Release Notice: X11R5 DG/UX™ X Window System™ for AViiON® Systems

System 5.4 Release 3.10

August 1994

Part number 085-600421-00

This release notice applies to the following model:

P001A

Release Notice:
X11R5
DG/UX™ X Window System™ for
AViiON® Systems
System 5.4 Release 3.10

085-600421-00

August 1994

Contents

1 Introduction

This Release Notice describes the DG/UX™ X Window System™ Revision 11 release 5 and its installation. It also includes information not currently available in the product manual, such as the product directory tree.

Between releases, Data General may issue updates to this product. An update is essentially a partial release. This mechanism reduces the time needed to fix problems by providing a level of correction short of releasing the complete product. Each update of a product supersedes the previous update.

Additional copies of this release notice can be printed. Use the file **/usr/opt/X11/release/X11_5.4R3.10.rn** on the release tape. If the on-line file and the hardcopy version of the release notice differ, the hardcopy notice takes precedence.

2 Product Description

The DG/UX X Windows System combines the X Window System Release 5 and Motif™ Version 1.2.2. The X Window System consists of an X server, a set of X clients, programming libraries, and on-line documentation. The X server manages a raster display and related input devices. X clients are applications that communicate with the X server to perform terminal emulation, window management, and other necessary functions. The subroutine libraries allow you to implement applications of your own.

Motif consists of the **mwm** window manager, a User Interface Language (uil) compiler, and subroutine libraries. Motif is a derivative product of the Open Software Foundation, Inc.'s OSF/Motif™.

3 Environment

3.1 Hardware

The DG/UX X Window System is based on a client/server model. This model allows you to run the X Window System in two distinct environments, with the X client and X server running on either the same machine or on different machines connected by a Local Area Network (LAN). In the first environment, where the X client and the X server are running on the same machine, they communicate using Unix sockets or shared memory. In this situation, the minimum configuration for an AViiON® workstation is a keyboard, mouse, raster display, disk and tape or fileserver.

Data General recommends a minimum of 16 MB for running X, and more if you use many graphical applications. Memory and CPU requirements are the same as those for the DG/UX system. Consult the DG/UX System 5.4 Release 3.10 Release Notice for these values.

In the second environment, the server and clients execute on different machines connected by a Local Area Network (LAN). The machine running the server must have, at a minimum, the standard AViiON workstation configuration. The machine running the X client can be any CPU connected to the AViiON workstation by the LAN.

3.2 Software

To determine environmental requirements, consult the DG/UX System 5.4 Release 3.10 Release Notice.

The remote machine presenting the displays generated by the DG/UX Window System must be equipped with support for TCP/IP LAN as well as an X server that supports the version 11 protocol.

4 Enhancements and Changes

4.1 Enhancements

4.1.1 mterm

The **mterm** command was enhanced to support an async menu option. This menu lets you use direct async line connections to change baud rate, parity, and stop bits "on the fly."

Since Release 5.4, UUCP builds the lock file for async usage using the following template:

```
/usr/spool/locks/LK.xxx.yyy.zzz
```

mterm now uses the same lock file template.

For more information, see the **cu**(1C) manual page.

Async lines used by UUCP are recognized as in use only by other programs building lock files the same way.

You can now select Binary or Normal logging without exiting and restarting **mterm**.

4.1.2 xresedit

The **xresedit** command was enhanced to display resource values more quickly. **xresedit** also displays explanations for Class and Instance.

4.1.3 xdm

By default, **xdm** now displays the host name in the login window.

4.2 Changes

4.2.1 Xdgmail

Xdgmail now saves files with the user's default groupid instead of "mail".

4.2.2 Motif 1.2.2

Version 1.2.2 of Motif, an update to version 1.2, is included in this update. Motif 1.2.2 contains only bug fixes and no new functionality.

4.2.3 mterm

The **mterm** fonts are delivered in both **.snf** and **.pcf** formats to correct for font problems experienced by some customers.

5 Notes and Warnings

5.1 Notes

5.1.1 X11R5 Release Organization

X11 packages were rearranged as follows into "run-time" and "development" environments. This change allows you to tailor your installation.

X11	Base X Window System package, containing executables, shared libraries, fonts, app-defaults, and bitmaps required for a running X Window System.
X11.man	X Window System runtime manual pages package, containing X and Motif man1 and man5 manual pages.
X11.sde	X Software Development Environment package, containing libraries, header files, and documentation needed to build X clients as well as customized X servers.
X11.sde.man	X Software Development Environment manual pages package, containing X and Motif man3 manual pages.

5.1.2 Font Server

A font server is included with X11R5. The font server is a separate process that communicates with the X server and allows multiple hosts to access a common set of fonts. It may be run, for example, on a file server with a lot of disk space available for font files. Other hosts may connect to the font server and request fonts that, due to local space limitations, cannot exist on their system. The font server can also perform font scaling calculations for its clients. This removes the computation burden from the local host resulting in an overall performance gain.

The font server is typically invoked from a system initialization file like **/etc/inittab** or a script under the **/etc/init.d** directory. See the **fs(1)** manual page for more detail.

To connect to the font server from another system, set the font path as follows:

```
   xset +fp tcp/hostname:7000
```

where hostname is the name of the host on which the font server is running.

In order to effectively run the font server, you must be **root**. Otherwise, the font server cannot open its error file (**fs-error**). If the error file does not exist, the font server reports its problems to the invoking console. For more information, see the **fs** (1) manual page.

5.1.3 Scalable Fonts

Scalable bitmap fonts and scalable outline fonts (Speedo fonts) are available in X11R5. To scale these fonts, enter values for the desired font's point size as well as **xdpi** and **ydpi** (the number of pixels per inch in the X and Y directions) in the font name and use the font name as normal. For example:

```
xterm -fn -bitstream-courier-medium-r-normal-*-17-0-75-75-m-0-iso8859-1
```

gives you a 17-point Speedo scaled font in an **xterm**.

5.1.4 Font Tools

Several font tools are available. These clients (except for **mkfontdir** and **bdftopcf**) work with the font server (see the "Font Server" section). In order to locate the correct font server, these tools expect the environment variable **FONTSERVER** to be set to **hostname:7000**. You can also set the **hostname:7000** value via a command line argument. For more details, see the associated manual pages for each client.

The font tools are as follows:

bdftopcf	Compiles fonts to **pcf** format; similar to **bdftosnf**.
mkfontdir	Same as in R4, but supports Speedo fonts.
showfont	Views font stats and glyphs on any terminal; similar to **showsnf**.
fsinfo	Finds out vital stats for a given font server.
fslsfonts	Lists fonts available on a font server.
fstobdf	Generates a **bdf** format font from any font server font.

In addition, the following R4 font tools are supported:

bdftosnf	Compiles fonts to **snf** format.
showsnf	Views font stats and glyphs of **snf** fonts.

The **snf** font format was replaced by MIT with the Portable Compiled Font (**pcf**) format. This format is portable across architectures, which was not true of the **snf** fonts. As a result of this change, the **snf** font tools will only be supported for a limited time. For more information, see the "PCF and SNF Font Support" section under "Notes".

5.1.5 Internationalization Support

Many routines were added to **libX11** and **libXt** to aid in programming internationalized X clients. These include simple "locale" management functions such as the following

- routines for internationalized text drawing
- Functions to support internationalized window manager and text properties
- Resource manager functions to support localized databases.

NOTE: Supported X clients have not incorporated any internationalization support.

5.1.6 X Color Management System (Xcms)

Until X11R5, color support was device-dependent; RGB values did not necessarily result in the same color from one display to another. R5 includes device-independent color with the X Color Management System (**Xcms**). **Xcms** functions were added to **Xlib** to support device-independent color spaces derivable from CIE XYZ color space (including CIE XYZ, xyY, L*u*v*, L*a*b*, and TekHVC). Monitors are characterized by data stored on **root** window properties. You can use the new program **xcmsdb** to set these properties. The default location of the "screen characterization database" loaded by **xcmsdb** is **/usr/lib/X11/dg6487.dcc**. A sample file is included with this release. However, even with this sample file, you will not get true device-independent color because physical characteristics of monitors change over time. Therefore, for accurate color reproduction, you should regularly calibrate your monitor and generate a similar file.

In addition, you can access client-side color name databases similar to **rgb.txt**, with the **Xcms** functions in **Xlib**. The default location for this file is **/usr/opt/X11/Xcms.txt**, although you can specify a different file in the **XCMSDB** environment variable.

5.1.7 Athena Widget Set

The R5 version of Athena widgets (as supplied by Data General) changed significantly from R4. R5 supports both the normal Athena "look and feel" (LAF) and the Motif™ LAF. You can select the LAF that a client (linked with the R5 **Xaw** library) uses at run time using the "lookAndFeel" resource. The three possible values for the resource are "athena", "motif", and "WindowManager". If "WindowManager" is specified and **mwm** is running when the client is invoked, the default LAF is "motif"; otherwise, it is "athena". Add one of the following lines to your **.Xdefaults** file to select a "look and feel":

```
*lookAndFeel: WindowManager
*lookAndFeel: athena
*lookAndFeel: motif
```

NOTE: The Motif LAF works only with Athena-based clients that are set up to use it. All Data General-supplied Athena-based X clients support the Motif LAF.

5.1.8 Dynamic Extensions

This release of X11R5 has a dynamic extension to the X server, the Input Extension (see below). Dynamic extensions are libraries of functions that are "linked" into the base package at runtime. For example, by default the X server does not provide the Input Extension to X. However, if you specify a command line option at server initialization the input extension is made available. The dynamic extension facility allows you to customize your X server based on local requirements.

If you want your site to use the input extension, specify it in the file **/var/X11/xdm/Xservers**. This file contains the information necessary to start the X System. For a complete discussion of the **Xservers** file format, refer to the **xdm(1X)** manual page.

The XINPUT extension can be supported independently. That is, you can run the X server with the extension turned on or off.

For example, specify

```
X -x XINPUT
```

to start the server with input extension support.

Specify

```
X
```

to start the server without the extension.

5.1.9 X Input Extension

To enable additional input devices via the X Input Extension, specify either of the following:

- **–device**<*device-spec*> ↵
- **–devicefile**<*filename*> ↵

The filename must be an ASCII file with one <*device-spec*> per line. A <*device-spec*> appears similar to one of the following:

- **geoball,dev=tty01,baud=9600**
- **tablet,dev=/dev/tty00,type=pointer**

A <*device-spec*> is a device handler name followed by one or more of the following, all separated by spaces, tabs and/or commas:

- **dev=**<*dev-path*>

  specifies the special device file to be opened. The */dev/* prefix on <*dev-path*> is optional.

- **baud=**<*baud*>

specifies the baud rate of the device. This may be ignored by some handlers (for example, the keyboard).

- **name=<*dev-name*>**

  is the name of the device used in the X11 Input Extension. This name is bound to an atom, and is used to identify extension devices. If it is not specified, it defaults to the handler name, except in uppercase characters.

- type=<*type*>

  specifies whether the device is to be the core pointer, core keyboard, or an extension device. The valid values for <*type*> are **pointer**, **keyboard**, **extensiondevice**, or **other** (the last two values are equivalent). You can abbreviate <*type*> to the first letter; the default is **extensiondevice.**

Two sample device handlers provided (in source code form) with the server are **tablet** and **geoball.** The tablet handler supports the Wacom SD-320 in Bit Pad II emulation mode. The geoball handler supports the Geoball by CIS Graphics, Inc. Additional handlers can be written and linked into the server using these two handlers as examples.

The X input extension is supported in X11R5, providing support for two additional input devices: "The Geometry Ball" manufactured by CIS Graphics Inc., and the WACOM graphics tablet manufactured by WACOM Inc. It has been provided as a shared object that the X server can conditionally load if you request the X input extension support. To do so, specify the XINPUT extension via the X server command line option **-x** in the file **/var/X11/xdm/Xservers**, as described above. The command portion of the line is similar to the following:

```
X /-x XINPUT
```

5.1.10 Xsess

Xsess is a newly developed screen manager designed to provide "rooms" functionality to X.desktop™. This client lets you logically group your X applications into rooms. Rooms contain related applications whose use you may desire at any given time. You can work with different groups of applications by simply changing from one room to another. Use **xsess** to promote efficient workspace organization and reduces screen clutter. For more information, consult the **xsess**(1) manual page.

LibXsess.so.1 is a shared library used by X.desktop that provides the communication layer between X.desktop and the **xsess** client.

5.1.11 Key Toggling

The manner in which the X server handles toggling keys was modified to correct an LED synchronization problem.

When an event arrives from the keyboard, the first determination made is whether the key should toggle. The key is considered a toggling key if one of the following conditions is true:

- The modifier mask for the key has the LockMask bit turned on.
- The key has one of the following KeySyms associated with it:
  - XK_Caps_Lock
  - XK_Shift_Lock
  - XK_Num_Lock
  - XK_Scroll_Lock
  - XK_Kana_Lock
- It is a Kanji keyboard in "Kana key is modifier" mode and the key has the XK_Mode_switch KeySym associated with it.
- If it is determined that the key should toggle, one of the following events occurs:
  - Key up events are discarded.
  - If the key has an LED, a KeyPress event is generated if the LED just switched on. A KeyRelease event is generated if the LED just switched off.
  - Otherwise, if the last event for this key was KeyPress, a KeyRelease event is generated. If the last event was KeyRelease, a KeyPress is generated.

The major change in the X server's behavior is that the lighted keys (Caps Lock, Num Lock, Scroll Lock and Kana Lock) will toggle if they have a "locking" KeySym, or the LockMask bit is turned on in the modifier mask. As a result, the LED on the key can no longer become out of sync with the logical state of the key, as it could have in previous X servers.

5.1.12 Keyboard Support

Keyboards for languages other than US English are handled differently in R5 (and R4) than they were in X11R3. X11R5 (and R4) use the new Mode_Shift feature to handle Alt key sequences. As a result, many applications built with X11R3 libraries will not interpret keyboard input correctly with foreign keyboards. To correct this problem, re-link the applications with the new libraries.

5.1.13 SHAPE and MIT_SUNDRY_NONSTANDARD Extensions

The MIT SHAPE and MIT_SUNDRY_NONSTANDARD extensions are supported in the extensions library **libXext**.

5.1.14 Shared Memory Communications Transport (DGShm)

X11R5 includes an implementation of shared memory transport for client/server communication when both the client and server are executing on the same computer. This feature provides performance improvements of up to 100% for many X protocol requests. For details on using the shared memory transport, consult the **XDGShm**(5) manual page.

5.1.15 Run-time Shared Objects

The following run-time shared objects are included in this release:

- **libX11.so.2**
- **libXt.so.2**
- **libXaw.so.2**
- **libXaw.so.1**
- **libXmu.so.2**
- **libXext.so.2**
- **libXi.so.2**.L1 **libXm.so.2**
- **libXsess.so.1**

A run-time shared object is a library loaded at run-time and shared by all the clients that use that library and that are linked dynamically. The following libraries are symbolic links to their version 2 counterparts:

- **libX11.so.1**
- **libXt.so.1**
- **libXm.so.1**.

The **libXaw.so.1** library is not a symbolic link because of incompatibilities introduced by MIT's X11R4 and X11R5 versions of the Athena widget set. The **.so.2** libraries implement both the X11R4 and X11R5 library calls (except for **libXaw.so.2**, which implements only X11R5); therefore, the **.so.1** can be a link to the **.so.2**.

New "skeleton" shared libraries for the X libraries are included in X11R5 to provide better standards compliance and to prevent the use of nonpublicly-available functions by applications. These include **libX11.so, libXt.so**, **libXaw.so**, **libXmu.so**, **libXext.so** and **libXi.so**. This may cause some applications that could be built with X11R4 to fail to link properly in X11R5. An application that fails to link is probably using a nonpublic function in the X libraries. Upward compatibility cannot be guaranteed when nonpublic functions are used by an application that uses shared libraries. However, if a nonpublic function is needed, the application can be linked statically.

In X11R5, **libXmu**, **libXext**, and **libXi** are provided in both static and dynamic library form. The dynamic libraries are **libXmu.so.2**, **libXext.so.2**, and **libXi.so.2**.

5.1.16 PCF and SNF Font Support

X11R5 uses Portable Compiled Fonts (**pcf**). These fonts are portable across most platforms. R5 continues support for Server Normal Fonts (**snf**) to provide backward compatibility.

Both **pcf** and **snf** versions of all the fonts are delivered in R5. However, **pcf** fonts are now the default fonts. **mkfontdir** chooses the **pcf** versions over **snf** unless you specify the **-r4** switch. With the **-r4** switch, **mkfontdir** only recognizes X11R4-compatible fonts. The switch is provided to ease the migration from X11R4 to R5; its support will be discontinued in a future release.

For example, suppose your site is upgrading its X server to R5, but you use R4 X Terminals. Because the X Terminals do not know about **pcf** fonts, the server must produce **snf** fonts. To force the server to use only **snf** fonts, issue the following command:

```
mkfontdir -r4 /usr/lib/X11/fonts /usr/lib/X11/fonts/misc/100dpi \
/usr/lib/X11/fonts/misc/75dpi
```

NOTE: For related X terminal information, see the section "AVX-30 X Terminal Support".

5.1.17 Home and End Key Interpretation in xterm

In DG/UX Release 5.4, **xterm** incorrectly interpreted the Home and End keys. Because the terminal **xterm** is emulating, a VT102, does not have either of the keys, **xterm** was changed so that it does not generate any escape sequences for these keys. You can override this default behavior with the "translations" resource. For details, see the **xterm**(1X) manual page.

5.1.18 mwm WM_SAVE_YOURSELF Protocol

With Motif 1.2 and later releases, **mwm** handles the WM_SAVE_YOURSELF protocol independently from the WM_DELETE_WINDOW protocol. A client expressing interest in both protocols is notified for each (first WM_DELETE_WINDOW, then WM_SAVE_YOURSELF) when you invoke **f.kill** on a window. A client expressing interest in neither protocol has its connection to the X server terminated. After **mwm** sends a WM_SAVE_YOURSELF client-message, it sets a timeout, the value of which is specified by **mwm**'s quitTimeout resource. The client's connection to the server is terminated when the timeout expires or when the client updates its WM_COMMAND property.

5.1.19 /usr/opt/X11 Structure

In release 5.4, the directories **/usr/bin/X11**, **/usr/lib/X11**, and **/usr/include/X11** were symbolic links to directories in the **/usr/opt/X11** file system. This caused problems when installing third-party software, for the following reasons:

- When **/usr/opt/X11** is remote-mounted, installation of third-party X11 software fails. Software installation is performed by the user **root**. Unfortunately, across an NFS mount, the user **root** is translated to the user **nobody** on the remote system. The user **nobody** does not have write access.

- Files added to these directories actually are added to the **/usr/opt/X11** file system instead of the **/usr** file system. This makes disk space instructions incorrect. For example, a third-party package may instruct you to have a certain amount of free space in the **/usr** file system, while the free space actually is required in **/usr/opt/X11**.

- Relative symbolic links (links starting with **..**) added to these directories could be incorrect. For example, symbolic links created in the directory **/usr/bin/X11** are actually created in **/usr/opt/X11/bin**. The problem is that **/usr/opt/X11/bin** is four directory levels below root (**/**), instead of three as **/usr/bin/X11** implies. The *Porting and Developing Applications on the DG/UX Systems* manual, referred to as the DG/UX Porting Guide in the rest of the document, recommends that you use absolute symbolic links (links starting with **/**) to avoid this problem.

Starting with DG/UX release 5.4.1, these directories are real directories in the **/usr** file system, and the individual files in the directories are linked, instead of the directories themselves. The following is the list of directories created in the **/usr** file system:

```
/usr/bin/X11
/usr/include/X11
/usr/include/X11/bitmaps
/usr/lib/X11
/usr/lib/X11/app-defaults
/usr/lib/X11/fonts
/usr/lib/X11/fonts/100dpi
/usr/lib/X11/fonts/75dpi
/usr/lib/X11/fonts/misc
```

The setup script "**/usr/sbin/setup.d/usr/X11__0.X11.do**" creates the directories listed above, and creates symbolic links for each of the files contained in the corresponding **/usr/opt/X11** directory (for example, **/usr/opt/X11/bin** for **/usr/bin/X11**).

You can use the following three methods to install third-party software:

- UNIX

- DG/UX Porting Guide

- Direct access of the **/usr/opt/X11** file system. The following examples use the package **foo** installing the file **bar** into the directory **/usr/bin/X11**.

5.1.19.1 UNIX

The traditional UNIX installation method views the **/usr** file system as one large file system containing all the software on a system. This method derives from earlier versions of UNIX, which did not have symbolic links, and had limited support for Logical Disk Units (LDU). You can copy, move, or use the **tar** command to place the files to be installed into the appropriate directory, as follows:

cp /mv/tar -> /usr/bin/X11/foo ↵

WARNING: This method fails on a Release 5.4 DG/UX system that remote mounts the **/usr/opt/X11** file system. Furthermore, because this method assumes the files are being added to the **/usr** file system, installation instructions may incorrectly describe disk space usage. Since Release 5.4.2, remote mounting and disk space usage are not issues, because the X11 directories, as this method assumes, are real directories in the **/usr** file system.

5.1.19.2 DG/UX Porting Guide Installation

The method described in the DG/UX Porting Guide installs the files of an optional package in the **opt** subdirectory of a file system, such as **/usr/opt/foo** for the package **foo**. Symbolic links are then added in "public" directories (for example, **/usr/bin/X11**) that point to files in the **opt** subdirectories. The DG/UX Porting Guide recommends that you use absolute symbolic links in the X11 directories. For example, you should use the following command to link the file **bar** in the package **foo**:

```
ln -s /usr/opt/foo/bar /usr/bin/X11/bar ↵
```

X11 directory change in DG/UX releases allow you to use relative symbolic links. However, use absolute symbolic links for downward compatibility with 5.4 systems. For example, all of the following **ln** commands work on a Release 5.4.2 or later system, but only the last works on a Release 5.4 system:

```
ln -s ../../opt/foo/bar /usr/bin/X11/bar ↵
ln -s ../../../usr/opt/foo/bar /usr/bin/X11/bar ↵
ln -s ../../../../usr/opt/foo/bar /usr/bin/X11/bar ↵
```

Also, a symbolic link in the following form works on a Release 5.4 system only:

```
ln -s ../../../opt/foo/bar /usr/bin/X11/bar ↵
```

5.1.19.3 /usr/opt/X11 Filesystem Installation

The third method of installation is to direct access the **/usr/opt/X11** file system, as follows:

```
cp/mv/tar -> /usr/opt/X11/bin/bar ↵
ln -s /usr/opt/foo/bar /usr/opt/X11/bin/bar ↵
ln -s ../../foo/bar /usr/opt/X11/bin/bar ↵
ln -s ../../../opt/foo/bar /usr/opt/X11/bin/bar ↵
ln -s ../../../../usr/opt/foo/bar /usr/opt/X11/bin/bar ↵
```

An optional package should NEVER write in the installation area of another package. The **/usr/opt/X11** file system is intended only to segregate the X11 package files from the base DG/UX system. If a third-party software package does install its files in this manner, you can re-run the setup script **/usr/sbin/setup.d/usr/X11__0.X11.do[ne]** to remedy the situation. This script creates symbolic links for all new files. Since this script automatically runs when a system is upgraded, you need to re-run it only if you install a package that loads directly into **/usr/opt/X11** after a system is upgraded.

5.1.20 Xlib and Xt Manual Pages

The **Xlib** and **Xt** manual pages included in this release are incomplete. Several functions in this release were provided by MIT without proper on-line documentation. For a list of functions without manual pages, refer to the file **/usr/opt/X11/release/manpages.missing**. The complete **Xlib** and **Xt** documentation is available in the O'Reilly X Window System manuals, volumes 2 and 5.

5.1.21 Console Terminal Emulator Prompts

When you terminate an X session and the screen returns to the console terminal emulator window, no prompts appear until you press <Enter>.

5.1.22 X11R5 and X11R3 Backward Compatibility Mode (bc)

By default, the X server is started in X11R3 backward compatibility mode. The default configuration files supplied with this release start the server in this state (see the file **/var/X11/xdm/Xservers**). This state disables certain types of error checking that allows R3 clients to run under an R5 server. Backward compatibility mode is specified by the **bc** command line option in **/var/X11/xdm/Xservers**. The **xset** utility can also turn **bc** mode on and off.

5.1.23 Font Tools

As part of the move from the **snf** font format to **pcf**, the font tools **bdftosnf** and **showsnf** were replaced by the X Consortium with **bdftopcf** and **showfont**. For a limited time, Data General will continue to ship **bdftosnf** and **showsnf** for backward compatibility. These commands will be removed in a future release.

5.1.24 Athena Widget Documentation

Although the Athena Widget set is included in X11R5, **Xaw** manual pages are not. For Athena widget documentation, refer to the O'Reilly X Window System manuals, volumes 4 & 5 (see the order form enclosed with your shipment).

5.1.25 editread and xterm stty Settings

If **editread** is turned on when an **xterm** is started in the background (for example, **xterm &**), the **stty** settings picked up by the background **xterm** could be those before or after **editread** sets them up. **xterm** reads the **stty** settings of the parent tty (**/dev/tty**) to establish the settings it will use. **editread**, however, sets up the **stty** settings during the re-initialization, after the background **xterm** is launched. Essentially, it is a race as to whether the **stty** settings are established by **xterm** or **editread**.

This condition appears often in the **csh**, but rarely in the **sh**. To work around this situation, specify a **.cshrc** file that resets your **stty** settings in **xterm** after the C shell comes up, or to turn **editread** off before the launch. If you turn off **editread** before the launch, you cannot turn it back on.

5.1.26 First Time X-terminal Installation Notice

The AVX30 Release Notice instructs the system administrator to create a symbolic link in the **/usr/opt/X11/lib** directory. Because of the change in the public X11 directory structure explained earlier, the command to create this symbolic link has changed. If you are installing X terminals for the first time, use the following command:

ln –s /usr/opt/X11/xtd /usr/lib/X11/ncd ↵

If the symbolic link was created before the system was upgraded to Release 5.4.2 using the AVX30 Release Notice instructions, no action is necessary. The link is automatically converted during the package setup.

NOTE: For more information, see the "AVX-30 X Terminal Support" section.

5.1.27 Re-running the X11 Setup Script

When you load an application (which loads files directly into **/usr/opt/X11**) onto a DG/UX 5.4R3.10 system, you must re-run (as **root**) the X11 setup script using the following command:

/usr/sbin/setup.d/usr/X11__0.X11.done / /usr ↵

This script creates the necessary symbolic links in **/usr/bin/X11**, **/usr/lib/X11**, and **/usr/include/X11** to let the package run correctly. For more details, see the section "/usr/opt/X11 Structure".

5.1.28 Shared Memory Transport Implementation

For information about the Data General shared memory transport implementation, refer to the **XDGShm(5X)** manual page.

5.1.29 mterm

In **mterm**, the "options|mode" menu selection produces undesirable results if you select it, and then switch to vt100 or vt52 mode after you start a child other than a shell. In particular, doing a change mode to vt100 or vt52 mode after a **telnet**, **vi**, or **rlogin** session is started produces undesirable results. **mterm** must change the line characteristics when switching from one mode to another. If a child process that resets the characteristics is started, incompatibilities occur.

Generally, it is safe to switch to a DG mode at any time, and to switch to VT mode if you have not started a child process other than your shell. You should put **mterm** in the mode of the target system before you start any child process necessary to connect you to the remote system. You may want to use the DG/UX **reset** command after changing modes.

5.1.30 xdgmail

Because of a problem with the **-iconic** switch, use the following resource to cause **xdgmail** to start in an iconic state:

```
XdgMail.xdgmail.iconic: true
```

5.2 Warnings

5.2.1 X Server Swap Usage

The X server can monopolize large quantities of system swap space. This is most notable when running clients that demand large numbers of small- to medium-sized pixmaps, or several large pixmaps. Pixmap usage is not the only way to consume memory; it is, however, an easy way.

To help combat the problem, restart the server regularly, using the "restart" button on the **xdm** login panel. You must restart the server for the system to reclaim the used memory.

5.2.2 Shared Memory Transport Implementation

With Data General's shared memory transport implementation, data can be written only to the client's output buffer via the standard macros provided in the X library. These include GetReq (for getting the next available X request packet) and Data (to place data in the buffer and pad appropriately). Writing directly to the buffer and manipulating the buffer pointers (**bufptr**, **buffer**, **bufmax**) without using the standard macros results in output buffer corruption and unpredictable failures. This applies mainly to writing X extensions.

5.2.3 BCS Compliance

You must start the server with the **bc** option to run 88Open BCS-certified clients. For information on where the **bc** option is specified, see the section "X11R5 and X11R3 Backward Compatibility Mode (bc)".

5.2.4 R3 Clients under an R5 Xserver

R3 clients probably will not run on the R5 Xserver. With the **bc** option on, however, the R3 clients once provided by Data General should run with the R5 server. However, user-defined or outside clients may require updating to be compatible with the R5 server, even with the **bc** switch turned on.

5.2.5 xdgmail

The following are known **xdgmail** problems:

- Displaying **xdgmail** to Remote Systems

  If you use **xdgmail** with the resource **XdgMail*editCmd** set to something other than **builtin**, you may not be able to create or edit messages. This happens when you start **xdgmail** on one machine and display it on another across the network. In this instance, you must set the shell DISPLAY variable set before you start **xdgmail**. For example, if you use **/bin/sh** to display **xdgmail** on the server "foo" to the workstation "bar," you must execute the following commands on the workstation ("foo"):

  env DISPLAY=bar:0 xdgmail & ↲

NOTE: Using the **-display** command line option will not work, since this applies only to **xdgmail** and not other processes (**xterm**) started from within **xdgmail**.

- Xdgmail and Application Modal Dialogue Boxes

  Because many of the dialogue boxes in **xdgmail** are application-modal (that is, they must be dismissed before **xdgmail** will respond to any other input), it may appear at times that **xdgmail** is inexplicably hung. If **xdgmail** stops responding to input, look for boxes related to **xdgmail** that are hidden by other windows. Also, make sure all boxes other than the main window and the most current dialogue box have been dismissed. Since the error and warning message boxes are relatively small, one or more of them is typically the culprit.

- Xdgmail and Control Buttons

  If you set the **XdgMail*buttons** resource to **none**, X displays the following warning:

```
Warning:
  Name: controlMenu
  Class: XmRowColumn
  failure of geometry request to "almost" reply
```

  This warning does not affect the behavior of **xdgmail**. (This problem will be corrected in a later release of **xdgmail**.) To avoid the problem, keep at least one button defined in the **XdgMail*buttons** resource.

- Missing Message View Window

  Sometimes when **xdgmail** starts up after a system reboot, it appears to be missing the message view window. Actually, the movable grip between with message button window and the message view window has moved to the bottom of the **xdgmail** main window. To correct this, identify the small square "grip" in the lower right corner of the **xdgmail** window and drag it up the screen with the mouse until the message view window and the button box are the desired size relative to one another.

5.2.6 Comments and Blanks in Resource Files

Do not use C style comments (**/* */**) and C preprocessor style comments (**#**) inside resource files. Instead, use exclamation marks (**!**) at the beginning of a line. This applies mainly to those resource files not read by **xrdb**. For more information, see the **xrdb** manual page.

Do not put or leave extra blanks at the end of resource definitions because trailing blanks are treated as part of the resource value.

5.2.7 X and Motif Library Changes

Because of changes by MIT and OSF in the X and Motif libraries, some applications that use shared libraries may no longer run on X11R5. Because MIT removed some undocumented functions from the public namespace, any application that references those functions will not work. Data General is attempting to provide compatibility between X11R4 applications and X11R5 shared libraries. If you experience dynamic loader errors when running an X11R4 based application on an X11R5 machine, contact Data General. To correct the problem, Data General will send you the X11R4 library. In general, this is how old versions of shared libraries are supported.

5.2.8 Skeleton Shared Libraries

Data General uses skeleton shared libraries for the X libraries in X11R5 to provide better standards compliance and to prevent the use of nonpublicly-available functions by applications. These libraries include the following:

- **libX11.so**
- **libXt.so**
- **libXaw.so**
- **libXmu.so**
- **libXext.so**
- **libXi.so.**

This may cause some applications that are normally built with X11R4 to fail to link properly in X11R5. An application that fails to link is probably using a nonpublic function in the X libraries. Upward compatibility cannot be guaranteed when nonpublic functions are used by an application that uses shared libraries. However, if a nonpublic function is required, the application can be linked statically, assuring upward compatibility of the executable.

5.2.9 Shared Library Versions

Because of the nature of shared libraries, modifications to the libraries in subsequent releases generally will not require clients that dynamically link them to be relinked. However, changes that alter the programming interface, as is the case above, will be released in a new version of the library, as indicated by the version suffix (**libX11.so.2**, for example). While these changes will be kept to a minimum, they are out of Data General's control; programming interface changes are dictated by the MIT X Consortium and/or OSF. Clients linked with the static libraries are not affected by any of the these changes. That is, while static clients cannot incorporate changes to libraries without being relinked, they also do not experience the problems associated with shared objects. Therefore, linking statically is a possible solution if you want to isolate a client from library changes.

5.2.10 Non-24-bit Applications on 24-bit AViiON Displays

The X Window System provided in this release functions correctly on 24-bit color AViiON systems. However, applications not specifically designated to run on 24-bit systems may not function correctly.

If you are using **xdm**, you can use the option [**–cc 3**] on the X Server startup line in **/var/X11/xdm/Xservers** to start the server on a 24-bit machine using the pseudo-color visual mode. This makes pseudo-color the default visual and **xdpyinfo** reports a color depth of eight bits.

The default display visual for the R4 Xserver on a 24-bit system is TrueColor. Applications that assume a display visual of PseudoColor may not work correctly.

5.2.11 Compressed Fonts

Compressed fonts are not supported in the font server; however, they continue to be supported in the X server.

5.2.12 tearOffModel Resource Converter

The tearOffModel resource converter is available only in applications that provide it. For details, see the **XmBulletinBoard**(3) manual page.

5.2.13 Mislabeled Fonts

The following fonts are classed as 8-bit iso8859-1 fonts. They are not really 8-bit, but are 7-bit fonts that have been misnamed by MIT. This means that only the lower 128 glyphs (characters) are accessible. Renaming the fonts or "completing" them would correct the situation, but would put Data General's implementation of X Windows at odds with MIT's. "Pretending" that they are 7-bit fonts is the common usage solution to this problem, because this font set has existed in this state since X11R4.

```
-misc-fixed-bold-r-normal--13-100-100-100-c-70-iso8859-1
-misc-fixed-bold-r-normal--13-100-100-100-c-80-iso8859-1
-misc-fixed-bold-r-normal--13-120-75-75-c-70-iso8859-1
-misc-fixed-bold-r-normal--13-120-75-75-c-80-iso8859-1
-misc-fixed-bold-r-semicondensed--13-100-100-100-c-60-iso8859-1
-misc-fixed-bold-r-semicondensed--13-120-75-75-c-60-iso8859-1
-misc-fixed-medium-r-normal--10-100-75-75-c-60-iso8859-1
-misc-fixed-medium-r-normal--10-70-100-100-c-60-iso8859-1
-misc-fixed-medium-r-normal--13-100-100-100-c-70-iso8859-1
-misc-fixed-medium-r-normal--13-100-100-100-c-80-iso8859-1
-misc-fixed-medium-r-normal--13-120-75-75-c-70-iso8859-1
-misc-fixed-medium-r-normal--13-120-75-75-c-80-iso8859-1
-misc-fixed-medium-r-normal--20-140-100-100-c-100-iso8859-1
-misc-fixed-medium-r-normal--20-200-75-75-c-100-iso8859-1
-misc-fixed-medium-r-normal--8-60-100-100-c-50-iso8859-1
-misc-fixed-medium-r-normal--8-80-75-75-c-50-iso8859-1
-misc-fixed-medium-r-normal--9-80-100-100-c-60-iso8859-1
-misc-fixed-medium-r-normal--9-90-75-75-c-60-iso8859-1
-misc-fixed-medium-r-semicondensed--12-110-75-75-c-60-iso8859-1
-misc-fixed-medium-r-semicondensed--12-90-100-100-c-60-iso8859-1
-misc-fixed-medium-r-semicondensed--13-100-100-100-c-60-iso8859-1
-misc-fixed-medium-r-semicondensed--13-120-75-75-c-60-iso8859-1
-schumacher-clean-bold-r-normal--10-100-75-75-c-60-iso8859-1
-schumacher-clean-bold-r-normal--10-100-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--12-120-75-75-c-60-iso8859-1
-schumacher-clean-bold-r-normal--12-120-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--13-130-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--14-140-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--15-150-75-75-c-90-iso8859-1
-schumacher-clean-bold-r-normal--16-160-75-75-c-80-iso8859-1
-schumacher-clean-bold-r-normal--8-80-75-75-c-80-iso8859-1
-schumacher-clean-medium-i-normal--12-120-75-75-c-60-iso8859-1
-schumacher-clean-medium-i-normal--8-80-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--10-100-75-75-c-50-iso8859-1
-schumacher-clean-medium-r-normal--10-100-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--10-100-75-75-c-70-iso8859-1
-schumacher-clean-medium-r-normal--10-100-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--12-120-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--12-120-75-75-c-70-iso8859-1
-schumacher-clean-medium-r-normal--12-120-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--13-130-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--13-130-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--14-140-75-75-c-70-iso8859-1
-schumacher-clean-medium-r-normal--14-140-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--15-150-75-75-c-90-iso8859-1
-schumacher-clean-medium-r-normal--16-160-75-75-c-80-iso8859-1
-schumacher-clean-medium-r-normal--6-60-75-75-c-40-iso8859-1
-schumacher-clean-medium-r-normal--6-60-75-75-c-50-iso8859-1
-schumacher-clean-medium-r-normal--6-60-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--8-80-75-75-c-50-iso8859-1
-schumacher-clean-medium-r-normal--8-80-75-75-c-60-iso8859-1
-schumacher-clean-medium-r-normal--8-80-75-75-c-70-iso8859-1
```

```
-schumacher-clean-medium-r-normal--8-80-75-75-c-80-iso8859-1
```

5.2.14 Loading User Supplied Items into /usr/opt/X11

Do not load user-supplied items into **/usr/opt/X11**. Do not place ANY applications, clients, fonts or files of any kind under the **/usr/opt/X11** directory structure. The **/usr/opt/X11** structure is provided as an official package release mechanism only.

While many of the files and directories under **/usr/opt/X11** are the actual locations for these entities, applications expect them to be under **/usr/lib/X11**, **/usr/bin/X11**, or **/usr/include/X11**. These are standard well-known locations, and as such are appropriate for adding user-supplied X files.

5.2.15 Running the Font Server

To run the font server effectively, you must be **root**. Otherwise, the font server cannot open its error file (**/var/X11/fs/fs-error**). If the error file does not exist, the font server reports its problems to the invoking console.

5.2.16 Security in Trivial File Transfer Protocol (TFTP)

Improved security in Trivial File Transfer Protocol (TFTP) allows access to the files in **/tftpboot** only. X terminals should use NFS® rather than TFTP to access configuration and font files. For more information, see the section "AVX-30 X Terminal Support".

5.2.17 AVX-30 X Terminal Support

Data General plans to remove software support for the AVX-30 X Terminal Support in the next revision of the DG/UX system.

5.2.18 Use of XrmParseCommand

In X11R4, **XrmParseCommand()** allowed you to pass a string of the form *AppName.class* as the fourth parameter (name), but the fourth parameter could only be the application's name. Since this use of **XrmParseCommand()** was not documented and is no longer available in X11R5, applications (such as the contributed client **xlock**) used this feature must be modified to work properly with X11R5. An application that has this problem will not act on some command line options. To resolve this problem, make changes to your **app-defaults** file instead of using the command line option.

5.2.19 mterm and editread

If you use **editread**, start an **mterm** session, change mode, and then press a character prior to pressing the Return (Enter) key, **editread** starts replicating the character you pressed and beeps uncontrollably. You can kill this **mterm** session with either the **mwm** window menu close button or **dg_kill(1)**. To avoid this problem, either disable **editread** before starting **mterm** or press the Return key after changing emulation mode.

5.2.20 xresedit

When you enter the **xresedit** edit window to edit a font resource, the window lists the wrong number of fonts that match the default font string. From the edit window, you can pick almost any value for each element of the font string even if that choice does not match any font. If you pick an invalid option, the window will not update properly until you create a valid font string. To avoid this problem, always pick one of the font elements and select the * choice or the already-selected choice. When you make this selection, the window lists valid options. In this way, you can avoid selecting invalid font elements.

Following is an example workaround of this problem. If you are editing a resource with the following font string:

```
-misc-fixed-medium-r-normal--15-140-75-75-c-90-iso8859-i
```

select the **fndry** pulldown menu and then select either * or **misc** to get a valid font. Now, you can select only valid options for each element.

5.2.21 Other Warnings

- The help switch on the **bdftosnf** command erroneously refers to the **–u** switch as the **–s** option. (It states "and # for –s is 1,2, or 4.")
- There is a conflict between **xdgmail** and mnemonics/accelerators. If you use a menu button mnemonic identical to an accelerator or another operation, you get both selections. For example, if you press "h" while in the menu bar, the help window AND the search dialog box pop up. This problem will be corrected in a later release of **xdgmail**.
- If the end of a mailbox is truncated, a currently-running **xdgmail** session may exit with a segmentation fault. You cannot restart **xdgmail** until one or more empty lines are appended to the mailbox.
- If the LANG variable is not set properly, **mwm** will not allow 8-bit characters to appear in the window title. Set the LANG environment variable to en_US to remedy the problem. The current setting of C is 7-bit. Because en_US is 8-bit, it allows **mwm** to display international characters in window titles. Do this only if you want 8-bit characters in the window title.
- When running Motif applications on an X terminal, turn off the option "Retain X Settings" for the applications to run properly.
- Motif appears to handle pop-up menus incorrectly under certain conditions. When running an application that supports pop-up menus, you might "lose" the keyboard by bringing up the

popup menu and pressing the Escape key. Switch to another window and then back to the original window to restore the focus.

- If you have an **mwm** key binding in your **.mwmrc** file for a key in a context other than the context you are in (for example, a binding for **root** exists, but you're in a window), **mwm** fails to pass the key event to the application in that context.

  If the following is in the **.mwmrc** file:

  ```
  RKeys MyKeyBindings {
          <Key>Scroll_Lock    root    f.nop }
  ```

  mwm should only consume the Scroll_Lock key events when the context is **root** with this definition.

  However, when the cursor is in another context (for example, a window), mwm also consumes the key and does not pass it to the application as it is supposed to. The application does see the FocusOut and FocusIn events when the key is pressed, indicating that mwm has intercepted the key event. The key event itself, however, is never passed to the application.

  Mwm passed on such keys in previous revisions (DG/UX 5.4 worked correctly), but with DG/UX 5.4.2 and later revisions, it no longer does so. You can remedy the problem by using a key binding for a key that is not in use.

- "OptionMenu" is now a default label for an option menu. Formerly, the default action (that is, not setting XmNlabelString, or setting it to NULL) was that no label appeared. Now, the default is for Motif to specify "OptionMenu" as the label.

- Indicators disappear when XmNlabelType is XmPixmap. When XmNlabelType is set to XmPixmap at the creation of a ToggleButton, but the actual pixmap is added later using SetValues, the indicator button disappears. This is not a problem if a pixmap is designated at creation time. It also is not a problem if one creates the ToggleButton with XmNlabelType set to the default, sets the pixmap, and then sets XmNlabelType to XmPixmap.

- **mwm** now requires that at least one key binding be present in a "Keys" definition in the **.mwmrc** file. In DG/UX 5.4, empty Keys definitions were legal and worked as expected.

  ```
  RKeys MyKeyBindings {

      !
      ! Mwm defaults deleted
      !

  }
  ```

  Along with an X resource of Mwm*keyBindings: MyKeyBindings, this had the desired effect of eliminating all **mwm** key bindings. However, in DG/UX 5.4.2 and later revisions, when the same definitions are used, **mwm** reports this error:

  ```
  Rmwm: Key bindings MyKeyBindings not found, using builtin key
  bindings
  ```

If you add one key definition to the Keys entry, the message is eliminated. Thus, **.mwmrc** now looks like this:

```
RKeys MyKeyBindings {

     !
     ! Mwm defaults deleted
     !

       <Key>F1     root     f.nop

}
```

This problem will be fixed in a later release of the DG/UX system.

- OptionMenu help does not behave as expected under certain conditions. In DG/UX System 5.4.2, 5.4R2.01 and 5.4R2.10, if you clicked on an OptionMenu widget, it did not matter how long you pressed the button. When the mouse button was pressed, the options displayed; when the mouse button was released, the display collapsed, but the widget remained selected. You could then get help by pressing F1.

  In DG/UX 5.4R3.10, if you click quickly on the widget, the options are displayed and remain displayed until you click somewhere again. If you click slowly, you get the old behavior. However, even though the widget appears to be selected, pressing F1 does nothing. But, if you click on the widget ***and hold the mouse button down***, and then with the mouse button down, press F1, you get help.

- **mwm** displays error messages in certain instances.

- XmList widget displays incorrect selections. When you select the top item in a list widget, a new item is added to the top of the list widget. When you then use the keyboard to select the newly-added item, the previously-selected item is still incorrectly highlighted.

- XmText module does not always redraw the screen after deletion.

  To work around this problem:

  1. Create a scrolled XmText widget containing enough text to force the widget to scroll.
  2. Use the mouse to highlight text from the middle of the scrolled region to the end of text.
  3. Scroll the window back to the top of the text.
  4. Press the Delete key.
     The highlighted text will be unhighlighted, but not erased.
  5. Run **xrefresh**.
     The previously highlighted text will disappear.

- In the Motif 1.2.2 version of **mwm** (in DG/UX System 5.4 Release 3.10), if you first move a window so that the frame border of the window being moved is beneath the popup box that shows the screen coordinates and then release the mouse button, you may get a short different color line in the window. This is a bug in **mwm** that will be fixed in a future Motif release. If you find the line annoying, issue an **xrefresh** command to redraw the window correctly.

5.2.22 Terminating an X Session

In DG/UX 5.4R3.10, you can no longer terminate an X session by killing the window manager. In previous revisions of the DG/UX system, the files **system.Xsession** and **sample.Xsession** had as their last line **exec mwm**. This allowed the current session to be terminated by killing **mwm.**

In 5.4R3.10, these files end with the lines **mwm &** and **wait.** This takes **mwm** out of the role of holding up the session and solves problems where *<defunct>* processes were accumulating. It also prohibits the termination of the current session by killing **mwm.**

Use the a root menu option Logout to terminate an X session. To use this option, you need to modify any **.Xsession** files based on the sample **.Xsession** of previous revisions of the DG/UX system as follows. Replace the **exec mwm** line with the two lines **mwm &** and **wait.**

6 Documentation

6.1 Manuals

The following documents ship with the X Window System released with the DG/UX™ operating system:

Publication	Part Number
X Window System Release Notice	085-600421
X Window System Programmer's Supplement for Release 5	069-100474
X Window System User's Guide, OSF/Motif Edition	069-100229
X Window System Document Order Form	069-100392
OSF/Motif Document Order Form	069-100411

Additional manuals can be ordered directly from O'Reilly & Associates and from Prentice Hall. Order forms with descriptions of books are included with your shipment.

See also various on-line documents in **/usr/opt/X11/doc.**

6.2 Documentation-changes files

There are no documentation-changes files associated with this release.

7 Software Distribution

7.1 Media

This release of the DG/UX™ Window System package is part of the general DG/UX operating system release package. Please see the DG/UX release notice for exact media details.

7.2 Files

For a complete list of files present in the DG/UX X Window System Package, see the following files under /usr/opt/X11/release after the system is loaded (associated package in brackets):

```
X11_5.4R3.10.fl           [X11]
X11.man_5.4R3.10.fl       [X11.man]
X11.sde_5.4R3.10.fl       [X11.sde]
X11.sde.man_5.4R3.10.fl   [X11.sde.man]
```

8 Installation Instructions

8.1 Introduction

This release of the DG/UX™ X Window System can be installed on both stand-alone workstations and OS server-client configurations using **sysadm**. This release contains Release 5 of the X11 product and replaces the current contents of **/usr/opt/X11**. Before proceeding, read these installation instructions CAREFULLY!

X11 was updated to Release 5. It will be henceforth referred to as *X11R5* or *R5*. Similarly, Releases 3 and 4 will be referred to as *X11R3* or *R3*, and *X11R4* or *R4*.

8.2 Restrictions

This release of the DG/UX X Window System is intended to run ONLY on DG/UX System 5.4 Release 3.10.

8.3 Installation Considerations

8.3.1 X11R5 Images

X11R5 is packaged into four images that are different from those delivered with X11R4. In general, X11R5 has been divided into two logical environments: run-time and development. That is, there is a distinction between what is needed to run X and what is needed to do X development (i.e., X clients and server customization). Each package has associated manual pages. The packages are described in detail below:

- X Window System Image [X11]

This required image contains the necessary files to support the DG/UX™ X Window System for a run-time environment. That is, it contains all X11 executables, shared objects, fonts, bitmaps, resource files, and other miscellaneous files needed for a running X environment. This image loads into the directory **/usr/opt/X11**. The current contents of **/usr/opt/X11** are overwritten during loading.

- X11 Manual Pages [X11.man]

This optional image contains all the section 1 (user commands) and section 5 (miscellaneous) manual pages supplied by MIT, OSF, and Data General. This image fits in the space allocated for the X11 package in **/usr/opt/X11**.

- X Software Development Environment Image [X11.sde]

This optional image contains all header files and static libraries needed to develop clients and customized X servers. This image also contains all on-line documentation (excluding manual pages) in both Postscript and printable formats.

- X Software Development Manual Pages [X11.sde.man]

  This image contains the X software development manual pages. This package contains all the section 3 (subroutine) manual pages supplied by MIT, OSF, and Data General.

8.3.2 Saving Locally-Tailored Files

Any files that were customized for your local site will be destroyed during the installation process. You should back up these files to tape before starting the installation process. This backup of selected files is in addition to a complete system backup discussed in a later section. For a convenient means of restoring your customizations without the cumbersome process of restoring a few files off a complete system backup, keep a separate copy of these files (made with **cpio** or **tar**).

The following is a suggested list of such files. It is not an exhaustive list, nor is it implied that the system administrator automatically save every item on the list. This list is a reminder of possible places where the system administrator may have customized the local environment:

NOTE: User sites should not modify any files in the directory structure. If modifications are necessary, find the symbolic link under **/usr/bin/X11**, **/usr/lib/X11**, or **/usr/include/X11** that has the same name as the actual file and delete it. Copy the original file from **/usr/opt/X11** into the same filename under **/usr/bin/X11** (or the other directories as necessary) and make the modifications there.

/usr/opt/X11/include/X11/bitmaps
(In this release, only the bitmaps received from the MIT distribution are included).

/usr/opt/X11/lib/fonts/
(In this release, only MIT distribution fonts are supplied).

/usr/opt/X11/xtd/*

/usr/lib/X11/app-defaults/*

/usr/lib/X11/rgb.*

/usr/opt/X11/lib/sample.Xdefaults

/usr/opt/X11/lib/xstart/xstart.sh

Other
Any files that were customized but are not included in the above list.

8.4 Installation Steps

Installation of the DG/UX X Window System is covered in the manual *Installing the DG/UX™ System.*

8.4.1 Starting Up and Using the DG/UX X Window System

Once the installation steps for your system are complete, use the **init** command to bring the system back to run level 3. Consult the manual *Managing the DG/UX System* for information on changing run levels. During the process of going to run level 3, the X server **(X)**, and the X Display Manager **(xdm)** are started. When the software packages were set up, an entry was automatically added into **/etc/inittab** to start the X Display Manager. **xdm** will present the **xdm** login window, which is used for login purposes. For more information about using and controlling **xdm**, consult the **xdm** and **XDMCP** manual pages.

Xdm has replaced the **xstart** interface as the preferred method for starting up an X session. When a user first logs on the system using **xdm**, his or her private **.xstart.sh** (or other such startup file) is not used. **Xdm** uses an **.Xsession** file to start up a tailored environment. You can reestablish your environment in one of two ways.

You can get backward compatibility with **xstart(1X)** by copying the file **/usr/lib/X11/xdm/xstart.Xsession** to your **$HOME** directory as the file **$HOME/.Xsession**. However, the preferred method is user to copy the file **/usr/lib/X11/xdm/sample.Xsession** to your **$HOME** directory as the file **$HOME/.Xsession**. Then, using an editor (such as **ed**(1), **vi**(1), etc.), enter your standard client set into this file, replacing the three clients (**xterm**, **xclock**, and **xbiff**) provided by default. For further information about these two files, consult the **Xsession**(5) manual page.

Before using the DG/UX X Window System, make sure that **/usr/bin/X11** is on your path. Your path contains the list of ordered directories to search for programs to be executed.

To check your path under the Bourne shell, execute the following:

```
echo $PATH
```

To check your path under the C shell, execute the following:

```
printenv PATH
```

If **/usr/bin/X11** does not appear in your environment variable, you must add **PATH** then **/usr/bin/X11** to your search path.

For the Bourne shell, use the following command to correctly set the **PATH** environment:

```
PATH=$PATH:/usr/bin/X11; export PATH
```

For the C shell, use the following command:

```
setenv PATH ${PATH}:/usr/bin/X11
```

You may want to edit your **.login** or **.cshrc** files to initialize this variable.

For information about how to log in and use the DG/UX X Window System, consult Appendix B in the *Using the DG/UX System* manual and Volume 3 of the O'Reilly X Window System Series.

User accounts created prior to installation of DG/UX 5.4R3.10 may use the Athena "lookAndFeel" by default (unless you set the resource ***lookAndFeel: motif**). New user accounts where **sysadm** is used to create the **$HOME** directory will have the motif "lookAndFeel" by default. To do this, set the resource ***lookAndFeel: motif** in the **.Xdefaults** file that is copied from the **/etc/skel** directory during creation of the **$HOME** directory by **sysadm**.

9 Preparing a Software Trouble Report (STR)

If you believe you have found an error in the DG/UX X Window System or its documentation, or if you have a suggestion for enhancing or improving the product, use a Data General Software Trouble Report (STR) to communicate this to Data General.

STR forms are available from the nearest Data General office or Data General representative, or the Software Support Center. On-line STR forms are available in **/usr/release/STR_form**.

If your contract permits, you may report the information called for in this section to your Data General representative. To help us process STRs quickly, **please include only one problem or suggestion on each STR form.** Please follow these guidelines when filling out your Software Trouble Report:

1. List the product name, model number, and revision number as shown on the title page of this release notice. If you are running an update or patch, include its number as well.
2. Decide what kind of STR you are writing:
   - Enhancement: describe the proposed enhancement clearly and tell why you want it. The better we understand your desire, the easier it is for us to evaluate your request.
   - Documentation Error: list the title and part number of the document or manual page and list the page and paragraph (or section) containing the error. Please state exactly why you think there is an error.
   - Software Problem: clearly and specifically state the problem so that support personnel can try to reproduce it. See the section **Software Problems** below for more details.
3. On the STR form provide all of the following information:
   - Date
   - Name and revision of the product
   - CPU type

- Hardware configuration (if relevant)
- Names and revisions of other software running on the system
- The command line or scenario that caused the problem
- The action(s) necessary to reproduce the problem
- How often the problem occurs and how serious it is

4. If the problem occurred soon after installing a new revision of software or new hardware, please note this.
5. If you received an error message, please write down the exact text (and number, if present) of the message.
6. Please see the DG/UX release notice for information on how to prepare a tape for STR submission.

9.1 Software Problems

Report any particular activity or program running on the system that seems to cause the problem. If the program is supplied by Data General, report in detail the exact steps used to reproduce the problem. If the program is supplied by another vendor or written by an installation, include a copy of the program and its source code if possible. Again, report in detail the exact steps used to reproduce the problem.

If your system panics, hangs or halts, see the DG/UX 5.4R3.10 Release Notice for instructions on taking system dumps and submitting DG/UX STRs.

End of DG/UX X Window System Release Notice