HP DECwindows Motif for OpenVMS Alpha

New Features

Order Number: AA–RT2CB–TE

October 2003

This manual describes new features and enhancements that pertain to the HP DECwindows Motif for OpenVMS Alpha Version 1.3–1 software.

Revision/Update Information:	This manual supersedes the *hp DECwindows Motif for hp OpenVMS Alpha New Features* for Version 1.3.
Operating System:	HP OpenVMS Alpha Version 7.3–2
Software Version:	HP DECwindows Motif for OpenVMS Alpha Version 1.3–1

Hewlett-Packard Company
Palo Alto, California

ZK6663

The HP DECwindows Motif for OpenVMS Alpha Version 1.3–1 documentation set is available on CD-ROM.

Contents

3 System Management Features

4 Programming Features

Index

Figures

Tables

Preface

This document describes the new features introduced with the HP DECwindows Motif for OpenVMS Alpha Version 1.3–1 (DECwindows Motif) software. For information about how these features might affect your system, read the release notes before you install, upgrade, or use the DECwindows Motif Version 1.3–1 software.

The features in this manual are cumulative from DECwindows Motif for OpenVMS Version 1.0 and indicate any undocumented items that still pertain to the software. For previous features, a label within the description indicates when the feature was introduced.

Intended Audience

This manual is intended for system managers, users, and programmers who work with the DECwindows Motif software.

Document Structure

This manual is structured as follows:

- Chapter 1 provides an overview of the current release.
- Chapter 2 describes features of interest to general users of the DECwindows Motif software.
- Chapter 3 describes features related to system and network management.
- Chapter 4 describes features that support application and system programmers.

Related Documents

For additional information about OpenVMS or DECwindows Motif products and services, visit the following web site:

`http://www.hp.com/go/openvms`

Reader's Comments

HP welcomes your comments on this manual. Please send comments to either of the following addresses:

Internet	**openvmsdoc@hp.com**
Mail	Hewlett-Packard Company OSSG Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

How To Order Additional Documentation

For information about how to order additional documentation, visit the following World Wide Web address:

`http://www.hp.com/go/openvms/doc/order`

Conventions

In this manual, references to OpenVMS are synonymous with the HP OpenVMS Alpha Operating System.

Unless otherwise specified, references to OpenVMS Clusters, VMSclusters, or clusters in this document are synonymous with HP OpenVMS Clusters.

All uses of DECwindows and DECwindows Motif refer to the HP DECwindows Motif for OpenVMS Alpha software; and all uses of X server and X display server refer to the DECwindows X11 Display Server. Additionally, all uses of DECwindows XUI (X User Interface) refer to the DECwindows product prior to DECwindows Motif Version 1.0.

The following conventions are also used in this manual:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.) In the HTML version of this document, this convention appears as brackets, rather than a box.
. . .	Horizontal ellipsis points in examples indicate one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	Vertical ellipsis points indicate the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In command format descriptions, parentheses indicate that you must enclose the choices in parentheses if you choose more than one.
[ ]	In command format descriptions, brackets indicate optional elements. You can choose one, none, or all of the options. (Brackets are not optional, however, in the syntax of a directory name in an OpenVMS file specification or in the syntax of a substring specification in an assignment statement.)
[|]	In command format descriptions, vertical bars separating items inside brackets indicate that you choose one, none, or more than one of the options.
{ }	In command format descriptions, braces indicate required elements; you must choose one of the options listed.

text style	This text style represents the introduction of a new term or the name of an argument, an attribute, or a reason. In the HTML version of this document, this convention appears as *italic text*.
italic text	Italic text emphasizes important information and indicates complete titles of manuals and variables. Variables include information that varies in system messages (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where *dd* represents the predefined code for the device type).
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Monospace type`	Monospace type indicates code examples and interactive screen displays. In the C programming language, monospace type in text identifies the following elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.
-	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1
Introduction

This chapter summarizes the features associated with the HP DECwindows Motif for OpenVMS Alpha Version 1.3–1 software. DECwindows Motif Version 1.3–1 is a minor release that includes restoration of the LAT transport interface and support for displaying customized welcome messages prior to logging into the New Desktop.

Table 1–1 lists all the items added during this release and cross-references the sections in which they are described.

For a detailed description of the corrections, changes, and known problems associated with this release, see the *HP DECwindows Motif for OpenVMS Alpha Release Notes* manual.

Table 1–1 Directory of Features for DECwindows Motif Version 1.3–1

Title	Section
Desktop Management Features	
Displaying Custom Messages Prior to Login	Section 3.4.1
X Display Server Management Features	
New Server Parameter for Setting Process Priority	Section 3.7.1
Transport Programming Features	
Support for the LAT Transport Interface Restored	Section 4.2.1

2
General User Features

This chapter provides information about new features that pertain to all users of DECwindows Motif.

2.1 OpenVMS Display Device and Layered Product Interfaces

This section contains release notes that pertain to the OpenVMS display device (SET DISPLAY) and DECwindows Motif layered product interfaces.

2.1.1 Enhanced DECwindows Motif Display Device Capabilities

V1.3

To support the new X display server access control and proxy features available as part of the X11R6.6 upgrade, many SET DISPLAY and SHOW DISPLAY command qualifiers have been either added or modified for use on systems running OpenVMS Alpha Version 7.3–1 or higher.

Table 2–1 lists these new and changed qualifiers that provide the following added functionality for DECwindows Motif display devices:

- Ability to define and view a set of named properties associated with a particular display device and assign their values. Named properties are intended to store such information as the network address or transport for the display device. Once defined, you can use these properties as the basis for subsequent DCL global symbol definitions.
- Access to a subset of commands from the X Authority utility (xauth). This utility enables you to manage the contents of one or more X authority files, which are used by the MIT-KERBEROS-5 and MIT-MAGIC-COOKIE-1 protocols to control access to an X server. Use the new qualifiers to create and use an alternate X authority file, manually generate or revoke authorization keys, or extract entries from the X authority file.
- Access to the proxy manager and Low-Bandwidth X (LBX) proxy server. The proxy manager and LBX proxy server assist in managing connections over low-bandwidth networks, such as the Internet. Use the new qualifiers to specify a standalone proxy server or to specify a proxy manager (and the manager's port and transport information). You can also enable authentication on proxy server connections and specify data values, such as a magic cookie, to be passed to the authentication protocol.

Table 2–1 New and Changed DCL SET DISPLAY and SHOW DISPLAY Command Qualifiers

Command	Qualifier	Description
SET DISPLAY	/GENERATE	Connects to the X server and generates a new authorization key.
	/LBXAUTHENTICATE /NOLBXAUTHENTICATE	Specifies the authentication protocol used to grant the proxy server access to the X server.
	/LBXDATA	Specifies a data value to be processed by the authentication protocol.
	/PMPORT	Indicates the port number for the proxy manager.
	/PMTRANSPORT	Specifies the network transport used by a display device to connect to the proxy manager.
	/PROXY	Specifies that a proxy manager be used to assign an LBX proxy server to act as an intermediary between the client and the X server specified in the command.
	/QUOTA	Modifies the value of the name count or data space quota for the display device.
	/REVOKE	Revokes the authorization key for the display device produced by the /GENERATE qualifier.
	/SERVER	Specifies the server number. This qualifier takes on the additional meaning of specifying a standalone LBX proxy server.
	/TRANSPORT	Specifies the network transport used to connect to the X server. When using the proxy manager, selects transport used between the selected proxy server and the X server. When using a standalone proxy server, selects the transport between the client and the proxy server. This qualifier also provides a default value for the /PMTRANSPORT qualifier.
	/VALUE	Sets, deletes, or modifies a named property value.
	/XAUTHORITY /NOXAUTHORITY	Specifies the location for an alternate X authority file. The /NOXAUTHORITY qualifier clears this setting.
SHOW DISPLAY	/ALL	Displays all named properties associated with the display device and their current values.
	/EXTRACT	Obtains the authorization data from the current X authority file and writes it to SYS$OUTPUT.
	/QUOTA	Displays the name count and data space quota values associated with the display device.
	/SYMBOLS	Defines a DCL symbol for each named property.
	/VALUES	Displays the set of values associated with the specified named properties.

For detailed information on these new qualifiers, see the online help for each command, or refer to the *OpenVMS DCL Dictionary: N–Z*.

2.1.2 Additional Display Device Logicals for Default Transport and Name Count Values

V1.3

Use the following logicals to override the normal defaults when creating display devices:

- DECW$SETDISPLAY_DEFAULT_TRANSPORT – Specifies the default transport to use when creating a display device without the /TRANSPORT qualifier.
- DECW$WS_DEFAULT_NAME_COUNT – Specifies the default name count quota for a new display device. This logical must be defined in executive mode and in the system logical name table.
- DECW$WS_DEFAULT_DATA_SPACE – Specifies the default data space quota for a new display device. This logical must be defined in executive mode and in the system logical name table.

2.2 General DECwindows Motif Environment

This section describes new features that are common to both the New Desktop and Traditional DECwindows Desktop environments.

2.2.1 Extended File Specification (EFS) Support

V1.2–5

In general, DECwindows Motif supports the Extended File Specifications (EFS) option provided in OpenVMS Alpha. The interface either supports the new ODS-5 file names or provides an error message indicating that the particular component does not support the new names.

2.2.1.1 File Selection Popup Window

The standard file selection popup window used by most DECwindows Motif applications fully supports ODS-5 style file names. The window supports deep directories, case preservation, and extended-length file names. The window supports entry of file names using the extended file name character and displays files using the extended file name character set using the circumflex character (^). See the current OpenVMS documentation for more specific information about using deep directories and the extended file name character set.

2.2.1.2 New Desktop and the FileView Application

With an ODS-5 volume, the FileView application supports deep directories, case preservation, and extended-length file names. FileView supports entry of file names using the extended file name character and displays files using the extended file name character set using the circumflex character (^). See the current OpenVMS documentation for more specific information about using deep directories and the extended file name character set.

Any custom FileView command extensions must be modified to support EFS.

2.2.1.3 Traditional Desktop and the File Manager Application

The File Manager application supports case preservation and creates files with the extended file name character set, but does not support any operations on its files.

The File Manager application supports deep directories and extended-length file names with the following restriction. ODS-5 volumes allow file names up to 236 8-bit characters in length and deep directory structures if the total file specification does not exceed 512 8-bit characters in length. The current version of the File Manager supports extended-length file names and deep directory structures with the additional restriction that a total file specification cannot exceed 235 8-bit characters in length. Exceeding this limit causes an error message for some menu items.

2.2.1.4 The Programming Libraries

The programming libraries fully support deep directories and extended-length file names. However, the libraries do not support case preservation or the extended file name character set.

2.2.1.5 Translated Image Support (TIS) Library

The translated image support (TIS) library has not been updated to support EFS.

2.2.2 Color Customizer Example Program

V1.2

The color customizer example program allows you to dynamically control the colors of your workstation environment. Window, icon, and window manager colors can be changed individually or as part of a palette switch. You can control mapping between resources and color cells, as well as the size and contents of the palette set. Also, automatic shadowing with the standard Motif shadowing algorithms is supported.

2.2.2.1 Supported Displays

The color customizer supports any display using pseudocolor or grayscale visuals. This includes most 4- and 8-plane workstation displays.

2.2.2.2 Supported Applications

The color customizer can affect the colors of any applications that use the current release of the DECwindows X Toolkit Library. Applications from other vendors and previous versions of the X Toolkit Library are unaffected.

Note

If the color customizer is used to control the colors of applications that have their own color customization dialog boxes (like the Session Manager, Window Manager, and DECwindows Mail), those application-specific color customization dialog boxes may not reflect the correct current color values while the customizer is running the application. This is normal; use the customizer instead of the application-specific dialog box to change these color values.

2.2.2.3 Building the Color Customizer on OpenVMS Systems

To build the color customizer on OpenVMS systems, perform the following steps:

1. Copy the files to a private directory. For example:

```
$ SET DEFAULT SYS$LOGIN
$ CREATE/DIRECTORY [.CUSTOMIZER]
$ SET DEFAULT [.CUSTOMIZER]
$ COPY DECW$EXAMPLES:CUSTOM.C []
$ COPY DECW$EXAMPLES:CUSTOM.UIL []
$ COPY DECW$EXAMPLES:CUSTOMIMAGE.DAT []
$ COPY DECW$EXAMPLES:XSETROOT_CUST.C []
$ COPY DECW$EXAMPLES:BUILD_CUSTOMIZER.COM []
```

2. Build the customizer using the following command:

```
$ @BUILD_CUSTOMIZER.COM
```

This command procedure creates the following output files:

CUSTOM.UID
CUSTOM.EXE
XSETROOT_CUST.EXE

2.2.2.4 Running the Color Customizer

To run the color customizer, perform the following steps:

1. Copy the files CUSTOM.UID and CUSTOM.EXE, which were created during the customizer build, to the directory where the customizer will be run. A typical location is the directory SYS$LOGIN or the directory DECW$USER_DEFAULTS.
2. Copy the files CUSTOM.DAT and DXMDEFAULTS.DAT from the directory DECW$EXAMPLES to the directory DECW$USER_DEFAULTS.
3. Run the executable file CUSTOM.EXE as follows:

```
$ RUN CUSTOM
```

Note

Only the colors of applications invoked after the customizer starts will be affected. For this reason, start the customizer as the first X application during the login process.

2.2.2.5 Modifying the DECW$LOGIN.COM File

As noted in Section 2.2.2.4, the color customizer should be the first X application started during the login process. Do this by starting it as a subprocess from within the DECW$LOGIN.COM file. Add a command to wait approximately 10 seconds between customizer startup and the startup of other applications.

For example, add the following lines to the DECW$LOGIN.COM file:

```
$! Starting the color customizer
$ DISPLAY = F$LOGICAL("DECW$DISPLAY")
$ SPAWN/NOWAIT/OUTPUT='DISPLAY' RUN SYS$LOGIN:CUSTOM.EXE
$ WAIT 0:0:10
```

See *Using DECwindows Motif for OpenVMS* and *Managing DECwindows Motif for OpenVMS Systems* for more information on the file DECW$LOGIN.COM.

2.2.2.6 Command Interface Summary

A box containing a list of available palettes is in the leftmost section of the Color Customizer window. Click on the desired palette to see the colors take affect.

Below the palettes are two arrays of colored buttons, representing the dynamically allocated color cells for normal and shadow colors. To find out what resources are affected by a color cell, click and hold the arrow button next to the color cell.

Hint

As a shortcut, you can click on the screen facsimile in the rightmost corner of the dialog box. If the portion you click on is colored by one of the resource values controlled by the customizer, the pop-up window for the appropriate color button is displayed.

To modify a single color cell, click on the corresponding color button. A colormix widget pops up; as you modify the color, these modifications are reflected in your workstation environment. Use the colormix widget reset button to return to the starting color at any time. You can also change the color cell you are modifying by clicking on a different color button while the colormix widget is displayed.

The automatic shadowing option causes shadow and select colors to be automatically updated when their corresponding background colors are changed. The standard Motif shadowing algorithms are used for these calculations.

Use the File menu to modify, add, and delete color palettes as follows:

- To modify an existing palette, select the palette, change the colors, and choose Save Palette from the File menu.
- To add a new palette, select an existing palette, modify the colors as necessary, and choose Save Palette As... from the File menu. A message box prompts you for the name of the new palette.
- To delete a palette, select the palette and choose Delete Palette from the File menu.

Changes made through the File menu automatically update the CUSTOM.DAT file, which contains the resource defaults.

The File menu Exit button causes the customizer application to exit. A warning dialog is displayed first. Note that the color cells allocated by the customizer (and used by the currently running applications) will be deallocated. After the customizer exits, if the colors of the currently running applications are not correct, the applications should be restarted to restore normal colors. Usually, there is no need to exit the color customizer; it is typically kept running at all times, like the Session Manager.

2.2.2.7 Changing the Mapping Between Color Resources and Color Cells

The file DXMDEFAULTS.DAT allows you to control how many dynamic color cells are allocated and what resources are affected. This file contains resource specifications like the following:

```
*background:       DXmDynamicWindowBackground
*foreground:       DXmDynamicWindowForeground
*topShadowColor:   DXmDynamicWindowTopShadow
```

When the customizer is started, the file DXMDEFAULTS.DAT is written to a property on the root window. Any application that is subsequently run and that uses the correct X Toolkit Library merges these resources with its normal resource database. Resource specifications in this file take precedence over specifications with equivalent resource names in other resource default files.

The resource values within the file DXMDEFAULTS.DAT have a special format. For each unique color value in this file that begins with the string "DXmDynamic", a color button is created in the color customizer. If the string "Shadow" is encountered in a name, the color button is placed in the shadow button box rather than the normal color button box. If a color value string ends with the suffix "Background", it is linked to any color buttons with identical prefixes and suffixes of "TopShadow", "BottomShadow", or "SelectColor" for purposes of automatic shadowing. If a color value named "DXmDynamicScreenBackground" is encountered, the color cell allocated is used by the customizer to set the root window background color.

You can edit the file DXMDEFAULTS.DAT and define resources to use the same color cells. You can have separate dynamic color cells, for scrollbar widgets or for your DECwindows Mail application, for example, by adding the following lines to the file DXMDEFAULTS.DAT:

```
Mail*background:          DXmDynamicMyMailBackground
Mail*foreground:          DXmDynamicMyMailForeground
Mail*topShadowColor:      DXmDynamicMyMailTopShadow
Mail*bottomShadowColor:   DXmDynamicMyMailBottomShadow
```

Adding the previous lines to the file DXMDEFAULTS.DAT and restarting the customizer causes four new color cells to be allocated and four new color buttons to be added to the customizer interface. These buttons are assigned default color values (usually black or white) for each palette. These defaults can then be modified for each palette through the customizer interface.

Note

The text of the DXMDEFAULTS.DAT file is read and parsed by the color customizer. The parsing algorithm does not allow comments, incorrect spacing, or incorrect resource specifications. If this file or the CUSTOM.DAT resource file become corrupt, the customizer cannot start correctly. To resolve the problem, copy the versions of CUSTOM.DAT and DXMDEFAULTS.DAT from the DECW$EXAMPLES directory into your login directory.

2.2.2.8 Using the Customizer with DECterm Windows

To change the colors of DECterm windows, copy the DECterm resource specifications from the file DXMDEFAULTS.DAT and add them to the DECterm resource defaults file DECW$USER_DEFAULTS:DECW$TERMINAL_DEFAULT.DAT. For example, add the following lines to the DECterm resource defaults file:

```
DECW$TERMINAL.main.terminal.background: DXmDynamicTerminalBackground
DECW$TERMINAL.main.terminal.foreground: DXmDynamicTerminalForeground
```

This allows the DECterm window colors to be customized with the color customizer.

2.2.2.9 Changing the Default Value of the Automatic Shadowing Toggle Button

The default value of the automatic shadowing toggle button is set using the Custom.autoShadow resource in the CUSTOM.DAT file as follows:

```
Custom.autoShadowing: False
```

The default value is True.

2.2.2.10 Using the Customizer on Multihead Systems

The color customizer affects only applications started on the same screen as the customizer. On most multihead systems, you can start a different color customizer for each screen and have a different palette in effect on each screen. On multihead systems using XINERAMA, a single instance of the color customizer affects all applications, since the screens function as a single logical screen.

The color customizer can be configured so that it is invoked once and affects all applications regardless of where they are started. This mode is invoked by modifying the Custom.multiScreen resource in the CUSTOM.DAT file as follows:

```
Custom.multiScreen: True
```

The default value is False.

2.2.2.11 Using the XSETROOT_CUST.EXE Demonstration Program

The XSETROOT_CUST.EXE demonstration program, created during the customizer build, is a modified version of the MIT utility program xsetroot that is used to set a bitmap on the root window. The XSETROOT_CUST.EXE program uses DXmDynamicScreenBackground and DXmDynamicScreenForeground as the background and foreground colors of the specified bitmap. If your DXMDEFAULTS.DAT file contains entries for these two dynamic colors, then use the customizer to dynamically modify the colors of your bitmap.

For example:

```
$ XSETROOT_CUST :== "$SYS$LOGIN:XSETROOT_CUST.EXE"
$ XSETROOT_CUST -BITMAP your_xbm_file.XBM
```

2.2.3 Drag and Drop Support

V1.2

The drag-and-drop feature lets you move or copy screen objects. This feature is provided primarily for programmers who choose to incorporate drag-and-drop into their applications. For example, you can move text from a text entry area and paste it elsewhere.

All DECwindows Motif applications except Notepad support the drag-and-drop feature. DECwindows Mail supports the drag-and-drop feature in all windows except the main message area, where DECwindows Mail has its own drag-and-drop; you can use MB2 to move messages around with the SVN interface.

To drag and drop text into a new location:

1. Select the text to be copied or moved with MB1.
2. To move the text, press and hold MB2; to copy the text, press and hold Ctrl/MB2.

   A move or copy icon appears.
3. Drag the icon to the location where you want to drop the text and release MB2.

If the object is highlighted as you drag the icon across it, you can drop the text into that location.

For a list of the widgets that support drag-and-drop functionality, see Section 4.7.1.

2.2.4 Tear-Off Menu Support

V1.2

Most DECwindows Motif applications allow you to tear off pull-down and popup menus. Tear-off menus let you keep frequently used menus displayed without repeatedly pulling them down or popping them up.

To tear off a menu:

1. Display a pull-down or popup menu.

   If the menu is a tear-off menu, a dotted line is displayed at the top of the menu.

2. Click on the dotted line with MB1.

   The menu remains active until it is closed or until the parent application is closed.

To close a tear-off menu:

1. Click on the Window menu button in the tear-off menu.
2. Choose the Close menu item.

2.3 New Desktop Environment

This section describes new features related to the New Desktop environment.

2.3.1 Support for UNIX-Style Filenames

V1.3

Starting with DECwindows Motif for OpenVMS Alpha Version 1.3, you have the ability to display file and device names in UNIX-style format in the File Selection widget and the File Manager (DTFILE).

When this feature is enabled, file and directory specifications are displayed according to UNIX pathname conventions, such as using slashes instead of square brackets to delimit directory trees. In addition, the case of device names is preserved when displaying UNIX-style pathnames versus being converted to uppercase.

The following sections briefly describe how to enable this feature.

2.3.1.1 Enabling in the File Selection Dialog Box

To enable the display of UNIX-style filenames in the File Selection dialog box, set one or more of the following logicals to a non-zero value:

DECC$FILENAME_UNIX_ONLY (CRTL mode)
DECW$XM_FORCE_UNIX_NAMES

These logicals can be defined system-wide by adding them to the SYS$MANAGER:SYLOGICALS.COM file, or defined on a per-user basis by adding them to each user's DECW$LOGIN.COM or LOGIN.COM file.

To force the File Selection dialog box to return selected filenames in OpenVMS format while displaying them in UNIX format, define the logical DECW$XM_UNIX_NAMES_TO_VMS. This enables other applications that rely on filenames in OpenVMS format to interact successfully with the File Selection dialog box while still displaying filenames in UNIX format.

2.3.1.2 Enabling in the File Manager (DTFILE)

To enable the display of UNIX-style filenames in the File Manager (DTFILE) set the logical CDE$DTFILE_UNIX_NAMES to a non-zero value.

This logical can be defined system-wide by adding it to the SYS$MANAGER:SYLOGICALS.COM file, or defined on a per-user basis by adding it to each user's DECW$LOGIN.COM or LOGIN.COM file.

2.3.2 Screen Saver and Screen Lock Support

V1.2–6

New Desktop now supports the MIT Screen Saver extension (MIT-SCREEN-SAVER), which is available on systems running on OpenVMS Alpha Version 7.1 or greater. This extension enables you to use the following features, which are available from the Style Manager Screen dialog box:

- **Screen saver**—Prevents screen burn-in by displaying one or more screen savers after a specific timeout period. You can set both the timeout period and the amount of time each screen saver is displayed onscreen.

  Sample screen savers are available from CDE$SYSTEM_DEFAULTS:[EXAMPLES.DTSCREEN]. To learn how to create additional screen savers and make them available to the Style Manager, see *Getting Started With the New Desktop*.

- **Screen lock**—Secures your current New Desktop session(s) by locking the Front Panel after a specific timeout period. Once locked, a user must enter the account password of the current session to unlock the desktop.

Note that these features are enabled at initial DECwindows Motif startup. You can disable the screen saver and screen lock functions from the Style Manager application.

For more information on using screen saver and screen lock, see the online help for the Style Manager application.

2.3.3 Updated Welcome Message

V1.2–6

The welcome message in the Login Screen now displays a host name regardless of the transport. If the DECnet transport is configured, the DECnet host name is displayed. If the TCP/IP transport is configured, the TCP/IP host name is displayed. If neither transport is configured, a default message of "Welcome to OpenVMS" is displayed.

2.3.4 Selecting Screens on Application Launch

V1.2–5

You can graphically select the screen on which a new application is displayed when launched from either the Front Panel, the File Manager, or the Application Manager. By default, the new application appears on the current screen (that is, the screen containing the mouse pointer). The feature allows you to drop an application icon from the File Manager or Application Manager onto one of the numbered screen controls of the Set Default Screen window. This starts the application on the selected screen without changing the current screen.

The Set Default Screen window is activated by selecting the "Set Default Screen" application in the Application Manager's Desktop Tools folder. You can start a separate instance of the Set Default Screen window on each screen. The highlighting of the default screen is synchronized across all instances of the Set Default Screen window.

2.3.5 Front Panel Icons Support MB3 Operations

V1.2–4

The New Desktop Front Panel supports mouse button 3 (MB3) operations. When the cursor is placed over a Front Panel icon and you press MB3, a subpanel or menu appears. The menu items are as follows:

- Top item–The label of the menu.
- Second item–The application that starts if you single click on the icon.
- Third item–Add or delete a subpanel, depending on whether a subpanel already exists for the control panel.

Caution

If the third item is "Delete Subpanel", this change is difficult to reverse without reinstalling the kit.

2.3.6 Detached Processes

V1.2–4

When you start an application (from the Front Panel or dtfile), a new detached process is created with a process name constructed from the user name, $CDE, and a three-digit numeric identifier.

For example, user SMITH starts an application whose process name is SMITH$CDE001. The next assigned process name would be SMITH$CDE002, unless SMITH$CDE001 has already terminated and is available for reuse.

2.3.7 Viewing Reference Pages

V1.2–4

DECwindows contains a collection of help files for the New Desktop called reference pages (also known as manpages). Reference pages are divided into sections and, on OpenVMS, the file extension indicates the section. Sections distributed with the release include the following:

Section	Purpose	Extension
1	Applications	*filename*.1
3	Libraries/programming	*filename*.3
4	Programming	*filename*.4
5	Include file formats	*filename*.5

A version of dthelpview has been set up with the appropriate action definition for manpage viewing. The process logical MANPATH has been defined to point to the CDE$SYSTEM_DEFAULTS:[MAN] directory that contains all of the reference page files.

You can use either of the following methods to start dthelpview and display reference pages:

Method 1

To start dthelpview from DECterm and view a reference page called dtaction.1:

1. At the DCL level or in a LOGIN.COM file, define dthelpview as a foreign command by entering the following:

```
$ dthelpview :== -
_"$ sys$sysdevice:[sys0.syscommon.cde$defaults.system.bin]dthelpview.exe"
```

2. At the DCL level, enter:

```
$ dthelpview -"manPage" dtaction.1
```

Method 2

To start dthelpview from Application Manager and view the reference page called dtaction.1:

1. Start Application Manager.
2. Double click on the Man Page Viewer icon located in the Desktop Apps application group. A dialog box appears.
3. Enter the reference page to be viewed and click on OK.

2.4 Traditional Desktop Environment

This section describes new features related to the traditional DECwindows desktop environment.

2.4.1 Resource Added for DECwindows XUI Applications

V1.0

The resource `Mwm*useDECMode` has been added to allow previous versions of DECwindows XUI applications to behave correctly with the Motif Window Manager. In particular, this resource is used to control focus, window placement, multiline icons, and the window's initial state (normal or minimized).

2.5 Applications

The following sections describe new features related to specific DECwindows Motif applications.

2.5.1 Bookreader

This section describes features related to the Bookreader application.

2.5.1.1 Bookreader Printing Improved

V1.2–4

The Bookreader application allows only draft-quality printing for books or topics. However, the Bookreader print function has been improved to eliminate such problems as missing lines and words, figures being overwritten by text, and poor leading of lines.

2.5.2 CDA Viewer

This section describes features related to the Compound Document Architecture (CDA) Viewer application.

2.5.2.1 Using the CDA Viewer to View Asian-Language Text

V1.2–3

You can use the CDA Viewer in two ways to view text files that contain Asian characters:

- Specify an options file to the CDA Viewer application.
- Define logical names at the DCL command level or in a LOGIN.COM file.

Refer to the *DECwindows Motif for OpenVMS Applications Guide* for information about using the CDA Viewer.

2.5.2.1.1 Specifying an Options File Specify an options file by including a one-line entry in the file in the following format:

```
TEXT TEXT_ENCODING text_encoding_value
```

- TEXT is the format.
- TEXT_ENCODING is the option you specify to CDA.
- *text_encoding_value* is the value of the codeset. (See Table 2–2 for a list of values.)

Table 2–2 shows the languages, codesets, and text-encoding values.

Table 2–2 Asian Language Codes for Options Files

Language	Codeset	Text Encoding Value
Japanese	DEC Kanji	DEC_KANJI
Japanese	Super DEC Kanji	SDECKANJI
Traditional Chinese	DEC Hanyu	DEC_HANYU
Simplified Chinese	DEC Hanzi	DEC_HANZI
Korean	DEC Korean	DEC_HANGUL

The following table shows examples of one-line entries.

Options File	One-Line Entry		
HANYU.CDA$OPTIONS	TEXT	TEXT_ENCODING	DEC_HANYU
HANZI.CDA$OPTIONS	TEXT	TEXT_ENCODING	DEC_HANZI
HANGUL.CDA$OPTIONS	TEXT	TEXT_ENCODING	DEC_HANGUL

To view the EXAMPLES_CUSTOMERS.TXT file that contains Japanese text in DEC Kanji, use your editor to create an options file called KANJI.CDA$OPTIONS. Add the following one-line entry to the file:

```
TEXT TEXT_ENCODING DEC_KANJI
```

When you access the file through the Options File dialog box with the CDA Viewer, the EXAMPLES_CUSTOMERS.TXT file is viewable in the DEC Kanji codeset (Japanese language).

2.5.2.1.2 Defining Logical Names The second option to enable viewing files in Asian languages is to specify the text file and encoding value by defining two logical names:

- DDIF$READ_TEXT_GL
- DDIF$READ_TEXT_GR

Table 2–3 shows the logical names and associated encoding values.

Table 2–3 Logical Names for Specifying Text Encoding

DDIF$READ_TEXT_GL	DDIF$READ_TEXT_GR	Encoding Value
LATIN1	MCS	MCS
LATIN1	LATIN1	ISO Latin–1
LATIN1	KATAKANA	ASCII–Kana
LATIN1	KANJI	DEC Kanji
ROMAN	MCS	Roman–MCS
ROMAN	LATIN1	Roman
ROMAN	KANJI	Roman–Kanji
ROMAN	KATAKANA	Roman–Kana
LATIN1	HANZI	DEC Hanzi
LATIN1	HANGUL	DEC Hangul
LATIN1	HANYU	DEC Hanyu

You can define the logical names on the DCL command line or in your LOGIN.COM file. For example:

```
$ DEFINE DDIF$READ_TEXT_GL LATIN1
$ DEFINE DDIF$READ_TEXT_GR KANJI
```

Note that this example defines the text encoding for DEC Kanji (see Table 2–3).

2.5.2.2 Converting Files That Contain Asian-Language Characters

V1.2–3

You can convert an Asian-language text file to another format by specifying an options file or by defining the logical names DDIF$READ_TEXT_GL and DDIF$READ_TEXT_GR as discussed in Section 2.5.2.1.1 and Section 2.5.2.1.2.

The format for converting a document from TEXT to another format is as follows:

```
$ CONVERT/DOCUMENT/OPTION=language.CDA$OPTIONS filename.TXT/FORMAT=TEXT -
_$ filename.output_extension/FORMAT=output_format
```

For example, to convert a traditional Chinese language text file to a DDIF file, enter the following command line:

```
$ CONVERT/DOCUMENT/OPTION=HANYU.CDA$OPTIONS -
_$ GUIDELINES_PERSONNEL.TXT/FORMAT=TEXT  GUIDELINES_PERSONNEL.DDIF
```

Note that this command line does not include the /FORMAT=DDIF qualifier; DDIF is the default.

The output file, GUIDELINES_PERSONNEL.DDIF, contains language data.

You can also create Asian language PostScript files from a DDIF, DTIF, or text (ASCII) file. For example, to convert a DDIF file to PostScript (.PS) format, enter the following command:

```
$ CONVERT/DOCUMENT filename.DDIF filename.PS/FORMAT=PS
```

Note

Convert only DDIF and DTIF files that contain language data to Asian language PostScript format.

When you print an Asian language PostScript file on a PostScript printer, ensure that the required language fonts are available on the printer. Otherwise, the PostScript file defaults to a basic set of fonts. If these fonts do not exist, the PostScript file defaults to Courier fonts. Table 2–4 shows the languages and their associated basic fonts.

Table 2–4 Languages and Associated Basic Fonts

Language	Basic Fonts
Japanese	Ryumin-Light-EUC-H or Ryumin-Light-Hankaku
Hanyu	Sung-Light-CNS11643, Sung-Light-DTSCS
Hangul	Munjo
Hanzi	XiSong-GB2312-80

Note

Vertical writing is not supported by the CDA converters. All vertical text is printed horizontally.

2.5.2.3 Dynamic Font Support

V1.2

As well as supporting a static-table for the fonts supported by the DECfonts Typeface Collection Version 1.2, the CDA Run-Time Services includes support for dynamic font lookup. This enables the CDA Viewer to use new fonts as they are installed on the system.

Dynamic font support is implemented using the WRITE$FONTS.INI file, which you can maintain using the Font utility provided with either DECwrite or DECpresent. If a document contains a font not found in the static tables, the CDA Viewer tries to open the WRITE$FONTS.INI file and search for the font. If the font is not found or if the system does not contain a WRITE$FONTS.INI file, the viewer uses a fallback font.

2.5.2.4 Enhanced Display Performance

V1.2

The current version of CDA Run-Time Services includes a performance enhancement that decreases the time it takes to display the first page of a CDA document. Other applications that use the CDA Viewer to view documents (for example, DECwindows Mail) also benefit from this enhancement.

The CDA Viewer enables this performance enhancement feature by default. You can disable the feature as follows:

```
$ DEFINE CDA_QUICK_FIRST_PAGE FALSE
```

The CDA Viewer might not display some documents correctly when this feature is enabled. If you encounter such a problem, disable the feature and invoke the CDA Viewer again.

2.5.2.5 Pack and Unpack Applications

V1.2

CDA Run-Time Services includes two standalone applications that can be used for transferring CDA documents across a network. The CDA Pack application packages a CDA document along with all of its externally referenced files into a single file that can be copied between systems or mailed to other users. The CDA Unpack application reads a file that is packaged by the CDA Pack application and creates a copy of the original document file and all its externally referenced files.

These applications allow you to copy CDA documents between systems without copying externally referenced files separately or correcting external file reference information after copying documents.

To use these applications, add the following lines to your LOGIN.COM file (or add the lines to the SYS$MANAGER:SYLOGIN.COM file):

```
$ PACK == "$SYS$SYSTEM:CDA$PACK.EXE"
$ UNPACK == "$SYS$SYSTEM:CDA$UNPACK.EXE"
```

These lines enable you to use the symbols PACK and UNPACK to invoke the Pack and Unpack applications, respectively.

2.5.2.5.1 Pack Application Syntax The CDA Pack application creates a single output file that contains the contents of a .DDIF or .DTIF input file. The single output file also includes the files that are referenced by the .DDIF or .DTIF input file.

The format of the PACK command is as follows:

```
$ PACK input-file-spec output-file-spec
```

The following sections explain the format of the PACK command.

input-file-spec
Specifies the name of the primary .DDIF or .DTIF input file.

output-file-spec
Specifies the name of the output file that is created by the PACK application. If you do not specify a device or directory, the output file is created in the current default directory.

Qualifiers:

/[NO]SKIP_MISSING
Controls whether the Pack application continues processing if it cannot find one or more of the files that are listed as external references in the input file. The names of any missing files are sent to SYS$ERROR when the Pack application is completed. If you specify /NOSKIP_MISSING, the Pack application does not create an output file if any of the externally referenced files are missing. The default is /SKIP_MISSING.

/[NO]CONTROLLED_COPY
Controls whether the output file includes only those external references that specify COPY_REFERENCE as the value of the ERF_CONTROL item in the input file. If you specify /NOCONTROLLED_COPY, the Pack application includes all referenced files, regardless of the value of the ERF_CONTROL item. The default is /NOCONTROLLED_COPY.

/ALWAYS_ENCODE
Controls whether an output file is created when there are no external references in the input file, or if none of the externally referenced files are found. The default is not to create an output file in these cases. If an output file is not created for these reasons, the Pack application returns the CDA_W_NOOUTFIL status code.

For example:

```
$ PACK MYFILE.DDIF TEST.PACK
```

2.5.2.5.2 Unpack Application Syntax The CDA Unpack application unpacks an input file created by the Pack application. The output files are the .DDIF or .DTIF file that is packed by the Pack application, as well as a file for each external reference in the .DDIF or .DTIF file. The Unpack application sends a list of created files to SYS$ERROR.

The format of the UNPACK command is as follows:

```
$ UNPACK input-file-spec
```

The following sections explain the format of the UNPACK command.

input-file-spec
Specifies the name of the input file that is created by the Pack application.

Qualifier:

/OUTPUT=output-file-spec
Specifies the file name and location of the files created by the Unpack application.

If you specify an output file name without a directory name, the Unpack application creates the main .DDIF or .DTIF file with the file name you specify in the current default directory. It also creates all externally referenced files in the current default directory.

If you specify a directory name without a file name, the Unpack application creates the main .DDIF or .DTIF file and all externally referenced files in the specified directory. The main .DDIF or .DTIF file has the same name as the file packed by the Pack application.

If you specify a directory name and a file name, the Unpack application creates the main .DDIF or .DTIF file and all the externally referenced files, in the specified directory. The main .DDIF or .DTIF file has the file name you specify.

For example:

```
$ UNPACK TEST.PACK
Output file DISK$:[SMITH]MYFILE.DDIF created.
Output file DISK$:[SMITH]FIGURE_1.DDIF created.

$ UNPACK TEST.PACK/OUTPUT=[SMITH.UNPACK]
Output file DISK$:[SMITH.UNPACK]MYFILE.DDIF created.
Output file DISK$:[SMITH.UNPACK]FIGURE_1.DDIF created.

$ UNPACK TEST.PACK/OUTPUT=[SMITH.UNPACK]NEW_FILE.DDIF
Output file DISK$:[SMITH.UNPACK]MYFILE.DDIF renamed NEW_FILE.DDIF
Output file DISK$:[SMITH.UNPACK]FIGURE_1.DDIF created.
```

2.5.2.5.3 Error Messages This section describes messages associated with the CDA Pack and Unpack applications.

FILESPEC, Missing filespec: *file-name*

Severity: Informational

Explanation: The Pack application cannot locate an external file included as an external reference in the .DDIF or .DTIF file or in one of the files referenced in the .DDIF or .DTIF file.

NOOUTFIL, No output file was created.

Severity: Warning

Explanation: The Pack application cannot find external references in the .DDIF or .DTIF document to be packed, and you did not specify the /ALWAYS_ENCODE qualifier.

OUTFILE, Output file created: *file-name*

Severity: Informational

Source: CDA_UNPACK

Explanation: The Unpack application created the specified file while unpacking a file created by the Pack application.

2.5.2.6 New CDA Viewer Error Message

V1.2

The CDA Viewer issues the following message if it is unable to create the application context:

DRMCTXFAIL, DVR could not create application context, aborting

Level: Error

Explanation: The CDA Viewer ends because an attempt to create the application context using the Resource Manager failed, which is usually caused by insufficient memory.

User Action: Reduce the system load and start the application again.

2.5.2.7 WRITE$FONTS Logical Name

V1.2

The WRITE$FONTS logical name references an initialization file used to provide font definitions to the CDA Viewer.

The default location for the WRITE$FONTS.INI file is SYS$LIBRARY, but, if the logical name WRITE$FONTS is defined, the CDA Viewer uses the logical name definition to search for the WRITE$FONTS.INI file.

Full path support is included, so any of the following definitions are valid:

WRITE$FONTS Logical Name	Resulting File
Undefined	SYS$LIBRARY:WRITE$FONTS.INI
DISK:[DIRECTORY]	DISK:[DIRECTORY]WRITE$FONTS.INI
SYS$LOGIN:	SYS$LOGIN:WRITE$FONTS.INI
.TMP	SYS$LIBRARY:WRITE$FONTS.TMP
DISK:[DIRECTORY]FILE	DISK:[DIRECTORY]FILE.INI

2.5.3 Clock

This section describes features that pertain to the Clock application.

2.5.3.1 DECsound Alarm Capability

V1.2

The Clock application includes an alarm feature that can be used if your system supports DECsound. On systems without sound capabilities, you can select only the keyboard bell. When you choose Alarm from the Options menu, a pop-up window appears. This pop-up window allows you to set the alarm time, choose the sound to be played, and indicate an alarm message. To see if your system supports this feature, invoke one of the sounds located in the DECW$EXAMPLES directory (for example, BELLS.AUD).

2.5.4 DECterm

This section describes features that pertain to the DECterm application.

2.5.4.1 Overlay Support

V1.2–3

The latest version of the Window Manager (MWM) supports overlays and utilizes additional planes of memory, which are available on some 3D graphics accelerators. The Window Manager places borders and banners for all the windows into these extra planes of memory and thereby reduces the number of expose events for your applications that use overlays.

You may need to modify your existing applications that use overlays to avoid potential problems with the colormap. HP recommends that you set up your system to share the overlay colormap with the Window Manager, as the hardware supports only one colormap for the overlay planes.

See the associated documentation for your 3D graphic accelerator to determine if overlays are supported.

Setting Up the Overlay Colormap

To modify your applications to share the overlay colormap with the Window Manager, query the server property name SERVER_OVERLAY_COLORMAPS. When you make the query, the server returns the 32-bit value for the overlay Colormap ID.

To set up your system to share the overlay colormap with the Window Manager, edit the files SYS$COMMON:[VUE$LIBRARY.SYSTEM]VUE$MWM.COM and SYS$COMMON:[SYSMGR]DECW$MWM.COM. Change the following line in each file:

```
$ mwm -multiscreen
```

Edit this line by adding the -Overlay command-line option as follows:

```
$ mwm -multiscreen "-Overlay"
```

Note that if you create and install your own colormap, the following problems can result:

- Colors flash on the screen when the colormap is changed.
- Border and banner colors also change when you change the colors of your colormap.

Restrictions

The following restrictions apply when you enable the Window Manager to use overlays:

- The Window Manager supports only single-screen systems and does not function correctly with multiple graphics devices (multihead).
- If you select a Matte Size value other than “None” from the Window Manager options list, the Matte color may not be correct; that is, the color does not match the selection and is occasionally transparent.
- If you select “Show feedback when moving or resizing windows” from the Workspace Options menu, the window with the feedback information causes expose events.

- When you move windows by showing the outline of the window, the outline appears to go below the window borders and banners.
- Window borders are occasionally and randomly displayed in clear or black. If this problem occurs, select the restart option from the Workspace menu to restart the Window Manager.

2.5.4.2 New Default Font Sizes

V1.2–4

In previous releases, the default DECterm font size for the "big" and "little" fonts were chosen by point size. This depended on whether 75 or 100 dpi fonts were installed first in the font path. Starting with DECwindows Motif for OpenVMS Version 1.2–4, DECterm chooses its default fonts by pixel size. The following table shows the two behaviors.

	Prior Releases		Since Version 1.2–4	
Default Font Size (in pixels)	**100**	**75[1]**	**100**	**75**
Big font	25	18	18	18
Little font	18	14	14	14

[1]75 dpi fonts or 100 dpi fonts on a 15-inch monitor

In addition, on 100-dpi displays DECterm now uses the big font by default. This results in DECterm using the same font size (18 pixels) on 100-dpi displays as it did in prior releases. On displays that are less than 325 mm wide, DECterm also now uses a bigger default font.

Follow these steps to use the same font as in previous releases (14 pixels):

1. From the Options menu, select the Window... item.
2. Choose Little Font. Click on Apply.
3. If the new default font is too small, click on the Big Font button and change the pixel size in the Other text entry field from 18 to 25.

2.5.4.3 Scrolling Using the Keyboard

V1.2–3

You can now scroll through text using the keyboard by pressing the Ctrl key and arrow keys or by pressing the Prev or Next key on the editing keypad.

2.5.4.4 ReGIS Input Cursors and Escape Sequences

V1.2–3

The DECterm application supports all ReGIS input cursors:

- Crosshair
- Diamond
- Rubber-band line
- Rubber-band rectangle

For a shape other than the diamond cursor when *n* is equal to 1, define the logical name DECW$TERM_REGIS_CURSOR as one of the numbers defined in the SYS$LIBRARY:DECW$CURSOR file.

V1.2

The following escape sequences are supported by DECterm:

- All page movement sequences (NP, PP, PPA, PPB, and PPR).
- One rectangular area operation sequence (DECCRA).
- The DECLFKC sequence.
- The ReGIS command S(C(In)) supports the rubber-band rectangle cursor and the diamond cursor.

See Section 4.7.3.1, ReGIS Input Cursors for additional information about escape sequences in DECwindows Motif software. See *HP DECwindows Motif for OpenVMS Alpha Release Notes* for details and restrictions on the use of these sequences.

2.5.4.5 Support for Local Echo Mode

V1.2

DECterm supports a local echo mode. In the Options/General dialog box, select Local Echo, which causes all character sequences generated locally to be echoed on the display and passed to the remote host. This feature is useful when connected to a host that does not echo typed characters.

2.5.4.6 Answerback Message Support

V1.2

A user interface is available to enter answerback messages. This answerback message field is for compatibility with HP terminals.

The answerback field is a buffer that contains up to thirty characters. The answerback field in earlier HP terminals contained a message used to identify itself to the host system. For DECterm windows, the answerback field can be used to store a sequence of characters that you can use for any repetitive purpose.

A field is provided in the Options/General dialog box to enter answerback text. Click on the answerback field and enter your text. To enter control characters, encode the control character as a two-digit hex ASCII code, preceded by a number sign (#).

For example, when you enter #0D in the answerback field, DECterm responds with a carriage return.

If two consecutive number sign characters are entered (##), a single number sign is transmitted.

If anything other than a valid two-digit hex code or another number sign is detected after an initial number sign, the number sign is treated as a normal text character. Refer to any ASCII table for a complete list of characters.

The answerback text can also be concealed. When the Conceal Answerback button is enabled, the answerback message is concealed. To deselect the Conceal Answerback button, click on the answerback text field, which erases the previous answerback message.

2.5.4.7 Seven-Bit Printer Support

V1.2

When the 7-Bit Printer button is selected in the Options/Printer dialog box, DECterm modifies printed text to be compatible with printers that do not support 8-bit characters. This includes modifying control sequence introducer (CSI) strings to use the format Escape-Left Bracket rather than the single 8-bit CSI character.

When the 8-Bit Printer button is selected, DECterm allows the use of 8-bit characters when printing. This mode can cause problems for older printers if they can not interpret 8-bit characters.

The default is 8-Bit.

2.5.4.8 Automatic Window Positioning

V1.1

A resource has been defined to manage repositioning a DECterm window when a resize operation forces part of the window off the screen. If a DECterm window is enlarged by using the Options/Window dialog box or by entering a SET TERMINAL/PAGE=*nn* or SET TERMINAL/WIDTH=*nn* command, the controller moves the newly resized DECterm window so that it can be viewed in its entirety. If you prefer DECterm not to move, add the following line to your DECW$TERMINAL_DEFAULT.DAT file:

```
DECW$TERMINAL.main.terminal.autoAdjustPosition: off
```

2.6 Tools and Utilities

The following sections describe new features related to specific X Window System utilities ported to DECwindows Motif.

2.6.1 AccessX Keyboard Utility (accessx)

V1.3

The AccessX Keyboard utility (accessx) is a client application that enables you to set one or more AccessX keyboard enhancements available with the X Keyboard extension (XKB). These enhancements make it easier for users with disabilities to interact with workstation input devices (keyboard and mouse).

Specifically, AccessX enhancements for XKB offer the following capabilities:

- **Sticky Keys** – Allows you to perform multikey operations with one hand, one finger, or a mouth stick. You can use this feature to enter certain uppercase letters and punctuation characters without having to hold down the Shift key. This feature also makes it easier to enter control sequences, such as Ctrl/C.
- **Mouse Keys (also known as Dead Mouse)** – Lets you map actions that you would perform with a mouse to keys on the numeric keyboard or other keys that you specify. With this feature, you can use one finger or a mouth stick to move the cursor to different areas of the screen, manipulate menus, and select, cut, and paste text.
- **Toggle Keys** – Provides audio feedback when the Shift Lock (Caps Lock) key is pressed. This feature helps users who might have difficulty seeing the keyboard light indicator for the Shift Lock key or users who are using a keyboard that does not provide light indicators for any keyboard settings.

- **Repeat Keys** – Allows you to adjust the auto-repeat keyboard mechanism speed or to turn it off entirely. With this feature turned on, you can set your keyboard so that holding down a key for a longer than average time does not cause a repeat entry of that character.
- **Slow Keys** – Makes the keys less likely to respond when brushed accidentally. With this feature turned on, the computer accepts only keystrokes that are held for a certain length of time. The computer ignores light keystrokes that are held only for a moment.
- **Bounce Keys** – Eliminates the problem of pressing a key and then accidentally pressing it again before moving to another key. You can set this feature to tell the computer not to process a second pressing of a key unless a certain length of time elapses between each pressing.
- **Time Out** – Shuts off the AccessX features (except for Repeat Keys) on a workstation after a specified period of time. If you are sharing a workstation and have set AccessX features, the settings are turned off automatically before the next use. To retain the AccessX settings at all times, you can turn off the Time Out feature.

The settings for these capabilities are stored as X resource specifications available from an **AccessX configuration file**. This utility reads the appropriate file (either for the client or server) and adjusts the resource settings depending upon the changes you make. See Section 2.6.1.1 and Section 2.6.1.2 to learn more about the configuration file and its default values.

Note

The accessx utility replaces the sample application formerly available with the AccessX extension. Starting with DECwindows Motif for OpenVMS Alpha Version 1.3, the capability provided by the AccessX extension is offered as part of XKB. For more information on enabling and using XKB and X Keyboard keymaps, see Section 3.5.2. For information regarding the AccessX extension, see the *HP DECwindows Motif for OpenVMS Alpha Release Notes*.

To run this utility, define accessx as a foreign command, and copy the UID file to DECW$USER_DEFAULTS:

```
$ accessx :== "$DECW$EXAMPLES:ACCESSX.EXE"
$ COPY DECW$EXAMPLES:ACCESSX.UID DECW$USER_DEFAULTS:ACCESSX.UID
```

The command format for accessx is as follows:

```
$ accessx [-options...]
```

You can run accessx by entering the command at the DCL prompt, with or without options. The options, described in Table 2–5, allow you to specify which configuration file to load and choose whether to display the status of the editing session. If no options are specified, the client configuration file is loaded by default.

Table 2–5 AccessX Keyboard Utility Options

Options	Description
-client	Instructs accessx to load and use the custom settings in the client configuration file. This is the default option.
-server	Instructs accessx to load and use the default, system-wide settings maintained in the server configuration file.
-status	Displays the status window when accessx is started.
-vmods	Uses the names for virtual modifiers in the status window. For example, with the **-vmods** option, accessx displays NumLock instead of Mod5.

Once invoked, the accessx dialog box is displayed, which lets you select the AccessX features that you want to enable. It also provides a test area that allows you to preview your settings before applying them to a DECwindows Motif session.

For detailed information on the controls and menu options in the dialog box, see the online help available from the Help menu option.

2.6.1.1 The AccessX Configuration File

The AccessX configuration file is an X resource file used to store and load specific AccessX settings for the client. The default client configuration file is located in DECW$USER_DEFAULTS:ACCESSX.DAT and is created the first time you choose Save Settings.

You can choose to edit the file directly or use the accessx utility (with the **-client** or **-server** option) to modify the settings. Note that any comments inserted in this file are deleted when you perform a save settings action.

The format for entries in this file follow the standard X resource specification format, which is:

**resource*:*value*

For example:

```
*BounceKeysToggle.set:False
```

Since an application can consist of a combination of input widgets (such as, push buttons and a scroll bar), you can use the widget class and name identifiers to create additional resource specifications to control these widgets.

2.6.1.2 Default Resource Settings

Table 2–6 lists the AccessX resources and their default values. These are default settings maintained by the server and also represent the initial settings in the client configuration file, prior to modification.

Table 2–6 Default AccessX Resource Settings

Resource	Value
*BounceKeysToggle.set	False
*EnableAccessXToggle.set	True

(continued on next page)

Table 2–6 (Cont.) Default AccessX Resource Settings

Resource	Value
*KRGDebounceScale.decimalPoints	1
*KRGDebounceScale.maximum	40
*KRGDebounceScale.minimum	1
*KRGDebounceScale.value	3
*KRGRepeatDelayScale.decimalPoints	2
*KRGRepeatDelayScale.maximum	400
*KRGRepeatDelayScale.minimum	1
*KRGRepeatDelayScale.value	66
*KRGRepeatRateScale.decimalPoints	2
*KRGRepeatRateScale.maximum	400
*KRGRepeatRateScale.minimum	1
*KRGRepeatRateScale.value	4
*KRGSlowKeysDelayScale.decimalPoints	1
*KRGSlowKeysDelayScale.maximum	40
*KRGSlowKeysDelayScale.minimum	1
*KRGSlowKeysDelayScale.value	3
*MouseKeysToggle.set	False
*MouseAccelScale.decimalPoints	1
*MouseAccelScale.maximum	40
*MouseAccelScale.minimum	1
*MouseAccelScale.value	20
*MouseDelayScale.decimalPoints	1
*MouseDelayScale.maximum	40
*MouseDelayScale.minimum	1
*MouseDelayScale.value	3
*MouseMaxSpeedScale.decimalPoints	0
*MouseMaxSpeedScale.maximum	500
*MouseMaxSpeedScale.minimum	1
*MouseMaxSpeedScale.value	300
*RepeatKeysToggle.set	True
*ShowStatusToggle.set	False
*SlowKeysOnAcceptToggle.set	True
*SlowKeysOnPressToggle.set	True
*SlowKeysToggle.set	False
*SoundOnOffToggle.set	True
*StickyKeysToggle.set	False
*StickyModSoundToggle.set	True
*StickyTwoKeysToggle.set	True

(continued on next page)

Table 2–6 (Cont.) Default AccessX Resource Settings

Resource	Value
*TimeOutScale.decimalPoints	0
*TimeOutScale.maximum	10
*TimeOutScale.minimum	1
*TimeOutScale.value	2
*TimeOutToggle.set	False
*ToggleKeysToggle.set	False

2.6.2 X Authority Utility (xauth)

V1.3

The X Authority utility (xauth) enables you to manage the contents of one or more X authority files. The X authority file contains information used to authorize client connections to the X server.

This utility is typically used to extract authorization records from one system and combine them with the records on another system, such as when granting access to additional users or enabling remote logins. The actual record entries vary depending on the authentication scheme currently in use.

In contrast to other X Window System utilities that are available with DECwindows Motif, xauth is included as a part of OpenVMS Alpha operating system. The xauth commands are case-insensitive and available directly from the DCL command line, xauth command line, or from a batch file.

The command format for xauth is as follows:

```
$ xauth [-f authfile] [-options...] [command]
```

Table 2–7 defines the available options.

Table 2–7 X Authority Utility Options

Options	Description
-f *authfile*	Specifies the name of the authority file. Version numbers are not allowed. If a display device is specified on the command line, xauth will use the X authority file referenced by the display device. Otherwise, xauth will use the default X authority file used by client applications. This file is the X authority file referenced by the DECW$DISPLAY display device, the DECW$XAUTHORITY logical, or SYS$LOGIN:DECW$XAUTHORITY.DECW$XAUTH.
-q	Specifies that xauth operate in quiet mode. Status messages are not displayed. This is the default setting if the output from xauth is not directed to a terminal.
-v	Specifies that xauth operate in verbose mode. Status message are printed. This is the default setting if the output from xauth is directed to a terminal.

(continued on next page)

Table 2–7 (Cont.) X Authority Utility Options

Options	Description
-i	Specifies that xauth ignore file locks. Normally, xauth will refuse to read or edit any files that have been locked by another program (such as, by another instance of xauth) and not timed out.
-b	Specifies that xauth break file locks before proceeding. Use this option only to clean up stale locks.

Table 2–8 defines the available commands.

Table 2–8 X Authority Utility Commands

Commands	Description
add	Adds or replaces the specified entries.
extract	Extracts and writes the specified entries to a new X authority file.
exit	Saves and closes the file and exits the xauth utility. (Available from the xauth command line only.)
remove	Deletes the specified entries.
merge	Appends entries from another X authority file.
nextract	Extracts the specified entries in numerical format.
nmerge	Merges the specified entries presented in numerical format.
list	Displays a listing of entries in the X authority file.
nlist	Displays a listing of entries in numerical format.
generate	Used to generate a new authorization key. Requires that DECwindows Motif be installed and the SECURITY extension be enabled on the X display server.
help	Displays information about the parameters and options for this utility. Subtopic help is also available by typing a question mark (?) at the command prompt.
info	Provides a brief overview of the X authority file.
quit	Closes the xauth utility without applying any changes. (Available from the xauth command line only.)
source	Runs xauth commands from a command file.

2.6.2.1 The X Authority File

The **X authority file** is a binary data file that contains information used to authorize connections to the X server on a system running DECwindows Motif for OpenVMS Alpha Version 1.3 or higher.

Each time an X Window System client application attempts to connect to an X server system that uses an authorization protocol, it references the current X authority file to determine the appropriate **authorization key** to apply in order to authenticate the connection. Each authorization key consists of the protocol name and token, which can be one of the following depending on the protocol in use:

- MIT-MAGIC-COOKIE-1 + random numeric code
- MIT-KERBEROS-5 + encrypted string (cached separately)

By default, an X authority file is created automatically the first time a user logs into a desktop on a system configured for MIT-MAGIC-COOKIE-1 or MIT-KERBEROS-5 authentication. The file is stored in that user's OpenVMS login directory (SYS$LOGIN:DECW$XAUTHORITY.DECW$XAUTH). Each time the user subsequently logs into a desktop on that system, a new authorization key is generated, passed to the X server, and written to the user's X authority file. This key controls access to the X server during the DECwindows Motif session.

A separate X authority file can be manually defined on a server level (using the DECW$SERVER_XAUTHORITY symbol) for those client applications that require access to the X server outside of the normal DECwindows Motif login process.

If the SECURITY extension is enabled, authorization keys can also be manually generated. Manually-generated keys can be used to further restrict server access. The generated key is stored in the X authority file on the client system overwriting any value already present for the specified display server. The key can be distributed to different client systems to allow connections to a specific server and can be revoked to stop subsequent connections.

Generated keys are assigned an **authorization ID** that associates the key with the user who generated the key. As a result, only the user who generated the key can revoke the key.

2.6.2.1.1 Format of an X Authority File Entry Each entry in an X authority file corresponds to a particular X display server and is composed of three main components:

display-name protocol token

display-name
Identifies the name of the X display to which you are authorizing access. The display name follows the format used by the X Window System:

[transport/][host][:]:server[.screen]

This format enables you to use a single X authority file to grant varying levels of access to different X display servers and connection families.

For example, the following entries grant access to the local display server on node HUBBUB and the remote display server on node ZEPHYR via the DECnet transport:

```
local/HUBBUB:0 MIT-MAGIC-COOKIE-1 cfcc5ef98f9718f90154f355c0ae9f62
decnet/ZEPHYR::0 MIT-MAGIC-COOKIE-1 cfcc5ef98f9718f90154f355c0ae9f62
```

- **transport/**
  Identifies the network transport used to connect to an X display server. Valid values are TCPIP/, DECNET/, or LOCAL/. If a transport value is not specified, the default value is interpreted from the format of the remaining portions of the display-name entry, for example:

  Host address and one colon (116.94.24.187:0) (TCP/IP)
  Two colons (::0 or ZEPHYR::0) (DECnet)
  No host name or address and one colon (:0) (local)

- **host[:]**
  Identifies the name of the host system where the X display server is located. A value of 0 is interpreted as the local host, which is the default. The type of host is determined by the transport value. The host name can be in the

format of a recognized IP address, DECnet node specification, or local host identifier, as follows:

HUBBUB.COMPANY.COM (IP host)
116.94.24.187 (IP address)
ZEPHYR: (DECnet node)
25.54: (DECnet address)
HUBBUB (local)

- **:server**
  Identifies the server. This value is required and must be preceded by a single colon (:). Typically the value for a single-server system is :0. If you are specifying a display on a multi-server system (such as when using a proxy server), additional values may apply depending on the number of servers in the configuration. If you have specified a display device (with the SET DISPLAY command), the server portion of the entry is assumed from the device specification.

- **.screen**
  Identifies the screen. On OpenVMS Alpha systems, the screen value is not held in the X authority file and is ignored when included in a command. All screens on a single server have the same authorization.

protocol

Indicates the authentication protocol in use. Valid values are MIT-MAGIC-COOKIE-1 and MIT-KERBEROS-5.

token

A random alphanumeric string that functions as a password authorizing a server connection. The format of the token depends on the authorization scheme in use. MIT-MAGIC-COOKIE-1 uses a 128-bit string known as a magic cookie. MIT-KERBEROS-5 uses an encrypted string to authorize server connections. This string is stored separately. The token entry in the X authority file represents the encoded location of the Kerberos keytab file and associated principal name, which is referenced by the server to locate the encrypted string.

2.6.2.1.2 Specifying an X Authority File By default, the X authority file referenced by client applications and the xauth utility is defined as SYS$LOGIN:DECW$XAUTHORITY.DECW$XAUTH. You can override this default and specify an alternate X authority file in either of the following ways:

- You can create alternate X authority files and switch between them using the DECW$XAUTHORITY logical. For example, the following command changes the X authority file in use for the current DECwindows Motif session to UNTRUSTED.DECW$AUTH:

  ```
  $  DEFINE DECW$XAUTHORITY SYS$MANAGER:[SYSMGR]UNTRUSTED.DECW$XAUTH
  ```

  The logical definition remains in use until it is redefined or an alternate value is specified using the SET DISPLAY/XAUTHORITY command.

- If a display device is used to create a client connection to an X server, you can specify an alternate X authority file using the SET DISPLAY/CREATE/XAUTHORITY command. Note that the file specified on this command line overrides both the default and any file referenced by the DECW$XAUTHORITY logical.

2.6.2.2 Invoking xauth and Entering Commands

You can choose to enter commands interactively from DCL, or enter the utility and issue commands from the xauth command line.

Note that SYS$LOGIN:DECW$XAUTHORITY.DECW$XAUTH is the default X authority file. If you want to work with an alternate file, use the -f option on the command line to specify the filename, as follows:

```
$ XAUTH -f SYS$SYSROOT:[SYSMGR]UNTRUSTED.DECW$XAUTH
Using authority file SYS$SYSROOT:[SYSMGR]UNTRUSTED.DECW$XAUTH

xauth>
```

Tips and Shortcuts

- If you are working with an X authority file other than the default, and plan to enter a series of commands, use the XAUTH -f command to enter the utility; then issue the subsequent commands from the utility command line. Otherwise, you will need to reenter the fully-qualified filename with each xauth command issued from the DCL command line.
- When adding a file entry, you can specify a period (.) in place of the value MIT-MAGIC-COOKIE-1. The period is replaced by the name of the authentication protocol.

2.6.2.3 Accessing Online Help

To display a brief list of the available xauth commands or a summary of their function, issue either the XAUTH ? or XAUTH HELP command.

2.6.2.4 Creating an X Authority File

Use the XAUTH -f ADD command to manually create an X authority file. You must manually create an X authority file for the server when enabling authentication outside of a DECwindows Motif session. You can also choose to create additional user X authority files to store alternate authentication settings, such as for authorizing untrusted network connections.

An X authority file name can consist of any characters currently supported by OpenVMS Alpha; however, the file extension is restricted to a maximum of 37 characters and version numbers are not allowed.

The -f option specifies the name of the X authority file, and the ADD command creates the file by adding an entry. If you do not enter a fully-qualified filename, the new X authority file is written to the current directory by default.

For example, the following command creates a new X authority file UNTRUSTED.DECW$XAUTH to be used to authorize untrusted network connections:

```
$ XAUTH -f UNTRUSTED.DECW$XAUTH ADD :0 .
cfcc5ef98f9718f90154f355c0ae9f62
```

2.6.2.5 Displaying File Information

To assist with debugging file access and write issues, xauth includes a command that displays summary information about a particular X authority file. Use the XAUTH INFO command to display information about an X authority file, such as the current lock status and change history.

For example, the following command displays summary information about the X authority file UNTRUSTED.DECW$XAUTH:

```
$  XAUTH -f SYS$SYSROOT:[SYSMGR]UNTRUSTED.DECW$XAUTH INFO
Authority file:      SYS$SYSROOT:[SYSMGR]UNTRUSTED.DECW$XAUTH
File new:            no
File locked:         yes
Number of entries:   2
Changes honored:     yes
Changes made:        no
Current input:       command line
```

2.6.2.6 Viewing and Editing X Authority Entries

Each X authority file assumes the default protections of the account and directory in which it resides. If you have the appropriate privileges, you can view or edit the contents of an X authority file. To ensure the appropriate level of security, access to this file is typically limited to either the local SYSTEM account, the file owner, or both.

Note

When an X authority file is open for viewing or editing, one or more lock files are created by adding -L or -C to the file extension (such as, *.DECW$XAUTH-C). This renders the X authority file locked from further use. When the file is closed, the lock is subsequently removed, and the lock files deleted.

If a DECwindows Motif session is terminated abruptly, one or more locked files can remain. Use the XAUTH command with options -b or -i to either break or ignore the locks and gain access to the file.

Displaying File Entries

Use the XAUTH LIST command to display the contents of an X authority file. Entries are displayed in the order in which they were added to the file (most recent, last).

For example, the following XAUTH command displays the entries in the X authority file UNTRUSTED.DECW$XAUTH:

```
$  XAUTH -f UNTRUSTED.DECW$XAUTH LIST
local/ZEPHYR:0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62
decnet/ZEPHYR::0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62
116.94.24.187:0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62
```

Note

TCP/IP is considered the default transport for X authority file entries. As a result, the transport portion of the display name is assumed and not displayed for entries that use the TCP/IP transport.

To limit the list to entries related to a particular display, enter the display name at the end of the XAUTH LIST command, as follows:

```
$ XAUTH -f UNTRUSTED.DECW$XAUTH LIST ZEPHYR::0
decnet/ZEPHYR::0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62
```

Adding and Removing File Entries

Use the XAUTH ADD and XAUTH REMOVE commands to add entries to or delete entries from an X authority file.

If you have created a display device (using the SET DISPLAY command), you can specify the device name on the xauth command line to insert or remove entries related to the display device. Typically, the X authority file entry for a display device corresponds to the display server specified by the SET DISPLAY command. However, if the SET DISPLAY command specifies that a proxy server be used, the file entry pertains to that proxy server.

For example, the following X authority file has a single entry for the LOCAL transport on node ZEPHYR. To use the same authorization key for the DECnet transport and to specify that Kerberos be used when connecting to remote node HUBBUB, you could add the following entries to the X authority file UNTRUSTED.DECW$XAUTH:

```
$  XAUTH -f UNTRUSTED.DECW$XAUTH
Using authority file untrusted.decw$xauth

xauth>  LIST
local/ZEPHYR:0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62

xauth>  ADD ::0 . cfcc5ef98f9718f90154f355c0ae9f62
xauth>  ADD HUBBUB::0 MIT-KERBEROS-5 ""

xauth>  LIST
local/ZEPHYR:0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62
decnet/ZEPHYR::0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62
decnet/HUBBUB::0  MIT-KERBEROS-5

xauth> EXIT
Writing X authority file untrusted.decw$xauth
```

Client applications running on systems in the same cluster share a single X authority file. As a result, in cluster configurations, adding an entry for the DECnet transport to the local system grants client applications running on other nodes in the cluster access to that system.

To discontinue remote access to HUBBUB, you could use the XAUTH REMOVE command to remove the entry, as follows:

```
$  XAUTH -f UNTRUSTED.DECW$XAUTH
Using authority file untrusted.decw$xauth

xauth> REMOVE HUBBUB::0
1 entries removed

xauth> LIST
local/ZEPHYR:0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62
decnet/ZEPHYR::0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62

xauth> EXIT
Writing X authority file untrusted.decw$xauth
```

Copying Entries Between X Authority Files

Use one or more of the following XAUTH commands to copy entries for a particular display from one X authority file to another.

This enables you to use an existing entry to grant another user access to a particular display or to obtain access to a remote host from the current display device.

- EXTRACT – Creates a new X authority file whose entries match those in the original file.

- MERGE – Appends the contents of one file to another, replacing entries for the same display name or adding entries for different names.
- NEXTRACT and NMERGE – These commands are designed to be used with the PIPE command. NEXTRACT extracts file entries in a text format that can then be used as input for the NMERGE command.

  For example, the following command extracts the X authority file entry for the local transport from the file UNTRUSTED.DECW$XAUTH and adds it to a new X authority file NEW_XAUTHORITY.DECW$XAUTH:

```
$  PIPE XAUTH -f UNTRUSTED.DECW$XAUTH NEXTRACT SYS$OUTPUT :0 | -
_$ XAUTH -f NEW_XAUTHORITY.DECW$XAUTH NMERGE SYS$INPUT
```

These commands can also be used with the rsh command to copy entries from an X authority file on an OpenVMS host to an X authority file on a remote UNIX system. For example, the following command extracts the entry for TCP/IP access (TCPIP/0:0) and adds it to the current file for user SMITH on the remote Tru64 UNIX system FLOPSY:

```
$  PIPE XAUTH -f UNTRUSTED.DECW$XAUTH NEXTRACT - TCPIP/0:0 | -
_$ rsh/user=smith/password=secret flopsy "xauth nmerge -"
```

Note

When using the PIPE and XAUTH commands to pass information to a Tru64 UNIX host, you must press Ctrl/C to terminate the connection to the Tru64 UNIX host and return control to OpenVMS.

2.6.2.7 Generating Authorization Keys

When the SECURITY extension is enabled on an X display server, you can generate additional authorization keys. Generated keys enable you to further manage server access and control the type of operations performed over the connection. For example, you can revoke a generated key at will, set it to expire after a certain time period, or use it to further grant or restrict the operations (at the X atom level) that can be performed over the connection.

Use the XAUTH GENERATE command to produce a new authorization key. Note that the generated key overwrites any existing key for the current session. To preserve the existing key, specify an alternate X authority file on the XAUTH command line.

For example, the following commands specify the alternate X authority file UNTRUSTED.DECW$XAUTH, generate and display a new key for the local display, and write the generated key to the alternate file:

```
$  XAUTH -f UNTRUSTED.DECW$XAUTH
Using authority file untrusted.decw$xauth

xauth> LIST :0
local/ZEPHYR:0  MIT-MAGIC-COOKIE-1  cfcc5ef98f9718f90154f355c0ae9f62

xauth> GENERATE :0

xauth> LIST :0
local/ZEPHYR:0  MIT-MAGIC-COOKIE-1  cfcc4ff77f3709c46222c355f0ea1c93

xauth> EXIT
Writing X authority file untrusted.decw$xauth
```

2.6.3 X Keyboard Compiler Utility (xkbcomp)

V1.3

The Keyboard Compiler utility (xkbcomp) compiles X Keyboard source files into loadable X Keyboard layout (.XKM) files. Using xkbcomp, you can customize the standard layouts provided with the X Window System by creating or modifying the component source files.

To run this utility, define xkbcomp as a foreign command:

```
$ xkbcomp == "$SYS$SYSTEM:DECW$XKBCOMP"
```

The command format for xkbcomp is as follows:

```
$ xkbcomp [-options...] input-file [output-file]
```

Table 2–9 Keyboard Compiler Options

Option	Description
-a	Specifies that all user actions be displayed.
-C	Creates a C header file during compilation.
-em1 *message*	Prints the specified message before printing any informational or error messages.
-emp *message*	Prints the specified message at the start of each line of messages.
-eml *message*	Prints the specified message after any informational or error messages.
-dflts	Specifies that the compiler generate default values for any missing parameters.
-I [*directory*]	Specifies the top level directory for include statements. A comma-separated list of multiple directories is allowed.
-l [*flags*]	Specifies that a list of matching keymap files be displayed, where *flags* can be one or more of the following options: f: lists fully-specified filenames h: lists hidden keymap files l: generates a long list p: lists partial keymap files r: lists recursive subdirectories The default of -l with no flags turns all options off.
-map *keymap file*	Specifies the keymap file to compile.
-merge	Merges the keymap file with the keymap currently residing on the server.
-o *file*	Specifies the fully-qualified name of the compiled keymap (.XKM) file.

(continued on next page)

Table 2–9 (Cont.) Keyboard Compiler Options

Option	Description
-optional *parts*	Specifies optional components of a keymap file, where *parts* can be any combination of: c: compatibility map g: geometry k: keycodes s: symbols t: types Note that errors in specifying optional components are not fatal.
-R [*DIRECTORY*]	Specifies the directory in which the component source files are located.
-synch	Forces keymap synchronization.
-w [*level*]	Sets the warning level for compiler errors, ranging from 0 (none) to all (10).
-xkb	Generates an X keyboard (.XKB) source file.
-xkm	Generates a compiled keymap (.XKM) file.

2.6.3.1 The X Keyboard Components Database

The X server maintains a database of keyboard components and common keyboard mappings. When combined, these components provide a complete description of a keyboard and its behavior.

The server loads the database from the compiled keymap file specified by the DECW$SERVER_XKEYBOARD_MAP parameter. This file is located in the directory defined by the DECW$SERVER_XKEYBOARD_COMPILED_DIR parameter. If the compiled keymap file does not exist, the server runs xkbcomp to compile the file from its component sources.

The following keyboard component source files comprise the database and are used to produce the loadable keymap files:

- **Keymap source files** – These are the upper-level source files that are specified as input files on the xkbcomp command line. These files are stored in the KEYMAP.DIGITAL subdirectory of the root directory specified by the DECW$SERVER_XKEYBOARD_DIRECTORY parameter. There is one keymap file for each supported language variant, for example:

  ```
  DECW$SYSCOMMON:[SYS$KEYMAP.XKB.KEYMAP.DIGITAL]US
  DECW$SYSCOMMON:[SYS$KEYMAP.XKB.KEYMAP.DIGITAL]JAPANESE
  ```

  The keymap source files reference the following component source files during the compilation to produce complete, loadable keymap (.XKM) file.

- **Keycode component source files** – These files specify the range and interpretation of the raw keycodes reported by the input device. They set the keycode symbolic names, the minimum and maximum legal keycodes for the keyboard, and the symbolic name for each key.

  The keycode files can also contain aliases for keys, symbolic names for indicators, and a description of which indicators are physically present.

The keycode component source files are stored in the KEYCODES.DIGITAL subdirectory of the root directory specified by the DECW$SERVER_ XKEYBOARD_DIRECTORY parameter, for example:

```
DECW$SYSCOMMON:[SYS$KEYMAP.XKB.KEYCODES.DIGITAL]LK
DECW$SYSCOMMON:[SYS$KEYMAP.XKB.KEYCODES.DIGITAL]PC
```

- **Types Source Files** – These files specify the layout types that can be associated with the various keyboard keys. They affect the types symbolic name and the list of layout types associated with the keyboard.

  The types component can also contain real modifier bindings and symbolic names for one or more virtual modifiers.

  These files are stored in the TYPES subdirectory under the root directory specified by the DECW$SERVER_XKEYBOARD_DIRECTORY parameter, for example:

  ```
  DECW$SYSCOMMON:[SYS$KEYMAP.XKB.TYPES]BASIC
  DECW$SYSCOMMON:[SYS$KEYMAP.XKB.TYPES]DEFAULT
  ```

- **Compatibility Map Source Files** – These files specify the rules used to assign actions to keyboard symbols (keysyms) based on the XKB capability (aware or unaware) between the client and server. The XKB capability is determined through the following compatibility transformations:

  - XKB extension state to core state
  - Core keyboard mapping to XKB keyboard mapping
  - XKB keyboard mapping to Core keyboard mapping

  The compatibility map component affects the compatibility symbolic name, the symbol compatibility map, and the group compatibility map. This component can also specify maps for indicators, as well as real modifier bindings and symbolic names of some virtual modifiers.

  The compatibility map source files are stored in the COMPAT subdirectory under the root directory specified by the DECW$SERVER_XKEYBOARD_ DIRECTORY parameter, for example:

  ```
  DECW$SYSCOMMON:[SYS$KEYMAP.XKB.COMPAT]BASIC
  DECW$SYSCOMMON:[SYS$KEYMAP.XKB.COMPAT]DEFAULT
  ```

- **Symbols Source Files** – These files specify the symbols bound to each keyboard key. They affect the symbols symbolic name, a key symbol mapping for each key, the keyboard modifier mapping, and the symbolic names for the keyboard symbol groups. The symbols component can also contain explicit actions and behaviors for some keys, or the real modifier bindings and symbolic names for some virtual modifiers.

  The symbols source files are stored in the SYMBOLS and SYMBOLS.DIGITAL subdirectories under the root directory specified by the DECW$SERVER_XKEYBOARD_DIRECTORY parameter, for example:

  ```
  DECW$SYSCOMMON:[SYS$KEYMAP.XKB.SYMBOLS]US
  DECW$SYSCOMMON:[SYS$KEYMAP.XKB.SYMBOLS.DIGITAL]SWISS_
  FRENCH
  ```

- **Geometry Source Files** – These files define the geometry of the keyboard. They define the geometry symbolic name and the keyboard geometry description. The geometry component can also contain aliases for keys or symbolic names for indicators and might affect the set of indicators that are physically present. Key aliases defined in the geometry component of a keyboard mapping override those defined in the keycodes component.

  These files are stored in the GEOMETRY subdirectory under the root directory specified by the DECW$SERVER_XKEYBOARD_DIRECTORY parameter, for example:

  ```
  DECW$SYSCOMMON: [SYS$KEYMAP.XKB.GEOMETRY.DIGITAL]LK
  DECW$SYSCOMMON: [SYS$KEYMAP.XKB.GEOMETRY.DIGITAL]PC
  ```

- **Other Directories** – The SEMANTICS subdirectory of the base directory DECW$SERVER_XKEYBOARD_DIRECTORY contains a set of files that define the default semantics for keyboard types and compatibility components.

  Files in the base directory DECW$SERVER_XKEYBOARD_DIRECTORY and in subdirectories TMP and RULES are used by the server and should not be modified.

2.6.4 Window Dump to Print File (xpr) Utility

V1.2

The Window Dump to Print File utility prints an X Window dump using the xpr program.

The xpr program receives as input a window dump file produced by the Window Dump utility (xwd) and formats it for output on the following printers:

- PostScript
- DIGITAL LN03 or LA100
- IBM PP3812 page printer
- HP LaserJet (or other PCL printers)
- HP PaintJet

To use the xpr program, define xpr as a user-defined command:

```
$ xpr == "$DECW$UTILS:XPR"
```

You must specify an input file. The xpr program prints the largest possible representation of the window on the output page. Options allow the user to add headers and trailers, specify margins, adjust the scale and orientation, and append multiple window dumps to a single output file.

Use the following command format:

```
$ xpr input_file [options...]
```

Options include:

```
-append filename  -noff  -output filename
-compact
-device {ln03 | la100 | ps | lw | pp | ljet | pjet | pjetxl}
-dump
-gamma correction
-gray {2 | 3 | 4}
-height inches  -width inches
-header string  -trailer string
-landscape  -portrait
-left inches  -top inches
-noposition
-nosixopt
-plane n
-psfig
-render type
-report
-rv
-scale scale
-slide
-split n-pages
```

Table 2–10 defines the available options.

Table 2–10 Window Dump to Print File Options

Option	Description	
-device *devtype*	Specifies the device on which the file is printed.	
	Currently supported devices:	
	la100	DIGITAL LA100.
	ln03	DIGITAL LN03.
	ljet	HP LaserJet series and other monochrome PCL devices such as ThinkJet, QuietJet, RuggedWriter, HP series, and HP-series printers.
	pjet	HP PaintJet (color mode).
	pjetxl	HP PaintJet XL Color Graphics Printer (color mode).
	pp	IBM PP3812.
	ps	PostScript printer.
	lw	LaserWriter is equivalent to -device ps and is provided only for backwards compatibility.
	The default is PostScript.	
-scale *scale*	Affects the size of the window on the page. The PostScript, LN03, and HP printers can translate each bit in a window pixel map into a grid of a specified size. For example, each bit might translate into a 3x3 grid. This would be specified by -scale 3. By default, a window is printed with the largest scale that will fit onto the page for the specified orientation.	
-height *inches*	Specifies the maximum height of the page.	
-width *inches*	Specifies the maximum width of the page.	
-left *inches*	Specifies the left margin in inches. Fractions are allowed. By default the window is centered in the page.	

(continued on next page)

Table 2–10 (Cont.) Window Dump to Print File Options

Option	Description
-top *inches*	Specifies the top margin for the picture in inches. Fractions are allowed.
-header *string*	Specifies a header string to be printed above the window.
-trailer *string*	Specifies a trailer string to be printed below the window.
-landscape	Forces the window to be printed in landscape mode. By default, a window is printed so that its longest side follows the long side of the paper.
-portrait	Forces the window to be printed in portrait mode. By default a window is printed so that its longest side follows the long side of the paper.
-plane *number*	Specifies which bit plane to use in an image. The default is to use the entire image and map values into black and white based on color intensities.
-gray	Uses a 2x2, 3x3, or 4x4 gray scale conversion on a color image, rather than mapping to strictly black and white. This doubles, triples, or quadruples the effective width and height of the image.
-rv	Forces the window to print in reverse video.
-compact	Uses run-length encoding for compact representation of windows with white pixels.
-output *filename*	Specifies an output file name.
-append *filename*	Specifies a file name previously produced by xpr to which the window is to be appended.
-noff	When specified in conjunction with -append, the window appears on the same page as the previous window.
-split *n-pages*	Allows the user to split a window onto several pages. This might be necessary for very large windows that would otherwise cause the printer to overload and print the page in an obscure manner.
-psfig	Suppresses translation of the PostScript picture to the center of the page.
-density *dpi*	Indicates dot-per-inch density to be used by the HP printer.
-cutoff *level*	Changes the intensity level where colors are mapped to either black or white for monochrome output on a LaserJet printer. The level is expressed as percentage of full brightness. Fractions are allowed.
-noposition	Causes header, trailer, and image positioning command generation to be bypassed for LaserJet, PaintJet and PaintJet XL printers.
-gamma *correction*	Changes the intensity of the colors printed by the PaintJet XL printer. The correction is a floating-point value in the range 0.00 to 3.00. Consult the operator's manual to determine the correct value for the specific printer.
-render *algorithm*	Allows the PaintJet XL printer to render the image with the best quality versus performance tradeoff. Consult the operator's manual to determine the available algorithms.

(continued on next page)

Table 2–10 (Cont.) Window Dump to Print File Options

Option	Description
-slide *filename*	Allows overhead transparencies to be printed using the PaintJet and PaintJet XL printers.

The program contains the following limitations:

- Support for PostScript output currently cannot use the -append, -noff, or -split options.
- The -compact option is only supported for PostScript output. It compresses white space but not black space, so it is not useful for reverse-video windows.
- For color images, map directly to PostScript image support.

Program limitations with an LN03 printer:

- The current version of xpr can print most X Windows that are not larger than two-thirds of the screen.

  For example, the LN03 prints a large Emacs window, but fails when trying to print the entire screen.
- The LN03 has memory limitations that cause it to incorrectly print large or complex windows. The two most common errors encountered are "band too complex" and "page memory exceeded" and are described as follows:
  - "band too complex"

    A window may have a particular six pixel row that contains too many changes (from black to white to black). This causes the printer to drop part of the line and possibly drop parts of the page. The printer flashes the number "1" on its front panel when this problem occurs. A possible solution to this problem is to increase the scale of the picture or to split the picture onto two or more pages.
  - "page memory exceeded"

    This occurs if the picture contains too much black space, or if the picture contains complex half-tones, such as the background color of a display. When this problem occurs, the printer automatically splits the picture onto two or more pages. The number "5" may flash on its front panel. As a possible solution to the problem, it might be necessary to either cut and paste or to rework the application to produce a less complex picture.

Program limitations with a LA100 printer:

- The picture is always printed in portrait mode.
- The scale is ignored.
- The scale factor will be different in the horizontal and vertical directions.

Program limitations with an HP printer:

- If the -density option is not specified, 300 dots-per-inch (dpi) is assumed for the ljet device and 90-dpi for the pjet device. The LaserJet printer supports 300-, 150-, 100-, and 75-dpi. Consult the operator's manual to determine the densities supported by other printers.
- If the -scale option is not specified, the image is expanded to fit the printable page area.

- The default printable page area is 8x10.5 inches. Other paper sizes can be accommodated using the -height and -width options.
- Note that a 1024x768 image fits the default printable area when processed at 100-dpi with scale=1; the same image can also be printed using 300-dpi with scale=3, but it requires more data to be transferred to the printer.
- The xpr program may be tailored for use with monochrome PCL printers other than the LaserJet. To print on a ThinkJet (HP 2225A) printer, invoke xpr as follows:

  `xpr -density 96 -width 6.667` *`filename`*

  To print black-and-white output on a PaintJet printer, invoke xpr as follows:

  `xpr -density 180` *`filename`*
- The monochrome intensity of a pixel is computed as 0.30*R + 0.59*G + 0.11*B. If the computed intensity of a pixel is less than the -cutoff *level*, it prints white. This maps light-on-dark display images to black-on-white hard copy. The default cutoff intensity is 50% of full brightness. For example, specifying -cutoff 87.5 means that a pixel will be displayed as black if the computed intensity is less than 85% of full brightness.
- A LaserJet printer must be configured with sufficient memory to print the image. To print a full page at 300-dpi, approximately 2 MB of printer memory is required.
- Color images are produced on the PaintJet printer at 90-dpi. The PaintJet is limited to 16 colors from its 330 color palette on each horizontal print line. The xpr program issues a warning message if more than 16 colors are encountered on a line. Xpr programs the PaintJet for the first 16 colors encountered on each line and uses the nearest matching programmed value for other colors on the line.
- Specifying the -rv option on the PaintJet printer causes black and white to be interchanged on the output image. No other colors are changed.
- Multiplane images must be recorded by xwd in ZPixmap format. Single-plane (monochrome) images may be in either XYPixmap or ZPixmap format.
- Some PCL printers do not recognize image positioning commands. Output for these printers is not centered on the page, and header and trailer strings may not appear where expected.
- The -gamma and -render options are supported only on the PaintJet XL printers.
- The -slide option is not supported on LaserJet printers.
- The -split option is not supported on HP printers.
- The -gray option is not supported on HP or IBM printers.

3
System Management Features

This chapter provides information about new features and enhancements related to DECwindows Motif system management.

3.1 Installation and Upgrade Information

The following sections describe features that pertain to installing and upgrading DECwindows Motif systems.

3.1.1 Version Checking Available for Command Files

V1.0

The DECwindows Motif kit contains version-checking command procedures that layered products can use during their installation procedure. The following three files are placed in the SYS$UPDATE directory during the installation of DECwindows Motif:

- DECW$GET_IMAGE_VERSION.COM

  A command procedure that extracts the image identification string from an image and places it into a user-defined symbol.

- DECW$COMPARE_VERSIONS.COM

  A command procedure that compares two image identification strings and assigns a value to a user-defined symbol with these possible results:

  - Facility codes do not match.
  - Identifiers are the same.
  - Second identifier is older than the first.
  - Second identifier is newer then the first.

- DECW$VERSIONS.COM

  A command procedure used to display the versions of several components of the DECwindows Motif layered product and the X11 display server. The DECW$VERSIONS.COM procedure uses the DECW$GET_IMAGE_ VERSION.COM command procedure to obtain the image idents of each file.

Use the following command to display the versions on sys$output:

```
$  @SYS$UPDATE:DECW$VERSIONS *
```

Component	Description
DECwindows ident	Xlib shareable image
DECwindows server	Server DIX file
DECwindows transport	Transport common
DECwindows Xlib	Xlib shareable image
DECwindows OSF/Motif Toolkit	OSF/Motif Xm Toolkit
DECwindows applications	DECwindows FileView
DECwindows programming	OSF/Motif UIL compiler

The output from the command procedure shows DW, the version number, and the date the image is created.

For example:

```
DW V1.2-4960312
```

is version 1.2-4 and was created on March 12, 1996.

3.2 System Tuning and Performance

This section describes features related to tuning and customizing the DECwindows Motif environment.

3.2.1 Setting the File Manager Refresh Rate

V1.2–6

You can now specify that the File Manager periodically update its view on the New Desktop by adjusting the `Dtfile.rereadTime` setting in the DTFILE.DAT resource file. The value of this setting represents the seconds elapsed between checking for changes in the viewed directories. Note that this setting does not work when viewing search lists.

3.3 Security and Authorization

The following sections describe features that pertain to maintaining system and network security of DECwindows Motif systems.

3.3.1 Enhanced Access Control

V1.3

DECwindows Motif offers support for additional security mechanisms that provide greater control over access to the server by remote applications. Both the DECwindows Motif client software and the DECwindows X11 Display Server have been modified to support the following:

- **Enhanced user-based access control** – The existing user-based authorization mechanism has been extended to support DECwindows Motif systems that operate without a login process, as described in Section 3.3.1.1.
- **Token-based access control** – This includes the use of the MIT-MAGIC-COOKIE-1 and MIT-KERBEROS-5 authentication protocols when clients connect to the X server, as described in Section 3.3.1.2.

- **Use of the SECURITY extension** – Using the SECURITY extension, you can further restrict or extend the access of certain client applications, as described in Section 3.3.1.5.

The following sections describe the available access control schemes and how to use them to manage access to the DECwindows X11 Display Server.

3.3.1.1 User-Based Access Control

User-based access control, as described in Chapter 12 of *Using DECwindows Motif for OpenVMS*, authorizes access to the X server based on the triplet of host, transport, and user name (such as, DECNET ZEPHYR JONES). The user name, node name, and transport information you provide acts as a filter to screen out all except a selected class of users.

User-based access control can be implemented one of two ways depending on your DECwindows Motif system environment:

- For access control outside a DECwindows Motif session, use the access allowed and access trusted files, as described in User-Based Access Control.
- For access control inside a DECwindows Motif session, use the Security Options dialog box to add users to the Authorized Users list, as described in User-Based Access Control.

User-based access control is always available, as long as there are entries in either the Authorized Users or access allowed list. Due to lack of encryption and the inability to specify usernames in the TCP/IP environment, this form of access control is the least secure and is recommended for authorizing access in the local or DECnet environment only.

3.3.1.2 Token-Based Access Control

Token-based access control authorizes access to the X server based on the presentation of a valid password or **token** by a client application during a connection request. The level to which the client is authenticated and the method of authentication varies depending on the protocol in use, which is specified in a user's **X authority file** (described in Section 2.6.2.1).

In general, each time a client application attempts to connect to an X server protected with token-based access control, it references the current X authority file to determine the appropriate protocol to apply and authentication method to follow in order to grant the connection.

Not only do token-based protocols offer greater protection for DECwindows X11 Display Server systems, but they also provide more control over the operations that can be performed over an open X server connection. For example, a token could be used to grant or deny trust privileges. Untrusted connections to an X server significantly restrict the operations that can be performed over the connection.

The token-based access control protocols supported by DECwindows Motif are Magic Cookie (MIT-MAGIC-COOKIE-1) and Kerberos (MIT-KERBEROS-5).

Note

MIT-MAGIC-COOKIE-1 and MIT-KERBEROS-5 are standard X Window System protocols. Third-party client applications can use these protocols to connect to protected DECwindows X display servers and DECwindows Motif clients can use them to connect to protected third-party X

display servers. Additional X Window System protocols, such as XDM-AUTHORIZATION-1 and SUN-DES-1, are not currently supported. Any third-party client applications using these protocols to access a DECwindows X display server will default to user-based access control.

3.3.1.2.1 Magic Cookie (MIT-MAGIC-COOKIE-1) The MIT-MAGIC-COOKIE-1 protocol was designed to provide a more secure alternative to the host-based security mechanism (xhost) available in previous releases of the X Window System. The first protocol to use a token-based approach, it provided the initial, standard means for restricting access to the X server on a user level.

Magic Cookie authorizes connections to an X server based on entries in the X authority file. Each entry for Magic Cookie access control specifies:

- the name of the X display ([transport/][host][:]:server[.screen])
- the protocol name (MIT-MAGIC-COOKIE-1)
- a random, 128-bit numeric code known as a **magic cookie**

Magic Cookie access control can be implemented one of two ways depending on your DECwindows Motif system environment:

- For access control outside a DECwindows Motif session, manually create the X authority file and reference that file using the DECW$SERVER_XAUTHORITY server parameter, as described in Magic Cookie Access Control.
- For access control inside a DECwindows Motif session, use the Security Options dialog box to enable access control, as described in Magic Cookie Access Control.

When Magic Cookie is used to authorize connections during a DECwindows Motif session, a cookie is generated each time a user successfully logs into their local DECwindows Motif desktop. The magic cookie authorizing the local connection, along with the device, transport, and protocol name is passed to the X server and stored in the current X authority file (SYS$LOGIN:DECW$XAUTHORITY.DECW$XAUTH).

Each time a client application attempts to connect to the X server during the session, the application must present a valid cookie to the X server along with the connection request. If the cookie matches the one maintained by the X server, the connection is authorized, access is granted to the X server, and the display is opened.

If the client does not present a valid cookie, the following message is displayed, and the connection is denied:

```
Xlib: connection to "0:0.0" refused by server
Xlib: Invalid MIT-MAGIC-COOKIE-1 key
X Toolkit Error:  Can't Open display
```

When the user logs out of the DECwindows Motif session, the server is reset, and the cookie is discarded.

The basic authorization process remains the same when Magic Cookie is used to authorize X server connections outside of a DECwindows Motif session. However, the file creation process is not. Both the X authority file and the magic cookie must be manually generated.

Due to the use of a randomly-generated token, Magic Cookie provides a more secure form of access control than the user-based scheme. However, the cookies are sent across the network unencrypted, leaving them prone to interception. As a result, this form of access control is recommended for authorizing connections in a local area network (LAN) or limited DECnet environment.

3.3.1.2.2 Kerberos (MIT-KERBEROS-5) Kerberos authorizes connections to an X server based on a combination of the following:

- the protocol name (MIT-KERBEROS-5) in the X authority file
- a list of valid Kerberos principals and their associated passwords
- presentation of valid credentials

Kerberos credentials, or **tickets**, are a set of electronic information that can be used to verify the identity of a **principal**. These principals are stored in an Authorized Principals list kept on the server system. With Kerberos, client applications run by a valid principal send requests for a ticket from the Kerberos **Key Distribution Center (KDC)** each time they attempt to connect to the Kerberos-protected X server.

Kerberos access control can be implemented one of two ways depending on your DECwindows Motif system environment:

- For access control outside a DECwindows Motif session, manually create the X authority file and reference that file using the DECW$SERVER_ XAUTHORITY server parameter, as described in Kerberos Access Control.
- For access control inside a DECwindows Motif session, use the Security Options dialog box to enable access control and add users to the Authorized Principals list, as described in Kerberos Access Control.

Once Kerberos access control is enabled on the server, a new ticket is requested from the KDC automatically each time a user logs into their local desktop. The KDC creates a **ticket-granting ticket** (TGT) associated with the user's principal name, encrypts it using the password as the key, and returns the encrypted TGT.

If the TGT is decrypted successfully, the user is authenticated and the TGT is cached. The TGT permits the authenticated principal to obtain additional tickets. These additional tickets grant access to specific services, in this case, access to the X server from other client applications. The requesting and granting of these additional tickets happens transparently.

With DECwindows Motif, user-to-user authentication is employed. In this model, both the client and server use a Kerberos client at each end of the connection to verify the identify of the user (principal). Once the principal is authenticated at both ends of the connection, access is granted to the X server.

By default, each TGT expires at a specified time. If a TGT has expired or been compromised, you can choose to revoke the current TGT and generate a new TGT by forcing a Kerberos login.

The basic authorization process remains the same when Kerberos is used to authorize X server connections outside of a DECwindows Motif session. However, the credential initialization is not. The user who is running the client application must force initialization using the Kerberos Initialization utility (kinit) or by forcing a login through OpenVMS.

Kerberos is the most secure form of access control since it encrypts the initial authentication information between the requesting client and the server system. Therefore, it is the recommended method for authorizing remote client connections over unsecure networks, such as the Internet.

Note

Kerberos is designed to generate a session key that can be used to encrypt all data transmitted over a network connection. The X Window System uses this key only to encrypt the initial authentication messages. Once the identity of the client has been reliably verified, all subsequent data is sent across the network channel unencrypted. As a result, the server itself can remain susceptible to some forms of network-level attacks.

3.3.1.3 Specifying X Server Access Control

When configuring access control for the X display server, you can choose to apply a traditional user-based scheme, a token-based scheme (such as Magic Cookie or Kerberos), or a combination of schemes depending upon your network environment. For example, you may choose to use Kerberos to authorize all remote server connections over TCP/IP and Magic Cookie to authorize LAN network connections.

When used in combination, the most restrictive access control scheme presented by the client always takes precedence. For example, if the server has all three schemes enabled, and the requesting client is using Magic Cookie, the server will attempt to authorize the connection via Magic Cookie. Note with Magic Cookie access control, user-based access is available by default. If the client attempts and fails to connect to the server using a token-based scheme and is also a member of the Authorized Users list, then access will be granted.

Before enabling access control, take the following actions:

Action	Description
Verify that an access trusted file exists.	In order to change access control settings, one or more OpenVMS Alpha users must hold trust privileges for the DECwindows Motif system. Before enabling authentication, ensure that an access trusted file exists and that at least one account (such as, SYSTEM) has been granted trust privileges. For information about the access trusted file, see The Access Trusted File.
Determine the appropriate method for the DECwindows Motif environment.	Select the authentication method most appropriate to your DECwindows Motif environment, and enable that method only. For example, for DECwindows Motif systems that run applications outside of a desktop, only enable authentication outside of a DECwindows Motif session. Combining schemes can result in a situation where the initial DECwindows Motif login process cannot login.

3.3.1.3.1 Enabling Outside a DECwindows Motif Session Enabling access control outside of a DECwindows Motif desktop session allows authorized OpenVMS users to run X Window System client applications on systems without a login process. This type of access control is used typically for systems that function as a standalone X server, versus an interactive DECwindows Motif workstation.

Use the server customization parameters and either the access allowed or X authority file to set access control, as described in the following sections.

The Access Allowed File

By default, access to the DECwindows X11 Display Server prior to login is limited to the local SYSTEM account via the DECnet or local transport. The **access allowed file** is an ASCII text file that grants additional OpenVMS users access to the X server automatically at server startup.

The access allowed settings remain in effect until a user logs into a DECwindows Motif desktop. Once a user logs into a desktop and begins a DECwindows Motif session, any security options defined with the Session Manager for that user are applied. If a token-based access control scheme has been enabled, additional information may need to be provided by a client application or user in order to gain access to the X server. See Section 3.3.1 for more information on token-based schemes currently supported by DECwindows Motif.

Once the user ends the session, the server is reinitialized, and the access allowed settings are restored.

Caution

The access allowed file is intended for use on workstations that do not normally use the DECwindows Motif login process. Do not use this file on workstations that rely on the DECwindows Motif login process to restrict access to the X server, as it can compromise the security of the DECwindows Motif system.

For example, a user granted access via the access allowed file could spoof a login window that captures the passwords of other users attempting to log into a DECwindows Motif desktop.

The Access Trusted File

Not to be confused with trusted network connections, as described in Section 2.6.2.7, trusted users are those who are authorized to change security settings. The **access trusted file** is an ASCII text file that identifies which OpenVMS users can change the access control settings for a particular DECwindows X11 Display Server.

By default, the local SYSTEM account is granted trust privileges (via the local or DECnet transport). However, when using token-based authentication, trust privileges are not assigned by default. You must manually assign these privileges using the access trusted file.

Entries in this file are automatically added to the access allowed list, unless a token-based authentication scheme is in place. In that case, trusted users must be granted access to the X server either through a manual entry to the access allowed list or via an entry in the appropriate X authority file. Similar to the settings in an access allowed file, access trusted settings remain in effect until a user logs into a DECwindows Motif desktop.

Format of File Entries

Depending on the access control method in place, the format of file entries can differ.

- **For User-Based or Magic Cookie Access Control:** Each entry in an access allowed or access trusted file follows the *transport-host-username* format currently used by the DECwindows Motif security options, as described in *Using DECwindows Motif for OpenVMS*.

  For example, the following entries in an access allowed file grant user JONES local access to the server as well as network access from node ZEPHYR via the DECnet transport:

  ```
  DECNET ZEPHYR JONES
  LOCAL 0 JONES
      .
      .
      .
  ```

  Note that when using TCP/IP as the network transport, access and trust privileges can only be assigned to a host versus to specific users. TCP/IP does not provide the user specification as part of the data provided on a remote connection.

  As a result, file entries for hosts using TCP/IP must contain an asterisk (*) for the user specification. This grants all users on a particular host system access or trust privileges when they connect to the X server using TCP/IP. For example, the following entry in an access allowed file grants access to all users on node ZEPHYR via the TCP/IP transport:

  ```
  TCPIP ZEPHYR *
      .
      .
      .
  ```

- **For Kerberos Access Control:** Entries are only allowed in the access allowed file. Each entry for a Kerberos principal in an access allowed file follows the *protocol-principal@realm-accessrights* format, where *accessrights* can be NONE, ALL, or *.

  For example, the following entry in an access allowed file grants principal JONES access to the server via the TCP/IP transport:

  ```
  KERBEROS jones@ORG.COMPANY.COM ALL
      .
      .
      .
  ```

User-Based Access Control

To apply user-based access control outside of a DECwindows Motif session, establish an access allowed and access trusted file, as follows:

1. Edit the file SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.COM.
2. Define the value of the DECW$SERVER_ACCESS_ALLOWED or DECW$SERVER_ACCESS_TRUSTED parameter so that it refers to the location where each file is stored, such as:

   ```
   $ DECW$SERVER_ACCESS_ALLOWED == "SYS$MANAGER:DECW$SERVER1_ACCESS_ALLOWED.DAT"
   $ DECW$SERVER_ACCESS_TRUSTED == "SYS$MANAGER:DECW$SERVER1_ACCESS_TRUSTED.DAT"
   ```

3. Save the file.

4. Create and edit the access allowed or access trusted file adding the appropriate user entries.

Note

Trust privileges do not automatically grant access to the X server when using token-based authentication. If you are creating an access trusted list on a system that has either Magic Cookie or Kerberos enabled, each user on that list must also have a valid entry in the related access allowed file or X authority file in order to access the X server.

5. Save the file and restart the server. The new access or trust privileges are applied automatically at startup.

Caution

Authorizing TCPIP host connections to an X server using an access allowed file entry provides unauthenticated access to a DECwindows Motif system. This could leave the system vulnerable to unwanted intrusion, Denial of Service (DoS) attacks, and possible data loss.

To ensure the proper level of system security, HP strongly recommends that a token-based scheme (such as, Magic Cookie or Kerberos) be used to authorize remote access to an X server via TCP/IP. Not only do these schemes provide greater system protection, they also allow you to grant (or deny) access on a per-user basis.

Magic Cookie Access Control

To apply Magic Cookie access control outside of a DECwindows Motif session, do the following:

1. Log into the SYSTEM account or another privileged account.
2. Edit the DECW$PRIVATE_SERVER_SETUP.COM file and define the value of the DECW$SERVER_XAUTHORITY parameter so that it refers to the location where the X authority file will be stored, such as:

```
$ DECW$SERVER_XAUTHORITY == "SYS$MANAGER:SERVER_ZEPHYR.DECW$XAUTH"
```

3. Exit and save the file.
4. Using xauth, manually create the X authority file for the server and add the appropriate entries. For example, the following command creates the new X authority file SERVER_ZEPHYR.DECW$XAUTH, adds the entry for the local transport, specifies the Magic Cookie protocol, and assigns a cookie value of 12345abcdef56789:

```
$ XAUTH -f SYS$SYSROOT:[SYSMGR]SERVER_ZEPHYR.DECW$XAUTH ADD -
_$ :0 MIT-MAGIC-COOKIE-1 12345abcdef56789
```

   The authorization key in this file is loaded into the X server at start up and can be used to authorize all client connections, regardless of display name.

5. Restart the server.
6. Propagate the key to all client systems using xauth, as described in Section 2.6.2.6.

Kerberos Access Control

In order to enable Kerberos, you or your system administrator must have first performed the following on the server system:

1. Installed and configured the TCP/IP for OpenVMS Alpha software with a domain name server.
2. Installed and configured the Kerberos Client for OpenVMS software, as described in the *Kerberos Client for OpenVMS Installation Guide and Release Notes*.
3. Obtained the following information:
   - Location of the KDC
   - The appropriate node, domain, and realm information for adding principals
   - Your principal name and password
4. Enable the TCP/IP transport by defining the DECW$SERVER_TRANSPORTS parameter in SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP and restarting the server.

To apply Kerberos access control outside of a DECwindows Motif session, do the following:

1. Invoke the Kerberos Administration utility, as follows:

   ```
   $ KERBEROS/INTERFACE=DECWINDOWS/ADMIN
   ```

2. Create the following principal, keytab file, and keytab file entry. Refer to your Kerberos Client for OpenVMS documentation for information on how to use the Kerberos Administration Utility.
   - Create the principal x0/*host@REALM*, for example:

     ```
     x0/system@ORG.COMPANY.COM
     ```

   - Create the keytab file SYS$SYSROOT:[SYSMGR]DECW$X0.KEYTAB.
   - Create an entry in that keytab file for principal x0.
3. Edit the DECW$PRIVATE_SERVER_SETUP.COM file and define the values of the following parameters so that they refer to the location where the X authority file, access allowed, and access trusted files will be stored, such as:

   ```
   $ DECW$SERVER_XAUTHORITY == "SYS$MANAGER:SERVER_ZEPHYR.DECW$XAUTH"
   $ DECW$SERVER_ACCESS_ALLOWED == "SYS$MANAGER:DECW$SERVER_ZEPHYR_ACCESS_ALLOWED.DAT"
   $ DECW$SERVER_ACCESS_TRUSTED == "SYS$MANAGER:DECW$SERVER_ZEPHYR_ACCESS_TRUSTED.DAT"
   ```

4. Exit and save the file.
5. Using xauth, manually create the X authority file for the server, and add the appropriate entries. For example, the following command creates the new X authority file SERVER_ZEPHYR.DECW$XAUTH, adds the entry for the local transport, specifies the Kerberos protocol, and assigns a value that identifies the keytab file:

   ```
   $  XAUTH -f SYS$SYSROOT:[SYSMGR]SERVER_ZEPHER.DECW$XAUTH -
   _$  ADD :0 MIT-KERBEROS-5 -
   _$  """CS:X0,SYS$SYSROOT:[SYSMGR]DECW$X0.KEYTAB"""
   ```

6. Manually create an access trusted file in the location specified by the DECW$SERVER_ACCESS_TRUSTED parameter, and add an entry for the SYSTEM account, as follows:

   ```
   * SYSTEM 0
   ```

7. Manually create an access allowed file in the location specified by the DECW$SERVER_ACCESS_ALLOWED parameter, and place an entry in the file for each Kerberos principal you want to grant access to the server.

8. Restart the server.

3.3.1.3.2 Enabling Inside a DECwindows Motif Session Use the Security Options dialog box to set the access control scheme in effect inside a DECwindows Motif session. The options in the dialog box enable you to set the access control scheme used by the local X display server, authorize other users access to the X server, and specify the scheme local client applications use when connecting to an X server.

Accessed from the Session Manager (Traditional Desktop) or Style Manager (New Desktop), the settings in Security Options dialog box are identical. Note, however, each desktop stores the settings differently:

- Settings in the Traditional Desktop are stored in the DECW$SMB_SECURITY.DAT file as soon as the changes are applied.
- Settings in the New Desktop are saved whenever a session is saved (such as, when saving a new home session or when saving the current session). Note that on the New Desktop, if you chose to restore the home session on next login, any changes will be lost unless you save them by updating the home session.

User-Based Access Control

To enable user-based access control and grant one or more authorized users access to your workstation display:

1. Do one of the following, depending on the desktop:
   - From the Traditional Desktop, choose Security... from Session Manager's Options menu.
   - From the New Desktop, click the Style Manager Security control.

   The Security Options dialog box is displayed.
2. Under Server Access Control, click Users... to display the Configure Users dialog box.
3. Type the node, the username, and the method of transport for the users you want to authorize.
4. Click on the Add button. The users are added to the Authorized Users list.
5. Click on OK to save and apply the changes and close the Configure Users dialog box.

To disable user-based access control, you must remove all users from the Authorized Users list.

To remove a user name, first click on the names you want to remove. Then click on the Remove button. Finally, click on OK or Apply. The users will no longer have authorized access to your workstation.

Magic Cookie Access Control

To enable Magic Cookie and grant one or more clients presenting a valid magic cookie access to your workstation display:

1. Do one of the following, depending on the desktop:
   - From the Traditional Desktop, choose Security... from Session Manager's Option menu.
   - From the New Desktop, click the Style Manager Security control.

   The Security Options dialog box is displayed.
2. Under Server Access Control, choose the Magic Cookie.
3. Click on OK to save and apply the changes and close the Security Options dialog box.
4. Once enabled, a cookie is generated each time you log into the desktop. To grant other users access to the X server, you must propagate the cookie to their X authority file using the xauth utility, as described in Section 2.6.2.6.

To disable Magic Cookie, deselect the Magic Cookie option and click OK or Apply

To prevent other users from accessing the current session using the current cookie value, click on the Create Cookie button. The new cookie value is added to your default X authority file.

Note

Any client applications that are connected to the X server when a new cookie is generated will remain connected. Authentication occurs only when initially connecting to the X server.

Kerberos Access Control

In order to enable Kerberos, you or your system administrator must have first performed the following on the server system:

1. Installed and configured the TCP/IP for OpenVMS Alpha software with a domain name server.
2. Installed and configured the Kerberos Client for OpenVMS software, as described in the *Kerberos Client for OpenVMS Installation Guide and Release Notes*.
3. Obtained the following information:
   - Location of the KDC
   - The appropriate node, domain, and realm information for adding principals
   - Your principal name and password
4. Enable the TCP/IP transport by defining the DECW$SERVER_TRANSPORT parameter in SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP and restarting the server.

To enable Kerberos, and grant one or more valid Kerberos principals access to your workstation display:

1. Do one of the following, depending on the desktop:
   - From the Traditional Desktop, choose Security... from Session Manager's Options menu.
   - From the New Desktop, click the Style Manager Security control.

   The Security Options dialog box is displayed.
2. Click on the Configure Principals button.
3. Enter the specification(s) for the Kerberos principal(s) you want to add to the Authorized Principals list.

   The format of a typical Kerberos principal is primary/instance@REALM.
4. Click on the Add button. The principal is added to the Authorized Principals box.
5. Click on OK to save and apply the changes and close the Configure Principals dialog box.
6. Under Server Access Control, choose Kerberos, and click OK.

   The Kerberos Login dialog box is displayed, and you are prompted to log in and verify your Kerberos credentials.
7. Enter your Kerberos principal name and password, and click OK. Note that principal names and passwords are case-sensitive.

To disable Kerberos, deselect the Kerberos option, remove all principals from the list, and click OK or Apply.

To prevent one or more principals from accessing your session, first click on the name(s) you want to remove. Then click on the Remove button. Finally, click on OK or Apply. The principal will no longer have authorized access to your workstation.

To prevent all principals from accessing your session, click on the Revoke Ticket button, and click OK or Apply.

3.3.1.4 Specifying Client Access Control

When a client application connects to an X server, the server determines which authentication protocol to use by accessing the current X Authority file. The X Authority file identifies the protocol to use based on the workstation to which the client is connecting. You can make changes to the X authority file using the Security Options dialog box or by directly using the X authority file, as described in Section 2.6.2.

To specify what access control scheme client applications on this workstation follow when connecting to an X server:

1. Do one of the following, depending on the desktop:
   - From the Traditional Desktop, choose Security... from Session Manager's Options menu.
   - From the New Desktop, click the Style Manager Security control.

   The Security Options dialog box is displayed.

2. Under Client Access Control, choose one of the following:

   Authorized Users List
   Kerberos
   Magic Cookie

3. Click on OK to save and apply the changes and close the Security Options dialog box.

   All subsequent client applications run from this system by the current user will apply this access control scheme when connecting to local X servers.

Note

Changes to client access control settings impact the contents of the default X authority file entries (local and DECnet) for the current user only, and do not impact any other access control settings in place on the system.

3.3.1.5 Using the SECURITY Extension

Using the SECURITY extension (described in Section 4.5.1.6), you can choose to manually generate authorization keys using xauth or the SET DISPLAY/GENERATE command. This allows you to specify one of the following additional attributes to apply to a server connection:

- UNTRUSTED – Indicates that this is a **untrusted connection**. An untrusted connection severely restricts the operations that can be performed over the connection. Client applications running over an untrusted connection are allowed limited access to X server extensions and are prevented from accessing windows other than those created by the application. This is the default attribute for all authorization keys.
- TRUSTED – Indicates that this is a **trusted connection**. A trusted connection allows all client operations to occur over the connection.
- TIMEOUT – Sets an expiration period for the token.
- GROUP – Indicates the application group to which the token applies.

Note

Client applications that have not been coded to allow for their use over an untrusted connection may behave unexpectedly. See the specification for the SECURITY extension from X.Org for a description of the limitations of an untrusted connection.

3.3.1.5.1 Enabling the SECURITY Extension To enable the SECURITY extension:

1. Edit the SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.COM file.
2. Search for and define the DECW$SERVER_EXTENSIONS parameter so that it includes a value of "SEC_XAG." For example:

   ```
   $ DECW$SERVER_EXTENSIONS == "SEC_XAG"
   ```

3. Save the file and restart the server.

3.3.1.5.2 Using the Security Policy File The **security policy file** enables you to configure the server to allow certain actions (at the X atom level) to be performed over untrusted network connections. This file establishes one or more site policies that specify the set of allowable actions through a series of field definitions.

A sample file has been provided with DECwindows Motif and is located in DECW$EXAMPLES:DECW$SECURITY_POLICY.TXT. Use this file as a template when creating a policy file. Security policies are described in the *Security Extension Specification Version 7.1* published by X.Org. Refer to this specification for details regarding the use and definition of security policies.

To establish a security policy file on a DECwindows Motif system, do the following:

1. Copy DECW$EXAMPLES:DECW$SECURITY_POLICY.TXT to another file, make the necessary changes, and save the file to an alternate location on the system.
2. Edit the DECW$PRIVATE_SERVER_SETUP.COM file.
3. Define the value of the parameter DECW$SECURITY_POLICY to point to the location where security policy file resides.
4. Save the file and restart the server.

3.4 Desktop Management Features

The following sections describe features that pertain to maintaining desktop applications.

3.4.1 Displaying Custom Messages Prior to Login

V1.3–1

DECwindows Motif for OpenVMS Alpha Version 1.3–1 allows system managers or other privileged users to display custom messages (such as greetings, security updates, and system broadcasts) prior to session log in.

At session startup, DECwindows Motif searches for the message file SYS$MANAGER:DECW$GREET.TXT. If the file is found, a window that contains the text from the message file is displayed in front of the login dialog box. Users must then press the Return key or click OK to continue and log into their New Desktop session. If the file is not found, the window is not displayed, and users can log into their New Desktop session directly.

To create a custom message, do the following:

1. Log into SYSTEM (or another privileged account).
2. Create the file DECW$GREET.TXT in one of the following locations:
   - For standalone systems, create the file in the directory SYS$SPECIFIC:[SYSMGR].
   - If you want to display a message across a cluster, create the file in the directory SYS$COMMON:[SYSMGR].
3. Enter the message text that you want displayed. The text is displayed according to the font family and language variant currently defined in CDE$SYSTEM_DEFAULTS:[CONFIG]XCONFIG.DAT and CDE$SYSTEM_

DEFAULTS:[CONFIG.%L]XRESOURCES.DAT. All printable characters supported by the current font family and variant are valid for display.

Also note that there are no explicit size requirements; the message window will size itself dynamically. Extremely long lines (those that are too long to fit on the screen itself) may be truncated.

Note

Lines that do not contain any text or formatting characters are ignored. To insert a blank line in the message file, enter at least one space character <sp> at the beginning of the line.

4. Save the file and restart the desktop session.

3.4.2 Disabling the Suggested Password List

V1.3

You can disable the display of the suggested password list when logging into New Desktop with an expired password.

To suppress the suggested password list, define the system logical CDE$NOGENPWD to a non-zero value, as follows:

```
$  DEFINE/SYSTEM CDE$NOGENPWD 1
```

3.4.3 Allowing Trusted Users to Unlock Paused Desktop Sessions

V1.3

You can now grant a DECwindows Motif user the ability to unlock a DECwindows Motif session paused using the Screen Lock function described in Section 2.3.2.

To specify the trusted user, define the system logical DECW$TRUSTED_ UNPAUSE logical, as follows, where *username* represents the name of an OpenVMS Alpha user:

```
$  DEFINE/SYSTEM DECW$TRUSTED_UNPAUSE "username"
```

3.4.4 Displaying an Expanded Welcome Message

V1.2–6

You can now enter a longer, customized welcome message to be displayed on the Login Screen of the New Desktop. The size of the welcome message string (Dtlogin*greeting.labelString) in XRESOURCES.DAT has been expanded allowing you to enter more than 8 lines of text.

Note that the actual number of lines you can enter and display is limited by the size of the screen and the selected font. However, a minimum of 25 lines is allowed on most display devices.

3.4.5 Displaying Console Messages

V1.2–3

DECwindows Motif for OpenVMS Version 1.2–3 introduced the feature of displaying console messages in the Console Window application. Previous versions of DECwindows Motif displayed the console window by default.

Note

The new default for displaying console messages starting with the DECwindows Motif for OpenVMS Version 1.2–3 release is DISABLE. The default in previous versions of DECwindows Motif was ENABLE. These values are discussed in greater detail later in this section. If the user selects the Alternate Console port for console communications, the DECwindows Console Window is disabled and the console broadcasts are enabled. Refer to the owner's guide for your workstation for information about selecting the Alternate Console port.

3.4.5.1 Display Options

Specify how to display messages by defining the global symbol DECW$CONSOLE_SELECTION in the customized startup file SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.COM. Enter one of the following values: WINDOW, DISABLE, or ENABLE.

- WINDOW

  Displays console messages in the Console Window application. This is a new application starting with the DECwindows Motif for OpenVMS Version 1.2–3 software. If you specify the WINDOW value, the Console Window is displayed in the lower right corner of the login screen by default and continues to be displayed after the user logs in to the system.

  The Console Window application shares the same executable file and looks similar to the Message Window. However, a menu bar is not displayed in the Console Window; it reads its resources from the DECW$CONSOLE.DAT file instead of from the DECW$MESSAGEPANEL.DAT file. Internally, the Console Window is invoked by running the DECW$MESSAGEPANEL.EXE executable with the command line option *-console*.

  To control the initial position of the Console Window and the classes of OPCOM output that are enabled, you can the define the DECW$CONSOLE_GEOMETRY global symbol in the file SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.COM.

  The DECW$CONSOLE_GEOMETRY symbol specifies the value of the *-geometry* option in the DECW$MESSAGEPANEL.EXE command line; this command is used to start the Console Window application. The default value is "-0-0", which specifies the location of the window in the lower right corner of the screen.

  To position the window at the lower left corner of the screen, for example, add the following line to the command file SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.COM:

  ```
  $ DECW$CONSOLE_GEOMETRY == "+0-0"
  ```

- DISABLE (default)

  Disables broadcasts to the OPA0: device. Console messages are not displayed.

- ENABLE

  Displays console messages in the console window. The console window is a six-line display area at the top of the workstation screen.

Note

Although ENABLE was the default value in previous releases of DECwindows Motif, it is recommended that you do not use this option with DECwindows Motif for OpenVMS Version 1.2–3 and later versions. Displaying console messages by default in the console window can corrupt the contents of the workstation display.

3.4.5.2 Global Symbols

The following list describes the related global symbols:

- DECW$CONSOLE_SELECTION

  Specifies how to display operator-messages options.

- DECW$CONSOLE_GEOMETRY

  Specifies the value of the -geometry option in the DECW$MESSAGEPANEL.EXE command line.

Refer to the chapter "Using DECwindows" in *Managing DECwindows Motif for OpenVMS Systems* for the complete list of global symbols. For information about defining global symbols in the file SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.COM, see *Managing DECwindows Motif for OpenVMS Systems*.

3.4.6 Customizing the Login Screen

V1.2

You can customize the DECwindows Motif login screen on the Traditional desktop to display alternate logos or screen colors. To customize the login screen, create a file named DECW$LOGIN.DAT in the SYS$MANAGER directory that contains your resource definitions. The custom resource definitions from SYS$MANAGER:DECW$LOGIN.DAT are merged with the resource definitions supplied by HP in SYS$COMMON:[DECW$DEFAULTS.SYSTEM]DECW$LOGIN.DAT to form the new login screen.

Keep customized versions of the DECW$LOGIN.DAT resource file in the SYS$MANAGER directory, and **not** in DECW$SYSTEM_DEFAULTS, to prevent your customized file from being overwritten when upgraded to a newer version of DECwindows Motif software. In addition, storing the file in the SYS$MANAGER directory prevents the custom file from superseding the file that is supplied by HP.

3.4.6.1 Customizing the Logo and Login Screen Colors

You can define the resources in Table 3–1 to control the position and colors of the logo and the color of the screen background in the Start Session screen.

Table 3–1 Moving the Logo and Changing Login Screen Colors

Resource	Description
rootColor	Color of the screen background.
logoColor	Color of the logo (default is burgundy).

(continued on next page)

Table 3–1 (Cont.) Moving the Logo and Changing Login Screen Colors

Resource	Description
logoX	x position of the logo (default is 0).
logoY	y position of the logo (default is 75).
centerLogoX	Boolean; if true (default), the logo is centered horizontally on the screen.

For example, to position the logo at x=100, y=600, add the following resource definitions to the SYS$MANAGER:DECW$LOGIN.DAT file:

```
decw$login.logoX: 100
decw$login.logoY: 600
decw$login.centerLogoX: false
```

3.4.6.2 Changing Positions of the Start Session and Set Password Dialog Boxes

You can define the resources in Table 3–2 to control the position of the Start Session and Set Password dialog boxes.

Table 3–2 Changing Position of the Start Session and Set Password Dialog Boxes

Resource	Description
centerStartSessionX	Boolean; if true (default), the Start Session dialog box is centered horizontally.
centerStartSessionY	Boolean; if true (default), the Start Session dialog box is centered vertically.
centerSetPasswordX	Boolean; if true (default), the Set Password dialog box for expired passwords is centered horizontally.
centerSetPasswordY	Boolean; if true (default), the Set Password dialog box is centered vertically.

For example, to position the Start Session dialog box at x=100, y=600, add the following resource definitions to the SYS$MANAGER:DECW$LOGIN.DAT file:

```
decw$login.centerStartSessionX: false
decw$login.centerStartSessionY: false
decw$login.HiddenShell.x: 100
decw$login.HiddenShell.y: 600
```

To position the Set Password dialog box at x=30, y=100, add the following resource definitions to the SYS$MANAGER:DECW$LOGIN.DAT file:

```
decw$login.centerSetPasswordX: false
decw$login.centerSetPasswordY: false
decw$login.SetPasswordShell.x: 30
decw$login.SetPasswordShell.y: 100
```

3.4.6.3 Disabling a Node Name Display in the Start Session Dialog Box

To prevent a node name from being displayed in the Start Session dialog box, add the following resource definition to the SYS$MANAGER:DECW$LOGIN.DAT file:

```
decw$login.displayNodeName: false
```

3.4.7 Displaying Customized Login Logos

V1.1

By default, if there is no DECwindows Motif license registered for the SYSTEM account, DECwindows does not display customized login logos. This is a problem on systems with DECwindows Motif personal-use licenses that do not include SYSTEM on the list of authorized DECwindows users.

To display a customized logo without a DECwindows Motif license for SYSTEM, add the following definition to the SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.COM file:

```
$ DECW$LOGINLOGOSUB == "TRUE"
```

Note

If the file does not exist, copy it from the file SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.TEMPLATE.

After editing the setup file, restart DECwindows Motif using the following command:

```
$ @SYS$MANAGER:DECW$STARTUP RESTART
```

DECwindows Motif login starts the logo process as a subprocess instead of as a detached process. The license check sees that the logo process is a child of the login process and that the X connection is opened.

3.5 Font and Keymap Management

The following sections describe features that pertain to font and keymap support.

3.5.1 Support for Euro Currency Symbol

V1.3

DECwindows Motif includes support for the euro currency symbol. Support for the euro symbol was formerly provided via a separate DECwindows Motif remedial kit (ALP_DWEURO_V0101). The euro font sets can now be installed during the base OpenVMS Alpha operating system installation. Choosing the euro option during installation allows you to then enable and display the euro symbol on all DECwindows Motif for OpenVMS Alpha Version 1.3 or greater systems.

3.5.1.1 Enabling Euro Support

To enable euro support, copy the following template files to separate command files, as follows:

```
$ COPY SYS$COMMON:[SYSMGR]DECW$EURO_APPS_SETUP.TEMPLATE -
_$ SYS$COMMON:[SYSMGR]DECW$EURO_APPS_SETUP.COM
$ COPY SYS$COMMON:[SYSMGR]DECW$EURO_SERVER_SETUP.TEMPLATE -
_$ SYS$COMMON:[SYSMGR]DECW$EURO_SERVER_SETUP.COM
```

Restart your DECwindows Motif system. The command files are run automatically as part of the DECwindows Motif startup procedure.

Support for the euro locale by the OpenVMS C Run-Time Library is not required for base DECwindows Motif euro support. However, if you want to run Motif applications in a euro locale, you must install Euro locale support, which is included in the OpenVMS Alpha kit.

Note

Once euro support is enabled, the character code 0xA4 may be displayed. This results from the euro sign and the key sequence to enter the euro sign always being in effect, regardless of the locale and codeset of the process.

3.5.1.2 Displaying the Euro Symbol in DECwindows Motif Applications

Once euro support is enabled on your system, you can display the euro sign with any ISO8859-1 bitmap font on your workstation, with no additional setup. DECwindows Motif applications that use standard ISO Latin-1 fonts to display text will automatically display the euro sign for character 0xA4. The character set portion of the XLFD name for these fonts is ISO8859-1.

To display the euro sign in a DECterm window, be sure the following two items are selected in the DECterm General Options dialog box:

UPSS ISO Latin-1
8-Bit Multinational Characters

3.5.1.3 Using the Keyboard to Manually Enter the Euro Symbol

Once euro support is enabled, you can manually enter the symbol using one of the following key sequences, depending on the type of keyboard attached to your system:

Keyboard Type	Keymap	Key Sequence

LK-style	*LK201* keymap	Use the same key sequence you use for the universal currency symbol. For example:
		`Compose+Space o x or Compose+Space O X`
		`Compose+Space x o or Compose+Space X O`
		`Compose+Space 0 x or Compose+Space 0 X`
		`Compose+Space x 0 or Compose+Space X 0`

LK-style	*LK401* keymap[1]	LeftCompose + E
PC-style[2]	*LK44* keymap	RightAlt + E

[1]Note that the RUSSIAN_LK401_BT keymap does not support the LeftCompose + E key sequence. The POLISH_LK401_BT keymap supports euro input using the LeftCompose + U key sequence.

[2]Such as, LK44*-**, PCXA*-**, or LK97W-**

3.5.1.4 DECTPU Character Set Qualifier

To display and edit the euro symbol in the DECTPU DECwindows interface, specify the ISO Latin-1 character set as follows:

```
$  EDIT/TPU/INTERFACE=DECWINDOWS/CHARACTER_SET=ISO_LATIN1
```

3.5.2 Support for X Keyboard Keymap Files

V1.3

The X Keyboard keymap files are the standard X Window System alternative to the proprietary keymaps currently provided with DECwindows Motif. They are intended to supplement, rather than replace, the DECwindows Motif keymap files, which will continue to be provided with the DECwindows Motif software.

You can compile X Keyboard layout files to create loadable keymaps using the X Keyboard Compiler utility (xkbcomp), as described in the following section, or the server will compile the files as needed.

Also, since the X Keyboard keymap format (.XKM) is the accepted, vendor-independent standard for loadable keyboards, you can choose to load .XKM files from other X11R6-based systems and X Window System software providers.

3.5.2.1 Creating a Modified Keymap File

To create a modified keymap file, do the following:

1. Edit the one or more component source files described in Section 2.6.3.1, and make the necessary changes. For example, to swap the left and right parenthesis for all US keymaps, edit SYS$COMMON:[SYS$KEYMAP.XKB.SYMBOLS.DIGITAL]US, as follows:

```
.
.
.
19       key <AE09> {       [              9,      parenleft ] };
20       key <AE10> {       [              0,     parenright ] };
.
.
.
```

2. Compile the component source files to create the modified keymap file. For example, to create a modified keymap file for DIGITAL_US_LK401, compile the sources as follows:

```
$  xkbcomp -RDECW$SYSCOMMON:[SYS$KEYMAP.XKB] -xkm -m lk401 -
_$  DECW$SYSCOMMON:[SYS$KEYMAP.XKB.KEYMAP.DIGITAL]us -
_$  -o SYS$COMMON:[SYS$KEYMAP.XKB.COMPILED]digital_us_lk401.xkm
```

You can then load the modified, compiled keymap file as described in Section 3.5.2.2.

3.5.2.2 Loading a Compiled Keymap File

To load a compiled X Keyboard keymap file, do the following:

1. Edit the DECW$PRIVATE_SERVER_SETUP.COM file.
2. Define the value of the parameter DECW$SERVER_EXTENSIONS so that it enables the use of the X Keyboard (XKB) extension, similar to the following:

```
$ DECW$SERVER_EXTENSIONS == "XKB,XINERAMA"
```

3. Define the value of the parameter DECW$SERVER_XKEYBOARD_LOAD_MAP to enable the use of X Keyboard keymaps:

```
$ DECW$SERVER_XKEYBOARD_LOAD_MAP=="1"
```

4. Define the value of the DECW$SERVER_XKEYBOARD_COMPILED_DIR parameter to point to where the keymap files are located. This directory is also where the server places any keymap files that it compiles on demand.

5. Define the value of the DECW$SERVER_XKEYBOARD_MAP parameter to point to the default X Keyboard keymap to load at server startup.
6. Save the file and restart the server.

Note

Some custom keyboard options are not available when using XKB and the X Keyboard keymaps. See the *HP DECwindows Motif for OpenVMS Alpha Release Notes* for a complete listing of X Keyboard keymap restrictions.

3.5.2.3 Enabling the AccessX Key Features

To enable the AccessX key features described in Section 2.6.1, do the following:

1. Edit the DECW$PRIVATE_SERVER_SETUP.COM file.
2. Search for and set the DECW$SERVER_ENABLE_ACCESSX parameter to a value of 1 (enabled).
3. Save the file, and restart the server.

You can then further configure the AccessX features using the accessx utility described in Section 2.6.1, or use the slow and sticky key functions, as follows:

To...	Perform This Action...
Toggle slow keys	Hold Shift key by itself for eight seconds
Toggle sticky keys	Press and release the left or right Shift key five times in a row, without any intervening key events and with less than 30 seconds delay between consecutive presses
Turn off sticky keys	Simultaneously press two or more modifier keys.

3.6 Proxy Server Management

The following sections describe features that pertain to managing the Low-Bandwidth X (LBX) proxy server and related proxy applications.

3.6.1 Support for Low-Bandwidth X (LBX)

Low-Bandwidth X (LBX) is an X server extension that performs compression of the X protocol. LBX was developed for those configurations where the display server is separated from the client by a slow speed line, such as a 56K dial-in modem or a wide-area network (WAN). When the X protocol was developed, the primary use of the protocol was over local area networks (LANs). Therefore, the X protocol was not optimized for low-speed connections. LBX addresses this shortcoming of the X protocol by using a compression and caching scheme designed to minimize the amount of data flow between the client and server.

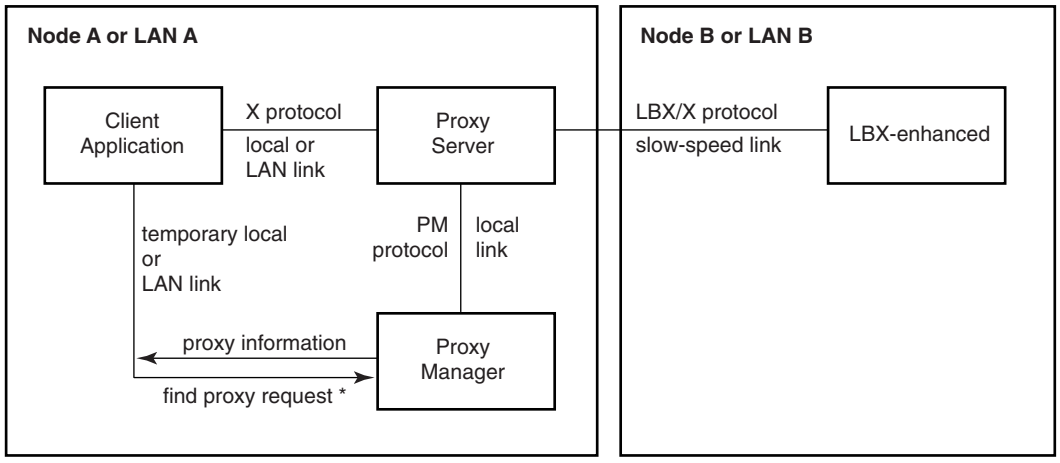
Note

Although LBX reduces data flow between systems, it is not recommended for a LAN-only environment. While it does reduce overall traffic flow, this comes at a cost of increased processing requirements. This generally results in a slight decrease in performance in a LAN-only environment.

The components of the LBX implementation in DECwindows Motif (see Figure 3–1) are as follows:

- **LBX-Enabled X Server**—The use of LBX requires that the X server be capable of interpreting the LBX protocol. On DECwindows Motif systems, you must enable the use of the LBX protocol through the DECW$SERVER_EXTENSIONS customization parameter. For more information about enabling LBX on an DECwindows Motif X server, see Section 3.7.2.
- **Proxy Server**—The proxy server appears to clients as any other X server. The proxy server accepts a connection request from a client program and acts as an intermediary between the client and the X server. Communication between the proxy server and the client uses the standard X protocol. Communication between the proxy server and the X server uses an LBX-enhanced X protocol.
- **Proxy Manager**—The proxy manager relieves clients from managing proxy servers. Instead of sending a request directly to a proxy server, the client sends a request to the proxy manager indicating the requested X server. The proxy manager is responsible for either using an existing proxy server or starting a new proxy server. Once the manager finds a proxy server, it returns the proxy server's address to the client. The proxy manager is optional.
- **DCL SET and SHOW DISPLAY Enhancements**—The SET DISPLAY command has been modified to include qualifiers that allow users to specify either a proxy manager or a proxy server. The changes also provide a method for supplying proxy authentication data. The SHOW DISPLAY command has been modified to display proxy information as well as proxy authentication information. For more information about the changes to the SET DISPLAY and SHOW DISPLAY commands, see the *OpenVMS DCL Dictionary: N–Z*.

Figure 3–1 LBX Components



* Normally performed using the SET DISPLAY command to refresh the DECW$DISPLAY logical. This connection is temporary and is used solely to find the address of the proxy server.

VM-1083A-AI

Note

Because the communication between the client and the proxy server uses the unoptimized X protocol, the client and the proxy server should always be on the same node or on the same LAN.

3.6.1.1 Proxy Server

Proxy servers can be one of three types:

- **Managed** – The proxy server is managed by a proxy manager. The server can be used by multiple clients to access multiple X servers. Clients do not need to know the proxy server's server number, they simply provide the requested X server to the proxy manager. The manager, in turn, either finds the appropriate existing proxy server or automatically starts a new instance of the proxy server automatically.
- **Unmanaged** – The proxy server is started manually. The proxy manager is aware of the server. The server can be used by multiple clients to access multiple display servers.
- **Standalone** – The proxy server is started manually. The proxy manager is not aware of the server. The server can be used by multiple clients to access a single X server. Clients need to know the proxy server's number.

Note the DECwindows Motif LBX proxy server is currently supported only as a managed or standalone configuration.

3.6.1.1.1 Starting LBX Proxy Servers How you start an LBX proxy server determines the proxy server's type and how a client accesses the proxy server.

Note

Before you start an LBX proxy server, ensure that the proxy server is properly authorized to connect to the X server. For more information about authentication in an LBX proxy environment, see Section 3.6.1.3.

Managed Proxy Servers

To start a managed LBX proxy server, place the following LBX service entry in the proxy manager's configuration file (see Section 3.6.1.2.1).

LBX MANAGED COMMAND SYS$MANAGER:DECW$LBXPROXY_SUB ["*qualifiers*"]

After the proxy manager is configured, no specific action is required to start the proxy server; the proxy manager starts the server when the manager receives the first client request.

Standalone Proxy Servers

You can start standalone LBX proxy servers either in the current process or as a detached process. To start a standalone proxy server in the current process, use the LBXPROXY command.

LBXPROXY [*qualifiers*]

For example, to start a proxy server in the current process, assign it server number 50, and have the server act as a proxy for the X server on node remote1.cmp.com, use the following command:

```
$ LBXPROXY /DISPLAY="REMOTE1.CMP.COM:0"/SERVER=50/FIXED_SERVER
```

To start a proxy server as a detached process, use the DECW$LBXPROXY command procedure.

@SYS$MANAGER:DECW$LBXPROXY ["*lbxproxy-qualifiers*"] ["*run-qualifiers*"]

For example, to start a proxy server as a detached process, assign it server number of 50, and have the server act as a proxy for the X server on node remote1.cmp.com, use the following command:

```
$ @SYS$MANAGER:DECW$LBXPROXY "/DISPLAY=""REMOTE1.CMP.COM:0""" + -
_$ "/SERVER=50/FIXED_SERVER"
```

Use the *run-qualifiers* parameter to pass any qualifiers to the RUN command used to invoke the LBXPROXY image. One use of this parameter might be to override the default LBXPROXY process characteristics or any values set by the logicals provided to modify these defaults.

Note

To start an LBX proxy server as a detached process requires the DETACH privilege or available maximum detached process quota. To modify the process quotas for a detached process requires the DETACH privilege.

Modifying the Default LBXPROXY Process Characteristics

Table 3–3 lists the logicals that are provided to override the default LBXPROXY process characteristics specified on the RUN command generated by SYS$MANAGER:DECW$LBXPROXY.

Table 3–3 LBXPROXY Process Characteristic Logicals

Logical	RUN Command Qualifier
DECW$LBX_AST_LIMIT	/AST_LIMIT
DECW$LBX_BUFFER_LIMIT	/BUFFER_LIMIT
DECW$LBX_DUMP	/DUMP
DECW$LBX_ENQUEUE_LIMIT	/ENQUEUE_LIMIT
DECW$LBX_EXTENT	/EXTENT
DECW$LBX_FILE_LIMIT	/FILE_LIMIT
DECW$LBX_IO_BUFFERED	/IO_BUFFERED
DECW$LBX_IO_DIRECT	/IO_DIRECT
DECW$LBX_LOG	/ERROR
DECW$LBX_MAXIMUM_WORKING_SET	/MAXIMUM_WORKING_SET
DECW$LBX_PAGE_FILE	/PAGE_FILE
DECW$LBX_PRIORITY	/PRIORITY
DECW$LBX_PROCESS_NAME	/PROCESS_NAME

(continued on next page)

Table 3–3 (Cont.) LBXPROXY Process Characteristic Logicals

Logical	RUN Command Qualifier
DECW$LBX_QUEUE_LIMIT	/QUEUE_LIMIT
DECW$LBX_WORKING_SET	/WORKING_SET

3.6.1.1.2 LBXPROXY Qualifiers Enter LBXPROXY command qualifiers in the same manner as you would for any other DCL command. For managed servers, specify these qualifiers on the LBX service line in the proxy manager's configuration file.

/ATOMS=file-specification

Specifies the file that the proxy server should use for atoms control. The file SYS$MANAGER:DECW$ATOMCONTROL.TEMPLATE contains an example of an atom control file. This qualifier cannot be specified if the NOATOMS or NOLBX option is specified for the /OPTION qualifier.

The default is /ATOMS=SYS$MANAGER:DECW$ATOMCONTROL.DAT. However, effectively the default is not to use atom control because the installation process does not convert the SYS$MANAGER:DECW$ATOMCONTROL.TEMPLATE file to the SYS$MANAGER:DECW$ATOMCONTROL.DAT file.

At startup, the LBX proxy server "pre-interns" the atoms specified in the atom control file. The atom control file also controls when the proxy server should delay sending data to the X server. This is done by specifying the following:

- Minimum data length required before the server delays any data.
- Which atoms should be delayed only when a window manager is running on the same connection.

The format of the atom control file is documented in the SYS$MANAGER:DECW$ATOMCONTROL.TEMPLATE file.

/CHEAT={ERRORS | EVENTS | NONE}

Specifies the level of cheating allowed on the X protocol for the sake of improved performance. The X protocol guarantees to the requesting party that all corresponding replies, events, or errors are returned to the requester in the same order as the original requests. The ERRORS option allows the proxy server to violate the X protocol with respect to errors. The EVENTS option allows the proxy server to violate the X protocol with respect to errors and events. The NONE option specifies that no protocol cheating is allowed.

The default is /CHEAT=NONE.

Warning

Some X applications may rely upon the correct ordering of events and errors. Enabling cheating may cause these applications to fail. Use this option at your own risk.

/DISPLAY={"network-address" | logical-name | device-name}

Specifies a network address, logical name, or device name that references the X server to which the proxy server should connect. A network address must be in the following form: "[*transport*/][*node*][:]:*display*[.*screen*]"

This option is ignored for managed proxy servers.

The default is /DISPLAY=DECW$DISPLAY.

/FIXED_SERVER
/NOFIXED_SERVER

Specifies that the proxy server should fail to start if the server number specified by the /SERVER qualifier is not available. See /SERVER qualifier for more information about server numbers. This option is useful for starting standalone servers as detached processes. In this case, the proxy server has no method to return the selected server number. This option is ignored for managed proxy servers.

The default is /NOFIXED_SERVER.

/MAXSERVER=value

Specifies the maximum number of X servers to which this proxy server can connect. This option is ignored for standalone servers. Specify a value from 1 to 63.

The default is /MAXSERVER=20.

/MOTION=value

Specifies the maximum number of pointer motion events that are allowed to remain unanswered between the proxy server and the X server. Specify a value from 1 to 32767.

The default is /MOTION=8.

/ONERROR={RECONNECT | TERMINATE}

Specifies the action taken when the proxy server encounters an internal error. This usually occurs when the proxy server loses its connection to the X server.

RECONNECT	Specifies that the proxy server should clean up its internal state information and await further requests. If the proxy server is a standalone server, this option also specifies that the proxy server should reconnect to the X server. For managed proxy servers with multiple connected X servers, the proxy server will try to reconnect each server connection when it fails.
TERMINATE	Specifies that the proxy server should exit. For managed proxy servers with multiple connected X servers, the proxy server will terminate only if all X server connections fail.

The default is /ONERROR=TERMINATE.

/ONEXIT={NOACTION | RESET | TERMINATE}

Specifies the action taken by this proxy server when the last client exits.

NOACTION	Specifies that the proxy server should continue running.
RESET	Specifies that the proxy server should clean up its internal state information and await further requests. If the proxy server is a standalone server, this option also specifies that the proxy server should reconnect to the X server.
TERMINATE	Specifies that the proxy server should exit.

The default is /ONEXIT=NOACTION.

/OPTIONS=(option-list)

Specifies the optimizations to use for this proxy server. With the exception of ALL and NONE, each option has a NO*option* form that disables the option. To enable a small number of options, use a combination of the NONE and the desired options. For example, /OPTIONS=(NONE,IMAGE) suppresses all optimization with the exception of image compression. To disable a small number of options, use a combination of the ALL and the undesired options. For example, /OPTIONS=(ALL,NOIMAGE,NOGRAPHICS) suppresses image and graphics optimization.

ALL	Enables all optimizations. The ALL and NONE options are mutually exclusive.
NONE	Disables all optimizations. The ALL and NONE options are mutually exclusive.
[NO]ATOMS	Enables [disables] reading of the atoms control file. The NOATOMS option is mutually exclusive with the /ATOMS qualifier.
[NO]GRABCMAP	Enables [disables] color map grabbing.
[NO]COMP	Enables [disables] stream compression.
[NO]DELTA	Enables [disables] delta request substitutions.
[NO]GRAPHICS	Enables [disables] reencoding of graphics requests (other than image-related requests).
[NO]IMAGE	Enables [disables] image compression.
[NO]INTERNSC	Enables [disables] short circuiting of InternAtom requests.
[NO]LBX	Enables [disables] all LBX optimizations (equivalent to [NO]ATOMS, [NO]GRABCMAP, [NO]GRAPHICS, [NO]IMAGE, [NO]INTERNSC, and [NO]WINATTR). The [NO]LBX option is mutually exclusive with any of the options controlled by [NO]LBX. The NOLBX option is mutually exclusive with the /ATOMS qualifier.
[NO]RGB	Enables [disables] color name to RGB mapping in the server. The NORGB option is mutually exclusive with the /RGB qualifier.
[NO]SQUISH	Enables [disables] squishing of X events.
[NO]TAGS	Enables [disables] use of tags.
[NO]WINATTR	Enables [disables] GetWindowAttributes/GetGeometry grouping into one round trip.
[NO]ZEROPAD	Enables [disables] zeroing out unused pad bytes in X requests, replies, and events.

The default is /OPTIONS=ALL.

/PARTIAL
/NOPARTIAL

Specifies that this proxy server is allowed to initialize even when it cannot open all transport sockets specified by the /TRANSPORT qualifier. These transport sockets are used to receive client requests. Note that if the proxy server detects that one of the sockets is in use by another task, the server fails to initialize with the server number in use, and the next server number is tried (as discussed in the description of the /SERVER qualifier).

The default is /NOPARTIAL.

/RGB=file-specification

Specifies the file describing the color name to RGB resolution in this proxy server. This qualifier cannot be specified if the NORGB option is specified for the /OPTION qualifier.

The default is /RGB=SYS$MANAGER:DECW$RGB.DAT.

/SERVER=server-number

Specifies the server number to assign to this proxy server. Specify an integer from 0 to 63.

The proxy server first attempts to initialize using the specified server number. If an active server is already using the specified number, the proxy server tries the next lower number. This process is repeated until an unused server number is found or until the proxy server tries the number 0 and fails. If the number 0 fails and the number specified was not 63, the server number wraps and attempts are tried starting with the number 63. The process continues until all numbers have been tried. If all numbers are in use, the proxy server fails to initialize and terminates.

For managed proxy servers, the search for a free server number always starts at 63; any number specified by this qualifier is ignored.

A standalone server started as a subprocess always prints out the server number used on SYS$ERROR. For standalone proxy servers that are detached processes, use the /FIXED_SERVER qualifier to force the server to use the specified server number or fail. Otherwise, there is no method to determine the server number actually used by the server.

The default is /SERVER=63.

/TAGCACHE=cache-size

Specifies the size of this proxy server's tag cache (in bytes). Specify the value 0 to disable tag caching.

The default is /TAGCACHE=1048576 (2^20).

/TRANSPORTS="transport-string"

Specifies the transports that this proxy server monitors for incoming client requests. If the /PARTIAL qualifier is not used, all transports specified by this qualifier must initialize successfully or the proxy server fails to initialize with the server number in use and the next server number is tried, as discussed in the description of the /SERVER qualifier.

The default is /TRANSPORTS="LOCAL,DECNET,TCPIP".

/ZLEVEL=compression-level

Specifies the Zlib compression level used for stream compression. Specify an integer from 1 (least compression) to 9 (most compression).

The default is /ZLEVEL=6.

3.6.1.1.3 Stopping LBX Proxy Servers You can stop LBX proxy servers either automatically or manually.

Stopping Servers Automatically

To stop an LBX proxy server automatically, use the /ONEXIT=TERMINATE qualifier when you start the server. For standalone proxy servers, specify this qualifier either on the LBXPROXY command line or in the *lbxproxy-qualifiers* parameter of the SYS$MANAGER:DECW$LBXPROXY command procedure.

For managed servers, specify this qualifier in the *parameters* argument in the LBX service definition in the proxy manager's configuration file.

Note

If not already terminated, all managed proxy servers automatically terminate when their proxy manager is terminated.

Stopping Servers Manually

To stop an LBX proxy server manually, use the DCL STOP command.

3.6.1.2 Proxy Manager

The proxy manager handles connection requests between client applications, the proxy server, and X server in managed configurations. The following sections describe how to configure and start the proxy manager.

3.6.1.2.1 The Proxy Manager Configuration File The proxy manager configuration file contains the information that the proxy manager needs to be able to locate proxy services. Each line in the configuration file can contain one of the following:

- **Comment** – Comment lines must begin with an exclamation character (!) in the first character position. All other characters in the line are ignored.
- **Managed service entry** – Managed service entries have the following format:

  service-name MANAGED COMMAND *command-file* [*parameters*]

  where:

service-name	Specifies the name of the managed service. The service name is case-insensitive. If the file contains multiple entries with the same service name, only the first occurrence has any effect. For the LBX service, the service name must be LBX. Service names must use characters in the X Portable Character Set with the exception of the Space, Tab, and Newline characters.
command-file	Specifies the name of the command procedure that the proxy manager should invoke to create a new instance of a proxy server for this service. For the LBX service, this argument is usually SYS$MANAGER:DECW$LBXPROXY_SUB.COM.
parameters	Specifies any parameters to pass to the command procedure specified in the *command-file* argument. All characters following the space after the *command-file* argument are passed as parameters to the command procedure. For the LBX service, the command procedure expects one quoted parameter: one or more command qualifiers.

  Currently, the only managed service supplied with DECwindows Motif is LBX.

- **Unmanaged service entry** – Unmanaged service entries have the following format:

  service-name UNMANAGED *address*

  where:

service-name	Specifies the name of the unmanaged service. The service name is case-insensitive. If the file contains multiple entries with the same service name, the manager tries each entry in order until an active and available proxy server is found. Service names must use characters in the X Portable Character Set with the exception of the Space, Tab, and Newline characters.
address	Specifies the address of the proxy server in Inter-Client Exchange (ICE) format.

  Currently, no unmanaged service is supplied with DECwindows Motif.

3.6.1.2.2 Starting the Proxy Manager You can configure the proxy manager to start automatically when DECwindows Motif starts or manually at a later time.

Starting at DECwindows Motif Startup

To start the proxy manager at DECwindows Motif startup, either edit the existing SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.COM file or copy the SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.TEMPLATE file to SYS$MANAGER:DECW$PRIVATE_APPS_SETUP.COM and edit the newly created file. For more information, see the section in the *Managing DECwindows Motif for OpenVMS Systems* manual that describes how to customize the Session Manager environment.

The DECW$PRIVATE_APPS_SETUP.TEMPLATE file contains global symbols that are used to control the behavior of a DECwindows session and its applications. Table 3–4 describes the symbols present in this file that control the proxy manager.

Table 3–4 Global Symbols Controlling the Proxy Manager

Symbol	Description
DECW$PROXY_MANAGER_CONFIG	Specifies the name of the proxy manager's configuration file. The definition of this symbol causes the DECwindows startup process to start the proxy manager. The file name specified by this symbol overrides any configuration file specified in the DECW$PROXY_ MANAGER_OPTIONS symbol. A default configuration file, SYS$MANAGER:DECW$LBXPROXY.DECW$PMCFG, is provided at installation time. This file has a single service entry for the LBX service.
DECW$PROXY_MANAGER_LOG	Specifies the name of the file that the proxy manager should use to log events. If the proxy manager starts and this symbol is undefined, the default log file is SYS$MANAGER:DECW$PROXYMANAGER.LOG.
DECW$PROXY_MANAGER_OPTIONS	Specifies any qualifiers that should be included on the XPROXYMANAGER command when the proxy manager is started. Note that the configuration file specified by the DECW$PROXY_MANAGER_CONFIG symbol always has precedence over any value specified by this symbol.

(continued on next page)

Table 3–4 (Cont.) Global Symbols Controlling the Proxy Manager

Symbol	Description
DECW$PROXY_MANAGER_QUOTAS	Specifies any qualifiers to include on the RUN command line used to start the proxy manager.

Notes

If you restart DECwindows Motif while a proxy manager process is running, the proxy manager does not restart automatically. To ensure that the proxy manager restarts (with any associated options) during DECwindows startup, stop the proxy manager process prior to restarting DECwindows, as described in Section 3.6.1.2.4.

Also note that when restarting the proxy manager as part of DECwindows startup, the owner of the proxy manager process is the user who issues the DECwindows Motif startup command. If DECwindows is started as part of system startup, the owner is the SYSTEM account. If DECwindows is started from another account, verify that the owner of that account has been granted access to the X display server.

Starting Manually

To start the proxy manager manually, use the XPROXYMANAGER command:

XPROXYMANAGER [*qualifiers*]

For example, to start a proxy manager using the configuration file SYS$MANAGER:DECW$LBXPROXY.DECW$PMCFG and the log file SYS$MANAGER:DECW$PM.LOG, use the following command:

```
$ XPROXYMANAGER/CONFIGURATION_FILE=SYS$MANAGER:DECW$LBXPROXY.DECW$PMCFG -
_$ /LOG=SYS$MANAGER:DECW$PM.LOG
```

3.6.1.2.3 Qualifiers Enter XPROXYMANAGER command qualifiers in the same manner as you would for any other DCL command. For proxy managers started at DECwindows startup, you can specify these qualifiers in the definition of the DECW$PROXY_MANAGER_OPTIONS symbol.

/CONFIGURATION_FILE=file-specification

Specifies the configuration file that the proxy manager should use to define the available proxy services.

The default is SYS$LOGIN:DECW$XPROXYMANAGER.DECW$PMCFG.

/LOG=file-specification
/NOLOG

Specifies the log file that the proxy manager should use to log errors. The proxy manager uses this file to record each request for a proxy server. The default is /NOLOG. If the /VERBOSE qualifier is specified, the default is /LOG. If /LOG is specified without a value, the output is sent to SYS$ERROR by default.

If an incomplete specification is entered, the following directory specification is used to complete the command:

```
SYS$SYSDEVICE:[]DECW$XPROXYMANAGER.LOG
```

/PORT=*integer*

Specifies the port number that this server monitors for incoming requests. For the TCP/IP transport, this qualifier specifies an IP port number from the port number space shared by all users of TCP/IP. For the DECnet and Local transports, this qualifier specifies a number that is used to create a unique resource name from the name space of all users of the Inter-Client Exchange (ICE) protocol. Specify a value from 1 to 16383.

The default port number is 6500.

/TRANSPORT="transport-string"

Specifies the transports which this proxy manager monitors for incoming client requests.

The default is /TRANSPORT="LOCAL,DECNET,TCPIP".

/VERBOSE
/NOVERBOSE

Specifies whether the proxy manager should log each proxy request to the proxy manager log file, in addition to error messages.

If the /LOG qualifier is specified, the default is /VERBOSE. If the /LOG qualifier is not specified, the default is /NOVERBOSE.

3.6.1.2.4 Stopping a Proxy Manager To stop a proxy manager, you must use the DCL STOP command. The STOP/EXIT=USER_MODE option (new to OpenVMS Alpha Version 7.3-1) is recommended.

Stopping the proxy manager also stops all managed proxy servers controlled by the proxy manager.

3.6.1.3 Authentication in an LBX Environment

When the proxy server connects to an X server, the proxy server undergoes authentication in the same manner as a client. How the proxy server obtains its authentication information depends on the type of proxy server.

A managed proxy server obtains its authentication information from the proxy manager. The proxy manager in turn receives the authentication information from the client. The client's default authentication information is contained in the client's X authority file. The client can control which X authority file is used by using the /XAUTHORITY qualifier to the SET DISPLAY command. The client can supply explicit authentication information on the SET DISPLAY command using the /LBXAUTHENTICATE and /LBXDATA qualifiers. The client also has the option of using the /NOLBXAUTHENTICATE qualifier to specify that the authentication information come from the proxy server's current X authority file. For more information about the SET DISPLAY command qualifiers for LBX, see *OpenVMS DCL Dictionary: N–Z*.

A standalone proxy server obtains its authentication information from the information present in the current X authority file.

Note

Clients connecting to an X server through an LBX proxy server must have a valid entry for the proxy server in the client X authority file. If the associated entry contains a generated, untrusted cookie, access to the X server is only granted for the initial proxy server connection. Subsequent

client connections cannot use the same cookie to gain access to the X server.

3.7 X Display Server Management

The following sections describe features that pertain to managing the DECwindows X11 Display Server.

3.7.1 New Server Parameter for Setting Process Priority

V1.3–1

The DECW$SERVER_PRIORITY parameter controls the priority of the X display server process. This parameter enables you to reduce the priority of the server process and improve system performance in request-intensive situations where response time is sluggish.

Some client applications send continuous requests that do not require a reply to the X display server. These request can impact the processing time allocated to other applications. This can sometimes slow the response time of applications such as the Window Manager so significantly that they appear locked.

To set the priority of the server process, do the following:

1. Estimate the optimal priority for the server process; the value must be in the range from 1 (low) to 15 (high). For the best results, HP recommends that you use a value of 4, 5, or 6 (default). Setting the priority too low can reduce the responsiveness of input devices (such as keyboard or mouse actions).
2. Edit the SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.COM file.
3. Define the DECW$SERVER_PRIORITY parameter so that it sets the priority to the desired value, similar to the following:

   ```
   $ DECW$SERVER_PRIORITY == "4"
   ```

4. Save the file and restart the server.

3.7.2 New Server Customization Parameters Available with X11R6.6 Upgrade

V1.3

In support of the enhanced X server device setup, keyboard, security, and error reporting capabilities provided with the X11R6.6 upgrade, the following parameters have been added to the DECW$PRIVATE_SERVER_SETUP.COM file:

Table 3–5 New DECwindows X11 Display Server Customization Parameters

Parameter	Type	Default Value	Range
DECW$SECURITY_POLICY	String		
DECW$SERVER_ACCESS_TRUSTED	String	SYS$MANAGER:DECW$SERVER_ACCESS_TRUSTED.DAT	
DECW$SERVER_ACCESS_ALLOWED	String	SYS$MANAGER:DECW$SERVER_ACCESS_ALLOWED.DAT	

(continued on next page)

Table 3–5 (Cont.) New DECwindows X11 Display Server Customization Parameters

Parameter	Type	Default Value	Range
DECW$SERVER_AUDIT_LEVEL	Integer	0	0,1,2,4
DECW$SERVER_DISABLESCREEN	Integer		0...15
DECW$SERVER_DISABLE_TEST	Boolean	False	
DECW$SERVER_EDGE_BOTTOM	Integer list		
DECW$SERVER_EDGE_LEFT	Integer list		
DECW$SERVER_EDGE_RIGHT	Integer list		
DECW$SERVER_EDGE_TOP	Integer list		
DECW$SERVER_ENABLE_ACCESSX	Boolean	0	
DECW$SERVER_ENABLESCREEN	Integer		0...15
DECW$SERVER_EXTENSIONS	String list	XIE,DEC-XTRAP,MULTI-BUFFERING,SEC_XAG	
DECW$SERVER_KEY_REPEAT_DELAY	Integer	660	0...1000
DECW$SERVER_KEY_REPEAT_INTERVAL	Integer	40	0...1000
DECW$SERVER_ONLYSCREEN	Integer		0...15
DECW$SERVER_SCREENS	String list		
DECW$SERVER_XAUTHORITY	String		
DECW$SERVER_XKEYBOARD_COMPILED_DIR	String list	SYS$COMMON:[SYS$KEYMAP.XKB.COMPILED]	
DECW$SERVER_XKEYBOARD_DIRECTORY	String list	DECW$SYSCOMMON:[SYS$KEYMAP.XKB]	
DECW$SERVER_XKEYBOARD_LOAD_MAP	Integer	0	0...1
DECW$SERVER_XKEYBOARD_MAP	String	DIGITAL_US_LK201	

If you plan on defining any of these parameters, do one of the following:

- Copy the updated version of SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.TEMPLATE to the file SYS$MANAGER:DECW$PRIVATE_SERVER_SETUP.COM and make the necessary modifications
- Enter the parameter definitions in your existing server startup file(s).

The following sections further describe each parameter. Note that these parameters are intended for use on or with OpenVMS Alpha Version 7.3–2 systems only. For a description of additional X server parameters, see *Managing DECwindows Motif for OpenVMS Systems*.

3.7.2.1 Extension Setup

While some extensions are a permanent part of the DECwindows X11 Display Server and are always enabled, some require activation through a parameter definition. Use the new parameters described in this section to set the range of active extensions on one or more DECwindows X11 Display Server systems.

See Section 3.7.3 for instructions on how to enable one or more X server extensions. For a brief description of the new extensions available with the DECwindows X11 Display Server, see Section 4.5.1.

DECW$SERVER_EXTENSIONS

This parameter determines which loadable server extensions are enabled and active. The valid values for DECW$SERVER_EXTENSIONS are:

D2DX-EXTENSIONS
DBE
DEC-XTRAP
LBX
MULTI-BUFFERING
SEC_XAG
XIE
XINERAMA
XKB

The default is "XIE,DEC-XTRAP,MULTI-BUFFERING,SEC_XAG." If you have user-written, third-party, or other HP X Window System extensions, you can use this parameter to enable the extensions at server startup.

Note

To prevent contention over resources or server requests, some combinations of extensions should not be loaded on the same display server system. See the *HP DECwindows Motif for OpenVMS Alpha Release Notes* for the list of unsupported combinations of server extensions.

The following parameter definition specifies the range of server extensions to enable:

Example

```
$ DECW$SERVER_EXTENSIONS == "XIE,DEC-XTRAP,XINERAMA,SEC_XAG,DBE"
```

DECW$SERVER_DISABLE_TEST

This parameter controls whether test extensions, XTEST and DEC-XTRAP, are enabled. Valid values for this parameter are T (True-disable) or F (False-enable) The default value is F.

The following parameter definition enables all test extensions:

Example

```
$ DECW$SERVER_DISABLE_TEST == "F"
```

3.7.2.2 Device Setup

The XINERAMA extension (formerly known as Panoramix) is used to construct a multiheaded X Window system and have it function as a single virtual display. Use the parameters in this section to define and enable the screens in the display, control their order, and set the boundary and shape of the display.

By default, all screens in the display are enabled. You can use DECW$SERVER_ONLYSCREEN, DECW$SERVER_DISABLESCREEN to selectively remove one or more screens from the from the display. Disabled screens are not initialized and are not assigned a screen number. For instructions on how to configure a multiheaded display using extension, see Section 3.7.4. For a brief description of the XINERAMA extension, see Section 4.5.1.9.

DECW$SERVER_SCREENS

With a multiheaded system based on the XINERAMA extension, screens are initialized in alphabetical order according to their device name versus their physical position. Use this parameter to change the order in which the screens are initialized.

The following parameter definition changes the initialization order in a four-screen multiheaded display:

Example

```
$ DECW$SERVER_SCREENS == "GYB0,GYA0,GYD0,GYC0"
```

DECW$SERVER_ENABLESCREEN

With a multiheaded system based on the XINERAMA extension, you can choose to re-enable disabled screens in the display individually. This parameter enables the specified screen(s). The valid value ranges from 0 to 15, which represent the maximum number of screens supported by XINERAMA.

The following example enables the second screen (1) in a four-screen (0,1,2,3) multiheaded display:

Example

```
$ DECW$SERVER_ENABLESCREEN == "1"
```

DECW$SERVER_DISABLESCREEN

With a multiheaded system based on the XINERAMA extension, you can choose to disable each screen in the display individually. This parameter disables the specified screen. The valid value ranges from 0 to 15, which represents the maximum number of screens supported by XINERAMA.

Once a screen is disabled, it is no longer initialized as part of the display and is not assigned a screen number. Note that this changes the existing screen order and alters the display of any predefined edge attachments.

The following parameter definition disables the third screen (2) in a four-screen (0,1,2,3) multiheaded display:

Example

```
$ DECW$SERVER_DISABLESCREEN == "2"
```

DECW$SERVER_ONLYSCREEN

With a multiheaded system based on the XINERAMA extension, you can choose to enable individual screens in the display at the exclusion of all others. This parameter explicitly enables the specified screen(s) and disables all others. The valid value ranges from 0 to 15, which represents the maximum number of screens supported by XINERAMA.

The following parameter definition enables the second screen (1) and disables all other screens (0,2,3) in a four-screen (0,1,2,3) multiheaded display:

Example

```
$ DECW$SERVER_ONLYSCREEN == "1"
```

DECW$SERVER_EDGE_LEFT

With a multiheaded system based on the XINERAMA extension, edge controls are used to define the boundaries of the virtual display. This parameter determines to what screen(s) the left boundary of the display is attached. The values are determined by screen number, for example:

left-screen#, index-screen#, right-screen#

where *index-screen#* represents the number of the screen to which you want the boundary attached, *left-screen#* indicates the number of the screen directly to the left of the index, and *right-screen#* indicates the number of the screen directly to the right of the index. Repeat this pattern for each screen you to which you want the border attached. A value of -1 equates to none.

The following parameter definition specifies the left edge of a square, four-screen display arranged in the following order:

2 3
0 1

where the left edge of the second and fourth screens (indices 1 and 3) are attached to the first and third screens (0,2):

Example

```
$ DECW$SERVER_EDGE_LEFT == "-1,0,-1,2"
```

DECW$SERVER_EDGE_RIGHT

With a multiheaded system based on the XINERAMA extension, edge controls are used to define the boundaries of the virtual display. This parameter determines to what screen the right boundary of the display is attached. The values are determined by screen number, for example:

right-screen#, index-screen#, left-screen#

where *index-screen#* represents the number of the screen to which you want the boundary attached, *right-screen#* indicates the number of the screen directly to the right of the index, and *left-screen#* indicates the number of the screen directly to the left of the index. Repeat this pattern for each screen you to which you want the border attached. A value of -1 equates to none.

The following parameter definition specifies the right edge of a square, four-screen display arranged in the following order:

2 3
0 1

where the right edges of the first and third screens (indices 0 and 2) are attached to the second and fourth screens (1,3):

Example

```
$ DECW$SERVER_EDGE_RIGHT == "1,-1,3,-1"
```

DECW$SERVER_EDGE_TOP

With a multiheaded system based on the XINERAMA extension, edge controls are used to define the boundaries of the virtual display. This parameter determines to what screen the top boundary of the display is attached. The values are determined by screen number, for example:

top-screen#, index-screen#, bottom-screen#

where *index-screen#* represents the number of the screen to which you want the boundary attached, *top-screen#* indicates the number of the screen directly above the index, and *bottom-screen#* indicates the number of the screen directly below the index. Repeat this pattern for each screen you to which you want the border attached. A value of -1 equates to none.

The following parameter definition specifies the top edge of a square, four-screen display arranged in the following order:

2 3
0 1

where the top edges of the first and second screens (indices 0 and 1) are attached to the third and fourth screens (2,3):

Example

```
$ DECW$SERVER_EDGE_TOP == "2,3,-1,-1"
```

DECW$SERVER_EDGE_BOTTOM

With a multiheaded system based on the XINERAMA extension, edge controls are used to define the boundaries of the virtual display. This parameter determines to what screen the bottom boundary of the display is attached. The values are determined by screen number, for example:

bottom-screen#, index-screen#, top-screen#

where *index-screen#* represents the number of the screen to which you want the boundary attached, *bottom-screen#* indicates the number of the screen directly below the index, and *top-screen#* indicates the number of the screen directly above the index. Repeat this pattern for each screen you to which you want the border attached. A value of -1 equates to none.

The following parameter definition specifies the bottom edge of a square, four-screen display arranged in the following order:

2 3
0 1

where the bottom edges of the third and fourth screens (indices 2 and 3) are attached to the first and second screens (0,1):

Example

```
$ DECW$SERVER_EDGE_BOTTOM == "-1,-1,0,1"
```

3.7.2.3 Keyboard

The X Keyboard extension (XKB) provides enhanced capabilities for defining the keyboard layout and audio feedback. It is a standard extension and includes all features previously provided by the proprietary AccessX extension. Use the new parameters in this section when using XKB to specify the settings for the X Keyboard layout files.

See Section 3.5.2 for instructions on how to load X Keyboard layout files. For a brief description of the extension, see Section 4.5.1.

DECW$SERVER_ENABLE_ACCESSX

This parameter enables the AccessX keyboard features for disabled users, such as sticky keys or slow keys. The valid values are 0 (disabled) or 1 (enabled). The default is 0.

The following example enables the AccessX features:

Example

```
$ DECW$SERVER_ENABLE_ACCESSX == "1"
```

DECW$SERVER_XKEYBOARD_COMPILED_DIR

When using XKB, this parameter specifies the default directory for all compiled X Keyboard files. This directory is also where the server places any keymap files that it compiles on demand. The default is SYS$COMMON:[SYS$KEYMAP.XKB.COMPILED].

The following parameter definition changes the root directory to SYS$COMMON:[SYS$KEYMAP.XKB.SERVER1]:

Example

```
$ DECW$SERVER_XKEYBOARD_COMPILED_DIR == "SYS$COMMON:[SYS$KEYMAP.XKB.SERVER1]"
```

DECW$SERVER_XKEYBOARD_DIRECTORY

When using XKB, this parameter specifies the default root directory for all X Keyboard files. All component source X Keyboard files are stored in subdirectories under this root directory. The default is DECW$SYSCOMMON:[SYS$KEYMAP.XKB].

The following parameter definition changes the root directory to SYS$COMMON:[SYS$KEYMAP.XKB]:

Example

```
$ DECW$SERVER_XKEYBOARD_DIRECTORY == "SYS$COMMON:[SYS$KEYMAP.XKB]"
```

DECW$SERVER_XKEYBOARD_LOAD_MAP

When using XKB, this parameter loads the X Keyboard layout specified by DECW$SERVER_XKEYBOARD_MAP. The valid values for this parameter are 0 (disabled) or 1 (enabled). The default is 0. When this parameter is disabled, the DECwindows keyboard maps are used.

The following parameter definition loads the default X Keyboard layout file:

Example

```
$ DECW$SERVER_XKEYBOARD_LOAD_MAP == "1"
```

DECW$SERVER_XKEYBOARD_MAP

When using XKB, this parameter specifies the default compiled X Keyboard layout file for your keyboard. The valid values for this parameter are the names of any compiled layout files that currently exist in the area specified by DECW$SERVER_XKEYBOARD_COMPILED_DIR. The default is DIGITAL_US_LK201.

The following parameter definition changes the X Keyboard layout to an alternate keyboard layout:

Example

```
$ DECW$SERVER_XKEYBOARD_MAP == "DIGITAL_US_LK401"
```

DECW$SERVER_KEY_REPEAT_DELAY

When using XKB, this parameter specifies the number of milliseconds before a keystroke is first repeated. The valid values for this parameter are 0 to 1000. The default is 660.

The following parameter specifies the delay for keystroke repetition:

Example

```
$ DECW$SERVER_KEY_REPEAT_DELAY == "800"
```

DECW$SERVER_KEY_REPEAT_INTERVAL

When using XKB, this parameter specifies the number of milliseconds between repeated keystrokes. The valid values for this parameter are 0 to 1000. The default is 40.

The following parameter specifies the interval for keystroke repetition:

Example

```
$ DECW$SERVER_KEY_REPEAT_INTERVAL == "20"
```

3.7.2.4 Security

The Security extension (SECURITY), along with the MIT-MAGIC-COOKIE-1 and MIT-KERBEROS-5 protocols, provides additional means for defining which clients are authorized to connect to the X server and what operations they can perform once connected. Use the new parameters in this section to specify the location of the files used with these mechanisms (security policy, X authority, access allowed, and access trusted files).

See Section 3.3.1 for details on defining and implementing an authentication scheme for the DECwindows X11 Display Server. For a brief description of the SECURITY extension, see Section 4.5.1.6.

DECW$SECURITY_POLICY

When using SECURITY, this parameter specifies the name of the security policy file. By default, no file is specified.

The following parameter specifies the security policy file SYS$MANAGER:DECW$SECURITY_POLICY.DAT:

Example

```
$ DECW$SECURITY_POLICY == "SYS$MANAGER:DECW$SECURITY_POLICY.DAT"
```

See Section 3.3.1.5.2 for a description of the security policy file.

DECW$SERVER_XAUTHORITY

This parameter specifies the name of the server X authority file. This file provides records used to authorize client connections to the server. By default, no file is specified. This allows access to the X server from the local SYSTEM account (via DECnet or the Local transport) without requiring additional authentication from the client.

Note that the settings in the X authority file specified by DECW$SERVER_ XAUTHORITY apply to server connections made before a user logs into the DECwindows desktop. Once a user logs into the desktop, the user's X authority settings are applied.

If a file is specified, the values from this file are loaded into the server and can be used by all client connections. To allow a normal login process to occur, trusted access must be explicitly granted using the DECW$SERVER_ACCESS_ TRUSTED.DAT file.

The following parameter specifies the X authority file SYS$MANAGER:DECW$XAUTH.DAT:

Example

```
$ DECW$SERVER_XAUTHORITY == "SYS$MANAGER:DECW$XAUTH.DAT"
```

See Section 2.6.2.1 for a description of the X authority file.

DECW$SERVER_ACCESS_TRUSTED

This parameter specifies the name of the trusted access file. This file lists those clients who maintain trusted access to the server. The default file is SYS$MANAGER:DECW$SERVER_ACCESS_TRUSTED.DAT.

Note that the settings in the trusted access file specified by DECW$SERVER_ ACCESS_TRUSTED apply to server connections made before a user logs into the DECwindows desktop. Once a user logs into the desktop, the user's access settings are applied.

The following parameter changes the trusted access file specification:

Example

```
$ DECW$SERVER_ACCESS_TRUSTED == "SYS$MANAGER:DECW$SERVER1_ACCESS_TRUSTED.DAT"
```

DECW$SERVER_ACCESS_ALLOWED

This parameter specifies the name of the access allowed file. This file lists those clients who are granted automatic access to the server without requiring additional authentication. The default file is SYS$MANAGER:DECW$SERVER_ACCESS_ALLOWED.DAT.

Note that the settings in the allowed access file specified by DECW$SERVER_ACCESS_ALLOWED apply to server connections made before a user logs into the DECwindows desktop. Once a user logs into the desktop, the user's access settings are applied.

The following parameter changes the allowed access file specification:

Example

```
$ DECW$SERVER_ACCESS_ALLOWED == "SYS$MANAGER:DECW$SERVER1_ACCESS_ALLOWED.DAT"
```

3.7.2.5 Error Reporting

The following new parameter replaces the symbol DECW$SERVER_CONNECT_LOG and provides additional options for controlling the content of the X server audit logs.

DECW$SERVER_AUDIT_LEVEL

This parameter controls whether normal client connect/disconnect messages are logged in the error log file for the server. Valid values for this parameter are:

0 (disabled)
1 (enabled)
2 (enabled with success messages)
4 (enabled with security logging)

The default value is 0.

The following parameter definition enables minimal audit logging:

Example

```
$ DECW$SERVER_AUDIT_LEVEL == "1"
```

3.7.3 Enhanced Support for Dynamically Loadable Extensions

V1.3

Since some combinations of X server extensions present a function or resource conflict if enabled concurrently, two new parameters (DECW$SERVER_EXTENSIONS and DECW$SERVER_DISABLE_TEST) have been added to the server startup file. These parameters allows you to control which groups of extensions are loaded and enabled on one or more servers. Each dynamically loadable extension specified by these symbols is converted to a shareable image, which is run at server startup.

To load and enable a set of extensions, modify the parameter definitions in the DECW$PRIVATE_SERVER_SETUP.COM file, and restart the server. For example, to enable XIE and XINERAMA, add the following line to the file:

```
$ DECW$SERVER_EXTENSIONS == "XIE,XINERAMA"
```

See Section 3.7.2.1 for a detailed description of the valid values for these parameters. For the current list of unsupported combinations of X server extensions, see the *HP DECwindows Motif for OpenVMS Alpha Release Notes*.

3.7.4 Support for Multihead Systems Using XINERAMA

V1.3

The XINERAMA extension enables you to connect multiple monitors to a single Alpha system running DECwindows Motif for OpenVMS Alpha Version 1.3 or higher to create a unified virtual display. In contrast to the traditional way of configuring multiheaded Alpha systems, described in *Managing DECwindows Motif for OpenVMS Systems*, XINERAMA provides more control over the arrangement of the screens and desktop. Under a multiheaded display that uses XINERAMA, you can customize the number, order, and configuration of each screen in the display, and drag windows and text from screen to screen on the desktop.

The following sections describe how to configure a multiheaded Alpha system using XINERAMA.

3.7.4.1 Hardware and Configuration Requirements

XINERAMA is supported only in a homogeneous graphics environment. Each multiheaded configuration must consist of common video cards, bit depths, visual classes, screen resolutions, and monitors of a similar size.

See the *HP DECwindows Motif for OpenVMS Alpha Software Product Description* for a list of the currently supported video graphics cards; see *Managing DECwindows Motif for OpenVMS Systems* for a description of the logicals you can use to change the default values for these graphics settings.

The X server supports up to 16 monitors in a multiheaded configuration. Note that the actual number of monitors you can use may be further limited by the number of available option card slots.

3.7.4.2 Setting Up a Multiheaded Alpha System

Configuring a multiheaded system using XINERAMA involves the following steps:

1. Disable VGA Services
2. Install the Video Cards
3. Enable XINERAMA
4. Arrange and Configure the Monitors

The following sections describe this process.

Step 1: Disable VGA Services

Some video cards can dynamically disable or enable VGA services as necessary, but others require that you manually disable VGA via a jumper setting on the video card. Refer to the documentation for your video cards to determine if this change is required. If so, make this change prior to installing the cards in your Alpha system.

Warning

If you install multiple video cards on a system without disabling VGA services on all but one of the cards, all of the cards will compete for control of the video subsystem at boot time, resulting in possible system damage.

Step 2: Install the Video Cards

Shut down the OpenVMS Alpha system and install the video cards, as instructed by the hardware documentation.

Turn the power back on and reboot the operating system. During startup, the OpenVMS Alpha operating system will verify that the video cards were installed correctly.

Step 3: Enable XINERAMA

Although this extension is part of the X server, it is not enabled by default. To enable XINERAMA:

1. Edit the DECW$PRIVATE_SERVER_SETUP.COM file.
2. Search for and define the parameter DECW$SERVER_EXTENSIONS so that it includes a value of "XINERAMA." For example:

```
$ DECW$SERVER_EXTENSIONS == "DEC-XTRAP,XINERAMA"
```

3. Save the file and restart the server.

Step 4: Arrange the Monitors

By default, the system uses the physical location of the video cards on the system bus to assign the device names (such as, GYA0, GYB0, etc.) and subsequently number the screens. For example in a four-monitor multihead configuration, if you have connected the cables to the video cards in the proper order and placed the monitors placed side-by-side, the screens could be numbered in either ascending (0, 1, 2, 3) or descending (3, 2, 1, 0) order.

If the screens are not in the desired order, you can do one of the following depending on your screen configuration:

- Physically move the monitors to the correct placement.
- Reconnect the cables in the correct order.
- Edit the DECW$PRIVATE_SERVER_SETUP.COM file and define the DECW$SERVER_SCREENS parameter so that it overrides the default screen order.

Once the screens are in the appropriate order, you can further customize the virtual display using the following edge attachment parameters in DECW$PRIVATE_SERVER_SETUP.COM:

DECW$SERVER_EDGE_LEFT
DECW$SERVER_EDGE_RIGHT
DECW$SERVER_EDGE_TOP
DECW$SERVER_EDGE_BOTTOM

These parameters, described in Section 3.7.2.2, control where each edge of the virtual display is attached.

When the setup process is complete, all the monitors should be active and organized in the proper arrangement. Once you restart DECwindows Motif, the login dialog box for the session is displayed at the center of the virtual display, and you should be able to open application windows and drag them from screen to screen.

4
Programming Features

This chapter describes new features relating to application and system programming in the DECwindows Motif environment. This includes extensions, libraries, and functions made available as part of the X11R6.6 implementation as well as those that are specific to the DECwindows Motif environment.

For information on how to program X Window System applications, see the *X Window System* and *X Window System Toolkit* (Scheifler and Gettys) series of manuals published by Butterworth-Heinemann.

4.1 General Run-Time and Programming Environment

The following sections describe features related to general DECwindows Motif programming environment.

4.1.1 Multithreading Support

V1.3

DECwindows Motif now supports multithreaded client applications. Client applications that use the HP POSIX Threads Library or HP Ada tasks are now fully supported in the DECwindows Motif for OpenVMS Alpha Version 1.3 or higher environment.

Additionally, each of the following libraries are now fully thread-safe, supporting simultaneous calls from multiple threads:

X11 library (Xlib)
X Toolkit intrinsics library (Xt)
X Extensions library
Inter-Client Exchange (ICE) library
Session Management Protocol (XSMP) library

4.1.2 Binary Compatibility

Client applications linked against previous versions of DECwindows Motif are binary compatible, with the exception of those applications that use any changed or retired Xlib entry points documented in this manual and the *HP DECwindows Motif for OpenVMS Alpha Release Notes*.

Existing applications require recompilation against the X11R6.6-compatible X Window libraries if they produce a shared image potentially used by other multithreaded applications and that image uses:

- Any of the following macros defined in DECW$INCLUDE:XLIBINT.H and is intended for use in a multithreaded environment:

  LockDisplay
  UnlockDisplay
  LockMutex
  UnlockMutex

- The ConnectionNumber macro or XConnectionNumber function and assumes the return value is an event flag.

Note that recompiling unmodified applications (those that use the previous version of the CompositeClassExtensionRec structure) against the updated library functions may generate Xt warning messages.

4.1.2.1 Use of Asynchronous System Traps (ASTs)

In the past, DECwindows Motif supported application calls from user-mode and normal-mode AST handlers. With DECwindows Motif for OpenVMS Alpha Version 1.3 and higher, applications calls from AST handlers are only supported for existing, unmodified applications (compiled against the X11R5 libraries).

New applications compiled against the X11R6.6 libraries should not call Xlib functions (other than XtNoticeSignal) from AST handlers even in single-threaded environments.

Applications compiled against the updated libraries should either use multiple threads if a higher level of concurrency is required or call XtNoticeSignal so that AST events are processed in the Xt main loop.

4.1.2.2 Levels of Thread Safety and Concurrency

New and existing DECwindows Motif shared images can be grouped according to the following levels of thread safety:

- **Thread-safe**–The image can be called concurrently from multiple threads.
- **Thread-aware**–The image can be used in an application that includes multiple threads. However, the application code must avoid making concurrent calls to the image, typically by using a global lock.
- **Thread-unsafe**–The image cannot be used in any application that has multiple thread support enabled.

Table 4–1 shows the current thread safety level for each of the DECwindows Motif shared images. Note that all images in the Translated Image Environment are thread-unsafe.

Table 4–1 Level of Thread Safety for DECwindows Motif Images

Image	Level of Thread Safety
DECW$AILSHR.EXE	unsafe
DECW$AILSHRR5.EXE	unsafe
DECW$BKRSHR.EXE	aware
DECW$BKRSHR12.EXE	aware
DECW$D2DXLIBSHR.EXE	aware
DECW$DWTLIBSHR.EXE	aware
DECW$DXMLIBSHR.EXE	aware
DECW$DXMLIBSHR12.EXE	aware
DECW$ICELIB.EXE	safe
DECW$LCNLIBSHR.EXE	safe
DECW$MAILSHR.EXE	unsafe

(continued on next page)

Table 4–1 (Cont.) Level of Thread Safety for DECwindows Motif Images

Image	Level of Thread Safety
DECW$MAILSHR12.EXE	unsafe
DECW$MRMLIBSHR12.EXE	aware
DECW$PRINTWGTSHR.EXE	aware
DECW$SMSHR.EXE	safe
DECW$TERMINALSHR.EXE	aware
DECW$TERMINALSHR12.EXE	aware
DECW$XEXTLIBSHR.EXE	safe
DECW$XLIBSHR.EXE	safe
DECW$XMLIBSHR.EXE	aware
DECW$XMLIBSHR12.EXE	aware
DECW$XMULIBSHR.EXE	aware
DECW$XMULIBSHRR5.EXE	aware
DECW$XTLIBSHRR5.EXE	safe
DECW$XTRAPLIBSHR.EXE	aware
DECW$XTRAPLIBSHRR5.EXE	aware
DECW$XTSHR.EXE	aware

In addition to thread safety, the updated libraries offer varying levels of concurrency when called from multiple threads:

- **X and X Extension libraries**
  For most operations, Xlib and X Extension libraries allow a single concurrent operation on each display connection. If XOpenDisplay is called twice to open two separate server connections, both connections can be operated upon at the same time. However, there are a few operations for which a global lock is needed to prevent corruption of global data.

  Note that XSelectAsyncInput and XSelectAsyncEvent are not supported if multithreading has been enabled by a call to XInitThreads. In general, use of XSelectAsyncEvent and XSelectAsyncInput is discouraged since they are non-standard functions. The equivalent functionality can be obtained by using threaded Xlib functions.

- **X Toolkit Intrinsics library**
  For most operations, Xt allows a single concurrent operation on each application context. There are a few operations for which a global lock is needed to prevent corruption of global data.

- **Inter-Client Exchange and Session Manager libraries**
  ICE and XSMP operations allow a single concurrent operation on each ICE connection.

- **Transport library**
  The transport library is used to communicate between the client application and the X server and between multiple client applications that use the ICE library. The final level of the transport code (which can communicate via global sections, DECnet, or TCP/IP) executes in OpenVMS inner mode, and as a result is serialized by the operating system.

Running on only one kernel thread at a time, this code temporarily blocks all other kernel threads making OpenVMS system calls that execute in inner mode. However, whenever the transport code is blocked, such as when waiting for a connection to open or for a reply to arrive, it is blocked at user mode allowing other threads to execute and use the transport.

4.1.2.3 Enabling Support for Multithreading

To enable multithreading, a client application must include initial calls to the multithreading functions in Table 4–2. The specific functions called by the application depend on the shared image(s) in use.

Table 4–2 Multithreading Functions

Function	In Image	Enables Multithreading For
DECW$LCN_ THREAD_INIT	DECW$LCNLIBSHR	Transport Interface
IceInitThreads	DECW$ICELIB	ICE and XSMP
XInitThreads	DECW$XLIBSHR	Xlib and X Extensions
XtToolkitThreadInitialize	DECW$XTLIBSHRR5	Xt

Note that IceInitThreads and XInitThreads implicitly call DECW$LCN_ THREAD_INIT. The interface to DECW$LCN_THREAD_INIT is described in Section 4.2.2.

These functions have no arguments and return a success status upon successful initialization. To ensure successful initialization, be sure to:

- Link the image with threads.
- Verify there is adequate process memory.
- Issue the initialization call prior to making any other call.

In addition to calling one of the multithreading functions, a client application must also be linked against the POSIX Threads Library. For example:

```
$  LINK THREAD_ICO/THREADS SYS$INPUT/OPT
SYS$LIBRARY:DECW$XLIBSHR/SHARE
SYS$LIBRARY:PTHREAD$RTL/SHARE
```

Explicit links against the threads library are not required if the application calls POSIX thread functions (such as, pthread_create).

4.1.2.4 Developing Applications with Thread-Aware Images

If a thread-aware image is used in a multithreaded application, the image must not accept concurrent calls nor make calls to other images that could change the state of the thread-aware image.

Note that if a thread-aware application uses the XtAppMainLoop or XtMainLoop function for dispatch handling, calls to the image will be made from callback functions. Xt makes these callbacks with an exclusive lock held on the application context. To avoid conflicts and deadlocks, applications that use a thread-aware image should include calls to XtAppLock before and XtAppUnlock after each call, or sequence of calls, to the image. The application must also call XtProcessLock and XtProcessUnlock to protect the thread-aware image against changes made by Xt to process global data.

In the following example, an application contains a background thread that constantly checks for error situations and displays an error message when a problem occurs. The main program thread first initializes thread support, creates the application context, creates the background thread, and then enters the Xtmain loop:

```
        static XtAppContext app_context;
int main ()
{
    .
    .
    .
    MrmInitialize ();
    XInitThreads ();
    XtToolkitThreadInitialize();
    XtToolkitInitialize();
    app_context = XtCreateApplicationContext();
    .
    .
    .
    pthread_create (&thread, 0, backgroundCode, 0);
    .
    .
    .
    XtAppMainLoop(app_context)
}
```

Code for the background thread is as follows:

```
void* backgroundCode (void* data)
{
   .
   .
   .
   if (problem_detected)
   {
       XtAppLock (app_context);
       XtProcessLock();
       if (! dlog ) dlog = XmcreateWarningDialog (...);
                           XtManageChild(dlog);
                           XSync(display, 0);
                           XtProcessUnlock();
                           XtAppUnlock(app_context);
    }
    .
    .
    .
}
```

Callbacks for handling the main events of the application do not require changes for multithreading, since they are called with the application context already locked.

Worker Threads

Each of the DECwindows Motif libraries can create worker threads to support multithreading. These threads are identified by their name, which begins with the string DECW$.

Worker threads typically operate at an elevated priority to prevent task inversion, where a high-priority application thread is waiting for the worker thread to complete its operation. Note that worker threads are typically used for short duration tasks, such as responding to an internal AST or sending a status broadcast to all threads waiting for a particular activity.

Upcalls and Kernel Threads

In general, DECwindows Motif for OpenVMS Alpha Version 1.3 supports client applications either with or without upcalls or multiple kernel threads enabled. However, to avoid problems with priority inversion, HP recommends that upcalls be enabled for all applications that use XtAppAddInput. If upcalls cannot be enabled, then HP recommends assigning the same priority to all threads that use DECwindows Motif.

For example, an application calls XtAppAddInput to request a response to an OpenVMS event flag. The worker thread executes a SYS$WFLOR system call to wait for the event flag. Without upcalls enabled, this thread remains available even though there is no event flag set. And as a result, lower priority threads would not be scheduled.

Cancellation Points

Although some calls in the thread-safe libraries include cancellation points, the action of canceling threads that are executing DECwindows Motif functions is not supported. Canceled threads may hold locks, which can block other threads.

Multiple Application Contexts

Note that multiple application contexts should not be used with multiple threads and thread-aware images. Thread-aware images may contain process global data that requires a single lock to control the data. However, multiple calls to thread-aware images may be made from Xt event handling functions prior to acquiring the lock.

4.2 Transport Programming

The following sections contain features related to DECwindows Motif transport interfaces.

4.2.1 Support for the LAT Transport Interface Restored

V1.3–1

With OpenVMS Alpha Version 7.3–2, support for the DECwindows Motif interface to the LAT transport, which was withdrawn with HP DECwindows Motif for HP OpenVMS Alpha Version 1.3, has been restored. This support enables users of X terminal systems, such as the VXT2000, to start LAT X sessions to communicate with systems running DECwindows Motif Version 1.3–1 or higher. It also allows single- and multithreaded client applications running on these DECwindows Motif systems to use the LAT transport to connect to X terminal systems.

Note that the restored LAT interface included with the OpenVMS Alpha Version 7.3-2 operating system can be used as a valid network transport for communication with the DECwindows Motif Version 1.3 and OpenVMS Alpha Version 7.3-2 display servers. However, use with any other communication protocols in the X11R6.6 environment is not supported. This includes communication by or with the following:

- Inter-Client Exchange (ICE) and Session Manager protocols
- Low-Bandwidth X (LBX) proxy servers
- Proxy manager applications
- Font servers

Additionally, HP does not support the use of a token-based authentication protocol (such as MIT-MAGIC-COOKIE-1 or MIT-KERBEROS-5) with the restored LAT transport interface.

4.2.2 Support for the Logical Connection Number (LCN) Interface

V1.3

DECwindows Motif for OpenVMS Alpha Version 1.3 introduces an interface for determining when an I/O channel is ready and available for use. The logical connection number (LCN) interface is now used to signal when DECwindows Motif I/O channels are available, including those for Inter-Client Exchange (ICE), local and remote X server, and for Input Method Server connections.

Previously, DECwindows Motif used an OpenVMS event flag number (EFN) to signal when input was received from the X server. However, EFNs cannot be used safely in a multithreaded environment. The LCN interface allows multiple threads to handle the same, or different, connections without any thrashing or unnecessary delays.

The following sections further describe the functions of the LCN interface and provide detailed information about the supported routines.

4.2.2.1 LCN Functions

The principal function of the LCN interface is to test the readiness of an I/O channel. The design of the interface is based on the UNIX select function, which tests the state of UNIX file descriptors and returns when one of them is ready or a timeout occurs.

On OpenVMS, the LCN routines perform the following operations:

- Initialize support for multithreading
- Allocate a connection number
- Query the status of a connection number
- Signal when input is available

4.2.2.1.1 Initializing Thread Support LCN routines can execute in a single-threaded environment using EFNs to signal input or in a multithreaded environment using POSIX Threads routines. Multithreading is enabled with the DECW$LCN_THREAD_INIT routine.

With multithreading enabled, the select routines (DECW$LCN_SELECT_ONE and DECW$LCN_SELECT) can be called concurrently from multiple kernel threads in user mode and one kernel thread in exec mode. Calls from user mode ASTs are not allowed.

With single threading, the select routines can be called from user mode and exec mode ASTs. Note, however, that the only concurrent calls allowed are one call from user mode followed by one call from an AST in user mode.

4.2.2.1.2 Allocating Connection Numbers LCNs are allocated to a connection using the DECW$LCN_ALLOCATE routine. Values for LCNs start at 64 to distinguish them from local event flags. The maximum number of concurrently allocated LCNs equals the open file limit of the process. If the quota is 0, a default value of 1023 is used. If the quota exceeds the maximum value, a value of 2047 is used.

Once an LCN is allocated, it is unavailable for reuse until freed by the DECW$LCN_FREE routine.

4.2.2.1.3 Querying Status and Signaling Input Each LCN has three status flags, which signify whether an LCN is ready and has received input from a particular operation. Each flag can be either set (1) using the DECW$LCN_SET_*x*_READY routine or cleared (0) using the DECW$LCN_CLEAR_*x*_READY routine.

Input is signaled by setting the appropriate ready flag. The following table lists each LCN flag and describes when it is typically set and cleared.

Flag	Description
read ready	Set when there is data available to read.
write ready	Set when there is space in internal buffers to which data can be written.
except ready	Set when there is high-priority (exceptional) input.

Each flag can be set individually, and a select operation can test any combination of them. There are two routines that essentially mirror the UNIX select function, and test the ready state of an LCN. DECW$LCN_SELECT selects and tests the status of a range of LCNs or EFNs. DECW$LCN_SELECT_ONE performs the same function, however only tests the status of a single LCN.

4.2.3 LCN Routines

This section describes each of the LCN routines, which are available from the library image DECW$LCNLIBSHR.EXE. To support use from protected images that cannot use the client library, some functions are also available as part of the X Transport system services (DECW$XPORT_SERVICES).

4.2.3.1 DECW$LCN_ALLOCATE

Assigns an LCN.

Format

DECW$LCN_ALLOCATE lcn

Returns

type: longword (unsigned)
access: write
mechanism: by value

Returns a longword condition value in R0. Condition values returned by this routine are listed under Condition Values Returned.

Arguments

lcn

type: longword
access: write only
mechanism: by reference

The value of the allocated LCN.

Description

DECW$LCN_ALLOCATE assigns a logical connection number. Initially, each allocated LCN is assigned with all ready flags (read, write, and except) in the clear state (0). The state of these flags can be changed using the DECW$LCN_SET_*x*_READY routines.

Once allocated, the LCN cannot be reused until it is released by DECW$LCN_FREE. DECW$LCN_ALLOCATE must be called before any query, wait, or signaling operations can be performed.

DECW$LCN_ALLOCATE is both thread- and AST-reentrant and is callable from exec and lower modes.

The equivalent function of DECW$LCN_ALLOCATE is also available as a system service (DECW$XPORT_LCN_ALLOCATE) from the set of transport-common routines (DECW$XPORT_SERVICES.EXE).

Condition Values Returned

SS$_NORMAL	Routine successfully completed.
DECW$_INSFMEM	There is insufficient memory to perform the operation.
DECW$_NOFREELCN	All LCNs are currently allocated.

4.2.3.2 DECW$LCN_CLEAR_*x*_READY

Changes the ready bit for read, write, or except operations to the clear state.

Format

DECW$LCN_CLEAR_READ_READY lcn [, prior]

DECW$LCN_CLEAR_WRITE_READY lcn [, prior]

DECW$LCN_CLEAR_EXCEPT_READY lcn [, prior]

Returns

type: longword (unsigned)
access: write
mechanism: by value

Returns a longword condition value in R0. Condition values returned by this routine are listed under Condition Values Returned.

Arguments

lcn

type: longword
access: read only
mechanism: by value

The value of the LCN whose ready bit for read, write, and except operations will be changed to the clear state (0).

[prior]

type: longword
access: write only
mechanism: by reference

The previous state of the associated ready flag, either clear (0) or set (1).

Description

DECW$LCN_CLEAR_*x*_READY clears the read, write, or except ready bit of an LCN. This indicates that the LCN is not available for input from the specified operations.

These routines are thread- and AST-reentrant and callable from exec and lower modes.

The equivalent functions of DECW$LCN_CLEAR_x_READY are also available as system services (DECW$XPORT_LCN_CLEAR_x) from the set of transport-common routines (DECW$XPORT_SERVICES.EXE). Note that when using the system service, the **prior** argument is required; use a 0 value to prevent the prior state from being returned.

Condition Values Returned

SS$_NORMAL	Routine successfully completed.
DECW$_NOT_INITIALIZED	The LCN has not been initialized; DECW$LCN_ ALLOCATE must be called prior to this operation.
DECW$_INVLCN	The LCN has not been allocated.

4.2.3.3 DECW$LCN_FREE

Deassigns an allocated LCN.

Format

DECW$LCN_FREE lcn

Returns

type: longword (unsigned)
access: write
mechanism: by value

Returns a longword condition value in R0. Condition values returned by this routine are listed under Condition Values Returned.

Arguments

lcn

type: longword
access: read only
mechanism: by value

The value of the LCN to be freed.

Description

DECW$LCN_FREE deassigns the specified LCN. Once freed, the LCN is available for immediate reallocation.

This routine is thread- and AST-reentrant and callable from exec and lower modes.

The equivalent function of DECW$LCN_FREE is also available as a system service (DECW$XPORT_LCN_FREE) from the set of transport-common routines (DECW$XPORT_SERVICES.EXE).

Note

If either DECW$LCN_SELECT_ONE or DECW$LCN_SELECT has been called to test a state of the LCN which has been freed, then the status DECW$_INVLCN is returned from the select call.

Condition Values Returned

SS$_NORMAL	Routine successfully completed.
SS$_INSFMEM	There is insufficient memory to perform the operation.
DECW$_NOT_INITIALIZED	The LCN has not been initialized; DECW$LCN_ ALLOCATE must be called prior to this operation.
DECW$_INVLCN	The LCN has not been allocated or is protected.

4.2.3.4 DECW$LCN_SELECT

Tests the ready state(s) of one or more LCNs and returns when one of the tested states is set, a timeout occurs, or a specified OpenVMS event flag is set.

Format

DECW$LCN_SELECT retcount, rmask, wmask, emask, [timeout], [efn], [efn_mask]

Returns

type: longword (unsigned)
access: write
mechanism: by value

Returns a longword condition value in R0. Condition values returned by this routine are listed under Condition Values Returned.

Arguments

retcount

type: longword
access: write only
mechanism: by reference

The total number of entries set in the three mask structures (rmask, wmask, emask).

rmask, wmask, emask

type: mask
access: read,write
mechanism: by reference

Specifies whether to check the read (rmask), write (wmask), or exception (emask) status of one or more LCNs. A value indicates that the status check be performed; a null value indicates that no check be made. On completion, the mask is updated to reflect which LCNs have their ready state set.

The format of each mask is an array of word values. The first entry is the number of remaining entries in the array. Each subsequent entry represents an LCN value.

timeout

type: quadword
access: read only
mechanism: by reference

The time by which the select operation will timeout if no input is received. The time value is expressed in OpenVMS binary delta-time format. A null value indicates no timeout. A value of 0 indicates the operation is in polling mode and will timeout immediately if none of the specified status bits are set.

efn

type: longword
access: read only
mechanism: by value

An event flag number (EFN) in the cluster to which the **efn_mask** argument applies. EFNs are typically used for single-threaded or inner-mode operations. In this environment, **efn** identifies an event flag for the wait operation.

If no EFN value is provided in single-thread mode, SYS$HIBER and SYS$WAKE are used. In these instances, SYS$HIBER must not be used concurrently within the process. In particular, POSIX Threads must not be loaded into the image, even if not in use.

For multithreaded, user mode operations, this argument can be optional depending on whether an EFN has been provided previously to DECW$LCN_THREAD_INIT. If the EFN was specified and the value of **efn_mask** is 0, the argument is optional. Otherwise the value of this argument is required and will be used as if it had been provided to DECW$LCN_THREAD_INIT.

efn_mask

type: longword
access: read only
mechanism: by value

A mask of EFNs to be tested. Requires the **efn** argument.

Description

DECW$LCN_SELECT waits until one of the specified LCN ready states has been set, timed out, or until the event flag condition is met. This routine checks whether the selected LCNs have been allocated and returns an error (DECW$_INVLCN) if one or more LCNs have either not been allocated or freed for reuse.

With multithreading enabled, this routine is thread-reentrant and callable from exec or lower modes. Calls from ASTs are not supported.

With single threading, the select routines can be called from user mode and exec mode ASTs. Note, however, that the only concurrent calls allowed are one call from user mode followed by one call from an AST in user mode.

Condition Values Returned

SS$_NORMAL	Routine successfully completed. One or more LCNs have their ready bit set as indicated in the updated mask values.
SS$_EXQUOTA	A process quota has been exceeded, this can be due to the timer entry or AST limit quota.
SS$_INSFMEM	There is insufficient memory to perform the operation.
SS$_UNASEFC	The process is not associated with the cluster that contains the specified event flag.

DECW$_BAD_EFN_CLUSTER	An event flag was not provided to DECW$LCN_THREAD_INIT, or the specified flag resides in a different event flag cluster.
DECW$_EFN_SET	One or more event flags in the mask have been set.
DECW$_INVARG	The array count in a read, write, or exception mask, or the timeout value is not valid.
DECW$_INVLCN	One or more LCNs have not been allocated or were freed during the operation.
DECW$_NOHIBER	This call was made from inner-mode with multithreading enabled. No EFN was specified.
DECW$_PTHREAD_INVALID	A POSIX Threads routine returned an unexpected error.
DECW$_TIMEDOUT	The end of the timeout period was reached.

4.2.3.5 DECW$LCN_SELECT_ONE

Tests the ready state of an LCN and returns when one of the tested states is set, a timeout occurs, or a specified OpenVMS event flag is set.

Format

DECW$LCN_SELECT_ONE lcn, read, write, except, [timeout], [efn], [efn_mask]

Returns

type: longword (unsigned)
access: write
mechanism: by value

Returns a longword condition value in R0. Condition values returned by this routine are listed under Condition Values Returned.

Arguments

lcn

type: longword
access: read only
mechanism: by value

The value of the LCN.

read, write, except

type: longword
access: write only
mechanism: by reference

Specifies whether to check the read, write, or exception status of an LCN. A non-zero value signifies that the status check be performed and the result stored in the referenced address on completion. A zero or null value indicates that no check be made.

timeout

type: quadword
access: read only
mechanism: by reference

The time by which the select operation will timeout if no input is received. The time value is expressed in OpenVMS binary delta-time format. A null value indicates no timeout. A value of 0 indicates the operation is in polling mode and will timeout immediately if none of the specified status bits are set.

efn

type: longword
access: read only
mechanism: by value

An event flag number (EFN) in the cluster to which the **efn_mask** argument applies. EFNs are typically used for single-threaded or inner-mode operations. In this environment, **efn** identifies an event flag for the wait operation.

If no EFN value is provided in single-thread mode, SYS$HIBER and SYS$WAKE are used. In these instances, SYS$HIBER must not be used concurrently within the process. In particular, POSIX Threads must not be loaded into the image, even if not in use.

For multithreaded, user mode operations, this argument can be optional depending on whether an EFN has been provided previously to DECW$LCN_THREAD_INIT. If the EFN was specified and the value of **efn_mask** is 0, the argument is optional. Otherwise the value of this argument is required and will be used as if it had been provided to DECW$LCN_THREAD_INIT.

efn_mask

type: longword
access: read only
mechanism: by value

A mask of EFNs to be tested. Requires the **efn** argument.

Description

DECW$LCN_SELECT_ONE waits until one of the specified LCN ready states has been set, timed out, or until the event flag condition is met. This routine checks whether the selected LCN has been allocated and returns an error (DECW$_INVLCN) if the LCN has not been allocated or freed for reuse.

With multithreading enabled, this routine is thread-reentrant and callable from exec or lower modes. Calls from ASTs are not supported.

With single threading, the select routines can be called from user mode and exec mode ASTs. Note, however, that the only concurrent calls allowed are one call from user mode followed by one call from an AST in user mode.

Condition Values Returned

SS$_NORMAL	Routine successfully completed. One or more LCNs have their ready bit set as indicated in the updated mask values.
SS$_EXQUOTA	A process quota has been exceeded, this can be due to the timer entry or AST limit quota.
SS$_INSFMEM	There is insufficient memory to perform the operation.
SS$_UNASEFC	The process is not associated with the cluster that contains the specified event flag.
DECW$_BAD_EFN_CLUSTER	An event flag was not provided to DECW$LCN_THREAD_INIT, or the specified flag resides in a different event flag cluster.
DECW$_INVARG	The timeout period or EFN is not accessible or within the range of valid values.

DECW$_INVLCN	The LCN has not been allocated or was freed during the operation.
DECW$_NOHIBER	This call was made from inner-mode with multithreading enabled. No EFN was specified.
DECW$_PTHREAD_INVALID	A POSIX Threads routine returned an unexpected error.
DECW$_TIMEDOUT	The end of the timeout period was reached.

4.2.3.6 DECW$LCN_SET_*x*_READY

Signals that input from a read, write, or exception operation has been received by changing the ready bit to the set state.

Format

DECW$LCN_SET_READ_READY lcn

DECW$LCN_SET_WRITE_READY lcn

DECW$LCN_SET_EXCEPT_READY lcn

Returns

type: longword (unsigned)
access: write
mechanism: by value

Returns a longword condition value in R0. Condition values returned by this routine are listed under Condition Values Returned.

Arguments

lcn

type: longword
access: read only
mechanism: by value

The value of the LCN whose ready bit will be changed to the set state (1).

Description

DECW$LCN_SET_*x*_READY sets the read, write, or except ready bit of an LCN. This signals input has been received from the specified operations.

These routines are thread- and AST-reentrant and callable from exec and lower modes. When any of the DECW$LCN_SET_*x*_READY routines are called from exec mode with multithreading enabled, the call declares a user mode AST. This AST performs the signaling for any user-mode processes that are currently in a wait state.

The equivalent functions of DECW$LCN_SET_*x*_READY are also available as system services (DECW$XPORT_LCN_SET_*x*) from the set of transport-common routines (DECW$XPORT_SERVICES.EXE).

Condition Values Returned

SS$_NORMAL	Routine successfully completed.
SS$_EXQUOTA	The current memory quota has been exceeded.
SS$_INSFMEM	There is insufficient memory to perform the operation.
DECW$_INVLCN	The LCN has not been allocated.

DECW$_NOT_INITIALIZED	The LCN has not been initialized; DECW$LCN_ALLOCATE must be called prior to this operation.

4.2.3.7 DECW$LCN_THREAD_INIT

Initializes multithreading support for LCN operations.

Format

DECW$LCN_THREAD_INIT [efn]

Returns

type: longword (unsigned)
access: write
mechanism: by value

Returns a longword condition value in R0. Condition values returned by this routine are listed under Condition Values Returned.

Arguments

efn

type: longword
access: read only
mechanism: by value

The value of the specified EFN. When multithreading is enabled, an EFN value must be specified when performing select operations (DECW$LCN_SELECT or DECW$LCN_SELECT_ONE) with event flag masks. The EFN is supplied either by DECW$LCN_THREAD_INIT or from the first select call that provides an **efn** argument.

The value of the **efn** argument must match the value supplied for any previous select operations or calls to DECW$LCN_THREAD_INIT. The event flag must also be in the same event flag cluster as the efn value supplied to subsequent select or initialization operations.

Description

DECW$LCN_THREAD_INIT enables multithreaded LCN operations. This routine is only callable from user mode with user mode ASTs enabled and can be called multiple times by a single process.

Condition Values Returned

SS$_NORMAL	Routine successfully completed.
DECW$_CHANGED_EFN	An event flag was specified that differs from that specified in a previous initialization or select call.
DECW$_INSFMEM	There is insufficient memory to perform the operation.
DECW$_NOPTHREADRTL	The POSIX Thread Library (PTHREAD$RTL) has not been loaded.
DECW$_NOTUSERMODE	This routine was not called from user mode.
DECW$_PTHREAD_INVALID	A POSIX Thread routine returned an unexpected error.

4.3 X Window System Library (Xlib)

The following sections describe features related to X Window System library (Xlib).

4.3.1 New Functions Available with X11R6.6 Upgrade

V1.3

As part of the core system upgrade to X11R6.6, the following new functions listed in Table 4–3 have been added to the version of Xlib available with DECwindows Motif for OpenVMS Alpha Version 1.3. Note that all functions are included in the current version of the DECW$XLIBSHR image.

Table 4–3 New Xlib Functions Supported for X11R6.6

Function Name	Description
_XAllocTemp	Thread-safe allocation of scratch data space for use by extension writers.
_XFreeTemp	Frees the scratch data space allocated by _XAllocTemp.
XCloseOM	Closes the specified output method.
XcmsSetCCCOfColormap	Sets the color conversion context for the specified colormap.
XAddConnectionWatch	Establishes a watch procedure callback for when internal connections are opened or closed.
XConvertCase	Obtains the uppercase and lowercase forms of a KeySym.
XContextualDrawing	Indicates whether text drawn with the current font set includes context-dependent drawing.
XCreateOC	Creates an output context within the specified output method.
XDestroyOC	Destroys an output context.
XDirectionalDependentDrawing	Indicates whether the drawing functions implement text directionality.
XDisplayOfOM	Returns the display associated with the specified output method.
XESetBeforeFlush	Defines a procedure that will be called just before data is sent to the X server.
XExtendedMaxRequestSize	Returns the maximum request size using extended length encoding (the BIG-REQUESTS extension).
XGetAtomNames	Returns the names associated with the specified X atoms.
XGetOCValues	Obtains the current output context values.
XGetOMValues	Obtains the current output method values.
XInitThreads	Initializes support for multiple threads.
XInternalConnectionNumbers	Returns a list of the internal connections open for a specified display.
XInternAtoms	Returns atoms for an array of names.

(continued on next page)

Table 4–3 (Cont.) New Xlib Functions Supported for X11R6.6

Function Name	Description
XLocaleOfOM	Returns the locale associated with the specified output method.
XLockDisplay	Locks a display to protect against concurrent access from multiple threads.
XOMOfOC	Returns the output method associated with the specified output context.
XOpenOM	Opens an X output method for the specified locale and modifiers.
XProcessInternalConnection	Processes input available on an internal connection.
XReadBitmapFileData	Reads a bitmap from a file and returns it as data.
XRegisterIMInstantiateCallback	Registers an input method callback.
XRemoveConnectionWatch	Removes a watch procedure established by XAddConnectionWatch.
XSetOCValues	Specifies one or more output context values.
XSetOMValues	Specifies one or more output method values.
XUnlockDisplay	Removes a lock established by XLockDisplay.
XUnregisterIMInstantiateCallback	Unregisters an input method instantiation callback.

See the *Xlib - C Language X Interface, X Consortium Standard, X Version 11, Release 6.4* specification available from X.Org for detailed information about each of these functions.

4.3.2 Updated Client-Side Extension Library

V1.3

The client-side extension library (DECW$XEXTLIBSHR) has been updated to support multithreading and new header files for the following new extensions available as part of the upgrade to X11R6.6:

- Application Group (XC-APPGROUP)
- Colormap Utilization Policy (TOG-CUP)
- Extended Visual Information (EVI)
- Low-Bandwidth X (LBX)
- Security (SECURITY)
- Synchronization (SYNC)
- X Double Buffer (DBE)
- XINERAMA (formerly PanoramiX)
- X Print (Xp)

All extensions in the library (new and existing) have been made thread-safe (as described in Section 4.1.1). In addition, the minor version of the library has been updated from 2,2 to 2,3 to prevent images linked against the updated DECW$XEXTLIBSHR from loading the incorrect version of the library.

Also, function names longer than 31 characters have been replaced by macro definitions compatible with the current version of the OpenVMS Linker.

See Section 4.5.1 for an overview of each of these extensions. For instructions on how to link to this library, see Section 4.3.8.

4.3.3 Support for LCNs

V1.3

Xlib now provides an alternate means of obtaining connection numbers for connections to DECwindows Motif for OpenVMS Alpha Version 1.3 or higher servers. The logical connection number (LCN) interface was specifically designed to support the communication needs of X11R6 systems and is intended as a replacement for the EFN mechanism.

The following functions and macros are designed for use with the new LCN interface:

- XAddConnectionWatch function (registers watch procedure)
- XInternalConnectionNumbers function
- XConnectionNumber function
- ConnectionNumber macro

These Xlib functions and macros are described in the *Xlib - C Language X Interface, X Consortium Standard, X Version 11, Release 6.4* specification available from X.Org.

Note for compatibility with DECwindows Motif for OpenVMS Version 1.2–6 and earlier clients, the existing event flag mechanism remains unchanged, and the XtAppAddInput function accepts both EFNs and LCNs. However, HP recommends that new applications, in particular ones that use multithreading, use LCNs. When Xlib has multithreading enabled, EFNs are not available. If a multithreaded application uses EFNs without multithreading enabled in Xlib, the EFN should be restricted to a single thread–the same thread used for all X calls.

For more information about the LCN interface and its available routines, see Section 4.2.2.

4.3.4 Updated X11 Environment Variable Parsing

V1.3

Xlib now accepts the equivalent X11 Release 6 (X11R6) POSIX-compliant forms of the following environment variables:

OpenVMS Form	X11R6 Form
DECW$DISPLAY	DISPLAY
DECW$RESOURCE_NAME	RESOURCE_NAME[1]

[1]Also requires the symbol DECW$VSW_COMPLIANT.

On connection to the X display server, Xlib checks the variable name. If the OpenVMS variable is not defined, Xlib checks for the X11R6 equivalent before returning a status value.

4.3.5 Additional Non-C Language Bindings Available with X11R6.6

Non-C language bindings (such as Fortran and Pascal) for the following new Xlib functions have been added to DECwindows Motif for OpenVMS Alpha Version 1.3. These bindings are in addition to those documented in the *DECwindows Motif for OpenVMS Guide to Non-C Bindings*.

X$CLOSE_OM
X$CONTEXTUAL_DRAWING
X$CONVERT_CASE
X$DESTROY_OC
X$DIRECTIONAL_DEPENDENT_DRAWING
X$DISPLAY_OF_OM
X$EXTENDED_MAX_REQUEST_SIZE
X$INIT_IMAGE
X$INIT_THREADS
X$INTERNAL_CONNECTION_NUMBERS
X$LOCALE_OF_OM
X$LOCK_DISPLAY
X$OPEN_OM
X$PROCESS_INTERNAL_CONNECTION
X$REGISTER_IM_INSTANTIATE_CB
X$SET_AUTHORIZATION
X$UNLOCK_DISPLAY
X$UNREGISTER_IM_INSTANTIATE_CB

4.3.5.1 CLOSE OM

OpenVMS Format

status_return = X$CLOSE_OM

(om)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
status_return	longword	longword	write	value
om	identifier	uns longword	read	reference

4.3.5.2 CONTEXTUAL DRAWING

OpenVMS Format

status_return = X$CONTEXTUAL_DRAWING

(font_set)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
status_return	cond_value	longword	write	value
font_set	identifier	uns longword	read	reference

4.3.5.3 CONVERT CASE

OpenVMS Format

X$CONVERT_CASE

(sym, lower, upper)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
sym	uns longword	uns longword	read	reference
lower	uns longword	uns longword	write	reference
upper	uns longword	uns longword	write	reference

4.3.5.4 DESTROY OC

OpenVMS Format

X$DESTROY_OC

(oc)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
oc	uns longword	uns longword	write	reference

4.3.5.5 DIRECTIONAL DEPENDENT DRAWING

OpenVMS Format

status_return = X$DIRECTIONAL_DEPENDENT_DRAWING

(font_set)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
status_return	cond_value	longword	write	value
font_set	identifier	uns longword	read	reference

4.3.5.6 DISPLAY OF OM

OpenVMS Format

display_return = X$DISPLAY_OF_OM

(om)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
display_return	identifier	uns longword	write	value
om	uns longword	uns longword	read	reference

4.3.5.7 EXTENDED MAX REQUEST SIZE

OpenVMS Format

req_size_return = X$EXTENDED_MAX_REQUEST_SIZE

(display)

Argument Information

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

4.3.5.8 INIT IMAGE

OpenVMS Format

status_return = X$INIT_IMAGE

(ximage)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
status_return	cond_value	longword	write	value
ximage	record	x$image	read	reference

4.3.5.9 INIT THREADS

OpenVMS Format

status_return = X$INIT_THREADS

Argument Information

Argument	Usage	Data Type	Access	Mechanism
status_return	cond_value	longword	write	value

4.3.5.10 INTERNAL CONNECTION NUMBERS

OpenVMS Format

status_return = X$INTERNAL_CONNECTION_NUMBERS

(display, fdptr, nptr)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
status_return	cond_value	longword	write	value
display	identifier	uns longword	read	reference
fdptr	longword	longword	write	reference
nptr	longword	longword	write	reference

4.3.5.11 LOCALE OF OM

OpenVMS Format

return_value = X$LOCALE_OF_OM

(om, return_string)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
return_value	longword	longword	write	value
om	uns longword	uns longword	read	reference
return_string	char_string	character string	write	descriptor

4.3.5.12 LOCK DISPLAY

OpenVMS Format

X$LOCK_DISPLAY

(display)

Argument Information

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

4.3.5.13 OPEN OM

OpenVMS Format

om_return = X$OPEN_OM

(display, db, str1, str2)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
om_return	uns longword	uns longword	write	reference
display	identifier	uns longword	read	reference
db	identifier	uns longword	read	reference
str1	char_string	character string	read	descriptor
str2	char_string	character string	read	descriptor

4.3.5.14 PROCESS INTERNAL CONNECTION

OpenVMS Format

X$PROCESS_INTERNAL_CONNECTION

(display, fdptr)

Argument Information

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

4.3.5.15 REGISTER IM INSTANTIATE Callback

OpenVMS Format

return_value = X$REGISTER_IM_INSTANTIATE_CB

(display, database, name_str, class_str, callback, client_data)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
return_value	longword	longword	write	value
display	identifier	uns longword	read	reference
database	identifier	uns longword	read	reference
name_str	char_string	character string	read	descriptor
class_str	char_string	character string	read	descriptor
callback	procedure	proc entry mask	read	value
client_data	char_string	character string	read	descriptor

4.3.5.16 SET AUTHORIZATION

OpenVMS Format

X$SET_AUTHORIZATION

(name, data)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
name	char_string	character string	read	descriptor
data	char_string	character string	read	descriptor

4.3.5.17 UNLOCK DISPLAY

OpenVMS Format

X$UNLOCK_DISPLAY

(display)

Argument Information

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

4.3.5.18 UNREGISTER IM INSTANTIATE Callback

OpenVMS Format

return_value = X$UNREGISTER_IM_INSTANTIATE_CB

(display, database, name_str, class_str, callback, client_data)

Argument Information

Argument	Usage	Data Type	Access	Mechanism
return_value	longword	longword	write	value
display	identifier	uns longword	read	reference
database	identifier	uns longword	read	reference
name_str	char_string	character string	read	descriptor
class_str	char_string	character string	read	descriptor
callback	procedure	proc entry mask	read	value
client_data	char_string	character string	read	descriptor

4.3.6 Support for Additional Fonts

V1.3

DECwindows Motif for OpenVMS Alpha Version 1.3 offers support for the following additional fonts and font technologies:

- **Agfa Monotype Windows-Compatible TrueType fonts** – To ensure fast, high-quality text rendering capabilities, DECwindows Motif for OpenVMS Alpha Version 1.3 includes the iType font rendering technology from Agfa Monotype Corporation along with a number of scalable fonts, including the Albany, Cumberland, Screen, and Thorndale type families. These fonts are identical in screen and printer metrics to the Windows core fonts Arial, Courier, and Times New Roman. Agfa Monotype's Windows-compatible fonts are part of the Enhanced Screen Quality (ESQ) line of TrueType fonts optimized for viewing at any resolution.

  For more information about the iType technology, visit the Agfa Monotype web site (http://www.agfamonotype.com).

- **X11R6.6 fonts** – As part of the standard X11R6.6 implementation, DECwindows Motif for OpenVMS Alpha Version 1.3 includes the 75- and 100-dpi versions of the Bitstream Charter and Adobe Utopia font families.

- **Previously undocumented fonts** – These fonts include the 75-dpi, 100-dpi, and common versions of the Lucida, Present Bullets, Fixed Width, Sun Open Look Glyph, and VT330 font families, and well as a set of language-specific and miscellaneous fonts. Also included are the Bitstream Speedo, Adobe Type1 Courier, and Utopia scalable fonts.

The following sections further describe the new font families, which can be loaded as described in Chapter 8 of the *VMS DECwindows Guide to Xlib (Release 4) Programming: MIT C Binding* manual. Each section provides the following information:

- Location and format of the font files
- List of the individual font file and font names in X Logical Font Description (XLFD) format

The file and font names are provided in a series of tables that are intended to supplement the existing font tables found in Appendix C of the *VMS DECwindows Guide to Xlib (Release 4) Programming: MIT C Binding* manual.

Note

File names containing consecutive underscore characters (_) or hyphens (-) may appear to contain a space between the consecutive characters. In all cases, the space is not present in the font name.

4.3.6.1 Additional 75-dpi Fonts

Table 4–4 lists the new and previously undocumented 75-dpi fonts and their file names. The files for these fonts are located in the following directory:

DECW$SYSCOMMON:[SYSFONT.DECW.75DPI]

Table 4–4 Additional 75-dpi Fonts (.PCF File Extension)

File Name	Font Name
Charter	
CHARTER08	-Bitstream-Charter-Medium-R-Normal- -8-80-75-75-P-45-ISO8859-1
CHARTER10	-Bitstream-Charter-Medium-R-Normal- -10-100-75-75-P-56-ISO8859-1
CHARTER12	-Bitstream-Charter-Medium-R-Normal- -12-120-75-75-P-67-ISO8859-1
CHARTER14	-Bitstream-Charter-Medium-R-Normal- -15-140-75-75-P-84-ISO8859-1
CHARTER18	-Bitstream-Charter-Medium-R-Normal- -19-180-75-75-P-106-ISO8859-1
CHARTER24	-Bitstream-Charter-Medium-R-Normal- -25-240-75-75-P-139-ISO8859-1
CHARTER_BOLD_ITALIC08	-Bitstream-Charter-Bold-I-Normal- -8-80-75-75-P-50-ISO8859-1
CHARTER_BOLD_ITALIC10	-Bitstream-Charter-Bold-I-Normal- -10-100-75-75-P-62-ISO8859-1
CHARTER_BOLD_ITALIC14	-Bitstream-Charter-Bold-I-Normal- -15-140-75-75-P-93-ISO8859-1
CHARTER_BOLD_ITALIC12	-Bitstream-Charter-Bold-I-Normal- -12-120-75-75-P-74-ISO8859-1
CHARTER_BOLD_ITALIC18	-Bitstream-Charter-Bold-I-Normal- -19-180-75-75-P-117-ISO8859-1
CHARTER_BOLD_ITALIC24	-Bitstream-Charter-Bold-I-Normal- -25-240-75-75-P-154-ISO8859-1
CHARTER_ITALIC08	-Bitstream-Charter-Medium-I-Normal- -8-80-75-75-P-44-ISO8859-1
CHARTER_ITALIC10	-Bitstream-Charter-Medium-I-Normal- -10-100-75-75-P-55-ISO8859-1
CHARTER_ITALIC12	-Bitstream-Charter-Medium-I-Normal- -12-120-75-75-P-65-ISO8859-1
CHARTER_ITALIC14	-Bitstream-Charter-Medium-I-Normal- -15-140-75-75-P-82-ISO8859-1
CHARTER_ITALIC18	-Bitstream-Charter-Medium-I-Normal- -19-180-75-75-P-103-ISO8859-1
CHARTER_ITALIC24	-Bitstream-Charter-Medium-I-Normal- -25-240-75-75-P-136-ISO8859-1
CHARTER_BOLD08	-Bitstream-Charter-Bold-R-Normal- -8-80-75-75-P-50-ISO8859-1
CHARTER_BOLD10	-Bitstream-Charter-Bold-R-Normal- -10-100-75-75-P-63-ISO8859-1
CHARTER_BOLD12	-Bitstream-Charter-Bold-R-Normal- -12-120-75-75-P-75-ISO8859-1

(continued on next page)

Table 4–4 (Cont.) Additional 75-dpi Fonts (.PCF File Extension)

File Name	Font Name
Charter	
CHARTER_BOLD14	-Bitstream-Charter-Bold-R-Normal- -15-140-75-75-P-94-ISO8859-1
CHARTER_BOLD18	-Bitstream-Charter-Bold-R-Normal- -19-180-75-75-P-119-ISO8859-1
CHARTER_BOLD24	-Bitstream-Charter-Bold-R-Normal- -25-240-75-75-P-157-ISO8859-1
Lucida	
LUCIDABRIGHT08	-B&H-LucidaBright-Medium-R-Normal- -8-80-75-75-P-45-ISO8859-1
LUCIDABRIGHT10	-B&H-LucidaBright-Medium-R-Normal- -10-100-75-75-P-56-ISO8859-1
LUCIDABRIGHT12	-B&H-LucidaBright-Medium-R-Normal- -12-120-75-75-P-68-ISO8859-1
LUCIDABRIGHT14	-B&H-LucidaBright-Medium-R-Normal- -14-140-75-75-P-80-ISO8859-1
LUCIDABRIGHT18	-B&H-LucidaBright-Medium-R-Normal- -18-180-75-75-P-103-ISO8859-1
LUCIDABRIGHT19	-B&H-LucidaBright-Medium-R-Normal- -19-190-75-75-P-109-ISO8859-1
LUCIDABRIGHT24	-B&H-LucidaBright-Medium-R-Normal- -24-240-75-75-P-137-ISO8859-1
LUCIDABRIGHT_ DEMIITALIC08	-B&H-LucidaBright-Demibold-I-Normal- -8-80-75-75-P-48-ISO8859-1
LUCIDABRIGHT_ DEMIITALIC10	-B&H-LucidaBright-Demibold-I-Normal- -10-100-75-75-P-59-ISO8859-1
LUCIDABRIGHT_ DEMIITALIC12	-B&H-LucidaBright-Demibold-I-Normal- -12-120-75-75-P-72-ISO8859-1
LUCIDABRIGHT_ DEMIITALIC14	-B&H-LucidaBright-Demibold-I-Normal- -14-140-75-75-P-84-ISO8859-1
LUCIDABRIGHT_ DEMIITALIC18	-B&H-LucidaBright-Demibold-I-Normal- -18-180-75-75-P-107-ISO8859-1
LUCIDABRIGHT_ DEMIITALIC19	-B&H-LucidaBright-Demibold-I-Normal- -19-190-75-75-P-114-ISO8859-1
LUCIDABRIGHT_ DEMIITALIC24	-B&H-LucidaBright-Demibold-I-Normal- -24-240-75-75-P-143-ISO8859-1
LUCIDABRIGHT_ITALIC08	-B&H-LucidaBright-Medium-I-Normal- -8-80-75-75-P-45-ISO8859-1
LUCIDABRIGHT_ITALIC10	-B&H-LucidaBright-Medium-I-Normal- -10-100-75-75-P-57-ISO8859-1
LUCIDABRIGHT_ITALIC12	-B&H-LucidaBright-Medium-I-Normal- -12-120-75-75-P-67-ISO8859-1
LUCIDABRIGHT_ITALIC14	-B&H-LucidaBright-Medium-I-Normal- -14-140-75-75-P-80-ISO8859-1
LUCIDABRIGHT_ITALIC18	-B&H-LucidaBright-Medium-I-Normal- -18-180-75-75-P-102-ISO8859-1
LUCIDABRIGHT_ITALIC19	-B&H-LucidaBright-Medium-I-Normal- -19-190-75-75-P-109-ISO8859-1
LUCIDABRIGHT_ITALIC24	-B&H-LucidaBright-Medium-I-Normal- -24-240-75-75-P-136-ISO8859-1
LUCIDABRIGHT_DEMI08	-B&H-LucidaBright-Demibold-R-Normal- -8-80-75-75-P-47-ISO8859-1
LUCIDABRIGHT_DEMI10	-B&H-LucidaBright-Demibold-R-Normal- -10-100-75-75-P-59-ISO8859-1
LUCIDABRIGHT_DEMI12	-B&H-LucidaBright-Demibold-R-Normal- -12-120-75-75-P-71-ISO8859-1
LUCIDABRIGHT_DEMI14	-B&H-LucidaBright-Demibold-R-Normal- -14-140-75-75-P-84-ISO8859-1
LUCIDABRIGHT_DEMI18	-B&H-LucidaBright-Demibold-R-Normal- -18-180-75-75-P-107-ISO8859-1
LUCIDABRIGHT_DEMI19	-B&H-LucidaBright-Demibold-R-Normal- -19-190-75-75-P-114-ISO8859-1

(continued on next page)

Table 4–4 (Cont.) Additional 75-dpi Fonts (.PCF File Extension)

File Name	Font Name
Lucida	
LUCIDABRIGHT_DEMI24	-B&H-LucidaBright-Demibold-R-Normal- -24-240-75-75-P-143-ISO8859-1
LUCIDATYPEWRITER_ BOLDSANS08	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-8-80-75-75-m-50-ISO8859-1
LUCIDATYPEWRITER_ BOLDSANS10	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-10-100-75-75-m-60-ISO8859-1
LUCIDATYPEWRITER_ BOLDSANS12	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-12-120-75-75-m-70-ISO8859-1
LUCIDATYPEWRITER_ BOLDSANS14	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-14-140-75-75-m-90-ISO8859-1
LUCIDATYPEWRITER_ BOLDSANS18	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-18-180-75-75-m-110-ISO8859-1
LUCIDATYPEWRITER_ BOLDSANS19	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-19-190-75-75-m-110-ISO8859-1
LUCIDATYPEWRITER_ BOLDSANS24	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-24-240-75-75-m-140-ISO8859-1
LUCIDATYPEWRITER_SANS08	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-8-80-75-75-m-50-ISO8859-1
LUCIDATYPEWRITER_SANS10	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-10-100-75-75-m-60-ISO8859-1
LUCIDATYPEWRITER_SANS12	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-12-120-75-75-m-70-ISO8859-1
LUCIDATYPEWRITER_SANS14	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-14-140-75-75-m-90-ISO8859-1
LUCIDATYPEWRITER_SANS18	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-18-180-75-75-m-110-ISO8859-1
LUCIDATYPEWRITER_SANS19	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-19-190-75-75-m-110-ISO8859-1
LUCIDATYPEWRITER_SANS24	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-24-240-75-75-m-140-ISO8859-1
LUCIDA_BOLDITALICSANS08	-B&H-Lucida-Bold-I-Normal-Sans-8-80-75-75-P-49-ISO8859-1
LUCIDA_BOLDITALICSANS10	-B&H-Lucida-Bold-I-Normal-Sans-10-100-75-75-P-67-ISO8859-1
LUCIDA_BOLDITALICSANS12	-B&H-Lucida-Bold-I-Normal-Sans-12-120-75-75-P-79-ISO8859-1
LUCIDA_BOLDITALICSANS14	-B&H-Lucida-Bold-I-Normal-Sans-14-140-75-75-P-92-ISO8859-1
LUCIDA_BOLDITALICSANS18	-B&H-Lucida-Bold-I-Normal-Sans-18-180-75-75-P-119-ISO8859-1
LUCIDA_BOLDITALICSANS19	-B&H-Lucida-Bold-I-Normal-Sans-19-190-75-75-P-122-ISO8859-1
LUCIDA_BOLDITALICSANS24	-B&H-Lucida-Bold-I-Normal-Sans-24-240-75-75-P-151-ISO8859-1
LUCIDA_BOLDSANS08	-B&H-Lucida-Bold-R-Normal-Sans-8-80-75-75-P-50-ISO8859-1
LUCIDA_BOLDSANS10	-B&H-Lucida-Bold-R-Normal-Sans-10-100-75-75-P-66-ISO8859-1
LUCIDA_BOLDSANS12	-B&H-Lucida-Bold-R-Normal-Sans-12-120-75-75-P-79-ISO8859-1
LUCIDA_BOLDSANS14	-B&H-Lucida-Bold-R-Normal-Sans-14-140-75-75-P-92-ISO8859-1
LUCIDA_BOLDSANS18	-B&H-Lucida-Bold-R-Normal-Sans-18-180-75-75-P-120-ISO8859-1
LUCIDA_BOLDSANS19	-B&H-Lucida-Bold-R-Normal-Sans-19-190-75-75-P-122-ISO8859-1
LUCIDA_BOLDSANS24	-B&H-Lucida-Bold-R-Normal-Sans-24-240-75-75-P-152-ISO8859-1
LUCIDA_ITALICSANS08	-B&H-Lucida-Medium-I-Normal-Sans-8-80-75-75-P-45-ISO8859-1
LUCIDA_ITALICSANS10	-B&H-Lucida-Medium-I-Normal-Sans-10-100-75-75-P-59-ISO8859-1
LUCIDA_ITALICSANS12	-B&H-Lucida-Medium-I-Normal-Sans-12-120-75-75-P-71-ISO8859-1

(continued on next page)

Table 4–4 (Cont.) Additional 75-dpi Fonts (.PCF File Extension)

File Name	Font Name
Lucida	
LUCIDA_ITALICSANS14	-B&H-Lucida-Medium-I-Normal-Sans-14-140-75-75-P-82-ISO8859-1
LUCIDA_ITALICSANS18	-B&H-Lucida-Medium-I-Normal-Sans-18-180-75-75-P-105-ISO8859-1
LUCIDA_ITALICSANS19	-B&H-Lucida-Medium-I-Normal-Sans-19-190-75-75-P-108-ISO8859-1
LUCIDA_ITALICSANS24	-B&H-Lucida-Medium-I-Normal-Sans-24-240-75-75-P-136-ISO8859-1
LUCIDA_SANS08	-B&H-Lucida-Medium-R-Normal-Sans-8-80-75-75-P-45-ISO8859-1
LUCIDA_SANS10	-B&H-Lucida-Medium-R-Normal-Sans-10-100-75-75-P-58-ISO8859-1
LUCIDA_SANS12	-B&H-Lucida-Medium-R-Normal-Sans-12-120-75-75-P-71-ISO8859-1
LUCIDA_SANS14	-B&H-Lucida-Medium-R-Normal-Sans-14-140-75-75-P-81-ISO8859-1
LUCIDA_SANS18	-B&H-Lucida-Medium-R-Normal-Sans-18-180-75-75-P-106-ISO8859-1
LUCIDA_SANS19	-B&H-Lucida-Medium-R-Normal-Sans-19-190-75-75-P-108-ISO8859-1
LUCIDA_SANS24	-B&H-Lucida-Medium-R-Normal-Sans-24-240-75-75-P-136-ISO8859-1
Present Bullets	
PRESENT_BULLETS8_75	-DEC-PresentBullets-Medium-R-Normal- -8-80-75-75-P-76-DEC-FontSpecific
PRESENT_BULLETS10_75	-DEC-PresentBullets-Medium-R-Normal- -10-100-75-75-P-96-DEC-FontSpecific
PRESENT_BULLETS12_75	-DEC-PresentBullets-Medium-R-Normal- -12-120-75-75-P-114-DEC-FontSpecific
PRESENT_BULLETS14_75	-DEC-PresentBullets-Medium-R-Normal- -14-140-75-75-P-134-DEC-FontSpecific
PRESENT_BULLETS18_75	-DEC-PresentBullets-Medium-R-Normal- -18-180-75-75-P-172-DEC-FontSpecific
PRESENT_BULLETS24_75	-DEC-PresentBullets-Medium-R-Normal- -24-240-75-75-P-229-DEC-FontSpecific
PRESENT_BULLETS36_75	-DEC-PresentBullets-Medium-R-Normal- -36-360-75-75-P-343-DEC-FontSpecific
PRESENT_BULLETS48_75	-DEC-PresentBullets-Medium-R-Normal- -48-480-75-75-P-458-DEC-FontSpecific
PRESENT_BULLETS72_75	-DEC-PresentBullets-Medium-R-Normal- -72-720-75-75-P-686-DEC-FontSpecific
Utopia	
UTOPIA10	-Adobe-Utopia-Regular-R-Normal- -10-100-75-75-P-56-ISO8859-1
UTOPIA12	-Adobe-Utopia-Regular-R-Normal- -12-120-75-75-P-67-ISO8859-1
UTOPIA14	-Adobe-Utopia-Regular-R-Normal- -15-140-75-75-P-79-ISO8859-1
UTOPIA18	-Adobe-Utopia-Regular-R-Normal- -19-180-75-75-P-101-ISO8859-1
UTOPIA24	-Adobe-Utopia-Regular-R-Normal- -25-240-75-75-P-135-ISO8859-1
UTOPIA_BOLD10	-Adobe-Utopia-Bold-R-Normal- -10-100-75-75-P-59-ISO8859-1
UTOPIA_BOLD12	-Adobe-Utopia-Bold-R-Normal- -12-120-75-75-P-70-ISO8859-1
UTOPIA_BOLD14	-Adobe-Utopia-Bold-R-Normal- -15-140-75-75-P-82-ISO8859-1
UTOPIA_BOLD18	-Adobe-Utopia-Bold-R-Normal- -19-180-75-75-P-105-ISO8859-1
UTOPIA_BOLD24	-Adobe-Utopia-Bold-R-Normal- -25-240-75-75-P-140-ISO8859-1
UTOPIA_BOLD_ITALIC10	-Adobe-Utopia-Bold-I-Normal- -10-100-75-75-P-58-ISO8859-1
UTOPIA_BOLD_ITALIC12	-Adobe-Utopia-Bold-I-Normal- -12-120-75-75-P-70-ISO8859-1
UTOPIA_BOLD_ITALIC14	-Adobe-Utopia-Bold-I-Normal- -15-120-75-75-P-70-ISO8859-1

(continued on next page)

Table 4–4 (Cont.) Additional 75-dpi Fonts (.PCF File Extension)

File Name	Font Name
Utopia	
UTOPIA_BOLD_ITALIC18	-Adobe-Utopia-Bold-I-Normal- -19-180-75-75-P-105-ISO8859-1
UTOPIA_BOLD_ITALIC24	-Adobe-Utopia-Bold-I-Normal- -25-240-75-75-P-140-ISO8859-1
UTOPIA_ITALIC10	-Adobe-Utopia-Regular-I-Normal- -10-100-75-75-P-55-ISO8859-1
UTOPIA_ITALIC12	-Adobe-Utopia-Regular-I-Normal- -12-120-75-75-P-67-ISO8859-1
UTOPIA_ITALIC14	-Adobe-Utopia-Regular-I-Normal- -15-140-75-75-P-79-ISO8859-1
UTOPIA_ITALIC18	-Adobe-Utopia-Regular-I-Normal- -19-180-75-75-P-100-ISO8859-1
UTOPIA_ITALIC24	-Adobe-Utopia-Regular-I-Normal- -25-240-75-75-P-133-ISO8859-1

4.3.6.2 Additional 100-dpi Fonts

Table 4–5 lists the new and previously undocumented 100-dpi fonts and their file names. The files for these fonts are located in the following directory:

DECW$SYSCOMMON:[SYSFONT.DECW.100DPI]

Table 4–5 Additional 100-dpi Fonts (.PCF File Extension)

File Name	Font Name
Charter	
CHARTER08_100DPI	-Bitstream-Charter-Medium-R-Normal- -11-80-100-100-P-61-ISO8859-1
CHARTER10_100DPI	-Bitstream-Charter-Medium-R-Normal- -14-100-100-100-P-78-ISO8859-1
CHARTER12_100DPI	-Bitstream-Charter-Medium-R-Normal- -17-120-100-100-P-95-ISO8859-1
CHARTER14_100DPI	-Bitstream-Charter-Medium-R-Normal- -19-140-100-100-P-106-ISO8859-1
CHARTER18_100DPI	-Bitstream-Charter-Medium-R-Normal- -25-180-100-100-P-139-ISO8859-1
CHARTER24_100DPI	-Bitstream-Charter-Medium-R-Normal- -33-240-100-100-P-183-ISO8859-1
CHARTER_BOLD08_100DPI	-Bitstream-Charter-Bold-R-Normal- -11-80-100-100-P-69-ISO8859-1
CHARTER_BOLD10_100DPI	-Bitstream-Charter-Bold-R-Normal- -14-100-100-100-P-88-ISO8859-1
CHARTER_BOLD12_100DPI	-Bitstream-Charter-Bold-R-Normal- -17-120-100-100-P-107-ISO8859-1
CHARTER_BOLD14_100DPI	-Bitstream-Charter-Bold-R-Normal- -19-140-100-100-P-119-ISO8859-1
CHARTER_BOLD18_100DPI	-Bitstream-Charter-Bold-R-Normal- -25-180-100-100-P-157-ISO8859-1
CHARTER_BOLD24_100DPI	-Bitstream-Charter-Bold-R-Normal- -33-240-100-100-P-206-ISO8859-1
CHARTER_BOLD_ITALIC08_ 100DPI	-Bitstream-Charter-Bold-I-Normal- -11-80-100-100-P-68-ISO8859-1
CHARTER_BOLD_ITALIC10_ 100DPI	-Bitstream-Charter-Bold-I-Normal- -14-100-100-100-P-86-ISO8859-1
CHARTER_BOLD_ITALIC12_ 100DPI	-Bitstream-Charter-Bold-I-Normal- -17-120-100-100-P-105-ISO8859-1
CHARTER_BOLD_ITALIC14_ 100DPI	-Bitstream-Charter-Bold-I-Normal- -19-140-100-100-P-117-ISO8859-1
CHARTER_BOLD_ITALIC18_ 100DPI	-Bitstream-Charter-Bold-I-Normal- -25-180-100-100-P-154-ISO8859-1

(continued on next page)

Table 4–5 (Cont.) Additional 100-dpi Fonts (.PCF File Extension)

File Name	Font Name
Charter	
CHARTER_BOLD_ITALIC24_100DPI	-Bitstream-Charter-Bold-I-Normal- -33-240-100-100-P-203-ISO8859-1
CHARTER_ITALIC08_100DPI	-Bitstream-Charter-Medium-I-Normal- -11-120-100-100-P-92-ISO8859-1
CHARTER_ITALIC10_100DPI	-Bitstream-Charter-Medium-I-Normal- -14-120-100-100-P-92-ISO8859-1
CHARTER_ITALIC12_100DPI	-Bitstream-Charter-Medium-I-Normal- -17-120-100-100-P-92-ISO8859-1
CHARTER_ITALIC14_100DPI	-Bitstream-Charter-Medium-I-Normal- -19-140-100-100-P-103-ISO8859-1
CHARTER_ITALIC18_100DPI	-Bitstream-Charter-Medium-I-Normal- -25-180-100-100-P-136-ISO8859-1
CHARTER_ITALIC24_100DPI	-Bitstream-Charter-Medium-I-Normal- -33-240-100-100-P-179-ISO8859-1
Lucida	
LUCIDABRIGHT08_100DPI	-B&H-LucidaBright-Medium-R-Normal- -11-80-100-100-P-63-ISO8859-1
LUCIDABRIGHT10_100DPI	-B&H-LucidaBright-Medium-R-Normal- -14-100-100-100-P-80-ISO8859-1
LUCIDABRIGHT12_100DPI	-B&H-LucidaBright-Medium-R-Normal- -17-120-100-100-P-96-ISO8859-1
LUCIDABRIGHT14_100DPI	-B&H-LucidaBright-Medium-R-Normal- -20-140-100-100-P-114-ISO8859-1
LUCIDABRIGHT18_100DPI	-B&H-LucidaBright-Medium-R-Normal- -25-180-100-100-P-142-ISO8859-1
LUCIDABRIGHT19_100DPI	-B&H-LucidaBright-Medium-R-Normal- -26-190-100-100-P-149-ISO8859-1
LUCIDABRIGHT24_100DPI	-B&H-LucidaBright-Medium-R-Normal- -34-240-100-100-P-193-ISO8859-1
LUCIDABRIGHT_DEMI08_100DPI	-B&H-LucidaBright-Demibold-R-Normal- -11-80-100-100-P-66-ISO8859-1
LUCIDABRIGHT_DEMI10_100DPI	-B&H-LucidaBright-Demibold-R-Normal- -14-100-100-100-P-84-ISO8859-1
LUCIDABRIGHT_DEMI12_100DPI	-B&H-LucidaBright-Demibold-R-Normal- -17-120-100-100-P-101-ISO8859-1
LUCIDABRIGHT_DEMI14_100DPI	-B&H-LucidaBright-Demibold-R-Normal- -20-140-100-100-P-118-ISO8859-1
LUCIDABRIGHT_DEMI18_100DPI	-B&H-LucidaBright-Demibold-R-Normal- -25-180-100-100-P-149-ISO8859-1
LUCIDABRIGHT_DEMI19_100DPI	-B&H-LucidaBright-Demibold-R-Normal- -26-190-100-100-P-155-ISO8859-1
LUCIDABRIGHT_DEMI24_100DPI	-B&H-LucidaBright-Demibold-R-Normal- -34-240-100-100-P-202-ISO8859-1
LUCIDABRIGHT_DEMIITALIC08_100DPI	-B&H-LucidaBright-Demibold-I-Normal- -11-80-100-100-P-66-ISO8859-1
LUCIDABRIGHT_DEMIITALIC10_100DPI	-B&H-LucidaBright-Demibold-I-Normal- -14-100-100-100-P-84-ISO8859-1
LUCIDABRIGHT_DEMIITALIC12_100DPI	-B&H-LucidaBright-Demibold-I-Normal- -17-120-100-100-P-101-ISO8859-1
LUCIDABRIGHT_DEMIITALIC14_100DPI	-B&H-LucidaBright-Demibold-I-Normal- -20-140-100-100-P-119-ISO8859-1
LUCIDABRIGHT_DEMIITALIC18_100DPI	-B&H-LucidaBright-Demibold-I-Normal- -25-180-100-100-P-149-ISO8859-1

(continued on next page)

Table 4–5 (Cont.) Additional 100-dpi Fonts (.PCF File Extension)

File Name	Font Name
Lucida	
LUCIDABRIGHT_DEMIITALIC19_100DPI	-B&H-LucidaBright-Demibold-I-Normal- -26-190-100-100-P-156-ISO8859-1
LUCIDABRIGHT_DEMIITALIC24_100DPI	-B&H-LucidaBright-Demibold-I-Normal- -34-240-100-100-P-203-ISO8859-1
LUCIDABRIGHT_ITALIC08_100DPI	-B&H-LucidaBright-Medium-I-Normal- -11-80-100-100-P-63-ISO8859-1
LUCIDABRIGHT_ITALIC10_100DPI	-B&H-LucidaBright-Medium-I-Normal- -14-100-100-100-P-80-ISO8859-1
LUCIDABRIGHT_ITALIC12_100DPI	-B&H-LucidaBright-Medium-I-Normal- -17-120-100-100-P-96-ISO8859-1
LUCIDABRIGHT_ITALIC14_100DPI	-B&H-LucidaBright-Medium-I-Normal- -20-140-100-100-P-113-ISO8859-1
LUCIDABRIGHT_ITALIC18_100DPI	-B&H-LucidaBright-Medium-I-Normal- -25-180-100-100-P-142-ISO8859-1
LUCIDABRIGHT_ITALIC19_100DPI	-B&H-LucidaBright-Medium-I-Normal- -26-190-100-100-P-148-ISO8859-1
LUCIDABRIGHT_ITALIC24_100DPI	-B&H-LucidaBright-Medium-I-Normal- -34-240-100-100-P-194-ISO8859-1
LUCIDATYPEWRITER_BOLDSANS08_100DPI	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-11-80-100-100-m-70-ISO8859-1
LUCIDATYPEWRITER_BOLDSANS10_100DPI	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-14-100-100-100-m-80-ISO8859-1
LUCIDATYPEWRITER_BOLDSANS12_100DPI	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-17-120-100-100-m-100-ISO8859-1
LUCIDATYPEWRITER_BOLDSANS14_100DPI	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-20-140-100-100-m-120-ISO8859-1
LUCIDATYPEWRITER_BOLDSANS18_100DPI	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-25-180-100-100-m-150-ISO8859-1
LUCIDATYPEWRITER_BOLDSANS19_100DPI	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-26-190-100-100-m-159-ISO8859-1
LUCIDATYPEWRITER_BOLDSANS24_100DPI	-B&H-LucidaTypewriter-Bold-R-Normal-Sans-34-240-100-100-m-200-ISO8859-1
LUCIDATYPEWRITER_SANS08_100DPI	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-11-80-100-100-m-70-ISO8859-1
LUCIDATYPEWRITER_SANS10_100DPI	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-14-100-100-100-m-80-ISO8859-1
LUCIDATYPEWRITER_SANS12_100DPI	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-17-120-100-100-m-100-ISO8859-1
LUCIDATYPEWRITER_SANS14_100DPI	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-20-140-100-100-m-120-ISO8859-1
LUCIDATYPEWRITER_SANS18_100DPI	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-25-180-100-100-m-150-ISO8859-1
LUCIDATYPEWRITER_SANS19_100DPI	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-26-190-100-100-m-159-ISO8859-1
LUCIDATYPEWRITER_SANS24_100DPI	-B&H-LucidaTypewriter-Medium-R-Normal-Sans-34-240-100-100-m-200-ISO8859-1

(continued on next page)

Table 4–5 (Cont.) Additional 100-dpi Fonts (.PCF File Extension)

File Name	Font Name
Lucida	
LUCIDA_BOLDITALICSANS08_100DPI	-B&H-Lucida-Bold-I-Normal-Sans-11-80-100-100-P-69-ISO8859-1
LUCIDA_BOLDITALICSANS10_100DPI	-B&H-Lucida-Bold-I-Normal-Sans-14-100-100-100-P-90-ISO8859-1
LUCIDA_BOLDITALICSANS12_100DPI	-B&H-Lucida-Bold-I-Normal-Sans-17-120-100-100-P-108-ISO8859-1
LUCIDA_BOLDITALICSANS14_100DPI	-B&H-Lucida-Bold-I-Normal-Sans-20-140-100-100-P-127-ISO8859-1
LUCIDA_BOLDITALICSANS18_100DPI	-B&H-Lucida-Bold-I-Normal-Sans-25-180-100-100-P-159-ISO8859-1
LUCIDA_BOLDITALICSANS19_100DPI	-B&H-Lucida-Bold-I-Normal-Sans-26-190-100-100-P-166-ISO8859-1
LUCIDA_BOLDITALICSANS24_100DPI	-B&H-Lucida-Bold-I-Normal-Sans-34-240-100-100-P-215-ISO8859-1
LUCIDA_BOLDSANS08_100DPI	-B&H-Lucida-Bold-R-Normal-Sans-11-80-100-100-P-70-ISO8859-1
LUCIDA_BOLDSANS10_100DPI	-B&H-Lucida-Bold-R-Normal-Sans-14-100-100-100-P-89-ISO8859-1
LUCIDA_BOLDSANS12_100DPI	-B&H-Lucida-Bold-R-Normal-Sans-17-120-100-100-P-108-ISO8859-1
LUCIDA_BOLDSANS14_100DPI	-B&H-Lucida-Bold-R-Normal-Sans-20-140-100-100-P-127-ISO8859-1
LUCIDA_BOLDSANS18_100DPI	-B&H-Lucida-Bold-R-Normal-Sans-25-180-100-100-P-158-ISO8859-1
LUCIDA_BOLDSANS19_100DPI	-B&H-Lucida-Bold-R-Normal-Sans-26-190-100-100-P-166-ISO8859-1
LUCIDA_BOLDSANS24_100DPI	-B&H-Lucida-Bold-R-Normal-Sans-34-240-100-100-P-216-ISO8859-1
LUCIDA_ITALICSANS08_100DPI	-B&H-Lucida-Medium-I-Normal-Sans-11-80-100-100-P-62-ISO8859-1
LUCIDA_ITALICSANS10_100DPI	-B&H-Lucida-Medium-I-Normal-Sans-14-100-100-100-P-80-ISO8859-1
LUCIDA_ITALICSANS12_100DPI	-B&H-Lucida-Medium-I-Normal-Sans-17-120-100-100-P-97-ISO8859-1
LUCIDA_ITALICSANS14_100DPI	-B&H-Lucida-Medium-I-Normal-Sans-20-140-100-100-P-114-ISO8859-1
LUCIDA_ITALICSANS18_100DPI	-B&H-Lucida-Medium-I-Normal-Sans-25-180-100-100-P-141-ISO8859-1
LUCIDA_ITALICSANS19_100DPI	-B&H-Lucida-Medium-I-Normal-Sans-26-190-100-100-P-147-ISO8859-1
LUCIDA_ITALICSANS24_100DPI	-B&H-Lucida-Medium-I-Normal-Sans-34-240-100-100-P-192-ISO8859-1
LUCIDA_SANS08_100DPI	-B&H-Lucida-Medium-R-Normal-Sans-11-80-100-100-P-63-ISO8859-1
LUCIDA_SANS10_100DPI	-B&H-Lucida-Medium-R-Normal-Sans-14-100-100-100-P-80-ISO8859-1
LUCIDA_SANS12_100DPI	-B&H-Lucida-Medium-R-Normal-Sans-17-120-100-100-P-96-ISO8859-1
LUCIDA_SANS14_100DPI	-B&H-Lucida-Medium-R-Normal-Sans-20-140-100-100-P-114-ISO8859-1

(continued on next page)

Table 4–5 (Cont.) Additional 100-dpi Fonts (.PCF File Extension)

File Name	Font Name
Lucida	
LUCIDA_SANS18_100DPI	-B&H-Lucida-Medium-R-Normal-Sans-25-180-100-100-P-142-ISO8859-1
LUCIDA_SANS19_100DPI	-B&H-Lucida-Medium-R-Normal-Sans-26-190-100-100-P-147-ISO8859-1
LUCIDA_SANS24_100DPI	-B&H-Lucida-Medium-R-Normal-Sans-34-240-100-100-P-191-ISO8859-1
Present Bullets	
PRESENT_BULLETS8_100	-DEC-PresentBullets-Medium-R-Normal- -8-80-100-100-P-105-DEC-FontSpecific
PRESENT_BULLETS10_100	-DEC-PresentBullets-Medium-R-Normal- -10-100-100-100-P-123-DEC-FontSpecific
PRESENT_BULLETS12_100	-DEC-PresentBullets-Medium-R-Normal- -12-120-100-100-P-154-DEC-FontSpecific
PRESENT_BULLETS14_100	-DEC-PresentBullets-Medium-R-Normal- -14-140-100-100-P-172-DEC-FontSpecific
PRESENT_BULLETS18_100	-DEC-PresentBullets-Medium-R-Normal- -18-180-100-100-P-229-DEC-FontSpecific
PRESENT_BULLETS24_100	-DEC-PresentBullets-Medium-R-Normal- -24-240-100-100-P-305-DEC-FontSpecific
PRESENT_BULLETS36_100	-DEC-PresentBullets-Medium-R-Normal- -36-360-100-100-P-458-DEC-FontSpecific
PRESENT_BULLETS48_100	-DEC-PresentBullets-Medium-R-Normal- -48-480-100-100-P-609-DEC-FontSpecific
PRESENT_BULLETS72_100	-DEC-PresentBullets-Medium-R-Normal- -72-720-100-100-P-952-DEC-FontSpecific
Utopia	
UTOPIA10_100DPI	-Adobe-Utopia-Regular-R-Normal- -14-100-100-100-P-75-ISO8859-1
UTOPIA12_100DPI	-Adobe-Utopia-Regular-R-Normal- -17-120-100-100-P-91-ISO8859-1
UTOPIA14_100DPI	-Adobe-Utopia-Regular-R-Normal- -19-140-100-100-P-105-ISO8859-1
UTOPIA18_100DPI	-Adobe-Utopia-Regular-R-Normal- -25-180-100-100-P-135-ISO8859-1
UTOPIA24_100DPI	-Adobe-Utopia-Regular-R-Normal- -33-240-100-100-P-180-ISO8859-1
UTOPIA_BOLD10_100DPI	-Adobe-Utopia-Bold-R-Normal- -14-100-100-100-P-78-ISO8859-1
UTOPIA_BOLD12_100DPI	-Adobe-Utopia-Bold-R-Normal- -17-120-100-100-P-93-ISO8859-1
UTOPIA_BOLD14_100DPI	-Adobe-Utopia-Bold-R-Normal- -19-140-100-100-P-108-ISO8859-1
UTOPIA_BOLD18_100DPI	-Adobe-Utopia-Bold-R-Normal- -25-180-100-100-P-140-ISO8859-1
UTOPIA_BOLD24_100DPI	-Adobe-Utopia-Bold-R-Normal- -33-240-100-100-P-186-ISO8859-1
UTOPIA_BOLDITALIC10_ 100DPI	-Adobe-Utopia-Bold-I-Normal- -14-100-100-100-P-78-ISO8859-1
UTOPIA_BOLDITALIC12_ 100DPI	-Adobe-Utopia-Bold-I-Normal- -17-120-100-100-P-93-ISO8859-1
UTOPIA_BOLDITALIC14_ 100DPI	-Adobe-Utopia-Bold-I-Normal- -19-140-100-100-P-109-ISO8859-1
UTOPIA_BOLDITALIC18_ 100DPI	-Adobe-Utopia-Bold-I-Normal- -25-180-100-100-P-139-ISO8859-1
UTOPIA_BOLDITALIC24_ 100DPI	-Adobe-Utopia-Bold-I-Normal- -33-240-100-100-P-186-ISO8859-1
UTOPIA_ITALIC10_100DPI	-Adobe-Utopia-Regular-I-Normal- -14-100-100-100-P-74-ISO8859-1
UTOPIA_ITALIC12_100DPI	-Adobe-Utopia-Regular-I-Normal- -17-120-100-100-P-89-ISO8859-1

(continued on next page)

Table 4–5 (Cont.) Additional 100-dpi Fonts (.PCF File Extension)

File Name	Font Name
Utopia	
UTOPIA_ITALIC14_100DPI	-Adobe-Utopia-Regular-I-Normal- -19-140-100-100-P-104-ISO8859-1
UTOPIA_ITALIC18_100DPI	-Adobe-Utopia-Regular-I-Normal- -25-180-100-100-P-134-ISO8859-1
UTOPIA_ITALIC24_100DPI	-Adobe-Utopia-Regular-I-Normal- -33-240-100-100-P-179-ISO8859-1

4.3.6.3 Additional Common Fonts

Table 4–6 lists previously undocumented Common fonts and their file names. The files for these fonts are located in the following directory:

DECW$SYSCOMMON:[SYSFONT.DECW.COMMON]

Table 4–6 Additional Common Fonts (.PCF File Extension)

File Name	Font Name
Fixed Width	
5X7	-Misc-Fixed-Medium-R-Normal- -7-70-75-75-C-50-ISO8859-1
7X14B	-Misc-Fixed-Bold-R-Normal- -14-130-75-75-C-70-ISO8859-1
7X14RK	-Misc-Fixed-Medium-R-Normal- -14-130-75-75-C-70-JISX0201.1976-0
8X16RK	-Sony-Fixed-Medium-R-Normal- -16-120-100-100-C-80-JISX0201.1976-0
12X24	-Sony-Fixed-Medium-R-Normal- -24-170-100-100-C-120-ISO8859-1
12X24RK	-Sony-Fixed-Medium-R-Normal- -24-170-100-100-C-120-JISX0201.1976-0
Sun Open Look Glyph	
OLGL10	-Sun-Open Look Glyph- - - - -10-100-75-75-P-101-SunOLGlyph-1
OLGL12	-Sun-Open Look Glyph- - - - -12-120-75-75-P-113-SunOLGlyph-1
OLGL14	-Sun-Open Look Glyph- - - - -14-140-75-75-P-128-SunOLGlyph-1
OLGL19	-Sun-Open Look Glyph- - - - -19-190-75-75-P-154-SunOLGlyph-1
VT330	
VT33018	-DEC-VT330-Medium-R-Normal- -20-180-75-75-C-100-ISO8859-1
VT33036	-DEC-VT330-Medium-R-Normal- -40-360-75-75-C-200-ISO8859-1
VT330_BOLD18	-DEC-VT330-Bold-R-Normal- -20-180-75-75-C-100-ISO8859-1
VT330_BOLD36	-DEC-VT330-Bold-R-Normal- -40-360-75-75-C-200-ISO8859-1
VT330_BOLD_DBLWIDE18	-DEC-VT330-Bold-R-Double Wide- -20-180-75-75-C-200-ISO8859-1
VT330_BOLD_DBLWIDE_ DECTECH18	-DEC-VT330-Bold-R-Double Wide- -20-180-75-75-C-200-DEC-DECTech
VT330_BOLD_DECTECH18	-DEC-VT330-Bold-R-Normal- -20-180-75-75-C-100-DEC-DECTech
VT330_BOLD_DECTECH36	-DEC-VT330-Bold-R-Normal- -40-360-75-75-C-200-DEC-DECTech
VT330_BOLD_NARROW18	-DEC-VT330-Bold-R-Narrow- -20-180-75-75-C-60-ISO8859-1
VT330_BOLD_NARROW36	-DEC-VT330-Bold-R-Narrow- -40-360-75-75-C-120-ISO8859-1

(continued on next page)

Table 4–6 (Cont.) Additional Common Fonts (.PCF File Extension)

File Name	Font Name
VT330	
VT330_BOLD_NARROW_ DECTECH18	-DEC-VT330-Bold-R-Narrow- -20-180-75-75-C-60-DEC-DECTech
VT330_BOLD_NARROW_ DECTECH36	-DEC-VT330-Bold-R-Narrow- -40-360-75-75-C-120-DEC-DECTech
VT330_BOLD_WIDE18	-DEC-VT330-Bold-R-Wide- -20-180-75-75-C-120-ISO8859-1
VT330_BOLD_WIDE_ DECTECH18	-DEC-VT330-Bold-R-Wide- -20-180-75-75-C-120-DEC-DECTech
VT330_DBLWIDE18	-DEC-VT330-Medium-R-Double Wide- -20-180-75-75-C-200-ISO8859-1
VT330_DBLWIDE_DECTECH18	-DEC-VT330-Medium-R-Double Wide- -20-180-75-75-C-200-DEC-DECTech
VT330_DECTECH18	-DEC-VT330-Medium-R-Normal- -20-180-75-75-C-100-DEC-DECTech
VT330_DECTECH36	-DEC-VT330-Medium-R-Normal- -40-360-75-75-C-200-DEC-DECTech
VT330_NARROW18	-DEC-VT330-Medium-R-Narrow- -20-180-75-75-C-60-ISO8859-1
VT330_NARROW36	-DEC-VT330-Medium-R-Narrow- -40-360-75-75-C-120-ISO8859-1
VT330_NARROW_DECTECH18	-DEC-VT330-Medium-R-Narrow- -20-180-75-75-C-60-DEC-DECTech
VT330_NARROW_DECTECH36	-DEC-VT330-Medium-R-Narrow- -40-360-75-75-C-120-DEC-DECTech
VT330_WIDE18	-DEC-VT330-Medium-R-Wide- -20-180-75-75-C-120-ISO8859-1
VT330_WIDE_DECTECH18	-DEC-VT330-Medium-R-Wide- -20-180-75-75-C-120-DEC-DECTech
Language-Specific Fonts	
HANGLG16	-Daewoo-Gothic-Medium-R-Normal- -16-120-100-100-C-160-KSC5601.1987-0
HANGLM16	-Daewoo-Mincho-Medium-R-Normal- -16-120-100-100-C-160-KSC5601.1987-0
HANGLM24	-Daewoo-Mincho-Medium-R-Normal- -24-170-100-100-C-240-KSC5601.1987-0
HEB6X13	-Misc-Fixed-Medium-R-SemiCondensed- -13-120-75-75-C-60-ISO8859-8
HEB8X13	-Misc-Fixed-Medium-R-Normal- -13-120-75-75-C-80-ISO8859-8
JISKAN16	-JIS-Fixed-Medium-R-Normal- -16-150-75-75-C-160-JISX0208.1983-0
JISKAN24	-JIS-Fixed-Medium-R-Normal- -24-230-75-75-C-240-JISX0208.1983-0
K14	-Misc-Fixed-Medium-R-Normal- -14-130-75-75-C-140-JISX0208.1983-0
Miscellaneous Fonts	
NIL2	-Misc-Nil-Medium-R-Normal- -2-20-75-75-C-10-Misc-FontSpecific

4.3.6.4 Bitstream Speedo Scalable Fonts

Table 4–7 lists the previously undocumented Bitstream Speedo scalable fonts and their file names. The files for these fonts are located in the following directory:

DECW$SYSCOMMON:[SYSFONT.DECW.SPEEDO]

Table 4–7 Bitstream Speedo Scalable Fonts (.SPD File Extension)

File Name	Font Name
Charter	
FONT0648	-Bitstream-Charter-Medium-R-Normal- -0-0-0-0-P-0-ISO8859-1
FONT0649	-Bitstream-Charter-Medium-I-Normal- -0-0-0-0-P-0-ISO8859-1
FONT0709	-Bitstream-Charter-Bold-R-Normal- -0-0-0-0-P-0-ISO8859-1
FONT0710	-Bitstream-Charter-Bold-I-Normal- -0-0-0-0-P-0-ISO8859-1
Courier	
FONT0419	-Bitstream-Courier-Medium-R-Normal- -0-0-0-0-m-0-ISO8859-1
FONT0582	-Bitstream-Courier-Medium-I-Normal- -0-0-0-0-m-0-ISO8859-1
FONT0583	-Bitstream-Courier-Bold-R-Normal- -0-0-0-0-m-0-ISO8859-1
FONT0611	-Bitstream-Courier-Bold-I-Normal- -0-0-0-0-m-0-ISO8859-1

4.3.6.5 Agfa Monotype TrueType Scalable Fonts

Table 4–8 lists the new Agfa Monotype TrueType scalable fonts and their file names. The files for these fonts are located in the following directory:

DECW$SYSCOMMON:[SYSFONT.DECW.TRUETYPE]

Table 4–8 Agfa Monotype TrueType Scalable Fonts (.TTF File Extension)

File Name	Font Name
Albany (Similar to Arial)	
ALBANYBD	-Agfa Monotype-Albany-Bold-R-Normal- -0-0-0-0-P-0-ISO8859-1
ALBANYBI	-Agfa Monotype-Albany-Bold-I-Normal- -0-0-0-0-P-0-ISO8859-1
ALBANYIT	-Agfa Monotype-Albany-Medium-I-Normal- -0-0-0-0-P-0-ISO8859-1
ALBANY_ _	-Agfa Monotype-Albany-Medium-R-Normal- -0-0-0-0-P-0-ISO8859-1
Cumberland (Similar to Courier)	
CUMBB	-Agfa Monotype-Cumberland-Bold-R-Normal- -0-0-0-0-M-0-ISO8859-1
CUMBBI	-Agfa Monotype-Cumberland-Bold-I-Normal- -0-0-0-0-M-0-ISO8859-1
CUMBI	-Agfa Monotype-Cumberland-Medium-I-Normal- -0-0-0-0-M-0-ISO8859-1
CUMBR	-Agfa Monotype-Cumberland-Medium-R-Normal- -0-0-0-0-M-0-ISO8859-1

(continued on next page)

Table 4–8 (Cont.) Agfa Monotype TrueType Scalable Fonts (.TTF File Extension)

File Name	Font Name
Screen	
SAN_M_21	-Agfa Monotype-Screen Sans-Medium-R-Normal- -0-0-0-0-M-0-ISO8859-1
SAN_P_21	-Agfa Monotype-Screen Sans-Medium-R-Normal- -0-0-0-0-P-0-ISO8859-1
SRF_M_21	-Agfa Monotype-Screen Serif-Medium-R-Normal- -0-0-0-0-M-0-ISO8859-1
SRF_P_21	-Agfa Monotype-Screen Serif-Medium-R-Normal- -0-0-0-0-P-0-ISO8859-1
Thorndale (Similar to Times New Roman)	
THOBI_ _ _	-Agfa Monotype-Thorndale-Bold-I-Normal- -0-0-0-0-P-0-ISO8859-1
THOB_ _ _ _	-Agfa Monotype-Thorndale-Bold-R-Normal- -0-0-0-0-P-0-ISO8859-1
THOI_ _ _ _	-Agfa Monotype-Thorndale-Medium-I-Normal- -0-0-0-0-P-0-ISO8859-1
THOR_ _ _ _	-Agfa Monotype-Thorndale-Medium-R-Normal- -0-0-0-0-P-0-ISO8859-1

4.3.6.6 Adobe Type1 Fonts

Table 4–9 lists previously undocumented Adobe Type1 fonts and their file names. The files for these fonts are located in the following directory:

DECW$SYSCOMMON:[SYSFONT.DECW.TYPE1]

Table 4–9 Adobe Type1 Scalable Fonts (.PCA File Extension)

File Name	Font Name
Courier	
COUR	-Adobe-Courier-Medium-R-Normal- -0-0-0-0-P-0-ISO8859-1
COURI	-Adobe-Courier-Medium-I-Normal- -0-0-0-0-P-0-ISO8859-1
COURB	-Adobe-Courier-Bold-R-Normal- -0-0-0-0-P-0-ISO8859-1
COURBI	-Adobe-Courier-Bold-I-Normal- -0-0-0-0-P-0-ISO8859-1
Utopia	
UTRG_ _ _ _	-Adobe-Utopia-Medium-R-Normal- -0-0-0-0-P-0-ISO8859-1
UTI_ _ _ _ _	-Adobe-Utopia-Medium-I-Normal- -0-0-0-0-P-0-ISO8859-1
UTB_ _ _ _ _	-Adobe-Utopia-Bold-R-Normal- -0-0-0-0-P-0-ISO8859-1
UTBI_ _ _ _	-Adobe-Utopia-Bold-I-Normal- -0-0-0-0-P-0-ISO8859-1

4.3.7 UIDPATH Environment Variable

V1.2–6

When opening a hierarchy, DECwindows Motif searches the DECW$USER_DEFAULTS and DECW$SYSTEM_DEFAULTS areas for the User Interface Definition (UID) file. On UNIX systems, the search path is defined using the UIDPATH variable and its fallbacks.

Now DECwindows Motif also checks for the UIDPATH variable if the UID file is not found using either of the OpenVMS symbols listed above. This variable references a UNIX-style pathname (for example, /foo/bar) and allows the substitutions strings as specified by X11 standards. For more information on the UIDPATH variable, see the *OSF/Motif Programmer's Reference*.

Note

The UIDPATH variable does not work with OpenVMS directory specifications. Use the DECW$*xxx*_DEFAULTS logicals to specify OpenVMS-style search paths.

4.3.8 Client Side Extension Library

V1.1

Starting with DECwindows Motif for OpenVMS Version 1.1, Xlib added a client side library, DECW$XEXTLIBSHR.EXE, that allows OpenVMS clients to issue Shape, XInput, Multibuffer, and Shared Memory extension requests to servers that provide these features.

You must modify the linking file options for client applications that issue these extension requests to link to the Xlib extensions shareable image in SYS$LIBRARY:DECW$XEXTLIBSHR.EXE. Add the following line to your linker options file:

```
SYS$LIBRARY:DECW$XEXTLIBSHR/SHARE
```

For more information on Shape, XInput, and Multibuffer extensions, see the following text files in SYS$HELP:

DECW$SHAPE.TXT
DECW$XINPUT.TXT
DECW$MULTIBUFFER.TXT

4.4 X Window System Toolkit (Xt)

The following sections describe features related to X Window System toolkit (Xt).

4.4.1 New Functions Available with X11R6.6 Upgrade

V1.3

The following new functions from X11R6.6 have been added to the version of Xt available with DECwindows Motif for OpenVMS Alpha Version 1.3.

Table 4–10 New Xt Functions Supported for X11R6.6

Function Name	Description
XtAppAddBlockHook	Registers a block hook procedure.
XtAppAddSignal	Registers a signal callback.
XtAppGetExitFlag	Supports controlled exit from main loop in a multithreaded application by returning the flag set by XtAppSetExitFlag.

(continued on next page)

Table 4–10 (Cont.) New Xt Functions Supported for X11R6.6

Function Name	Description
XtAppLock	Locks the application context in a multithreaded application.
XtAppSetExitFlag	Supports controlled exit from main loop in a multithreaded application by setting a flag in the application context.
XtAppUnlock	Releases an application context lock.
XtCancelSelectionRequest	Cancels a multiple selection request.
XtChangeManagedSet	Simultaneously removes from and adds to the geometry managed set of a composite widget.
XtCreateSelectionRequest	Adds to a multiple selection request.
XtDispatchEventToWidget	Dispatches an event to a specified widget.
XtGetClassExtension	Locates a class extension record of an object class.
XtGetDisplays	Lists the open displays associated with an application context.
XtGetKeyboardFocusWidget	Determines which widget would be the end result of keyboard event forwarding for a keyboard event on a specified widget.
XtGetSelectionParameters	Gets target parameters needed to perform a selection conversion.
XtHooksOfDisplay	Retrieves the hook registration object for the specified display.
XtInsertEventTypeHandler	Registers an event handler procedure by event type.
XtIsSessionShell	Widget subclass verification function for the session shell widget.
XtLastEventProcessed	Retrieves the last event processed by XtDispatchEvent.
XtNoticeSignal	Notifies the X Toolkit that a signal has occurred.
XtOpenApplication	Convenience function to initialize intrinsics, create an application context, open a display connection, and create an application shell.
XtProcessLock	Locks the X Toolkit process lock.
XtProcessUnlock	Releases the X Toolkit process lock.
XtRegisterDrawable	Associates a drawable with a widget so that the drawable receives events as if part of the widget.
XtRegisterExtensionSelector	Registers a procedure to receive extension events for a widget.
XtRemoveBlockHook	Discontinues use of a block hook procedure.
XtRemoveEventTypeHandler	Removes a registration created by XtInsertEventTypeHandler.
XtRemoveSignal	Removes a registered signal callback.
XtReleasePropertyAtom	Releases a reservation made by XtReservePropertyAtom.
XtReservePropertyAtom	Reserves a unique atom for selection requests on a widget.

(continued on next page)

Table 4–10 (Cont.) New Xt Functions Supported for X11R6.6

Function Name	Description
XtSendSelectionRequest	Sends a multiple selection request.
XtSessionGetToken	With the new session shell widget, gets an additional token for a save callback response with a deferred outcome.
XtSessionReturnToken	Returns a token obtained using XtSessionGetToken when checkpoint processing is complete.
XtSetEventDispatcher	Registers the event dispatcher procedure for events of the specified type.
XtSetSelectionParameters	Associates target parameters with a selection.
XtToolkitThreadInitialize	Initializes multithreaded support.
XtUnregisterDrawable	Removes an association set by XtRegisterDrawable.
XtVaOpenApplication	Convenience function to initialize intrinsics, create an application context, open a display connection, and create an application shell.

In addition, two new variables are provided in support of the new session shell widget:

sessionShellClassRec
sessionShellWidgetClass

See the *X Toolkit Intrinsics - C Language Interface, X Window System, X Version 11 Release 6.4* specification available from X.Org for detailed information about each of these functions and variables.

4.4.2 Support for Easy Resource Configuration

V1.3

Setting and changing resources in X Window System applications can be difficult for both the application developer and the end user. Resource Configuration Management (RCM) addresses this problem by changing the X Intrinsics to immediately modify a resource for the specified widget and each child widget in the hierarchy. As a result:

- No sourcing of resource files is required.
- The application does not need to be restarted for the new resource values to take effect.
- The change occurs immediately.

RCM was made available as part of the X11R6.4 release and is now available with DECwindows Motif for OpenVMS Alpha Version 1.3. However, note that RCM is not a standard part of the X Toolkit Intrinsics. It is neither an X Consortium standard nor part of the X Window System specifications. As a result, there are currently no public customization tools that take advantage of this feature.

If you are interested in learning more about RCM, see the *X Toolkit Intrinsics - C Language Interface, X Window System, X Version 11 Release 6.4 Release Notes* available from X.Org.

4.4.3 New Option for CompositeClassExtensionRec

V1.3

With X11R6, some modifications were made to the widget internals, as described in the *X Window System Toolkit* manual.

In particular, a new option in the Composite Class extension record enables you to make bundled changes to the managed set of a Composite widget. Widgets that define a change-managed procedure that performs additions and deletions to the managed set of children in a single invocation should set allows_change_ managed_set option to TRUE in the extension record.

For more information about the impact this new option may have on existing applications, see the *HP DECwindows Motif for OpenVMS Alpha Release Notes*.

4.4.4 New Default Format for XtResolvePathname

V1.2–6

In XtResolvePathname, the default pathname is required to have certain properties when either no other path information is present in the call, or when it is referenced by the environment variable XFILESEARCHPATH. The former default OpenVMS format of the pathname consisted of a type-name-suffix substitution. The modified pathname now reflects the 6-part fallback, as specified by X11 Release 6.

The new pathname behavior is enabled by setting the DECW$VSW_COMPLIANT variable, as follows:

```
$ DEFINE DECW$VSW_COMPLIANT 1
```

4.4.5 XtAppMainLoop Routine

V1.2–5

Previously, if a program entered its event loop, (for example, by calling XtAppMainLoop) without having opened a display or specified a timer or event flag for the program to wait for (by calling XtAppAddTimeout or XtAppAddInput), Xlib terminated the program with the following error message:

```
X Toolkit Error: Error in XMultiplexInput
```

Starting with DECwindows Motif for OpenVMS Version 1.2–5, if there is nothing to wait for, Xlib stalls waiting for input instead of terminating with an error status.

To allow Xlib to process events at a later time, applications should provide some means of regaining control, such as specifying an event flag (on DECwindows Motif for OpenVMS Version 1.2–6 and previous systems) or a logical connection number (on DECwindows Motif for OpenVMS Alpha Version 1.3 and higher systems) by calling XtAppAddInput.

4.5 X Window System Extensions and Protocols

The following sections describe features related to X Window System extensions, protocols, and their libraries.

4.5.1 Additional X Display Server Extensions Supported with X11R6.6

V1.3

The following X11R6.6 protocol X Window System extensions have been integrated into the DECwindows X11 Display Server and are now supported by DECwindows Motif:

- Application Group (XC-APPGROUP)
- Big Requests (BIG-REQUESTS)
- Colormap Utilization Policy (TOG-CUP)
- Extended Visual Information (EVI)
- Low-Bandwidth X (LBX)
- Security (SECURITY)
- Synchronization (SYNC)
- XC-MISC
- X Double Buffer (DBE)
- XINERAMA (formerly PanoramiX)
- X Keyboard (XKB)

BIG-REQUESTS, EVI, SYNC, TOG-CUP, and XC-MISC are a permanent part of the DECwindows X11 Display Server and are always enabled. DBE, LBX, SECURITY, XC-APPGROUP, XINERAMA, and XKB are dynamically loadable using the DECW$SERVER_EXTENSIONS parameter defined in the DECW$PRIVATE_SERVER_SETUP.COM file. With this symbol, each extension is converted to a shareable image, which is run at server startup. Note that some combinations of extensions can result in conflict; see Section 3.7.3 for instructions on enabling these extensions.

To access these these extensions, link applications against one or more of the following libraries:

Extension	Library
DBE EVI LBX SECURITY SYNC TOG-CUP XC-APPGROUP XINERAMA	DECW$XEXTLIBSHR
BIG-REQUESTS XC-MISC XKB	DECW$XLIBSHR

For more information about the updates made to the client-side extension library in support of X11R6.6, see Section 4.3.2.

The following sections briefly describe each extension, its function, and any variances from the standard X11R6.6 implementation provided by X.Org. For a detailed description of each extension protocol and the available server requests, see the *X Window System* (Scheifler and Gettys) series of manuals published by Butterworth-Heinemann, or visit the X.Org web site (http://www.x.org) for the X Window System protocol and library specifications.

4.5.1.1 Application Group Extension (XC-APPGROUP)

XC-APPGROUP enables multiple programs to manage X Window applications on the desktop. This extension allows X applications to be embedded into the window of another program, such as a web browser.

Sets of one or more applications, known as an Application Group, are managed by a controlling application, known as the Application Group Leader. The group shares the *Substructure-Redirect* attribute of the window with the Application Manager and one or more Application Group Leaders.

Code that uses XC-APPGROUP must include the following header files:

```
# include "DECW$INCLUDE:Xlib.h"
# include "DECW$INCLUDE:Xag.h"
```

This extension is dynamically loadable (along with the SECURITY extension) at server startup; see Section 3.7.3. Call the following routine to check if XC-APPGROUP is available on the server system:

```
Bool XagQueryVersion (
       Display *dpy,
       int     *major_version_return,
       int     *minor_version_return
);
```

The following table lists each argument and its description.

Argument	Description
dpy	An input parameter that contains the current display.
major_version_return	Major version number of the extension implementation. Returned by XagQueryVersion.
minor_version_return	Minor version number of the extension implementation. Returned by XagQueryVersion.

4.5.1.2 Big Requests Extension (BIG-REQUESTS)

BIG-REQUESTS enables a client application to extend the length field of a protocol request from 2^{18} bytes to a 32-bit value. This is useful for clients and other extensions that frequently transmit complex information to the display server.

The only callable function associated with this extension is XExtendedMaxRequestSize, which has been incorporated into Xlib. As such, it is always available when connected to an X Window system that offers this extension.

4.5.1.3 Colormap Utilization Policy Extension (TOG-CUP)

TOG-CUP provides the following colormap management capabilities to the display server:

- A mechanism for a special application (such as a colormap manager) to recognize special colormap requirements. For example, this extension enables an application to locate and initialize a default colormap.
- A policy that encourages colormap sharing and reduces colormap flashing on low-end 8-bit frame buffers.
- A behavior in the color allocation scheme that reduces colormap flashing when colormaps are not shared.

Specifically, the TOG-CUP protocol provides methods that query the server for a list of reserved colormap entries and initialize shareable colormap entries at specific locations. If the core protocol does not contain information about the returned pixel values, the AllocColor and AllocNamedColor requests look in the default colormap for a matching color. If a match is found and the corresponding cell in the private colormap is empty, the color is allocated to that cell in the private colormap rather than the first available location. This minimizes colormap flashing when the main window's default visual class is using a private colormap and is set to GrayScale, PseudoColor, or DirectColor.

Code that uses the TOG-CUP extension must include the following header files:

```
# include "DECW$INCLUDE:Xlib.h"
# include "DECW$INCLUDE:Xcup.h"
```

This extension is a fixed part of the display server and is always enabled. Call the following routine to check if the TOG-CUP extension is available on the server system:

```
Bool XcupQueryVersion (display, &major, &minor)
       Display *display,
       int     major,minor;
```

Note that client applications must call XcupQueryVersion before calling any other TOG-CUP function.

The following table lists each argument and its description.

Argument	Description
display	An input parameter that contains the current display.
major	Major version number of the extension implementation. Returned by XcupQueryVersion.
minor	Minor version number of the extension implementation. Returned by XcupQueryVersion.

4.5.1.4 Extended Visual Information Extension (EVI)

EVI enables a client to query the display server for additional information about core X visuals, such as colormap information and framebuffer levels. Note that this extension only provides support for client applications and not other X Window System extensions.

Code that uses EVI must include the following header files:

```
# include "DECW$INCLUDE:Xlib.h"
# include "DECW$INCLUDE:Xevi.h"
```

This extension is a fixed part of the display server and is always enabled. Call the following routine to check if EVI is available on the server system:

```
Bool XeviQueryExtension (
      Display *dpy
);
```

The following table lists each argument and its description.

Argument	Description
dpy	An input parameter that contains the current display.

4.5.1.5 Low-Bandwidth X Extension (LBX)

LBX is a network-transparent protocol for running X Window System applications over transport channels whose bandwidth and latency are significantly lower than that available in local area networks. LBX combines a variety of caching and reencoding techniques that reduce the volume of data sent over the network.

By using a proxy server as an intermediary between the client applications and the X server, low-bandwidth/high-latency communication is maintained between the proxy and X server. The proxy server reencodes and compresses requests, events, replies and errors, as well as the resulting data stream. Additionally, the proxy can cache information from the server to provide low-latency replies to client applications.

A proxy can serve multiple client applications and does not prevent clients from connecting directly to the server. The proxy can combine calls from multiple client applications into a single data stream.

Use of LBX is transparent to clients. The only interface to LBX available to client code is a query to check the availability of LBX. Code that uses this query must include the following header files:

```
# include "DECW$INCLUDE:Xlib.h"
# include "DECW$INCLUDE:Xlbx.h"
```

This extension is dynamically loadable at server startup; see Section 3.7.3. Call the following routines to check if LBX has been loaded and is available on the server system:

```
Bool XLbxQueryVersion(
          Display *display,
          int     *major_version_return,
          int     *minor_version_return
);
```

The following table lists each argument and its description.

Argument	Description
display	An input parameter that contains the current display.
major_version_return	Major version number of the extension implementation. Returned by XLbxQueryVersion.
minor_version_return	Minor version number of the extension implementation. Returned by XLbxQueryVersion.

4.5.1.6 Security Extension (SECURITY)

SECURITY contains a new protocol that provides for enhanced X server security. This extension adds the concepts of **trusted** and **untrusted** client connections to the X Window System protocol. The trust status of a client is determined by the authorization method used during the startup of a connection. All clients using host- or user-based authorization are considered trusted. Clients using token-based authorization protocols may be either trusted or untrusted depending on the authorization data included in the connection request.

The requests in SECURITY permit a trusted client to create multiple authorization entries related to a single authorization protocol. Each entry is tagged with a trust status, which is then associated with any client using that authorization entry.

When a connection identifying an untrusted client is accepted, the client is restricted from performing certain operations that would steal or modify data that is held by the server for trusted clients. An untrusted client performing a disallowed operation will receive protocol errors.

When a client is untrusted, the server can also limit the extensions that are available to the client. Each X protocol extension is responsible for defining what operations are permitted to untrusted clients. By default, the entire extension is hidden to untrusted clients.

With DECwindows Motif, the following extensions (standard and non-standard) are defined as secure:

BIG-REQUESTS
LBX
XC-MISC

All other extensions are considered insecure. See Section 3.3.1 for more information on how to select an appropriate authentication method and specify trusted or untrusted connections.

Code that uses SECURITY must include the following header files:

```
# include "DECW$INCLUDE:Xlib.h"
# include "DECW$INCLUDE:security.h"
```

This extension is dynamically loadable (along with the XC-APPGROUP extension) at server startup; see Section 3.7.3. Call the following routine to check if SECURITY is available on the server system:

```
Bool XSecurityQueryExtension (
       Display *dpy,
       int     *major_version_return,
       int     *minor_version_return
);
```

The following table lists each argument and its description.

Argument	Description
dpy	An input parameter that contains the current display.
major_version_return	Major version number of the extension implementation. Returned by XSecurityQueryExtension.
minor_version_return	Minor version number of the extension implementation. Returned by XSecurityQueryExtension.

4.5.1.7 XC-MISC Extension

XC-MISC allows client applications to retrieve previously-used resource ID ranges from the X server. Xlib handles this function automatically. This extension is useful for long-running applications that use many resource IDs over their runtime life.

Since the XC-MISC functions are part of Xlib, they are a standard part of the client. As such, they are are always available when connected to an X Window system that offers this extension.

4.5.1.8 X Double Buffer Extension (DBE)

DBE provides a way to display flicker-free animation on an X Window system and is intended as a replacement to the Multibuffering extension. Successive frames of an animation sequence are rendered into the back buffer while the previously rendered frame is displayed in the front buffer. When a new frame is ready, the back and front buffers swap roles, making the new frame visible. Only completely rendered frames are shown; these frames remain visible during the entire time it takes to display the new frame.

Code that uses DBE must include the following header files:

```
# include "DECW$INCLUDE:Xlib.h"
# include "DECW$INCLUDE:Xdbe.h"
```

This extension is dynamically loadable at server startup; see Section 3.7.3. Call the following routine to check if the extension has been loaded and is available on the server system:

```
Bool XdbeQueryExtension (
       Display *dpy,
       int     *major_version_return,
       int     *minor_version_return
);
```

The following table lists each argument and its description.

Argument	Description
dpy	An input parameter that contains the current display.
major_version_return	Major version number of the extension implementation. Returned by XdbeQueryExtension.
minor_version_return	Minor version number of the extension implementation. Returned by XdbeQueryExtension.

4.5.1.9 XINERAMA Extension

XINERAMA (formerly known as PanoramiX) enables a system configured with multiple video monitors (multiheaded system) to function as a single large screen. This extension allows application windows and cursor movement to span multiple screens and move from one screen to another.

The overall size of the composite screen equals the combined size of all screens. Monitor configurations can be easily modified by enabling this extension in conjunction with the associated screen symbols (such as DECW$SERVER_SCREENS). See Section 3.7.2.2 for the complete list of logicals associated with this extension; see Section 3.7.4 for instructions on how to setup and configure a multiheaded system that uses XINERAMA.

Note

This extension is only supported in a homogeneous graphics environment, which consists of common display devices, visual classes, depths, resolutions, etc. In addition, there may be some restrictions if operating in 3D mode (such as with the OpenGL layered product). See *HP DECwindows Motif for OpenVMS Alpha Release Notes* for the current restrictions regarding this extension.

4.5.1.10 X Keyboard Extension (XKB)

XKB enhances the control and customization of the keyboard under the X Window System by providing the following:

- Support for the ISO9996 standard for keyboard layouts
- Compatibility with the core X keyboard handling
- Standard methods for handling keyboard LEDs and locking modifiers
- Support for keyboard geometry

Note that all AccessX extension features for people with physical impairments have been incorporated into XKB. These accessibility features include StickyKeys, SlowKeys, BounceKeys, MouseKeys, and ToggleKeys, as well as complete control over the autorepeat delay rate.

Code that uses XKB must minimally include the following header files:

```
# include "DECW$INCLUDE:Xlib.h"
# include "DECW$INCLUDE:XKBlib.h"
```

To modify keyboard geometry descriptions, the names and identifiers of the predefined bells, or X Keyboard map definitions, additionally include the following header files:

```
# include "DECW$INCLUDE:XKBgeom.h"
# include "DECW$INCLUDE:XKBbells.h"
# include "DECW$INCLUDE:XKM.h"
# include "DECW$INCLUDE:XKMformat.h"
# include "DECW$INCLUDE:XKBfile.h"
# include "DECW$INCLUDE:XKBrules.h"
# include "DECW$INCLUDE:XKBconfig.h"
```

This extension is dynamically loadable at server startup; see Section 3.7.3. Call the following routine to check if the extension has been loaded and is available on the server system:

```
Bool XkbQueryExtension(
       Display  *dpy,
       int      *opcodeReturn,
       int      *eventBaseReturn,
       int      *errorBaseReturn,
       int      *majorReturn,
       int      *minorReturn
);
```

The following table lists each argument and its description.

Argument	Description
dpy	An input parameter that contains the current display.
opcodeReturn	Major operation code of the extension. Returned by XkbQueryExtension.
eventBaseReturn	Base event code of the extension. Returned by XkbQueryExtension.
errorBaseReturn	Base error code of the extension. Returned by XkbQueryExtension.
majorReturn	Major version number of the extension implementation. Returned by XkbQueryExtension.
minorReturn	Minor version number of the extension implementation. Returned by XkbQueryExtension.

4.5.1.11 X Synchronization Extension (SYNC)

SYNC provides primitive calls that synchronize requests from multiple clients on different hosts running different operating systems. This extension enables applications to make the best use of buffering resources within the client, server, and network and eliminates network errors that can occur when two systems are running a distributed application.

Multimedia applications can use this extension to synchronize streams of audio, video, and graphics data. For example, simple animation applications can be implemented without having to use round-trip requests.

Code that uses SYNC must include the following header files:

```
# include "DECW$INCLUDE:Xlib.h"
# include "DECW$INCLUDE:sync.h"
```

This extension is a fixed part of the display server and is always enabled. Call the following routines to check if the extension has been loaded and is available on the server system:

```
Bool XSyncQueryExtension(
       Display *dpy,
       int     *event_base_return,
       int     *error_base_return
);

Status XSyncInitialize(
       Display *dpy,
       int     *major_version_return,
       int     *minor_version_return
);
```

The following table lists each argument and its description.

Argument	Description
dpy	An input parameter that contains the current display.
event_base_ return	An output parameter that indicates the base event code for the extension.
error_base_ return	An output parameter that indicates the base error code for the extension.
major_version_ return	Major version number of the extension implementation. Returned by XSyncInitialize.

Argument	Description
minor_version_return	Minor version number of the extension implementation. Returned by XSyncInitialize.

4.5.2 Server Extensions Updated for X11R6.6

V1.3

The following existing X Window System extensions have been updated for DECwindows Motif for OpenVMS Alpha Version 1.3:

- DEC XTrap (DEC-XTRAP) and X Test (XTEST) – Now disabled with the DECW$SERVER_DISABLE_TEST parameter
- MIT Screen Saver (MIT-SCREEN-SAVER) – Updated to work with XINERAMA
- Multibuffering (MBE) – New XmbufClearBufferArea function
- MIT Miscellaneous (MIT-SUNDRY-NONSTANDARD)
- MIT Shared Memory (MIT-SHM)
- Non-Rectangular Window Shape (SHAPE)
- X Image Extension (XIE) – Supports V3.0 and not the adopted standard of V5.0

4.5.3 Inter-Client Exchange (ICE) Protocol Support

V1.3

The Inter-Client Exchange (ICE) Protocol provides support for direct X Window System client-to-client communication without using the X server. This means that DECwindows Motif client applications can use ICE rather than connect to the X server. The standard protocol provides the basic mechanisms for establishing and closing network transport connections, performing authentication, negotiating versions, and reporting errors. The protocols running within an ICE connection are known a **subprotocols**, of which Session Manager (described in Section 4.5.4) is a member.

A new client-side library, DECW$ICELIB, is provided. Code that uses ICE must include the following header file:

```
# include "DECW$INCLUDE:Icelib.h"
```

The following sections describe the implementation of ICE provided with DECwindows Motif, highlighting any variances from or restrictions posed by the standard implementation. For a detailed description of the ICE protocol and the available server requests, see the *X Window System* (Scheifler and Gettys) series of manuals published by Butterworth-Heinemann, or visit the X.Org web site (http://www.x.org) for the X Window System protocol and library specifications.

4.5.3.1 Multithreading Considerations

The ICE library supports multithreading after IceInitThreads has been called. IceInitThreads must be the first call on the ICE library if multithreading is required. Programs that call IceInitThreads must have been linked against PTHREAD$RTL.

The following sections further describe issues with using ICE functions in a multithreaded environment.

Lock Nesting

Locks held by IceLockConn and IceAppLockConn are recursive. The corresponding unlock routine must be called the same number of times as the lock routine.

Deleting IceConn Objects

IceConn objects can be deleted by:

- IceProcessMessages returning IceProcessMessagesConnectionClosed
- IceCloseConnection returning IceClosedNow

In these cases, the IceConn object is freed without validation even though locks may still be held. To avoid race conditions, ensure that the deleted IceConn object is not being used by another thread.

Non-Atomic Functions and Macros

The following subset of the ICE functions that prepare and read messages are not atomic and do not acquire locks:

IceGetHeader
IceGetHeaderExtra
IceSimpleMessage
IceErrorHeader
IceWriteData
IceWriteData16
IceWriteData32
IceSendData
IceWritePad
IceReadSimpleMessage
IceReadCompleteMessage
IceDisposeCompleteMessage
IceReadMessageHeader
IceReadData
IceReadData16
IceReadData32
IceReadPad

Any multithreaded application that uses one or more of these macros or functions must explicitly acquire a lock on the connection before creating a message, and release the lock after the message is prepared. For read operations, this action is not required since the ICE process callbacks automatically lock the connection.

For example, the following is sample code for creating a message:

```
IceAppLockConn (iceConn);
IceGetHeaderExtra (iceConn, _SmcOpcode, SM_RegisterClient,
          SIZEOF (smRegisterClientMsg), WORD64COUNT (extra),
          smRegisterClientMsg, pMsg, pData);

*((CARD32 *) pData) = len;
pData += 4;

memcpy (pData, previousId, len);
pData += (len + 3) & (~3);
IceAppUnLockConn (iceConn);
```

Since an ICE connection can be shared between protocols, every protocol must use these locks, even if the protocol can only be used by a single thread.

Opening Connections

DECwindows Motif for OpenVMS Alpha Version 1.3 restricts multithreaded applications from concurrently calling IceOpenConnection and IceCloseConnection. IceOpenConnection can accept concurrent calls to itself as long as IceCloseConnection is not called at the same time.

ICE can maintain two open connections for the same protocol by using a major opcode check to the IceOpenConnection call. Since a protocol is registered only after it calls IceProtocolSetup, a conflict can occur if two threads simultaneously establish ICE connections for the same protocol and request that the connection is not shared.

To prevent this conflict from occurring, code for opening an ICE connection with a major opcode check should follow a format similar to the following:

```
IceConn conn;
IceProtocolSetupStatus status;
while (1) {
     conn = IceOpenConnection (....)
     if (conn == 0) break;
     status = IceProtocolSetup (...)
     if (status != IceProtocolAlreadyActive) break;
     IceCloseConnection (conn);
     /* Try again as another thread set up the protocol on this connection */
}
```

4.5.3.2 Differences from the Standard Implementation

The following sections describe differences from and issues in the standard ICE implementation provided with X11R6.6.

Connection and Protocol Authentication

The implementation of ICE included with DECwindows Motif for OpenVMS Alpha Version 1.3 does not include any authentication mechanisms for ICE connections. All listen objects must use IceSetHostBasedAuthProc to register host-based authentication.

For protocol authentication, all authentication schemes provided when the protocol is registered are permitted. This differs from the standard ICE implementation, where only those those schemes defined in the ICE Authority (IceAuth) file are allowed.

Object Name Changes

The sample implementation of ICE provided by X.Org contained objects whose name differed from that described in the ICE specification. The following table lists those objects and specifies which name was used in the DECwindows Motif for OpenVMS Alpha Version 1.3 implementation of ICE.

Documented Object Name	Implemented Object Name
IceGetContext	IceConnectionGetContext
major_opcode	majorOpcode
minor_opcode	minorOpcode

IceGetHeaderExtra Structure

The header structure used with IceGetHeaderExtra must have a sizeof value that is a multiple of 8 bytes.

4.5.4 X Session Management Protocol (XSMP) Support

V1.3

The X Session Management Protocol (XSMP) provides a standard way for users to save client sessions. Each session is controlled by a network service known as the **session manager**. The session manager issues commands that direct client applications to save their state information for use during subsequent sessions.

This protocol is built on top of ICE, which manages the client connections to the session manager server.

Code that uses XSMP must include the following header files:

```
# include "DECW$INCLUDE:SM.h"
# include "DECW$INCLUDE:SMlib.h"
# include "DECW$INCLUDE:SMproto.h"
```

The following sections describe the implementation of XSMP provided with DECwindows Motif for OpenVMS Alpha Version 1.3, highlighting any variances from or restrictions posed by the standard implementation. For a detailed description of the XSMP protocol and the available server requests, see the *X Window System* (Scheifler and Gettys) series of manuals published by Butterworth-Heinemann, or visit the X.Org web site (http://www.x.org) for the X Window System protocol and library specifications.

4.5.4.1 Multithreading Considerations

The implementation of XSMP is thread safe, using locks on the underlying ICE connection as needed. All send message operations are thread cancellation points; all callback operations are made by locking the associated ICE connection.

When SmcOpenConnection is called, it opens an ICE connection and processes messages until the session manager registers the client. The open connection subsequently causes a series of ICE watch procedures to be called. Typically, these procedures add the connection to a list monitored for input. IceProcessMessages is called when input to the list arrives. The thread issuing the IceProcessMessages calls will be blocked if it tries to handle a new connection.

4.5.4.2 Differences from the Standard Implementation

The following sections describe differences from and issues in the standard XSMP implementation provided with X11R6.6.

SmcCloseConnection and SmsCleanUp

In the standard implementation, SmcCloseConnection disables shutdown negotiation for an ICE connection, which results in abrupt termination of the connection. This can prevent the session manager from receiving all SmCloseConnection messages.

In the DECwindows Motif implementation, shutdown negotiation is enabled. SmcCloseConnection returns SmcClosedASAP, and the connection is closed only after the session manager calls SmsCleanUp.

Note also that the sample session manager code does not specify whether SmsCleanUp closes an ICE Connection. In the DECwindows Motif implementation, an IceCloseConnection call is issued.

POSIX Property Names and Data Type Definitions

The standard specification defines the data types and property names supported for POSIX. The DECwindows Motif implementation specifies the property names; however, the data types definitions are not provided, since they may vary based on the use of session manager in the OpenVMS Alpha environment.

4.5.5 MIT Shared Memory Extension (MIT-SHM) Support

V1.2

On OpenVMS Alpha systems, shared memory extension support provides the capability to share memory XImages. This is a version of the XImage interface where the actual image data is stored in a shared-memory segment. Consequently, the image does not need to be moved through the Xlib interprocess communication channel. For large images, use of this extension can result in dramatic performance increases.

Support for shared memory pixmaps is also provided. Shared memory pixmaps are two-dimensional arrays of pixels in a format specified by the X server, where the image data is stored in the shared memory segment. Through the use of shared memory pixmaps, you can change the contents of these pixmaps without using any Xlib routines.

These routines are included in the client side extension library. See Section 4.3.8 for details on linking this library.

4.5.5.1 How to Use Shared Memory Extension

Code that uses the shared memory extension must include the following header files:

```
# include "DECW$INCLUDE:Xlib.h"
# include "DECW$INCLUDE:shm.h"
# include "DECW$INCLUDE:XShm.h"
```

Any code that uses the shared memory extension should first check that the server provides the extension. In some cases, such as running over the network, the extension does not work.

To check if the shared memory extension is available on your system, call one of the following routines:

```
Status XShmQueryExtension (display)
       Display *display

Status XShmQueryVersion (display, major, minor, pixmaps)
       Display *display;
       int *major, *minor;
       Bool *pixmaps
```

The following table lists each argument and its description.

Argument	Description
display	The current display. If the shared memory extension is used, the return value from either function is True. Otherwise, your program operates using conventional Xlib calls.
major	Major version number of the extension implementation. Returned by XShmQueryVersion.
minor	Minor version number of the extension implementation. Returned by XShmQueryVersion.
pixmaps	True, indicates that shared memory pixmaps are supported.

4.5.5.2 Using Shared Memory XImages

The following sequence shows the process for creating and using shared memory XImages:

1. Create the shared memory XImage structure.
2. Create a shared memory segment to store the image data.
3. Attach the shared memory segment.
4. Inform the server about the shared memory segment.
5. Use the shared memory XImage.

The following sections explain each step in this process:

Step 1—Creating a Shared Memory XImage Structure

To create a shared memory XImage, use the XShmCreateImage routine, which has the following format:

```
XImage *XShmCreateImage (display, visual, depth, format, data,
            shminfo, width, height)
       Display *display;
       Visual *visual;
       unsigned int depth, width, height;
       int format;
       char *data;
       XShmSegmentInfo *shminfo;
```

Most of the arguments are the same as for XCreateImage (See the *X Window System* for a description of the XCreateImage routine.) Note that there are no *offset*, *bitmap_pad*, or *bytes_per_line* arguments. These quantities are set by the server, and your code needs to abide by them. Unless you have already allocated the shared memory segment (see step 2), you pass in NULL for the *data* pointer.

The argument *shminfo* is a pointer to a structure of type XShmSegmentInfo. Allocate one of these structures so that it has a lifetime at least as long as that of the shared memory XImage. There is no need to initialize this structure before the call to XShmCreateImage.

If successful, an XImage structure is returned, which you can use for the subsequent steps.

Step 2—Creating the Shared Memory Segment

Create the shared memory segment after the creation of the XImage because the XImage returns information that indicates how much memory to allocate.

The following example illustrates how to create the segment:

```
shminfo.shmid = shmget (IPC_PRIVATE,
        image->bytes_per_line * image->height, IPC_CREAT|0777);
```

This example assumes that you called your shared memory XImage structure. A return value of 0 indicates the shared memory allocation has failed. Use the *bytes_per_line* field, not the *width* you used to create the XImage, as they may be different.

Note that the shared memory ID returned by the system is stored in the shminfo structure. The server needs that ID to attach itself to the segment.

Step 3—Attaching the Shared Memory Segment

Attach the shared memory segment to your process as in the following example:

```
shminfo.shmaddr = image->data = shmat (shminfo.shmid, 0, 0);
```

The address returned by shmat is stored in *both* the XImage structure and the shminfo structure.

To finish supplying arguments in the shminfo structure, decide how you want the server to attach to the shared memory segment, and set the *shminfo.readOnly* field as follows:

```
shminfo.readOnly = False;
```

If you set the structure to True, the server cannot write to this segment, and XShmGetImage calls fail.

Note

The shared memory segment routines are provided with DECwindows Motif. Using global sections, these routines emulate the shared memory routines on UNIX systems.

Step 4—Informing the Server About the Shared Memory Segment

Tell the server to attach to your shared memory segment as in the following example:

```
Status XShmAttach (display, shminfo);
```

If successful, a nonzero status is returned, and your XImage is ready for use.

Step 5—Using the Shared Memory XImage

To write a shared memory XImage into an X drawable, use the XShmPutImage routine. The XShmPutImage routine uses the following format:

```
XShmPutImage (display, d, gc, image, src_x, src_y,
                    dest_x, dest_y, width, height, send_event)
       Display *display;
       Drawable d;
       GC gc;
       XImage *image;
       int src_x, src_y, dest_x, dest_y;
       unsigned int width, height;
       Bool send_event;
```

The interface is identical to the XPutImage routine (see the *X Window System*), except for one additional parameter, *send_event*. If this parameter is passed as True, the server generates a completion event when the image write is complete. This allows your program to know when it is safe to begin manipulating the shared memory segment again.

The completion event is of the type XShmCompletionEvent, which is defined as follows:

```
typedef struct {
    inttype;              /* of event */
    unsigned long serial; /* # of last request processed */
    Bool send_event;      /* true if came from a SendEvent request */
    Display *display;     /* Display the event was read from */
    Drawable drawable;    /* drawable of request */
    int major_code;       /* ShmReqCode */
    int minor_code;       /* X_ShmPutImage */
    ShmSeg shmseg;        /* the ShmSeg used in the request */
    unsigned long offset; /* the offset into ShmSeg used */
} XShmCompletionEvent;
```

To determine the event type value that is used at run time, use the XShmGetEventBase routine as in the following example:

```
int CompletionType = XShmGetEventBase (display) + ShmCompletion;
```

Note

If you modify the shared memory segment before the arrival of the completion event, the results may be inconsistent.

To read image data into a shared memory XImage, use the XShmGetImage routine, which uses the following format:

```
Status XShmGetImage (display, d, image, x, y, plane_mask)
       Display *display;
       Drawable d;
       XImage *image;
       int x, y;
       unsigned long plane_mask;
```

The following table lists each argument and its description.

Argument	Description
display	The display of interest.
d	The source drawable.
image	The destination XImage.
x	X-offset within the source drawable.
y	Y-offset within the source drawable.
plane_mask	The planes that are to be read.

To destroy a shared memory XImage, first instruct the server to detach from it, then destroy the segment itself. The following example illustrates how to destroy a shared memory XImage:

```
XShmDetach (display, shminfo);
XDestroyImage (image);
shmdt (shminfo.shmaddr);
shmctl (shminfo.shmid, IPC_RMID, 0);
```

4.5.5.3 Using Shared Memory Pixmaps

Unlike X images, for which any image format is usable, the shared memory extension supports only a single format for the data stored in a shared memory pixmap (XYPixmap or ZPixmap). This format is independent of the depth of the image and independent of the screen. (For 1-bit pixmaps the format is irrelevant.)

The XShmPixmapFormat routine returns the shared memory pixmap format for the server. The XShmPixmapFormat routine has the following format:

```
int XShmPixmapFormat (display)
       Display *display;
```

Your application can only use shared memory pixmaps in the format returned by the XShmPixmapFormat routine (including bits-per-pixel). To create a shared memory pixmap do the following:

- Create a shared memory segment and shminfo structure exactly the same way as is listed for shared memory XImages steps 1 through 4 (see Section 4.5.5.2). While it is not necessary to create an XImage first (step 1), doing so incurs little overhead and provides an appropriate bytes_per_line value to use.
- Call the XShmCreatePixmap routine, which has the following format:

  ```
  Pixmap XShmCreatePixmap (display, d, data, shminfo, width,
                           height, depth);
         Display *display;
         Drawable d;
         char *data;
         XShmSegmentInfo *shminfo;
         unsigned int width, height, depth;
  ```

  The arguments are the same as for XCreatePixmap (see the *X Window System*) except for two additional parameters, *data* and *shminfo*. The *data* parameter is the pointer to the shared memory segment and is the same as the shminfo.shmaddr field. The *shminfo* parameter is the same as the previous structure.

If successful, a pixmap is returned, which you can manipulate. You can manipulate its contents directly through the shared memory segment. Shared memory pixmaps are destroyed with the XFreePixmap routine, although you should detach and destroy the shared memory segment (see step 4 in Section 4.5.5.2).

4.5.6 X Image Extension (XIE) Support

V1.1

Starting with DECwindows Motif for OpenVMS Version 1.1, DECwindows Motif supports the X Image Extension (XIE). XIE allows image display processing using resources on the server side of the X client-server model. XIE eliminates the need to transmit image data repeatedly from the client to the server and also allows data to be transmitted in compressed form, reducing the network load.

DECwindows Motif includes the XIE client side shareable library (XIE$SHRLIB.EXE) and C language header files. These allow applications to communicate with any X11 server that supports the XIE extension.

An XIE program uses a structure called the XIEImage to describe image data on the client side. This general mechanism describes data that the destination server is incapable of processing. Consult the documentation for the server system for information on what data types and sizes are supported. Unless the documentation specifies different limits, the server is capable of processing unsigned byte (UdpK_DTypeBU), unaligned bit field (UdpK_DTypeVU), and aligned bit field (UdpK_DTypeV) data, with a maximum depth of 8 bits per pixel per component. The XIE client library supports these data types, as well as unsigned word (UdpK_DTypeWU), and a depth of up to 16 bits per pixel per component.

Although the XIE protocol and programming interface have been standardized for X11R6, DECwindows Motif has not yet migrated to the latest implementation of this protocol.

4.6 DECwindows Extensions to Motif

The following sections describe features related to DECwindows extensions to the X Window System.

4.6.1 SVN Widget Supports Extended Selection

V1.2–6

The Structured Visual Navigation (SVN) widget now allows users to extend a range of selection using the Shift+Down-Arrow key sequence. Note that this change has also been applied to the sample program SVNMSAMPLE.C.

4.6.2 DXmCSText Input Method Support

V1.2

X11R5 input method support was added to the DXmCSText widget. Specify input methods using the vendor shell XmNinputMethod resource. However, to maintain backward compatibility, the existing input method resources DXmNinputMethod and DXmNinputMethodType are still available.

4.7 Application Programming

The following sections describe features related to application programming.

4.7.1 Drag-and-Drop Enabled Widgets

V1.2

The drag-and-drop feature lets you move or copy information between widgets. This feature is provided primarily for programmers to incorporate the feature into their applications.

All DECwindows Motif for OpenVMS Version 1.2 and higher applications support the drag-and-drop feature, with the exception of Notepad. DECwindows Mail supports drag-and-drop in all windows except the main message area, where DECwindows Mail has its own drag-and-drop feature; you can use MB2 to move messages around with the SVN interface.

Drag-and-drop functionality has been implemented in the widgets listed in Table 4–11.

Table 4–11 Drag-and-Drop Widgets

Widget	Drag Operation	Drop Operation
XmText	copy and move	copy and move
XmTextField	copy and move	copy and move
XmLabel	copy	
XmPushButton	copy	
XmToggleButton	copy	
XmList	copy	

For information about how to include additional drag-and-drop functionality in applications and for an example of a drag-and-drop program, see the *Open Software Foundation: OSF/Motif Programmer's Guide, Revision 1.2* manual.

4.7.2 CDA Programming

This section describes features and changes related to CDA programming.

4.7.2.1 Changes to the CDA Programming Interface

V1.2

This section describes the changes to the programming interface for this version of CDA Run-Time Services.

This version provides a new set of header files that define CDA constants, types, and routines using portable naming conventions. By using these new naming conventions, you can use a wider variety of C compilers to minimize the amount of system-specific code in your CDA applications.

The names of the new set of header files are the same as the names of the previous set of header files, except that the dollar sign ($) has been removed. For example, the cda$msg.h include file is now called cdamsg.h. Other examples include the following: The DDIF$K_DSC_MAJOR_VERSION symbol is now declared as DDIF_K_DSC_MAJOR_VERSION, and the CDA$_NORMAL status value is now defined as CDA_NORMAL.

The previous set of header files is also included in this version, but these files will no longer be updated. Changes introduced since the release of DECwindows Motif Version 1.1 (for example, the new definitions for audio support), are available only in the new set of header files. To use the new CDA features, change the file names in your source code.

The new set of header files supplements the previous set of header files. If you want to write ANSI-compliant applications using CDA definitions and CDA Toolkit calls, use the new set of header files. However, you can continue to use the header files that define symbols containing the dollar sign ($) provided you choose a non-ANSI compilation mode.

By using the previous set of header files, you can successfully build existing source code that uses the previous naming conventions.

See Table 4–12 for a list of new header file names.

Table 4–12 New Header File Names

Previous Name	New Name
cda$def.h	cdadef.h
cda$msg.h	cdamsg.h
ddif$def.h	ddifdef.h
dtif$def.h	dtifdef.h
cda$ptp.h	cdaptp.h
cda$typ.h	cdatyp.h
dvr$msg.h	dvrmsg.h
dvr$cc_def.h	dvrccdef.h
dvr$cc_ptp.h	dvrccptp.h
dvr$decw_def.h	dvrwdef.h
dvr$decw_ptp.h	dvrwptp.h

4.7.2.2 Changes to CDA External Reference Processing

V1.2

CDA Run-Time Services supports relative file specifications for external references. Relative references are also supported: that is, a reference where the directory path is not fully specified but is relative to the directory path of the parent document.

4.7.2.3 Restructuring CDA Shareable Images

V1.2

The CDA Viewer includes two shareable images to allow installation on systems where DECwindows is not installed.

In DECwindows Motif for OpenVMS Version 1.2 software, shareable images that use the X services were renamed. Table 4–13 lists the shareable images in the various versions of the CDA Viewer.

Table 4–13 Names of Shareable Images

CDA Version	Image Name	Description
Version 1.6	CDA$ACCESS	CDA Run-Time Services shareable image.
	DDIF$VIEWSHR	Callable viewer widget.
Version 1.7	CDA$ACCESS	CDA Run-Time Services shareable image.
	DDIF$VIEWSHR DDIF$DECW_ VIEWSHR DDIF$CC_VIEWSHR	The DDIF$DECW_VIEWSHR widget is a callable viewer widget that uses the LIB$FIND_IMAGE_SYMBOL routine to invoke the DDIF$DECW_VIEWSHR (DECwindows interface) and DDIF$CC_ VIEWSHR (character-cell interface) widgets.
Version 1.8A or later	CDA$ACCESS	CDA Run-Time Services shareable image.
	DDIF$VIEWSHR12	The DDIF$VIEWSHR12 widget is a callable viewer widget that uses LIB$FIND_IMAGE_SYMBOL to invoke the DDIF$DECW_VIEWSHR12 (DECwindows interface) and DDIF$CC_VIEWSHR (character-cell interface) widgets.

By using the LIB$FIND_IMAGE_SYMBOL routine to reference the entry points to the DDIF$DECW_VIEWSHR, DDIF$DECW_VIEWSHR12, and DDIF$CC_ VIEWSHR images, an application can dynamically determine whether it can execute in a given environment. The DDIF$VIEW.EXE application now replaces this routine.

The previous DDIF$VIEWSHR.EXE shareable image is still included to maintain compatibility with applications linked against it. However, new applications (and previous applications that take advantage of new features) should use the new shareable images.

4.7.3 DECterm Programming

This section describes features and changes related to DECterm programming.

4.7.3.1 ReGIS Input Cursors

V1.2–3

DECterm supports the following input cursors: cross-hair, rubber-band line, diamond, and rubber-band rectangle. To select input cursors use the S(C(I*n*)) command. Table 4–14 shows the values of *n*.

Table 4–14 ReGIS Input Cursors—Cursor styles and Values

Cursor Style	Variable *n*
Cross-hair	Omitted
Cross-hair (default)	0
Diamond	1
Cross-hair	2
Rubber-band line	3
Rubber-band rectangle	4

Note

If a shape other than the diamond cursor is desired when *n* is equal to 1, define the logical name DECW$DECTERM_REGIS_CURSOR to be one of the numbers defined in the SYS$LIBRARY:DECW$CURSOR.H file.

4.7.3.2 Page-Movement Escape Sequences

V1.2

The following page-movement escape sequences are implemented in DECterm:

```
NP      <CSI> Pn U      Next Page
PP      <CSI> Pn V      Previous Page
PPA     <CSI> Pn P      Page Position Absolute
PPB     <CSI> Pn R      Page Position Backward
PPR     <CSI> Pn Q      Page Position Relative
```

Note that "Pn" is the number of pages to move; the exception is PPA, where "Pn" is the actual page number.

Note

DECterm does not support cursor coupling; the cursor is always bound to the current (displayed) page.

Index

E

F

G

H

I

K

L

M

N

O