUS	uns byte	False except for radio boxes, which default to true
menu_entry_class	DWT$C_NMENU_ENTRY_CLASS	identifier	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DWT$C_NMENU_HISTORY	uns longword	Null
entry_callback	DWT$C_NENTRY_CALLBACK	uns longword	Null
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
help_widget	DWT$C_NHELP_WIDGET	uns longword	Null

A.16 Message Box

See Table A–16 for a summary of message box widget attributes.

Table A–16 Message Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	5 pixels
height	DWT$C_NHEIGHT	uns word	5 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Dialog Box Pop-Up Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	Not supported
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null

(continued on next page)

A.16 Message Box

Table A–16 (Cont.) Message Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Dialog Box Pop-Up Attributes			
units	DWT$C_NUNITS	uns byte	Not supported
title	DWT$C_NTITLE	uns longword	Widget name
style	DWT$C_NSTYLE	uns byte	DWT$C_MODAL
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Not supported
margin_width	DWT$C_NMARGIN_WIDTH	uns word	12 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	10 pixels
default_position	DWT$C_NDEFAULT_POSITION	uns byte	False
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	Not supported
resize	DWT$C_NRESIZE	uns byte	DWT$C_SHRINK_WRAP
take_focus	DWT$C_NTAKE_FOCUS	uns byte	True for modal dialog box; false for modeless dialog box
no_resize	DWT$C_NNO_RESIZE	uns byte	True
auto_unmanage	DWT$C_NAUTO_UNMANAGE	uns byte	True
default_button	DWT$C_NDEFAULT_BUTTON	uns longword	Not supported
cancel_button	DWT$C_NCANCEL_BUTTON	uns longword	Not supported
Widget-Specific Attributes			
label	DWT$C_NLABEL	uns longword	Widget name
ok_label	DWT$C_NOK_LABEL	uns longword	"Acknowledged"
yes_callback	DWT$C_NYES_CALLBACK	uns longword	Null

A.17 Option Menu

See Table A–17 for a summary of option menu widget attributes.

Table A–17 Option Menu Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	As large as necessary to hold all child widgets
height	DWT$C_NHEIGHT	uns word	As large as necessary to hold all child widgets
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False

(continued on next page)

Table A–17 (Cont.) Option Menu Attributes

Attribute	VAX Name	VAX Data Type	Default
Common Widget Attributes			
font	DWT$C_NFONT	uns longword	The default XUI Toolkit font; used only by gadget children
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Menu Attributes			
orientation	DWT$C_NORIENTATION	uns byte	DWT$C_ORIENTATION_ VERTICAL
spacing	DWT$C_NSPACING	uns word	0 pixels
adjust_margin	DWT$C_NADJUST_MARGIN	uns byte	True
margin_width	DWT$C_NMARGIN_WIDTH	uns word	3 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	0 pixels
entry_border	DWT$C_NENTRY_BORDER	uns word	0 pixels
menu_alignment	DWT$C_NMENU_ALIGNMENT	uns byte	True
entry_alignment	DWT$C_NENTRY_ALIGNMENT	uns byte	DWT$C_ALIGNMENT_ BEGINNING
menu_packing	DWT$C_NMENU_PACKING	uns byte	DWT$C_MENU_PACKING_ TIGHT for all menu types except for radio boxes. Radio boxes default to DWT$C_MENU_PACKING_ COLUMN.
menu_num_columns	DWT$C_NMENU_NUM_COLUMNS	uns longword	1
menu_radio	DWT$C_NMENU_RADIO	uns byte	False except for radio boxes, which default to true
radio_always_one	DWT$C_NRADIO_ALWAYS_ONE	uns byte	True
menu_is_ homogeneous	DWT$C_NMENU_IS_ HOMOGENEOUS	uns byte	False except for radio boxes, which default to true
menu_entry_class	DWT$C_NMENU_ENTRY_CLASS	identifier	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DWT$C_NMENU_HISTORY	uns longword	Null
entry_callback	DWT$C_NENTRY_CALLBACK	uns longword	Null
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
help_widget	DWT$C_NHELP_WIDGET	uns longword	Null

(continued on next page)

Table A–17 (Cont.) Option Menu Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
label	DWT$C_NLABEL	uns longword	Widget name
sub_menu_id	DWT$C_NSUB_MENU_ID	uns longword	Zero

A.18 Pull Down Menu Entry

See Table A–18 for a summary of pull-down menu entry widget attributes.

Table A–18 Pull-Down Menu Entry Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	Width of the label or pixmap, plus the hotspot width, plus two times **margin_width**
height	DWT$C_NHEIGHT	uns word	Height of the label or pixmap, plus two times **margin_height**
border_width	DWT$C_NBORDER_WIDTH	uns word	0 pixels
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window

(continued on next page)

Table A–18 (Cont.) Pull-Down Menu Entry Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Label Attributes			
label_type	DWT$C_NLABEL_TYPE	uns byte	DWT$C_CSTRNG
label	DWT$C_NLABEL	uns longword	Widget name
margin_width	DWT$C_NMARGIN_WIDTH	uns word	2 pixels for text; 0 pixels for pixmap
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	2 pixels for text; 0 pixels for pixmap
alignment	DWT$C_NALIGNMENT	uns byte	DWT$C_ALIGNMENT_CENTER
pixmap	DWT$C_NPIXMAP	uns longword	Null
margin_left	DWT$C_NMARGIN_LEFT	uns word	0 pixels
margin_right	DWT$C_NMARGIN_RIGHT	uns word	0 pixels
margin_top	DWT$C_NMARGIN_TOP	uns word	0 pixels
margin_bottom	DWT$C_NMARGIN_BOTTOM	uns word	0 pixels
conform_to_text	DWT$C_NCONFORM_TO_TEXT	uns byte	True if width and height are zero; false if width and height are not zero

(continued on next page)

Table A–18 (Cont.) Pull-Down Menu Entry Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
sub_menu_id	DWT$C_NSUB_MENU_ID	uns longword	Null
activate_callback	DWT$C_NACTIVATE_CALLBACK	uns longword	Null
pulling_callback	DWT$C_NPULLING_CALLBACK	uns longword	Null
hot_spot_pixmap	DWT$C_NHOT_SPOT_PIXMAP	uns longword	Null

A.19 Push Button

See Table A–19 for a summary of push button widget attributes.

Table A–19 Push Button Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	Width of the label or pixmap, plus two times **margin_ width**
height	DWT$C_NHEIGHT	uns word	Height of the label or pixmap, plus two times **margin_height**
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null

(continued on next page)

Table A–19 (Cont.) Push Button Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Label Attributes			
label_type	DWT$C_NLABEL_TYPE	uns byte	DWT$C_CSTRING
label	DWT$C_NLABEL	uns longword	Widget name
margin_width	DWT$C_NMARGIN_WIDTH	uns word	2 pixels for text; 0 pixels for pixmap
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	2 pixels for text; 0 pixels for pixmap
alignment	DWT$C_NALIGNMENT	uns byte	DWT$C_CENTER_ ALIGNMENT
pixmap	DWT$C_NPIXMAP	uns longword	Null
margin_left	DWT$C_NMARGIN_LEFT	uns word	0 pixels
margin_right	DWT$C_NMARGIN_RIGHT	uns word	0 pixels
margin_top	DWT$C_NMARGIN_TOP	uns word	0 pixels
margin_bottom	DWT$C_NMARGIN_BOTTOM	uns word	0 pixels
conform_to_text	DWT$C_NCONFORM_TO_TEXT	uns byte	True if width and height are zero; false if width and height are not zero

(continued on next page)

Table A–19 (Cont.) Push Button Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
border_highlight	DWT$C_NBORD_HIGHLIGHT	uns byte	False
fill_highlight	DWT$C_NFILL_HIGHLIGHT	uns byte	False
activate_callback	DWT$C_NACTIVATE_CALLBACK	uns longword	Null
arm_callback	DWT$C_NARM_CALLBACK	uns longword	Null
disarm_callback	DWT$C_NDISARM_CALLBACK	uns longword	Null
accelerator_text	DWT$C_NACCELERATOR_TEXT	uns longword	Null
button_accelerator	DWT$C_NBUTTON_ ACCELERATOR	uns longword	Null
shadow	DWT$C_NSHADOW	uns byte	True

A.20 Radio Box

See Table A–20 for a summary of radio box widget attributes.

Table A–20 Radio Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	As large as necessary to hold all child widgets
height	DWT$C_NHEIGHT	uns word	As large as necessary to hold all child widgets
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes

(continued on next page)

A.20 Radio Box

Table A–20 (Cont.) Radio Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Widget-Specific Attributes			
orientation	DWT$C_NORIENTATION	uns byte	DWT$C_ORIENTATION_ VERTICAL
spacing	DWT$C_NSPACING	uns word	1 pixel
adjust_margin	DWT$C_NADJUST_MARGIN	uns byte	True
margin_width	DWT$C_NMARGIN_WIDTH	uns word	3 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	3 pixels
entry_border	DWT$C_NENTRY_BORDER	uns word	0 pixels
menu_alignment	DWT$C_NMENU_ALIGNMENT	uns byte	True
entry_alignment	DWT$C_NENTRY_ALIGNMENT	uns byte	DWT$C_ALIGNMENT_ BEGINNING

(continued on next page)

Table A–20 (Cont.) Radio Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
menu_packing	DWT$C_NMENU_PACKING	uns byte	DWT$C_MENU_PACKING_TIGHT for all menu types except for radio boxes. Radio boxes default to DWT$C_MENU_PACKING_COLUMN.
menu_num_columns	DWT$C_NMENU_NUM_COLUMNS	uns longword	1
menu_radio	DWT$C_NMENU_RADIO	uns byte	False except for radio boxes, which default to true
radio_always_one	DWT$C_NRADIO_ALWAYS_ONE	uns byte	True
menu_is_homogeneous	DWT$C_NMENU_IS_HOMOGENEOUS	uns byte	False except for radio boxes, which default to true
menu_entry_class	DWT$C_NMENU_ENTRY_CLASS	identifier	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DWT$C_NMENU_HISTORY	uns longword	Null
entry_callback	DWT$C_NENTRY_CALLBACK	uns longword	Null
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
help_widget	DWT$C_NHELP_WIDGET	uns longword	Null

A.21 Scale

See Table A–21 for a summary of scale widget attributes.

Table A–21 Scale Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	Calculated based on the scale width or height, the label widths, and the orientation

(continued on next page)

Table A–21 (Cont.) Scale Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
height	DWT$C_NHEIGHT	uns word	Calculated based on the scale width or height, the label widths, and the orientation
border_width	DWT$C_NBORDER_WIDTH	uns word	0 pixels
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null

(continued on next page)

Table A–21 (Cont.) Scale Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
value	DWT$C_NVALUE	uns longword	Zero
title	DWT$C_NTITLE	uns longword	Null
orientation	DWT$C_NORIENTATION	uns byte	DWT$C_ORIENTATION_ HORIZONTAL
scale_width	DWT$C_NSCALE_WIDTH	uns word	20 pixels for vertical scales; calculated for horizontal scales.
scale_height	DWT$C_NSCALE_HEIGHT	uns word	20 pixels for horizontal scales; calculated for vertical scales
min_value	DWT$C_NMIN_VALUE	uns longword	Zero
max_value	DWT$C_NMAX_VALUE	uns longword	100
decimal_points	DWT$C_NDECIMAL_POINTS	uns byte	Zero
show_value	DWT$C_NSHOW_VALUE	uns byte	True
slider_pixmap	DWT$C_NSLIDER_PIXMAP	uns longword	Null
drag_callback	DWT$C_NDRAG_CALLBACK	uns longword	Null
value_changed_ callback	DWT$C_NVALUE_CHANGED_ CALLBACK	uns longword	Null

A.22 Scroll Bar

See Table A–22 for a summary of scroll bar widget attributes.

Table A–22 Scroll Bar Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	For vertical scroll bars, 17 pixels; for horizontal scroll bars, the width of the parent minus 17 pixels
height	DWT$C_NHEIGHT	uns word	For horizontal scroll bars, 17 pixels; for vertical scroll bars, the height of the parent minus 17 pixels

(continued on next page)

A.22 Scroll Bar

Table A–22 (Cont.) Scroll Bar Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	Not supported
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Widget-Specific Attributes			
int_value	DWT$C_NVALUE	uns longword	Zero
min_value	DWT$C_NMIN_VALUE	uns longword	Zero
max_value	DWT$C_NMAX_VALUE	uns longword	100

(continued on next page)

Table A–22 (Cont.) Scroll Bar Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
orientation	DWT$C_NORIENTATION	uns byte	DWT$C_ORIENTATION_ HORIZONTAL
translations1	DWT$C_NTRANSLATIONS1	uns longword	Null
translations2	DWT$C_NTRANSLATIONS2	uns longword	Null
shown	DWT$C_NSHOWN	uns longword	10 units
inc	DWT$C_NINC	uns longword	10 units
page_inc	DWT$C_NPAGE_INC	uns longword	10 units
unit_inc_callback	DWT$C_NUNIT_INC_CALLBACK	uns longword	Null
unit_dec_callback	DWT$C_NUNIT_DEC_CALLBACK	uns longword	Null
page_inc_callback	DWT$C_NPAGE_INC_CALLBACK	uns longword	Null
page_dec_callback	DWT$C_NPAGE_DEC_CALLBACK	uns longword	Null
to_top_callback	DWT$C_NTO_TOP_CALLBACK	uns longword	Null
to_bottom_callback	DWT$C_NTO_BOTTOM_ CALLBACK	uns longword	Null
drag_callback	DWT$C_NDRAG_CALLBACK	uns longword	Null
value_changed_ callback	DWT$C_NVALUE_CHANGED_ CALLBACK	uns longword	Null

A.23 Scroll Window

See Table A–23 for a summary of scroll window widget attributes.

Table A–23 Scroll Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	Widget specific
height	DWT$C_NHEIGHT	uns word	Widget specific
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color

(continued on next page)

A.23 Scroll Window

Table A–23 (Cont.) Scroll Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	Not supported
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Not supported

(continued on next page)

Table A–23 (Cont.) Scroll Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
h_scroll	DWT$C_NHORIZONTAL_SCROLL_BAR	uns longword	Null
v_scroll	DWT$C_NVERTICAL_SCROLL_BAR	uns longword	Null
work_window	DWT$C_NWORK_WINDOW	uns longword	Null
shown_value_automatic_horiz	DWT$C_NSHOWN_VALUE_AUTOMATIC_HORIZ	uns byte	True
shown_value_automatic_vert	DWT$C_NSHOWN_VALUE_AUTOMATIC_VERT	uns byte	True

A.24 Selection

See Table A–24 for a summary of selection widget attributes.

Table A–24 Selection Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Centered in the parent window
y	DWT$C_NY	longword	Centered in the parent window
width	DWT$C_NWIDTH	uns word	Width of the list box, plus the width of the push buttons, plus three times **margin_width**
height	DWT$C_NHEIGHT	uns word	Height of the list box, plus the height of the text edit field, plus the height of the label, plus three times **margin_height**
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True

(continued on next page)

Table A–24 (Cont.) Selection Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Dialog Box Pop-Up Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
units	DWT$C_NUNITS	uns byte	DWT$C_FONT_UNITS
style	DWT$C_NSTYLE	uns byte	DWT$C_MODELESS
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Null
margin_width	DWT$C_NMARGIN_WIDTH	uns word	5 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	5 pixels
default_position	DWT$C_NDEFAULT_POSITION	uns byte	False
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	True
resize	DWT$C_NRESIZE	uns byte	DWT$C_RESIZE_GROW_ ONLY
no_resize	DWT$C_NNO_RESIZE	uns byte	True
title	DWT$C_NTITLE	uns longword	"Open"

(continued on next page)

Table A–24 (Cont.) Selection Attributes

Attribute	VAX Name	VAX Data Type	Default
Dialog Box Pop-Up Attributes			
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
take_focus	DWT$C_NTAKE_FOCUS	uns byte	True for modal dialog box; false for modeless dialog box
auto_unmanage	DWT$C_NAUTO_UNMANAGE	uns byte	True
default_button	DWT$C_NDEFAULT_BUTTON	uns longword	Null
cancel_button	DWT$C_NCANCEL_BUTTON	uns longword	Null
Widget-Specific Attributes			
label	DWT$C_NLABEL	uns longword	"Items"
value	DWT$C_NVALUE	uns longword	Null string
selection_label	DWT$C_NSELECTION_LABEL	uns longword	"Selection"
ok_label	DWT$C_NOK_LABEL	uns longword	"OK"
cancel_label	DWT$C_NCANCEL_LABEL	uns longword	"Cancel"
activate_callback	DWT$C_NACTIVATE_CALLBACK	uns longword	Null
cancel_callback	DWT$C_NCANCEL_CALLBACK	uns longword	Null
no_match_callback	DWT$C_NNO_MATCH_CALLBACK	uns longword	Null
visible_item_count	DWT$C_NVISIBLE_ITEMS_COUNT	uns longword	8 items
items	DWT$C_NITEMS	uns longword	Null
item_count	DWT$C_NITEMS_COUNT	uns longword	0 items
must_match	DWT$C_NMUST_MATCH	uns byte	False

A.25 Separator

See Table A–25 for a summary of separator widget attributes.

Table A–25 Separator Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	3 pixels

(continued on next page)

A.25 Separator

Table A–25 (Cont.) Separator Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
height	DWT$C_NHEIGHT	uns word	3 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	0 pixels
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Not supported
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	Not supported
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Not supported

(continued on next page)

Table A–25 (Cont.) Separator Attributes

Attribute	VAX Name	VAX Data Type	Default
Label Attributes			
label_type	DWT$C_NLABEL_TYPE	uns byte	DWT$C_CSTRING
label	DWT$C_NLABEL	uns longword	Widget name
margin_width	DWT$C_NMARGIN_WIDTH	uns word	2 pixels for text; 0 pixels for pixmap
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	2 pixels for text; 0 pixels for pixmap
alignment	DWT$C_NALIGNMENT	uns byte	Center alignment
pixmap	DWT$C_NPIXMAP	uns longword	Null
margin_left	DWT$C_NMARGIN_LEFT	uns word	0 pixels
margin_right	DWT$C_NMARGIN_RIGHT	uns word	0 pixels
margin_top	DWT$C_NMARGIN_TOP	uns word	0 pixels
margin_bottom	DWT$C_NMARGIN_BOTTOM	uns word	0 pixels
conform_to_text	DWT$C_NCONFORM_TO_TEXT	uns byte	True if width and height are zero; false if width and height are not zero
Widget-Specific Attributes			
orientation	DWT$C_NORIENTATION	uns byte	DWT$C_ORIENTATION_HORIZONTAL

A.26 S Text

See Table A–26 for a summary of simple text widget attributes.

Table A–26 S Text Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	As large as necessary to display the **rows** and **cols** with the given **margin_width** and **margin_height**

(continued on next page)

Table A–26 (Cont.) S Text Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
height	DWT$C_NHEIGHT	uns word	As large as necessary to display the **rows** and **cols** with the given **margin_width** and **margin_height**
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_ managed	DWT$C_NMAPPED_WHEN_ MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Widget-Specific Attributes			
margin_width	DWT$C_NMARGIN_WIDTH	uns word	2 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	2 pixels
cols	DWT$C_NCOLS	uns longword	20 characters
rows	DWT$C_NROWS	uns longword	1 character
top_position	DWT$C_NTOP_POSITION	uns longword	Zero
word_wrap	DWT$C_NWORD_WRAP	uns byte	False
scroll_vertical	DWT$C_NSCROLL_VERTICAL	uns byte	False
resize_height	DWT$C_NRESIZE_HEIGHT	uns byte	True
resize_width	DWT$C_NRESIZE_WIDTH	uns byte	True
value	DWT$C_NVALUE	char string	null string
editable	DWT$C_NEDITABLE	uns byte	True
max_length	DWT$C_NMAX_LENGTH	uns longword	2^{31-1}

(continued on next page)

Table A–26 (Cont.) S Text Attributes

Attribute	VAX Name	VAX Data Type	Default
Widget-Specific Attributes			
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
lost_focus_callback	DWT$C_NLOST_FOCUS_ CALLBACK	uns longword	Null
value_changed_ callback	DWT$C_NVALUE_CHANGED_ CALLBACK	uns longword	Null
insertion_point_visible	DWT$C_NINSERTION_POINT_ VISIBLE	uns byte	True
auto_show_insert_ point	DWT$C_NAUTO_SHOW_INSERT_ POINT	uns byte	True
insertion_position	DWT$C_NINSERTION_POSITION	uns longword	Zero
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
font	DWT$C_NFONT	uns longword	Current server font list
blink_rate	DWT$C_NBLINK_RATE	uns longword	500 milliseconds
scroll_left_side	DWT$C_NSCROLL_LEFT_SIDE	uns byte	False
half_border	DWT$C_NHALF_BORDER	uns byte	True
pending_delete	DWT$C_NPENDING_DELETE	uns byte	True

A.27 Toggle Button

See Table A–27 for a summary of toggle button widget attributes.

Table A–27 Toggle Button Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	Width of the label or pixmap, plus two times **margin_ width**
height	DWT$C_NHEIGHT	uns word	Height of the label or pixmap, plus two times **margin_height**
border_width	DWT$C_NBORDER_WIDTH	uns word	0 pixels
border	DWT$C_NBORDER	uns longword	Default foreground color

(continued on next page)

A.27 Toggle Button

Table A–27 (Cont.) Toggle Button Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
Label Attributes			
label_type	DWT$C_NLABEL_TYPE	uns byte	DWT$C_CSTRING
label	DWT$C_NLABEL	uns longword	Widget name
margin_width	DWT$C_NMARGIN_WIDTH	uns word	2 pixels for text; 0 pixels for pixmap
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	2 pixels for text; 0 pixels for pixmap

(continued on next page)

Table A–27 (Cont.) Toggle Button Attributes

Attribute	VAX Name	VAX Data Type	Default
Label Attributes			
alignment	DWT$C_NALIGNMENT	uns byte	Center alignment
pixmap	DWT$C_NPIXMAP	uns longword	Null
margin_left	DWT$C_NMARGIN_LEFT	uns word	0 pixels
margin_right	DWT$C_NMARGIN_RIGHT	uns word	0 pixels
margin_top	DWT$C_NMARGIN_TOP	uns word	0 pixels
margin_bottom	DWT$C_NMARGIN_BOTTOM	uns word	0 pixels
conform_to_text	DWT$C_NCONFORM_TO_TEXT	uns byte	True if width and height are zero; false if width and height are not zero
Widget-Specific Attributes			
shape	DWT$C_NSHAPE	uns byte	DWT$C_RECTANGULAR
visible_when_off	DWT$C_NVISIBLE_WHEN_OFF	uns byte	True
spacing	DWT$C_NSPACING	uns longword	4 pixels
pixmap_on	DWT$C_NPIXMAP_ON	uns longword	Null
pixmap_off	DWT$C_NPIXMAP_OFF	uns longword	Null
value	DWT$C_NVALUE	uns byte	False
arm_callback	DWT$C_NARM_CALLBACK	uns longword	Null
disarm_callback	DWT$C_NDISARM_CALLBACK	uns longword	Null
value_changed_ callback	DWT$C_NVALUE_CHANGED_ CALLBACK	uns longword	Null
indicator	DWT$C_NINDICATOR	uns byte	True when label is text; false when label is pixmap
accelerator_text	DWT$C_NACCELERATOR_TEXT	uns longword	Null
button_accelerator	DWT$C_NBUTTON_ ACCELERATOR	char string	Null

A.28 Window

See Table A–28 for a summary of window widget attributes.

Summary of Widget Attributes (VAX Binding)

A.28 Window

Table A–28 Window Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	0 pixels
height	DWT$C_NHEIGHT	uns word	0 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null
Common Widget Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	False
font	DWT$C_NFONT	uns longword	Not supported
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_translations	DWT$C_NGRAB_MERGE_TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Not supported
Widget-Specific Attribute			
expose_callback	DWT$C_NEXPOSE_CALLBACK	uns longword	Null

A.29 Work Box

See Table A–29 for a summary of work box widget attributes.

Table A–29 Work Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Core Widget Attributes			
x	DWT$C_NX	longword	Determined by the geometry manager
y	DWT$C_NY	longword	Determined by the geometry manager
width	DWT$C_NWIDTH	uns word	5 pixels
height	DWT$C_NHEIGHT	uns word	5 pixels
border_width	DWT$C_NBORDER_WIDTH	uns word	1 pixel
border	DWT$C_NBORDER	uns longword	Default foreground color
border_pixmap	DWT$C_NBORDER_PIXMAP	uns longword	Null
background	DWT$C_NBACKGROUND	uns longword	Default background color
background_pixmap	DWT$C_NBACKGROUND_PIXMAP	uns longword	Null
colormap	DWT$C_NCOLORMAP	uns longword	Default color map
sensitive	DWT$C_NSENSITIVE	uns byte	True
ancestor_sensitive	DWT$C_NANCESTOR_SENSITIVE	uns byte	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DWT$C_NACCELERATORS	uns longword	Null
depth	DWT$C_NDEPTH	uns longword	Depth of the parent window
translations	DWT$C_NTRANSLATIONS	uns longword	Null
mapped_when_managed	DWT$C_NMAPPED_WHEN_MANAGED	uns byte	True
screen	DWT$C_NSCREEN	uns longword	The parent screen
destroy_callback	DWT$C_NDESTROY_CALLBACK	uns longword	Null

(continued on next page)

A.29 Work Box

Table A–29 (Cont.) Work Box Attributes

Attribute	VAX Name	VAX Data Type	Default
Dialog Box Pop-Up Attributes			
foreground	DWT$C_NFOREGROUND	uns longword	Default foreground color
highlight	DWT$C_NHIGHLIGHT	uns longword	Default foreground color
highlight_pixmap	DWT$C_NHIGHLIGHT_PIXMAP	uns longword	Null
user_data	DWT$C_NUSER_DATA	uns longword	Null
direction_r_to_l	DWT$C_NDIRECTION_R_TO_L	uns byte	Not supported
font	DWT$C_NFONT	uns longword	XUI Toolkit font
grab_key_syms	DWT$C_NGRAB_KEY_SYMS	uns longword	Tab key
grab_merge_ translations	DWT$C_NGRAB_MERGE_ TRANSLATIONS	uns longword	See DIALOG BOX CREATE.
help_callback	DWT$C_NHELP_CALLBACK	uns longword	Null
units	DWT$C_NUNITS	uns byte	Not supported
title	DWT$C_NTITLE	uns longword	Widget name
style	DWT$C_NSTYLE	uns byte	DWT$C_MODAL
map_callback	DWT$C_NMAP_CALLBACK	uns longword	Null
unmap_callback	DWT$C_NUNMAP_CALLBACK	uns longword	Null
focus_callback	DWT$C_NFOCUS_CALLBACK	uns longword	Null
text_merge_ translations	DWT$C_NTEXT_MERGE_ TRANSLATIONS	uns longword	Not supported
margin_width	DWT$C_NMARGIN_WIDTH	uns word	12 pixels
margin_height	DWT$C_NMARGIN_HEIGHT	uns word	10 pixels
default_position	DWT$C_NDEFAULT_POSITION	uns byte	False
child_overlap	DWT$C_NCHILD_OVERLAP	uns byte	Not supported
resize	DWT$C_NRESIZE	uns byte	DWT$C_SHRINK_WRAP
take_focus	DWT$C_NTAKE_FOCUS	uns byte	True for modal dialog box; false for a modeless dialog box
no_resize	DWT$C_NNO_RESIZE	uns byte	True
auto_unmanage	DWT$C_NAUTO_UNMANAGE	uns byte	True
default_button	DWT$C_NDEFAULT_BUTTON	uns longword	Not supported
cancel_button	DWT$C_NCANCEL_BUTTON	uns longword	Not supported
Widget-Specific Attributes			
label	DWT$C_NLABEL	uns longword	Widget name
cancel_label	DWT$C_NCANCEL_LABEL	uns longword	"Cancel"
cancel_callback	DWT$C_NCANCEL_CALLBACK	uns longword	Null

B Summary of Widget Attributes (C Binding)

This appendix contains tables listing all attributes for each widget. The attributes are listed according to the widget class hierarchy beginning with the superclass widget. Any widget-specific exceptions to the inherited defaults are listed in the defaults column.

Each table contains the following information:

- Attribute name (same for VAX and C binding)
- C name for the attribute
- C data type
- Default value

B.1 Attached Dialog Box

See Table B–1 for a summary of attached dialog box widget attributes.

Table B–1 Attached Dialog Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	5 pixels
height	DwtNheight	Dimension	5 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null

(continued on next page)

Summary of Widget Attributes (C Binding)

B.1 Attached Dialog Box

Table B–1 (Cont.) Attached Dialog Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Dialog Box Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	DwtFontUnits
style	DwtNstyle	unsigned char	DwtWorkarea
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
text_merge_translations	DwtNtextMergeTranslations	XtTranslations	Null
margin_width	DwtNmarginWidth	Dimension	1 pixel
margin_height	DwtNmarginHeight	Dimension	1 pixel
default_position	DwtNdefaultPosition	Boolean	False
child_overlap	DwtNchildOverlap	Boolean	True
resize	DwtNresize	unsigned char	DwtResizeGrowOnly
grab_key_syms	DwtNgrabKeySyms	KeySym	The default array contains the Tab key symbol.
grab_merge_translations	DwtNgrabMergeTranslations	XtTranslations	See DIALOG BOX CREATE for the default syntax.

(continued on next page)

Table B–1 (Cont.) Attached Dialog Box Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
default_horizontal_ offset	DwtNdefaultHorizontalOffset	int	0 pixels
default_vertical_offset	DwtNdefaultVerticalOffset	int	0 pixels
rubber_positioning	DwtNrubberPositioning	Boolean	False
fraction_base	DwtNfractionBase	int	100

B.2 Attached Dialog Box Popup

See Table B–2 for a summary of attached dialog box pop-up widget attributes.

Table B–2 Attached Dialog Box Pop-Up Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	0 pixels
height	DwtNheight	Dimension	0 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window

(continued on next page)

B.2 Attached Dialog Box Popup

Table B–2 (Cont.) Attached Dialog Box Pop-Up Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Dialog Box Pop-Up Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	DwtFontUnits
style	DwtNstyle	unsigned char	DwtDwtModeless
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
text_merge_translations	DwtNtextMergeTranslations	XtTranslations	Null
margin_width	DwtNmarginWidth	Dimension	3 pixels
margin_height	DwtNmarginHeight	Dimension	3 pixels
default_position	DwtNdefaultPosition	Boolean	False
child_overlap	DwtNchildOverlap	Boolean	True
resize	DwtNresize	unsigned char	DwtResizeGrowOnly
no_resize	DwtNnoResize	Boolean	True
title	DwtNtitle	DwtCompString	Widget name
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
take_focus	DwtNtakeFocus	Boolean	True for modal dialog box; false for modeless dialog box
auto_unmanage	DwtNautoUnmanage	Boolean	True

(continued on next page)

Table B–2 (Cont.) Attached Dialog Box Pop-Up Attributes

Attribute	C Name	C Data Type	Default
Dialog Box Pop-Up Attributes			
default_button	DwtNdefaultButton	Widget	Null
cancel_button	DwtNcancelButton	Widget	Null
grab_key_syms	DwtNgrabKeySyms	KeySym	The default array contains the Tab key symbol.
grab_merge_ translations	DwtNgrabMergeTranslations	XtTranslations	See DIALOG BOX CREATE for the default syntax.
Widget-Specific Attributes			
default_horizontal_ offset	DwtNdefaultHorizontalOffset	int	0 pixels
default_vertical_offset	DwtNdefaultVerticalOffset	int	0 pixels
rubber_positioning	DwtNrubberPositioning	Boolean	False
fraction_base	DwtNfractionBase	int	100

B.3 Caution Box

See Table B–3 for a summary of caution box widget attributes.

Table B–3 Caution Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	5 pixels
height	DwtNheight	Dimension	5 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True

(continued on next page)

Table B–3 (Cont.) Caution Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Dialog Box Pop-Up Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	Not supported
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	Not supported
title	DwtNtitle	DwtCompString	Widget name
style	DwtNstyle	unsigned char	DwtModal
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
text_merge_translations	DwtNtextMergeTranslations	XtTranslations	Not supported
margin_width	DwtNmarginWidth	Dimension	12 pixels
margin_height	DwtNmarginHeight	Dimension	10 pixels
default_position	DwtNdefaultPosition	Boolean	False
child_overlap	DwtNchildOverlap	Boolean	Not supported
resize	DwtNresize	unsigned char	DwtResizeShrinkWrap
take_focus	DwtNtakeFocus	Boolean	True for modal dialog box; false for modeless dialog box

(continued on next page)

Table B–3 (Cont.) Caution Box Attributes

Attribute	C Name	C Data Type	Default
Dialog Box Pop-Up Attributes			
no_resize	DwtNnoResize	Boolean	True
auto_unmanage	DwtNautoUnmanage	Boolean	True
default_button	DwtNdefaultButton	Widget	Not supported
cancel_button	DwtNcancelButton	Widget	Not supported
Widget-Specific Attributes			
label	DwtNlabel	DwtCompString	Widget name
yes_label	DwtNyesLabel	DwtCompString	"Yes"
no_label	DwtNnoLabel	DwtCompString	"No"
cancel_label	DwtNcancelLabel	DwtCompString	"Cancel"
default_push_button	DwtNdefaultPushButton	unsigned char	Yes button
yes_callback	DwtNyesCallback	DwtCallbackPtr	Null
no_callback	DwtNnoCallback	DwtCallbackPtr	Null
cancel_callback	DwtNcancelCallback	DwtCallbackPtr	Null

B.4 Command Window

See Table B–4 for a summary of command window widget attributes.

Table B–4 Command Window Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	0 pixels
height	DwtNheight	Dimension	0 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map

(continued on next page)

Table B–4 (Cont.) Command Window Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Dialog Box Pop-Up Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	Not supported
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	Not supported
title	DwtNtitle	DwtCompString	Widget name
style	DwtNstyle	unsigned char	DwtModal
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
text_merge_translations	DwtNtextMergeTranslations	XtTranslations	Not supported
margin_width	DwtNmarginWidth	Dimension	12 pixels
margin_height	DwtNmarginHeight	Dimension	10 pixels
default_position	DwtNdefaultPosition	Boolean	True
child_overlap	DwtNchildOverlap	Boolean	Not supported
resize	DwtNresize	unsigned char	Not supported

(continued on next page)

Table B–4 (Cont.) Command Window Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
value	DwtNvalue	char *	Null
prompt	DwtNprompt	DwtCompString	">"
lines	DwtNlines	int	2 lines
history	DwtNhistory	char *	Null string
command_entered_ callback	DwtNcommandEnteredCallback	DwtCallbackPtr	Null
value_changed_ callback	DwtNvalueChangedCallback	DwtCallbackPtr	Null
t_translation	DwtNtTranslation	XtTranslations	Null

B.5 Dialog Box

See Table B–5 for a summary of dialog box widget attributes.

Table B–5 Dialog Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	5 pixels
height	DwtNheight	Dimension	5 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null

(continued on next page)

Table B–5 (Cont.) Dialog Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	DwtFontUnits
style	DwtNstyle	unsigned char	DwtWorkarea
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
text_merge_translations	DwtNtextMergeTranslations	XtTranslations	Null
margin_width	DwtNmarginWidth	Dimension	1 pixel
margin_height	DwtNmarginHeight	Dimension	1 pixel
default_position	DwtNdefaultPosition	Boolean	False
child_overlap	DwtNchildOverlap	Boolean	True
resize	DwtNresize	unsigned char	DwtResizeGrowOnly
grab_key_syms	DwtNgrabKeySyms	KeySym	The default array contains the Tab key symbol.
grab_merge_translations	DwtNgrabMergeTranslations	XtTranslations	See DIALOG BOX CREATE for the default syntax.

B.6 Dialog Box Popup

See Table B–6 for a summary of dialog box pop-up widget attributes.

Table B–6 Dialog Box Pop-Up Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	0 pixels
height	DwtNheight	Dimension	0 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	DwtFontUnits
style	DwtNstyle	unsigned char	DwtModeless
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null

(continued on next page)

Table B–6 (Cont.) Dialog Box Pop-Up Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
text_merge_ translations	DwtNtextMergeTranslations	XtTranslations	Null
margin_width	DwtNmarginWidth	Dimension	3 pixels
margin_height	DwtNmarginHeight	Dimension	3 pixels
default_position	DwtNdefaultPosition	Boolean	False
child_overlap	DwtNchildOverlap	Boolean	True
resize	DwtNresize	unsigned char	DwtResizeGrowOnly
no_resize	DwtNnoResize	Boolean	True
title	DwtNtitle	DwtCompString	Widget name
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
take_focus	DwtNtakeFocus	Boolean	True for modal dialog box; false for modeless dialog box
auto_unmanage	DwtNautoUnmanage	Boolean	True
default_button	DwtNdefaultButton	Widget	Null
cancel_button	DwtNcancelButton	Widget	Null
grab_key_syms	DwtNgrabKeySyms	KeySym	The default array contains the Tab key symbol.
grab_merge_ translations	DwtNgrabMergeTranslations	XtTranslations	See DIALOG BOX CREATE for the default syntax.

B.7 File Selection

See Table B–7 for a summary of file selection widget attributes.

Table B–7 File Selection Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Centered in the parent window
y	DwtNy	Position	Centered in the parent window
width	DwtNwidth	Dimension	Width of the list box, plus the width of the push buttons, plus three times **margin_ width**

(continued on next page)

Table B–7 (Cont.) File Selection Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
height	DwtNheight	Dimension	Height of the list box, plus the height of the text edit field, plus the height of the label, plus three times **margin_height**
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Dialog Box Pop-Up Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	DwtFontUnits
style	DwtNstyle	unsigned char	DwtModeless
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
text_merge_translations	DwtNtextMergeTranslations	XtTranslations	Null

(continued on next page)

Table B–7 (Cont.) File Selection Attributes

Attribute	C Name	C Data Type	Default
Dialog Box Pop-Up Attributes			
margin_width	DwtNmarginWidth	Dimension	5 pixels
margin_height	DwtNmarginHeight	Dimension	5 pixels
default_position	DwtNdefaultPosition	Boolean	False
child_overlap	DwtNchildOverlap	Boolean	True
resize	DwtNresize	unsigned char	DwtResizeGrowOnly
no_resize	DwtNnoResize	Boolean	True
title	DwtNtitle	DwtCompString	"Open"
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
take_focus	DwtNtakeFocus	Boolean	True for modal dialog box; false for modeless dialog box
auto_unmanage	DwtNautoUnmanage	Boolean	True
default_button	DwtNdefaultButton	Widget	Null
cancel_button	DwtNcancelButton	Widget	Null
Selection Attributes			
label	DwtNlabel	DwtCompString	"Items"
value	DwtNvalue	DwtCompString	Null string
selection_label	DwtNselectionLabel	DwtCompString	"Files in"
ok_label	DwtNokLabel	DwtCompString	"OK"
cancel_label	DwtNcancelLabel	DwtCompString	"Cancel"
activate_callback	DwtNactivateCallback	DwtCallbackPtr	Null
cancel_callback	DwtNcancelCallback	DwtCallbackPtr	Null
no_match_callback	DwtNnoMatchCallback	DwtCallbackPtr	Null
visible_item_count	DwtNvisibleItemsCount	int	8 items
items	DwtNitems	DwtCompString *	Null
item_count	DwtNitemsCount	int	0 items
must_match	DwtNmustMatch	Boolean	False

(continued on next page)

Table B–7 (Cont.) File Selection Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
filter_label	DwtNfilterLabel	DwtCompString	"File filter"
apply_label	DwtNapplyLabel	DwtCompString	"Filter"
dir_mask	DwtNdirMask	DwtCompString	"*.*"
dir_spec	DwtNdirSpec	DwtCompString	Null string
file_search_proc	DwtNfileSearchProc	VoidProc	Default file search procedure
list_updated	DwtNlistUpdated	Boolean	False

B.8 Help

See Table B–8 for a summary of help widget attributes.

Table B–8 Help Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	Cannot be set by the caller
height	DwtNheight	Dimension	Cannot be set by the caller
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null

(continued on next page)

Table B–8 (Cont.) Help Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
about_label	DwtNaboutLabel	DwtCompString	"About"
add_topic_label	DwtNaddTopicLabel	DwtCompString	"Additional topics:"
application_name	DwtNapplicationName	DwtCompString	Null
badframe_message	DwtNbadframeMessage	DwtCompString	"Couldn't find frame %s in %x library\n"
badlib_message	DwtNbadlibMessage	DwtCompString	"Couldn't open %s library \n"
cols	DwtNcols	int	55 character cells
copy_label	DwtNcopyLabel	DwtCompString	"Copy"
default_position	DwtNdefaultPosition	Boolean	True
dismiss_label	DwtNdismissLabel	DwtCompString	"Dismiss"
edit_label	DwtNeditLabel	DwtCompString	"Edit"
erroropen_message	DwtNerroropenMessage	DwtCompString	"Error opening file %s\n"
exit_label	DetNexitLabel	DwtCompString	"Exit"
file_label	DwtNfileLabel	DwtCompString	"File"
first_topic	DwtNfirstTopic	DwtCompString	Null
glossary_label	DwtNglossaryLabel	DwtCompString	"Glossary"
glossary_topic	DwtNglossaryTopic	DwtCompString	Null

(continued on next page)

Table B–8 (Cont.) Help Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
goback_label	DwtNgobackLabel	DwtCompString	"Go Back"
goover_label	DwtNgooverLabel	DwtCompString	"Go to Overview"
goto_label	DwtNgotoLabel	DwtCompString	"Go To"
help_font	DwtNhelpFont	DwtFontList	The default help font
help_label	DwtNhelpLabel	DwtCompString	"Help"
helpmessage_title	DwtNhelpmessageTitle	DwtCompString	"Message"
helpmessage_title_type	DwtNhelpmessageTitleType	unsigned char	DwtCString
history_label	DwtNhistoryLabel	DwtCompString	"History..."
historybox_label	DwtNhistoryboxLabel	DwtCompString	"Help Topic History"
keyword_label	DwtNkeywordLabel	DwtCompString	"Keyword..."
keywords_label	DwtNkeywordsLabel	DwtCompString	"Keyword:"
library_spec	DwtNlibrarySpec	DwtCompString	Null
library_type	DwtNlibraryType	DwtCompString	DwtTextLibrary
nokeyword_message	DwtNnokeywordMessage	DwtCompString	"Couldn't find keyword %s\n"
notitle_message	DwtNnotitleMessage	DwtCompString	"No title to match string%s\n"
nulllib_message	DwtNnulllibMessage	DwtCompString	"No library specified\n"
nulltopic_message	DetNnulltopicMessage	DwtcompString	"No first topic and overview topic specified\n"
overview_topic	DwtNoverviewTopic	DwtCompString	Null
rows	DwtNrows	int	20 lines
saveas_label	DwtNsaveasLabel	DwtCompString	"Save As..."
search_apply_label	DwtNsearchApplyLabel	DwtCompString	"Apply"
searchkeywordbox_label	DwtNsearchkeywordboxLabel	DwtCompString	"Search Topic Keywords"
search_label	DwtNsearchLabel	DwtCompString	"Search"
searchtitlebox_label	DwtNsearchtitleboxLabel	DwtCompString	"Search Topic Titles"
selectall_label	DwtNselectallLabel	DwtCompString	"Select All"
title_label	DwtNtitleLabel	DwtCompString	"Title..."
titles_label	DwtNtitlesLabel	DwtCompString	"Title:"
topictitles_label	DwtNtopictitlesLabel	DwtCompString	"Topic Titles:"

(continued on next page)

Table B–8 (Cont.) Help Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
view_label	DwtNviewLabel	DwtCompString	"View"
visitglos_label	DwtNvisitglosLabel	DwtCompString	"Visit Glossary"
visit_label	DwtNvisitLabel	DwtCompString	"Visit"
unmap_callback	DwtNunmapCallback	DwtCompString	Null

B.9 Label

See Table B–9 for a summary of label widget attributes.

Table B–9 Label Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	Width of the label or pixmap, plus two times **margin_width**
height	DwtNheight	Dimension	Height of the label or pixmap, plus two times **margin_height**
border_width	DwtNborderWidth	Dimension	0 pixels
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null

(continued on next page)

Table B–9 (Cont.) Label Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
label_type	DwtNlabelType	unsigned char	Compound string
label	DwtNlabel	DwtCompString	Widget name
margin_width	DwtNmarginWidth	Dimension	2 pixels for text; 0 pixels for pixmap
margin_height	DwtNmarginHeight	Dimension	2 pixels for text; 0 pixels for pixmap
alignment	DwtNalignment	unsigned char	Center alignment
pixmap	DwtNpixmap	Pixmap	Null
margin_left	DwtNmarginLeft	Dimension	0 pixels
margin_right	DwtNmarginRight	Dimension	0 pixels
margin_top	DwtNmarginTop	Dimension	0 pixels
margin_bottom	DwtNmarginBottom	Dimension	0 pixels
conform_to_text	DwtNconformToText	Boolean	True if width and height are zero; false if width and height are not zero

B.10 List Box

See Table B–10 for a summary of list box widget attributes.

Table B–10 List Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimenision	As large as necessary to hold the longest item without exceeding the size of its parent
height	DwtNheight	Dimension	As large as necessary to hold the number of items specified by **visible_item_count** without exceeding the size of its parent
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null

(continued on next page)

Table B–10 (Cont.) List Box Attributes

Attribute	C Name	C Data Type	Default
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	Not supported
help_callback	DwtNhelpCallback	DwtCallbackPtr	Not supported
Scroll Window Attributes			
h_scroll	DwtNhorizontalScrollBar	Widget	Null
v_scroll	DwtNverticalScrollBar	Widget	Null
work_window	DwtNworkWindow	Widget	Null
shown_value_ automatic_horiz	DwtNshownValueAutomaticHoriz	Boolean	True
shown_value_ automatic_vert	DwtNshownValueAutomaticVert	Boolean	False
Widget-Specific Attributes			
margin_width	DwtNmarginWidth	Dimension	10 pixels
margin_height	DwtNmarginHeight	Dimension	4 pixels
spacing	DwtNspacing	Dimension	1 pixel
items	DwtNitems	DwtCompString *	Null
item_count	DwtNitemsCount	int	0 items
selected_items	DwtNselectedItems	DwtCompString *	Null
selected_items_count	DwtNselectedItemsCount	int	0 items
visible_item_count	DwtNvisibleItemsCount	int	As many items as can fit in the core attribute **height**
single_selection	DwtNsingleSelection	Boolean	True
resize	DwtNresize	unsigned char	DwtGrowOnly
horiz	DwtNhorizontal	Boolean	False
single_callback	DwtNsingleCallback	DwtCallbackPtr	Null

(continued on next page)

Table B–10 (Cont.) List Box Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
single_confirm_ callback	DwtNsingleConfirmCallback	DwtCallbackPtr	Null
extend_callback	DwtNextendCallback	DwtCallbackPtr	Null
extend_confirm_ callback	DwtNextendConfirmCallback	DwtCallbackPtr	Null

B.11 Main Window

See Table B–11 for a summary of main window widget attributes.

Table B–11 Main Window Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	5 pixels
height	DwtNheight	Dimension	5 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window

(continued on next page)

Table B–11 (Cont.) Main Window Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Not supported
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Not supported
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	Not supported
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
command_window	DwtNcommandWindow	Widget	Null
work_window	DwtNworkWindow	Widget	Null
menu_bar	DwtNmenuBar	Widget	Null
horizontal_scroll_bar	DwtNhorizontalScrollBar	Widget	Null
vertical_scroll_bar	DwtNverticalScrollBar	Widget	Null
accept_focus	DwtNacceptFocus	Boolean	False
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null

B.12 Menu Bar

See Table B–12 for a summary of menu bar widget attributes.

Summary of Widget Attributes (C Binding)
B.12 Menu Bar

Table B–12 Menu Bar Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	16 pixels
height	DwtNheight	Dimension	Number of lines needed to display all entries
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	Used only by gadget children
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null

(continued on next page)

Table B–12 (Cont.) Menu Bar Attributes

Attribute	C Name	C Data Type	Default
Menu Attributes			
orientation	DwtNorientation	unsigned char	DwtOrientationVertical
spacing	DwtNspacing	Dimension	1 pixel
adjust_margin	DwtNadjustMargin	Boolean	True
margin_width	DwtNmarginWidth	Dimension	3 pixels
margin_height	DwtNmarginHeight	Dimension	3 pixels
entry_border	DwtNentryBorder	Dimension	0 pixels
menu_alignment	DwtNmenuAlignment	Boolean	True
entry_alignment	DwtNentryAlignment	unsigned char	DwtAlignmentBeginning
menu_packing	DwtNmenuPacking	unsigned char	DwtMenuPackingTight for all menu types except for radio boxes. Radio boxes default to DwtMenuPackingColumn.
menu_num_columns	DwtNmenuNumColumns	short	1 row or column
menu_radio	DwtNmenuRadio	Boolean	False, except for radio boxes, which default to true
radio_always_one	DwtNradioAlwaysOne	Boolean	True
menu_is_ homogeneous	DwtNmenuIsHomogeneous	Boolean	False except for radio boxes, which default to true
menu_entry_class	DwtNmenuEntryClass	WidgetClass	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DwtNmenuHistory	Widget	Null
entry_callback	DwtNentryCallback	DwtCallbackPtr	Null
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
menu_help_widget	DwtNmenuHelpWidget	Widget	Null

B.13 Menu

See Table B–13 for a summary of menu widget attributes.

Summary of Widget Attributes (C Binding)
B.13 Menu

Table B–13 Menu Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	If **orientation** is vertical, **width** is the maximum entry **width** or 16 pixels. If **orientation** is horizontal, **width** is the sum of **width** and **spacing** or 16 pixels.
height	DwtNheight	Dimension	If **orientation** is vertical, **height** is the sum of **height** and **spacing** or 16 pixels. If **orientation** is horizontal, **height** is the maximum entry **height** or 16 pixels.
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_ managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null

(continued on next page)

Table B–13 (Cont.) Menu Attributes

Attribute	C Name	C Data Type	Default
Common Widget Attributes			
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
orientation	DwtNorientation	unsigned char	DwtOrientationVertical
spacing	DwtNspacing	Dimension	1 pixel
adjust_margin	DwtNadjustMargin	Boolean	True
margin_width	DwtNmarginWidth	Dimension	3 pixels
margin_height	DwtNmarginHeight	Dimension	3 pixels
entry_border	DwtNentryBorder	Dimension	0 pixels
menu_alignment	DwtNmenuAlignment	Boolean	True
entry_alignment	DwtNentryAlignment	unsigned char	Alignment at the beginning
menu_packing	DwtNmenuPacking	unsigned char	DwtMenuPackingTight for all menu types except for radio boxes. Radio boxes default to DwtMenuPackingColumn.
menu_num_columns	DwtNmenuNumColumns	int	1 column
menu_radio	DwtNmenuRadio	Boolean	False except for radio boxes, which default to true
radio_always_one	DwtNradioAlwaysOne	Boolean	True
menu_is_ homogeneous	DwtNmenuIsHomogeneous	Boolean	False except for radio boxes, which default to true
menu_entry_class	DwtNmenuEntryClass	WidgetClass	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DwtNmenuHistory	Widget	Null
entry_callback	DwtNentryCallback	DwtCallbackPtr	Null
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
menu_help_widget	DwtNmenuHelpWidget	Widget	Null

B.14 Menu Popup

See Table B–14 for a summary of menu pop-up widget attributes.

Table B–14 Menu Popup Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	As large as necessary to hold all child widgets
height	DwtNheight	Dimension	As large as necessary to hold all child widgets
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null

(continued on next page)

Table B–14 (Cont.) Menu Popup Attributes

Attribute	C Name	C Data Type	Default
Common Widget Attributes			
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
orientation	DwtNorientation	unsigned char	DwtOrientationVertical
spacing	DwtNspacing	Dimension	1 pixel
adjust_margin	DwtNadjustMargin	Boolean	True
margin_width	DwtNmarginWidth	Dimension	3 pixels
margin_height	DwtNmarginHeight	Dimension	3 pixels
entry_border	DwtNentryBorder	Dimension	0 pixels
menu_alignment	DwtNmenuAlignment	Boolean	True
entry_alignment	DwtNentryAlignment	unsigned char	DwtAlignmentBeginning
menu_packing	DwtNmenuPacking	unsigned char	DwtMenuPackingTight for all menu types except for radio boxes. Radio boxes default to DwtMenuPackingColumn.
menu_num_columns	DwtNmenuNumColumns	short	1 column
menu_radio	DwtNmenuRadio	Boolean	False except for radio boxes, which default to true
radio_always_one	DwtNradioAlwaysOne	Boolean	True
menu_is_ homogeneous	DwtNmenuIsHomogeneous	Boolean	False except for radio boxes, which default to true
menu_entry_class	DwtNmenuEntryClass	WidgetClass	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DwtNmenuHistory	Widget	Null
entry_callback	DwtNentryCallback	DwtCallbackPtr	Null
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
menu_help_widget	DwtNmenuHelpWidget	Widget	Null

B.15 Menu Pulldown

See Table B–15 for a summary of menu pull-down widget attributes.

Summary of Widget Attributes (C Binding)
B.15 Menu Pulldown

Table B–15 Menu Pulldown Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Not supported
y	DwtNy	Position	Not supported
width	DwtNwidth	Dimension	As large as necessary to hold all child widgets
height	DwtNheight	Dimension	As large as necessary to hold all child widgets
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null

(continued on next page)

Table B–15 (Cont.) Menu Pulldown Attributes

Attribute	C Name	C Data Type	Default
Menu Attributes			
orientation	DwtNorientation	unsigned char	DwtOrientationVertical
spacing	DwtNspacing	Dimension	1 pixel
adjust_margin	DwtNadjustMargin	Boolean	True
margin_width	DwtNmarginWidth	Dimension	3 pixels
margin_height	DwtNmarginHeight	Dimension	3 pixels
entry_border	DwtNentryBorder	Dimension	0 pixels
menu_alignment	DwtNmenuAlignment	Boolean	True
entry_alignment	DwtNentryAlignment	unsigned char	DwtAlignmentBeginning
menu_packing	DwtNmenuPacking	unsigned char	DwtMenuPackingTight for all menu types except for radio boxes. Radio boxes default to DwtMenuPackingColumn.
menu_num_columns	DwtNmenuNumColumns	int	1 column
menu_radio	DwtNmenuRadio	Boolean	False except for radio boxes, which default to true
radio_always_one	DwtNradioAlwaysOne	Boolean	True
menu_is_ homogeneous	DwtNmenuIsHomogeneous	Boolean	False except for radio boxes, which default to true
menu_entry_class	DwtNmenuEntryClass	WidgetClass	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DwtNmenuHistory	Widget	Null
entry_callback	DwtNentryCallback	DwtCallbackPtr	Null
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
menu_help_widget	DwtNmenuHelpWidget	Widget	Null

B.16 Message Box

See Table B–16 for a summary of message box widget attributes.

Table B–16 Message Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager

(continued on next page)

B.16 Message Box

Table B–16 (Cont.) Message Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	5 pixels
height	DwtNheight	Dimension	5 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Dialog Box Pop-Up Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	Not supported
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	Not supported
title	DwtNtitle	DwtCompString	Widget name
style	DwtNstyle	unsigned char	DwtModal
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null

(continued on next page)

Table B–16 (Cont.) Message Box Attributes

Attribute	C Name	C Data Type	Default
Dialog Box Pop-Up Attributes			
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
text_merge_ translations	DwtNtextMergeTranslations	XtTranslations	Not supported
margin_width	DwtNmarginWidth	Dimension	12 pixels
margin_height	DwtNmarginHeight	Dimension	10 pixels
default_position	DwtNdefaultPosition	Boolean	False
child_overlap	DwtNchildOverlap	Boolean	Not supported
resize	DwtNresize	unsigned char	DwtShrinkWrap
take_focus	DwtNtakeFocus	Boolean	True for modal dialog box; false for modeless dialog box
no_resize	DwtNnoResize	Boolean	True
auto_unmanage	DwtNautoUnmanage	Boolean	True
default_button	DwtNdefaultButton	Widget	Not supported
cancel_button	DwtNcancelButton	Widget	Not supported
Widget-Specific Attributes			
label	DwtNlabel	DwtCompString	Widget name
ok_label	DwtNokLabel	DwtCompString	"Acknowledged"
yes_callback	DwtNyesCallback	DwtCallbackPtr	Null

B.17 Option Menu

See Table B–17 for a summary of option menu widget attributes.

Table B–17 Option Menu Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	As large as necessary to hold all child widgets

(continued on next page)

Table B–17 (Cont.) Option Menu Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
height	DwtNheight	Dimension	As large as necessary to hold all child widgets
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	Default XUI Toolkit font; used only by gadget children
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Menu Attributes			
orientation	DwtNorientation	unsigned char	DwtOrientationVertical
spacing	DwtNspacing	Dimension	1 pixel
adjust_margin	DwtNadjustMargin	Boolean	True
margin_width	DwtNmarginWidth	Dimension	3 pixels

(continued on next page)

Table B–17 (Cont.) Option Menu Attributes

Attribute	C Name	C Data Type	Default
Menu Attributes			
margin_height	DwtNmarginHeight	Dimension	3 pixels
entry_border	DwtNentryBorder	Dimension	0 pixels
menu_alignment	DwtNmenuAlignment	Boolean	True
entry_alignment	DwtNentryAlignment	unsigned char	DwtAlignmentBeginning
menu_packing	DwtNmenuPacking	unsigned char	DwtMenuPackingTight for all menu types except for radio boxes. Radio boxes default to DwtMenuPackingColumn.
menu_num_columns	DwtNmenuNumColumns	int	1 column
menu_radio	DwtNmenuRadio	Boolean	False except for radio boxes, which default to true
radio_always_one	DwtNradioAlwaysOne	Boolean	True
menu_is_ homogeneous	DwtNmenuIsHomogeneous	Boolean	False except for radio boxes, which default to true
menu_entry_class	DwtNmenuEntryClass	WidgetClass	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DwtNmenuHistory	Widget	Null
entry_callback	DwtNentryCallback	DwtCallbackPtr	Null
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
menu_help_widget	DwtNmenuHelpWidget	Widget	Null
Widget-Specific Attributes			
label	DwtNlabel	DwtCompString	Widget name
sub_menu_id	DwtNsubMenuId	Widget	Zero

B.18 Pull Down Menu Entry

See Table B–18 for a summary of pull-down menu entry widget attributes.

Summary of Widget Attributes (C Binding)

B.18 Pull Down Menu Entry

Table B–18 Pull Down Menu Entry Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	Width of the label or pixmap, plus the hotspot width, plus two times **margin_width**
height	DwtNheight	Dimension	Height of the label or pixmap, plus two times **margin_height**
border_width	DwtNborderWidth	Dimension	0 pixels
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null

(continued on next page)

Table B–18 (Cont.) Pull Down Menu Entry Attributes

Attribute	C Name	C Data Type	Default
Label Attributes			
label_type	DwtNlabelType	unsigned char	DwtCString
label	DwtNlabel	DwtCompString	Widget name
margin_width	DwtNmarginWidth	Dimension	2 pixels for text; 0 pixels for pixmap
margin_height	DwtNmarginHeight	Dimension	2 pixels for text; 0 pixels for pixmap
alignment	DwtNalignment	unsigned char	DwtCenterAlignment
pixmap	DwtNpixmap	Pixmap	Null
margin_left	DwtNmarginLeft	Dimension	0 pixels
margin_right	DwtNmarginRight	Dimension	0 pixels
margin_top	DwtNmarginTop	Dimension	0 pixels
margin_bottom	DwtNmarginBottom	Dimension	0 pixels
conform_to_text	DwtNconformToText	Boolean	True if width and height are zero; false if width and height are not zero
Widget-Specific Attributes			
sub_menu_id	DwtNsubMenuId	Widget (*)	Null
activate_callback	DwtNactivateCallback	DwtCallbackPtr	Null
pulling_callback	DwtNpullingCallback	DwtCallbackPtr	Null
hot_spot_pixmap	DwtNhotSpotPixmap	Pixmap	Null

B.19 Push Button

See Table B–19 for a summary of push button widget attributes.

Table B–19 Push Button Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	Width of the label or pixmap, plus two times **margin_ width**

(continued on next page)

Table B–19 (Cont.) Push Button Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
height	DwtNheight	Dimension	Height of the label or pixmap, plus two times **margin_height**
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Label Attributes			
label_type	DwtNlabelType	unsigned char	DwtCString
label	DwtNlabel	DwtCompString	Widget name
margin_width	DwtNmarginWidth	Dimension	2 pixels for text; 0 pixels for pixmap

(continued on next page)

Table B–19 (Cont.) Push Button Attributes

Attribute	C Name	C Data Type	Default
Label Attributes			
margin_height	DwtNmarginHeight	Dimension	2 pixels for text; 0 pixels for pixmap
alignment	DwtNalignment	unsigned char	DwtCenterAlignment
pixmap	DwtNpixmap	Pixmap	Null
margin_left	DwtNmarginLeft	Dimension	0 pixels
margin_right	DwtNmarginRight	Dimension	0 pixels
margin_top	DwtNmarginTop	Dimension	0 pixels
margin_bottom	DwtNmarginBottom	Dimension	0 pixels
conform_to_text	DwtNconformToText	Boolean	True if width and height are zero; false if width and height are not zero
Widget-Specific Attributes			
border_highlight	DwtNborderHighlight	Boolean	False
fill_highlight	DwtNfillHighlight	Boolean	False
activate_callback	DwtNactivateCallback	DwtCallbackPtr	Null
arm_callback	DwtNarmCallback	DwtCallbackPtr	Null
disarm_callback	DwtNdisarmCallback	DwtCallbackPtr	Null
accelerator_text	DwtNacceleratorText	DwtCompString	Null
button_accelerator	DwtNbuttonAccelerator	char *	Null
shadow	DwtNshadow	Boolean	True

B.20 Radio Box

See Table B–20 for a summary of radio box widget attributes.

Table B–20 Radio Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	As large as necessary to hold all child widgets

(continued on next page)

B.20 Radio Box

Table B–20 (Cont.) Radio Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
height	DwtNheight	Dimension	As large as necessary to hold all child widgets
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
orientation	DwtNorientation	unsigned char	DwtOrientationVertical
spacing	DwtNspacing	Dimension	1 pixel
adjust_margin	DwtNadjustMargin	Boolean	True
margin_width	DwtNmarginWidth	Dimension	3 pixels

(continued on next page)

Table B–20 (Cont.) Radio Box Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
margin_height	DwtNmarginHeight	Dimension	3 pixels
entry_border	DwtNentryBorder	Dimension	0 pixels
menu_alignment	DwtNmenuAlignment	Boolean	True
entry_alignment	DwtNentryAlignment	unsigned char	DwtrAlignmentBeginning
menu_packing	DwtNmenuPacking	unsigned char	DwtMenuPackingTight for all menu types except for radio boxes. Radio boxes default to DwtMenuPacking column.
menu_num_columns	DwtNmenuNumColumns	int	1 column
menu_radio	DwtNmenuRadio	Boolean	False except for radio boxes, which default to true
radio_always_one	DwtNradioAlwaysOne	Boolean	True
menu_is_ homogeneous	DwtNmenuIsHomogeneous	Boolean	False except for radio boxes, which default to true
menu_entry_class	DwtNmenuEntryClass	WidgetClass	Null except for radio boxes, which default to *togglebuttonwidgetclass*
menu_history	DwtNmenuHistory	Widget	Null
entry_callback	DwtNentryCallback	DwtCallbackPtr	Null
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
menu_help_widget	DwtNmenuHelpWidget	Widget	Null

B.21 Scale

See Table B–21 for a summary of scale widget attributes.

Table B–21 Scale Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager

(continued on next page)

Table B–21 (Cont.) Scale Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
width	DwtNwidth	Dimenson	Calculated based on the scale width or height, the label widths, and the orientation
height	DwtNheight	Dimension	Calculated based on the scale width or height, the label widths, and the orientation
border_width	DwtNborderWidth	Dimension	0 pixels
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null

(continued on next page)

Table B–21 (Cont.) Scale Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
value	DwtNvalue	int	Zero
title	DwtNtitle	DwtCompString	DwtCString
orientation	DwtNorientation	unsigned char	DwtOrientationHorizontal
scale_width	DwtNscaleWidth	Dimension	20 pixels for vertical scales; calculated for horizontal scales
scale_height	DwtNscaleHeight	Dimension	20 pixels for horizontal scales; calculated for vertical scales
min_value	DwtNminValue	int	Zero
max_value	DwtNmaxValue	int	100
decimal_points	DwtNdecimalPoints	short	Zero
show_value	DwtNshowValue	Boolean	True
slider_pixmap	DwtNsliderPixmap	Pixmap	Null
drag_callback	DwtNdragCallback	DwtCallbackPtr	Null
value_changed_ callback	DwtNvalueChangedCallback	DwtCallbackPtr	Null

B.22 Scroll Bar

See Table B–22 for a summary of scroll bar widget attributes.

Table B–22 Scroll Bar Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	For vertical scroll bars, 17 pixels; for horizontal scroll bars, the width of the parent minus 17 pixels
height	DwtNheight	Dimension	For horizontal scroll bars, 17 pixels; for vertical scroll bars, the height of the parent minus 17 pixels
border_width	DwtNborderWidth	Dimension	1 pixel

(continued on next page)

Table B–22 (Cont.) Scroll Bar Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	Not supported
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
int_value	DwtNvalue	int	Zero
min_value	DwtNminValue	int	Zero
max_value	DwtNmaxValue	int	100
orientation	DwtNorientation	unsigned char	DwtOrientationVertical
translations1	DwtNtranslations1	XtTranslations	Null
translations2	DwtNtranslations2	XtTranslations	Null
shown	DwtNshown	int	10 units

(continued on next page)

Table B–22 (Cont.) Scroll Bar Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
inc	DwtNinc	int	10 units
page_inc	DwtNpageInc	int	10 units
unit_inc_callback	DwtNunitIncCallback	DwtCallbackPtr	Null
unit_dec_callback	DwtNunitDecCallback	DwtCallbackPtr	Null
page_inc_callback	DwtNpageIncCallback	DwtCallbackPtr	Null
page_dec_callback	DwtNpageDecCallback	DwtCallbackPtr	Null
to_top_callback	DwtNtoTopCallback	DwtCallbackPtr	Null
to_bottom_callback	DwtNtoBottomCallback	DwtCallbackPtr	Null
drag_callback	DwtNdragCallback	DwtCallbackPtr	Null
value_changed_ callback	DwtNvalueChangedCallback	DwtCallbackPtr	Null

B.23 Scroll Window

See Table B–23 for a summary of scroll window widget attributes.

Table B–23 Scroll Window Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	Widget specific
height	DwtNheight	Dimension	Widget specific
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes

(continued on next page)

B.23 Scroll Window

Table B–23 (Cont.) Scroll Window Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	Not supported
help_callback	DwtNhelpCallback	DwtCallbackPtr	Not supported
Widget-Specific Attributes			
h_scroll	DwtNhorizontalScrollBar	Widget	Null
v_scroll	DwtNverticalScrollBar	Widget	Null
work_window	DwtNworkWindow	Widget	Null
shown_value_automatic_horiz	DwtNshownValueAutomaticHoriz	Boolean	True
shown_value_automatic_vert	DwtNshownValueAutomaticVert	Boolean	True

B.24 Selection

See Table B–24 for a summary of selection widget attributes.

Table B–24 Selection Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Centered in the parent window
y	DwtNy	Position	Centered in the parent window
width	DwtNwidth	Dimension	Width of the list box, plus the width of the push buttons, plus three times **margin_width**
height	DwtNheight	Dimension	Height of the list box, plus the height of the text edit field, plus the height of the label, plus three times **margin_height**
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Dialog Box Popup Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False

(continued on next page)

Table B–24 (Cont.) Selection Attributes

Attribute	C Name	C Data Type	Default
Dialog Box Popup Attributes			
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	DwtFontUnits
style	DwtNstyle	unsigned char	DwtModeless
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
text_merge_ translations	DwtNtextMergeTranslations	XtTranslations	Null
margin_width	DwtNmarginWidth	Dimension	5 pixels
margin_height	DwtNmarginHeight	Dimension	5 pixels
default_position	DwtNdefaultPosition	Boolean	False
child_overlap	DwtNchildOverlap	Boolean	True
resize	DwtNresize	unsigned char	DwtResizeGrowOnly
no_resize	DwtNnoResize	Boolean	True
title	DwtNtitle	DwtCompString	"Open"
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
take_focus	DwtNtakeFocus	Boolean	True for modal dialog box; false for modeless dialog box
auto_unmanage	DwtNautoUnmanage	Boolean	True
default_button	DwtNdefaultButton	Widget	Null
cancel_button	DwtNcancelButton	Widget	Null
Widget-Specific Attributes			
label	DwtNlabel	DwtCompString	"Items"
value	DwtNvalue	DwtCompString	Null string
selection_label	DwtNselectionLabel	DwtCompString	"Selection"
ok_label	DwtNokLabel	DwtCompString	"OK"
cancel_label	DwtNcancelLabel	DwtCompString	"Cancel"
activate_callback	DwtNactivateCallback	DwtCallbackPtr	Null
cancel_callback	DwtNcancelCallback	DwtCallbackPtr	Null
no_match_callback	DwtNnoMatchCallback	DwtCallbackPtr	Null

(continued on next page)

Table B–24 (Cont.) Selection Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
visible_item_count	DwtNvisibleItemsCount	int	8 items
items	DwtNitems	DwtCompString *	Null
item_count	DwtNitemsCount	int	0 items
must_match	DwtNmustMatch	Boolean	False

B.25 Separator

See Table B–25 for a summary of separator widget attributes.

Table B–25 Separator Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	3 pixels
height	DwtNheight	Dimension	3 pixels
border_width	DwtNborderWidth	Dimension	0 pixels
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Not supported
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null

(continued on next page)

B.25 Separator

Table B–25 (Cont.) Separator Attributes

Attribute	C Name	C Data Type	Default
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	Not supported
help_callback	DwtNhelpCallback	DwtCallbackPtr	Not supported
Label Attributes			
label_type	DwtNlabelType	unsigned char	DwtCString
label	DwtNlabel	DwtCompString	Widget name
margin_width	DwtNmarginWidth	Dimension	2 pixels for text; 0 pixels for pixmap
margin_height	DwtNmarginHeight	Dimension	2 pixels for text; 0 pixels for pixmap
alignment	DwtNalignment	unsigned char	DwtCenterAlignment
pixmap	DwtNpixmap	Pixmap	Null
margin_left	DwtNmarginLeft	Dimension	0 pixels
margin_right	DwtNmarginRight	Dimension	0 pixels
margin_top	DwtNmarginTop	Dimension	0 pixels
margin_bottom	DwtNmarginBottom	Dimension	0 pixels
conform_to_text	DwtNconformToText	Boolean	True if width and height are zero; false if width and height are not zero
Widget-Specific Attributes			
orientation	DwtNorientation	unsigned char	DwtOrientationHorizontal

B.26 S Text

See Table B–26 for a summary of simple text widget attributes.

Table B–26 S Text Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	As large as necessary to display the **rows** and **cols** with the given **margin_width** and **margin_height**
height	DwtNheight	Dimension	As large as necessary to display the **rows** and **cols** with the given **margin_width** and **margin_height**
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Widget-Specific Attributes			
margin_width	DwtNmarginWidth	Dimension	2 pixels
margin_height	DwtNmarginHeight	Dimension	2 pixels
cols	DwtNcols	Dimension	20 characters
rows	DwtNrows	Dimension	1 character
top_position	DwtNtopPosition	long int	Zero
word_wrap	DwtNwordWrap	Boolean	False

(continued on next page)

Table B–26 (Cont.) S Text Attributes

Attribute	C Name	C Data Type	Default
Widget-Specific Attributes			
scroll_vertical	DwtNscrollVertical	Boolean	False
resize_height	DwtNresizeHeight	Boolean	True
resize_width	DwtNresizeWidth	Boolean	True
value	DwtNvalue	char *	Null string (" ")
editable	DwtNeditable	Boolean	True
max_length	DwtNmaxLength	int	2^{31-1}
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
lost_focus_callback	DwtNlostFocusCallback	DwtCallbackPtr	Null
value_changed_ callback	DwtNvalueChangedCallback	DwtCallbackPtr	Null
insertion_point_visible	DwtNinsertionPointVisible	Boolean	True
auto_show_insert_ point	DwtNautoShowInsertPoint	Boolean	True
insertion_position	DwtNinsertionPosition	int	Zero
foreground	DwtNforeground	Pixel	Default foreground color
font	DwtNfont	FontList	Current server DwtFontList
blink_rate	DwtNblinkRate	int	500 milliseconds
scroll_left_side	DwtNscrollLeftSide	Boolean	False
half_border	DwtNhalfBorder	Boolean	True
pending_delete	DwtNpendingDelete	Boolean	True

B.27 Toggle Button

See Table B–27 for a summary of toggle button widget attributes.

Table B–27 Toggle Button Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	Width of the label or pixmap, plus two times **margin_ width**

(continued on next page)

Table B–27 (Cont.) Toggle Button Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
height	DwtNheight	Dimension	Height of the label or pixmap, plus two times **margin_height**
border_width	DwtNborderWidth	Dimension	0 pixels
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	XUI Toolkit font
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
Label Attributes			
label_type	DwtNlabelType	unsigned char	DwtCString
label	DwtNlabel	DwtCompString	Widget name
margin_width	DwtNmarginWidth	Dimension	2 pixels for text; 0 pixels for pixmap

(continued on next page)

Table B–27 (Cont.) Toggle Button Attributes

Attribute	C Name	C Data Type	Default
Label Attributes			
margin_height	DwtNmarginHeight	Dimension	2 pixels for text; 0 pixels for pixmap
alignment	DwtNalignment	unsigned char	DwtCenterAlignment
pixmap	DwtNpixmap	Pixmap	Null
margin_left	DwtNmarginLeft	Dimension	0 pixels
margin_right	DwtNmarginRight	Dimension	0 pixels
margin_top	DwtNmarginTop	Dimension	0 pixels
margin_bottom	DwtNmarginBottom	Dimension	0 pixels
conform_to_text	DwtNconformToText	Boolean	True if width and height are zero; false if width and height are not zero
Widget-Specific Attributes			
shape	DwtNshape	unsigned char	DwtRectangular
visible_when_off	DwtNvisibleWhenOff	Boolean	True
spacing	DwtNspacing	short	4 pixels
pixmap_on	DwtNpixmapOn	Pixmap	Null
pixmap_off	DwtNpixmapOff	Pixmap	Null
value	DwtNvalue	Boolean	False
arm_callback	DwtNarmCallback	DwtCallbackPtr	Null
disarm_callback	DwtNdisarmCallback	DwtCallbackPtr	Null
value_changed_ callback	DwtNvalueChangedCallback	DwtCallbackPtr	Null
indicator	DwtNindicator	Boolean	True when label is text; false when label is pixmap
accelerator_text	DwtNacceleratorText	DwtCompString	Null
button_accelerator	DwtNbuttonAccelerator	char *	Null

B.28 Window

See Table B–28 for a summary of window widget attributes.

Table B–28 Window Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	0 pixels
height	DwtNheight	Dimension	0 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Common Widget Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	False
font	DwtNfont	DwtFontList	Not supported
help_callback	DwtNhelpCallback	DwtCallbackPtr	Not supported
Widget-Specific Attribute			
expose_callback	DwtNexposeCallback	DwtCallbackPtr	Null

B.29 Work Box

See Table B–29 for a summary of work box widget attributes.

Table B–29 Work Box Attributes

Attribute	C Name	C Data Type	Default
Core Widget Attributes			
x	DwtNx	Position	Determined by the geometry manager
y	DwtNy	Position	Determined by the geometry manager
width	DwtNwidth	Dimension	5 pixels
height	DwtNheight	Dimension	5 pixels
border_width	DwtNborderWidth	Dimension	1 pixel
border	DwtNborder	Pixel	Default foreground color
border_pixmap	DwtNborderPixmap	Pixmap	Null
background	DwtNbackground	Pixel	Default background color
background_pixmap	DwtNbackgroundPixmap	Pixmap	Null
colormap	DwtNcolormap	Colormap	Default color map
sensitive	DwtNsensitive	Boolean	True
ancestor_sensitive	DwtNancestorSensitive	Boolean	The bitwise AND of the parent widget's **sensitive** and **ancestor_sensitive** attributes
accelerators	DwtNaccelerators	XtTranslations	Null
depth	DwtNdepth	int	Depth of the parent window
translations	DwtNtranslations	XtTranslations	Null
mapped_when_managed	DwtNmappedWhenManaged	Boolean	True
screen	DwtNscreen	Screen *	The parent screen
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
destroy_callback	DwtNdestroyCallback	DwtCallbackPtr	Null
Dialog Box Pop-Up Attributes			
foreground	DwtNforeground	Pixel	Default foreground color
highlight	DwtNhighlight	Pixel	Default foreground color
highlight_pixmap	DwtNhighlightPixmap	Pixmap	Null
user_data	DwtNuserData	Opaque *	Null
direction_r_to_l	DwtNdirectionRToL	Boolean	Not supported
font	DwtNfont	DwtFontList	XUI Toolkit font

(continued on next page)

Table B–29 (Cont.) Work Box Attributes

Attribute	C Name	C Data Type	Default
Dialog Box Pop-Up Attributes			
help_callback	DwtNhelpCallback	DwtCallbackPtr	Null
units	DwtNunits	unsigned char	Not supported
title	DwtNtitle	DwtCompString	Widget name
style	DwtNstyle	unsigned char	DwtModal
map_callback	DwtNmapCallback	DwtCallbackPtr	Null
unmap_callback	DwtNunmapCallback	DwtCallbackPtr	Null
focus_callback	DwtNfocusCallback	DwtCallbackPtr	Null
text_merge_ translations	DwtNtextMergeTranslations	XtTranslations	Not supported
margin_width	DwtNmarginWidth	Dimension	12 pixels
margin_height	DwtNmarginHeight	Dimension	10 pixels
default_position	DwtNdefaultPosition	Boolean	False
child_overlap	DwtNchildOverlap	Boolean	Not supported
resize	DwtNresize	unsigned char	DwtResizeShrinkWrap
take_focus	DwtNtakeFocus	Boolean	True for modal dialog box; false for modeless dialog box
no_resize	DwtNnoResize	Boolean	True
auto_unmanage	DwtNautoUnmanage	Boolean	True
default_button	DwtNdefaultButton	Widget	Not supported
cancel_button	DwtNcancelButton	Widget	Not supported
Widget-Specific Attributes			
label	DwtNlabel	DwtCompString	Widget name
cancel_label	DwtNcancelLabel	DwtCompString	"Cancel"
cancel_callback	DwtNcancelCallback	DwtCallbackPtr	Null

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-DEC-DEMO (800-332-3366) using a 1200- or 2400-baud modem. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local DIGITAL subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	———	Local DIGITAL subsidiary or approved distributor
Internal[1]	———	SDC Order Processing - WMO/E15 *or* Software Distribution Center Digital Equipment Corporation Westminster, Massachusetts 01473

[1]For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

VMS DECwindows Toolkit Routines Reference Manual
Part I: AA–MG23A–TE
Part II: AA–MK88A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

-- Do Not Tear - Fold Here and Tape ----------

digital™

No Postage
Necessary
if Mailed
in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01-3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here ----------