X Window System, Version 11
Release 6.3

Release Notes

X Consortium, Inc.

December 23, 1996

Table of Contents

1. What Is Release 6.3

This is the last X Consortium implementation of the X Window System. X is a vendor-neutral, system-architecture neutral network-transparent window system and user interface standard. X runs on a wide range of computing and graphics machines. For an overview of X, see the *X* manual page.

R6.3 is an update to R6.1. It is compatible with R6 and R6.1 at the source and protocol levels in all respects, and binaries are upward-compatible.

What about Release 6.2? Release 6.2 is a proper subset of Release 6.3 produced at the request of the OSF Common Desktop Environment program. It was produced by the X Consortium and is being released by OSF simultaneously with CDE 2.1. Release 6.2 contains only the print extension and the Xlib implementation of vertical writing and user-defined character support.

The X Consortium was an independent, not-for-profit membership corporation formed in 1993 as the successor to the MIT X Consortium and dissolved at the end of 1996. Refer to the *Consortium* man page for additional details about the X Consortium.

See **xc/INSTALL.PS** (PostScript) or **xc/INSTALL.TXT** (plain text) for instructions on how to build and install this software.

1.1. Overview of the X Consortium Release

The X Consortium software and documentation in Release 6.3 is in directory **xc/** and contains the following:

X Consortium Standards
: The X Consortium produced standards: documents which define network protocols, programming interfaces, and other aspects of the X environment. See the *XStandards* manual page for a list of standards.

Implementations
: For most of our standards, we provide high-quality implementations to demonstrate proof of concept and to give early adopters and vendors a base to use. These are not *reference* implementations; the written specifications define the standards.

Fonts
: A collection of bitmap and outline fonts are included in the distribution, contributed by various individuals and companies.

Utility Libraries
: A number of libraries, such as *Xmu* and the *Athena Widget Set*, are included. These are not standards, but are used in building X Consortium applications and may be useful in building other applications.

Programs
: We also provide a number of application programs. A few of these programs, such as *xdm* (or its equivalent), should be considered essential in almost all environments. The rest of the applications carry no special status; they are simply programs that have been developed and/or maintained by X Consortium staff. In some cases, you will find better substitutes for these programs contributed by others.

1.2. Supported Systems

We built and tested this release on the following systems:

AIX 4.2
Digital Unix 4.0A
HP-UX 10.01
IRIX 6.2
Solaris 2.5
UNIX System V/386 Release 4.2 (Novell UnixWare) Version 2.02

We also built this release on the following and did some minimal testing:

FreeBSD 2.1.6
Linux 1.2.13 (Yggdrasil) and 2.0.0 (Slackware 3.1)
SCO Open Server 5.0
SunOS 4.1.4
Windows NT 4.0

In all cases except SunOS we have used the vendor's compiler. On SunOS we build with *gcc*.

1.2.1. Supported Display Devices

This release includes the necessary device-dependent support to build a native X server for the following platforms:

AIX: Xibm with Skyway display adapter
HP-UX: Xhp
Digital Unix: Xdec on Alpha AXP with PMAG-B frame buffer
SunOS/Solaris: Xsun -- see the Xsun man page for supported frame buffers
Ultrix[1] :Xdec
XFree86: See the XF_* man pages for supported video cards

In addition to the above, the Xvfb and Xnest servers can be built on most platforms.

Native servers are not built on IRIX or Microsoft Windows NT.

1.3. The XC Tree

The general layout under **xc/** is as follows:

[1] this is not a full ANSI C and POSIX platform and the full release will no longer build here. But we have not intentionally removed any old support. I.e. *no guarantees* but we hope we've left open the possibility of someone else making the release work again on Ultrix.

config/	config files, *imake*, *makedepend*, build utilities
doc/	all documentation other than per-program manual pages
fonts/	BDF, Speedo, Type1 fonts
include/	include files shared by multiple directories
lib/	all libraries
nls/	national language support files
programs/	all programs, including the X server and *rgb*
util/	*patch*, *compress*, other utilities
bug-report	bug reporting template
registry	X Registry

This file is **xc/RELNOTES.***, in various formats. The documentation source files **RELNOTES.ms** and **INSTALL.ms** are in the **xc/doc/misc/** directory.

1.4. X Registry

The X Consortium maintained a registry of certain X-related items to aid in avoiding conflicts and to aid in sharing of such items.

The registry is in the file **xc/registry** in the distribution. The latest version may also be available by sending a message to xstuff@x.org. The message can have a subject line and no body, or a single-line body and no subject; in either case the line should look like this:

    send docs registry

1.5. Extensions Supported

The core distribution includes the following extensions: BIG-REQUESTS, DOUBLE-BUFFER, LBX, MIT-SHM, MIT-SUNDRY-NONSTANDARD, Multi-Buffering, RECORD, SECURITY, SHAPE, SYNC, X3D-PEX, XC-APPGROUP, XC-MISC, XFree86-VidModeExtension, XIE, XInputExtension, XKEYBOARD, XpExtension (printing), XTEST, and XTestExtension1.

Not all of these extensions are standards; see the *XStandards* manual page. Some of these extensions are not supported on all platforms.

1.6. Implementation Parameters

Some of the specifications define some behavior as implementation-dependent. Implementations of X Consortium standards need to document how those parameters are implemented; this section does so.

XFILESEARCHPATH default

This default can be set at build time by setting the *imake* variables XFileSearchPathDefault, XAppLoadDir, XFileSearchPathBase, and ProjectRoot in **site.def**. See **xc/config/cf/README** for instructions and **xc/config/cf/X11.tmpl**[2] for details of how these configuration variables are used.

By default ProjectRoot is /usr/X11R6.3 and XFILESEARCHPATH has these components:

    /usr/X11R6.3/lib/X11/%L/%T/%N%C%S
    /usr/X11R6.3/lib/X11/%l/%T/%N%C%S

[2] This file was renamed from Project.tmpl in R6.1 in order to support multiple projects (e.g. CDE/Motif) in the same config directory.

/usr/X11R6.3/lib/X11/%T/%N%C%S
/usr/X11R6.3/lib/X11/%L/%T/%N%S
/usr/X11R6.3/lib/X11/%l/%T/%N%S
/usr/X11R6.3/lib/X11/%T/%N%S

XUSERFILESEARCHPATH default
If the environment variable XAPPLRESDIR is defined, the default value of XUSERFILESEARCHPATH has the following components:

$XAPPLRESDIR/%L/%N%C
$XAPPLRESDIR/%l/%N%C
$XAPPLRESDIR/%N%C
$HOME/%N%C
$XAPPLRESDIR/%L/%N
$XAPPLRESDIR/%l/%N
$XAPPLRESDIR/%N
$HOME/%N

Otherwise it has these components:

$HOME/%L/%N%C
$HOME/%l/%N%C
$HOME/%N%C
$HOME/%L/%N
$HOME/%l/%N
$HOME/%N

XKEYSYMDB default
Defaults to **/usr/X11R6.3/lib/X11/XKeysymDB**, assuming **ProjectRoot** is set to **/usr/X11R6.3**.

XCMSDB default
Defaults to **/usr/X11R6.3/lib/X11/Xcms.txt**, assuming **ProjectRoot** is set to **/usr/X11R6.3**.

XLOCALEDIR default
Defaults to the directory **/usr/X11R6.3/lib/X11/locale**, assuming **ProjectRoot** is set to **/usr/X11R6.3**. The XLOCALEDIR variable can contain multiple colon-separated pathnames.

XErrorDB location
The Xlib error database file is **/usr/X11R6.3/lib/X11/XErrorDB**, assuming **ProjectRoot** is set to **/usr/X11R6.3**.

XtErrorDB location
The Xt error database file is **/usr/X11R6.3/lib/X11/XtErrorDB**, assuming **ProjectRoot** is set to **/usr/X11R6.3**.

Supported Locales
X locales supported are in **locale.dir**; the mapping between various system locale names and X locale names is in **locale.alias**. Both files are shipped in the **xc/nls/X11/locale/** directory and installed in the XLocaleDir directory (e.g. **/usr/X11R6.3/lib/X11/locale/**).

Input Methods supported
The core distribution does not include any input method servers. However, Xlib supplies a default built-in input method that supports compose processing in 8-bit locales. Compose files are provided for Latin-1 and Latin-2. The built-in input method can support other locales, given suitable compose files. See **xc/nls/X11/locale/Compose/iso8859-*** for the supported compositions.

There are input method servers available on the net.

2. What is Unchanged in Release 6.3

As this is an update release, there is a great deal of stability in the standards, libraries, and clients. No existing standards other than the ICE library specification have changed in a material way, though several documents have been updated with editorial improvements. There is one new interface added to the ICE library *libICE*; see below. The extension library, *libXext*, is updated to include the LBX, security, and application group extension interfaces. All previous interfaces in these and all other libraries are unchanged.

3. What Is New in Release 6.3

This section describes changes in the X Consortium distribution since Release 6.1.

All libraries, protocols, and servers are compatible with Release 6 and Release 6.1. That is, R6 and R6.1 clients and applications will work with R6.3 libraries and servers. Most R6.3 clients will work with R6.1 and R6 libraries except those that use the new interfaces in libICE, libXext, and libXp.

The major new functionality in R6.3 is support for World Wide Web integration, protection of data from "untrusted" client connections, a bandwidth- and latency-optimized protocol for using X across the Internet, a print protocol following the Xlib API, and support for vertical text writing and user-defined characters in the Xlib implementation.

3.1. OS Support

The following platforms have a newer operating system version supported:

System	R6.1	R6.3
AIX	4.1.4	4.2
Digital Unix	3.2C	4.0A
HP-UX	10.01	
IRIX	5.3	6.2
Solaris	2.4	2.5
UnixWare	2.02	

We also built on the following platforms, however full support is not guaranteed:

System	R6.1	R6.3
FreeBSD	2.1.0	2.1.6
Linux	1.2.13	2.0
SCO Open Server	5.0	
SunOS	4.1.3	4.1.4
Windows NT	3.5	4.0

3.2. New Standards

The following are new X Consortium standards in Release 6.3. Each is described in its own section below.

Low Bandwidth X Extension
RX: X Remote Execution MIME type
Security Extension
Application Group Extension
Print Extension
Proxy Management Protocol

3.3. Low Bandwidth X Extension

The Low Bandwidth X extension (LBX) defines several compression and local caching techniques to improve performance on wide area networks and also on slower-speed connections. These reduce the amount of protocol data transported over the network and reduce the number of client-to-server roundtrips required for common application startup operations.

LBX was referred to as X.fast in some materials but we elected to not go through the implementation and change all the names. To avoid any confusion with an external name different from the internal name in the implementation, we elected to drop the ‘‘X.fast’’ moniker.

LBX is implemented in two pieces; an X server extension and a proxy application. The X server extension provides the new optimized protocol. The proxy application, *lbxproxy*, translates a normal client X protocol stream into an LBX stream. This permits any existing application to gain the benefit of the optimized protocol with no changes. The proxy is especially useful when multiple applications are running on the same local area network separated from the X server by a slower network. In this case the full benefit of the local cache is shared by each application using the same proxy process.

The specification for LBX is in **xc/doc/specs/Xext/lbx.mif** (FrameMaker interchange source) and **xc/doc/hardcopy/Xext/lbx.PS.Z** (compressed PostScript).

3.4. RX: X Remote eXecution

The remote execution (RX) service specifies a MIME format for invoking applications remotely, for example via a World Wide Web browser. This RX format specifies a syntax for listing network services required by the application, for example an X display server. The requesting Web browser must identify specific instances of the services in the request to invoke the application.

The distribution contains a helper program (*xrx*) and a Netscape Navigator plug-in (*libxrx*) that demonstrate this protocol. The plug-in requires Navigator 3.0.

We have only been able to test the plug-in on HP-UX, IRIX, Digital Unix, and Solaris2. Netscape Navigator binaries for other platforms are either not available at all or were not available in time to be included in the testing for this release.

The specification for the RX mime type is in **xc/doc/specs/RX/RX.mif** (FrameMaker interchange source) and **xc/doc/hardcopy/RX/RX.PS.Z** (compressed PostScript).

The following section describes the procedure to set up your environment and try the examples provided in this distribution.

3.4.1. Preparing Your Web Server

In order to demonstrate the RX helper program and the RX Netscape plug-in you need to have access to an HTTP server to install "common gateway interface" (CGI) scripts. While CGI programs can be written in any compiled or interpreted language, the sample CGI programs in the distribution are written in perl.

If you don't currently have a web server the NCSA server is a good one to try. Binaries for various systems are available at:

http://hoohoo.ncsa.uiuc.edu/docs/setup/PreExec.html

If you don't have perl you can get the source code from:

ftp://prep.ai.mit.edu/pub/gnu/perl-4.036.tar.gz

You need to install the HTML, RX, and CGI sample files into your server's HTML and CGI directories. The process can be partially automated by adding the following definitions to your **site.def** or **host.def** file:

WebServer defines the hostname and port of your web server, for example

```
#define WebServer www.myorg.org:8001
```

HtmlDir defines the path at which HTML and RX documents are installed, for example

```
#define HtmlDir /usr/local/etc/httpd/htdocs
```

CgiBinDir defines the path at which CGI programs are installed, for example

```
#define CgiBinDir /usr/local/etc/httpd/cgi-bin
```

ProxyManager defines the transport scheme, hostname, and port for CGI programs to contact the Proxy Manager. See the *proxymngr* man pages for further details. Typically the proxy manager host will be the same as your web server, for example:

```
#define ProxyManager tcp/www.myorg.org:6500
```

Then make the Makefiles and build the directories with the following command sequence:

```
cd xc/programs/xrx/htdocs
xmkmf ../../.. programs/xrx/htdocs
make
make install
cd ../cgi-bin
xmkmf ../../.. programs/xrx/cgi-bin
make
make install
```

These directories are not automatically built or installed by the top level Makefile because they install outside the ProjectRoot.

You also need to configure your web server so that files with the extension name "rx" are of the MIME type "application/x-rx". See your HTTP server's configuration documentation for the right procedure to do so.

3.4.2. The RX Helper Program

The helper program, *xrx*, may be used with any Web browser to interpret the new RX document type.

The RX helper program is installed in <ProjectRoot>/bin (e.g. /usr/X11R6.3/bin/). You will need to configure your web browser to use it for RX documents by adding a line to your $HOME/.mailcap:

```
application/x-rx; /X11/bin/xrx %s
```

You may need to refer to your web browser's documentation for exact instructions on configuring helper applications.

The helper program is activated by your browser as soon as you retrieve any document of the MIME type application/x-rx. All you need to do is to point your browser at the URL:

```
http://your.web.server/xload.rx
```

The application (i.e. *xload*) should appear on your DISPLAY as a new top-level client. The client will be running on your web server host and connected to your X server. If your X server supports the SECURITY extension the client will be running as an untrusted client.

3.4.3. The RX Netscape Navigator Plug-in

The Navigator plug-in supports all the functions of *xrx* and in addition uses the new XC-APPGROUP extension, if your X server provides it, to cause the remotely launched application to be embedded within the browser page from which it was launched.

The HTML page links to an RX document via the EMBED tag, a Netscape extension to HTML. The RX document provides the plug-in with the list of services the application wants to use. Based on this information, the plug-in sets the various requested services, including creating authorization keys, and passes the relevant data to the application through an HTTP GET request of the associated CGI script. The Web server then executes the CGI script to start the application.

To be able to use the RX plug-in you need Netscape Navigator 3.0. Binaries for various systems can be found at:

```
http://home.netscape.com/comprod/mirror/client_download.html
```

To complete the installation of the Netscape plug-in, find the file named *libxrx.so.6.3* or *libxrx.sl.6.3* (or similar, depending on your platform) in <ProjectRoot>/lib (e.g. **/usr/X11R6.3/lib**) and copy it to either /usr/local/lib/netscape/plugins or $HOME/.netscape/plugins. Do not install the symlinks *libxrx.so* or *libxrx.sl*; they may confuse Netscape.

You should remove or comment out the line you may have previously added in your mailcap file to use the RX helper program, otherwise the plug-in will not be enabled. (The usual comment character for mailcap is "#".)

If you are already running Netscape Navigator, you need to exit and restart it after copying the plug-in library so the new plug-in will be found. Once this is done you can check that Navigator has successfully loaded the plug-in by checking the "About Plug-ins" page from the Help menu. This should show something like:

RX Plug-in

File name: /usr/guest/netscape/plugins/libxrx.sl.6.3

X Remote Activation Plug-in

Mime Type	Description	Suffixes	Enabled
application/x-rx	X Remote Activation Plug-in	xrx	Yes

The plug-in will be activated by Netscape Navigator as soon as you retrieve any document of the MIME type application/x-rx. Several samples are included in the distribution. The most basic one is xload. All you need to do is point your browser at the page:

```
http://your.web.server/xload.html
```

If something goes wrong check on the all the previous steps listed above and try again. Once xload is working you can try some of the other examples in the distribution such as bitmap.html or dtcm.html.

3.4.4. Trying Embedding With an Old X Server

The Netscape Navigator plug-in, *libxrx*, will work with an X server that does not contain the application group or security extensions. The application will be started as a separate top-level client.

If you wish to try out the embedding facilities without replacing your desktop X server, you may use the *Xnest* server.

A typical *Xnest* session would look like the following:

```
% Xnest :11
% xterm -display :11
```

These two commands start a ‘‘nested’’ server and a terminal emulator within that server. Your favorite window manager and Netscape Navigator can now be executed from the nested xterm window. You may wish to first disable access control in the nested server by running ‘‘xhost +’’ in the nested xterm.

3.4.5. Setting Up Your Own Applications To Run Over The Web

Based on the examples provided in the distribution it should be easy to set up your web server to run your own applications. Every application requires 3 additional files to identify it to Web browsers:

myapp.html	An HTML page to present the application embedded
myapp.rx	The RX document describing the application
myapp.pl	The CGI script to start the application

Note that the separate ‘‘.rx’’ file could be omitted by implementing the CGI script such that if it is invoked without a QUERY_STRING it will return the RX content. We decided not to do so in the distributed examples for purpose of clarity.

The xload demo provides a good starting point. Simply make a copy of each of the files xload.rx, xload.html, and xload.pl. Then look inside them for every instance of ‘‘xload’’ and change it to whatever is appropriate for your application.

You will not be able to run the dtcm demo unless you have *dtcm* (a CDE component) installed on your web server host. This example shows how a CGI script would look when an X Print server is requested.

The script *dtcm.pl* is, for that reason, slightly more complicated than other examples.

3.5. Security Extension

The SECURITY extension contains new protocol needed to provide enhanced X server security. This extension adds to the X protocol the concepts of ‘‘trusted’’ and ‘‘untrusted’’ clients. The trust status of a client is determined by the authorization used at connection setup. All clients using host-based authorization are considered ‘‘trusted’’. Clients using other authorization protocols may be either trusted or untrusted depending on the data included in the connection authorization phase.

The requests in the security extension permit a trusted client to create multiple authorization entries for a single authorization protocol. Each entry is tagged with the trust status to be associated with any client presenting that authorization.

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. Such a client may be written to catch these errors and continue operation.

When a client is untrusted, the server will 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.

The specification for the SECURITY extension is in **xc/doc/specs/Xext/security.tex** (LaTeX source) and **xc/doc/hardcopy/Xext/security.PS.Z** (compressed PostScript).

3.5.1. Untrusted Application Behavior

Most applications work normally when run as untrusted clients, but since the security extension changes the semantics of certain parts of the X protocol, it is no surprise that some clients behave differently when untrusted. We note the following significant behavior changes, separated into two categories: changes that we expect could disappear or mutate if the implementation were improved in a future release, and changes we expect are permanent, legitimate defenses against data loss or leakage.

3.5.1.1. Behaviors That Are Implementation-Dependent

The following behaviors when running the respective applications as untrusted are not mandated by the security design but are side effects of limitations in the current implementation.

oclock is square because the SHAPE extension hasn’t been marked secure yet. Similarly, Xaw applications that use oval buttons will have rectangular buttons instead.

Any application that depends on an extension other than XC-MISC, LBX, or BIG-REQUESTS will have different behavior, as no other extensions are currently marked secure. The core clients affected are *xieperf* and all the xkb utilities.

emacs exits with a Window error when trying to use the QueryPointer request on the root window when you click in a buffer.

FrameMaker, and *xwd -root* both exit with a Window error when trying to use the GetWindowAttributes request on a window manager frame window.

All the remaining changes are involved in some way with window properties. Some of these behaviors can be modified with changes to the SecurityPolicy file; see the *Xserver* man page.

Several clients exit with a Window error when trying to use the DeleteProperty request on various properties on the root window. These include *xcmsdb -remove*, *xprop -root -remove*, and *xstdcmap -delete*.

xprop exits with an Atom error when attempting to access protected properties.

The following two changes require, in addition, a “trusted selection intermediary” to provide selection transfer from untrusted to trusted clients (and vice-versa). R6.3 does not include such a trusted intermediary.

xterm exits with an Atom error when it tries to store the property value during a selection transfer (paste) to a trusted selection requester.

The “copy 0 to PRIMARY” button of *xcutsel* does not work.

Selection transfer from untrusted clients to trusted clients fails when the untrusted client attempts to use SendEvent to generate the SelectionNotify event for the requester. Most requesters will treat this as a transfer timeout and continue. Xt-based applications will create an additional Atom each time such a transfer is attempted.

3.5.1.2. Behaviors That Are Not Likely To Change

The following behaviors represent actions performed by the applications that are disallowed by design.

editres will fail when pointed at a trusted client when it tries to read window properties on a window owned by that client.

Xnest exits on startup with an Access error as it tries to use the ChangeKeyboardControl request.

The new *generate* option to *xauth* fails because untrusted applications are not allowed to create additional authorizations.

xhost cannot be used to modify the host access list.

xmag gets an unending stream of Drawable errors as it tries to use the PolyRectangle request on the root window. If you click to select a location to magnify, xmag gets a Drawable error as it tries to use the GetImage request on the root window. *xmag* could be modified to exit gracefully under these conditions.

netscape exits on startup with a Drawable error when trying to use the GetImage request on the root window.

xmodmap exits with an Access error when trying to use the ChangeKeyboardMapping request.

xset with the *b*, *c*, *led*, or *r* options exits with an Access error when trying to use the ChangeKeyboard-Control request. With the *bc* option, it can’t find the MIT-SUNDRY-NONSTANDARD extension and exits gracefully.

xsetroot exits with a Window error when trying to use the ChangeWindowAttributes request on the root window.

3.6. Application Group Extension

The application group extension (XC-APPGROUP) provides new protocol to implement Application Groups (“AppGroups”). The AppGroup facility allows other clients to share the SubstructureRedirect mechanism with the window manager. This allows another client called the “application group leader”, such as a web browser, to intercept a MapRequest made by a third application and reparent its window into the web browser before the window manager takes control. The AppGroup leader may also limit the screens and visuals available to the applications in the group.

Users who have an XC-APPGROUP enhanced X server and an RX plug-in for their Netscape Navigator web browser can run programs remotely over the web and have the output appear as part of the presentation in their web browser.

The only way for an application to become a member of an AppGroup is by using an authorization generated using the new security extension. Whenever an application connects to the server, the authorization that it used to connect is tested to see if it belongs to an AppGroup. This means that the Authorization data must be transmitted to the remote host where the application will be run. In the case of RX, HTTP is used to send the Authorization. Sites who have concerns about sending unencrypted authorization data such as MIT-MAGIC-COOKIE-1 via HTTP should configure their web servers and web browsers to use SHTTP or SSL.

The specification for the XC-APPGROUP extension is in **xc/doc/specs/Xext/AppGroup.mif** (FrameMaker interchange source) and **xc/doc/hardcopy/Xext/AppGroup.PS.Z** (compressed PostScript).

3.7. Print Extension

The print extension supports output to hardcopy devices using the core X drawing requests. The print extension adds requests for job and page control and defines how specific printer attributes are communicated between the server and printing clients. Printer attribute specifications are modeled after the ISO 10175 specification.

An X client that wants to produce hardcopy output will typically open a second connection to an X print server, produce a print job, and then close the print server connection. The print server may be the same process as the display server (the term ‘‘video server’’ is sometimes used) although the implementation provided in R6.3 does not completely support video and print servers in the same binary.

The specification for the print extension is in **xc/doc/specs/XPRINT/xp_proto.mif** (FrameMaker interchange source) and **xc/doc/hardcopy/XPRINT/xp_proto.PS.Z** (compressed PostScript). The library API specification is in **xc/doc/specs/XPRINT/xp_library.mif** (FrameMaker interchange source) and **xc/doc/hardcopy/XPRINT/xp_library.PS.Z** (compressed PostScript).

3.7.1. Running an X Print Server

The print server is simply an X server with the print extension and special DDX implementations. The X Print Server is started like any other X server.

Here is a sample command line for use with a typical configuration:

```
% Xprt :1 -ac
```

The options used in the example are:

:1 On a host that is running a video display server you will need to specify a different display from the default.

-ac Disable access control, since no simple mechanism for sharing keys is provided.

The X print server supports the following additional options:

-XpFile Points to the directory containing the print server configuration files.

XPCONFIGDIR
Environment variable specifying alternative location of the print server configuration files.

The print server, *Xprt*, is built only if the config option XprtServer is YES. Four printer DDXen are provided, each with a separate config option to control whether or not it will be included: XpRasterDDX, XpColorPclDDX, XpMonoPclDDX, XpPostScriptDDX; see **xc/config/cf/README**. XprtServer defaults to the value of BuildServer (i.e. *Xprt* will be built by default on all platforms that build a full X server). XpRasterDDX and XpMonoPclDDX default to NO. XpColorPclDDX and XpPostScriptDDX default to YES.

The print server is configured through a directory of configuration files that define printer model types and instances of printer models. An example configuration tree is provided in **xc/programs/Xserver/XpConfig/**. See also **xc/doc/specs/Xserver/Xprt.mif** (FrameMaker interchange source) and **xc/doc/hardcopy/Xserver/Xprt.PS.Z** (compressed PostScript) for further instructions on configuring *Xprt*.

3.7.2. Specifying The Print Server To A Client

By convention, clients locate the print server using the environment variable XPRINTER. The syntax of XPRINTER is an augmented DISPLAY; i.e.

    printerName@host:display

where ‘‘printerName’’ is one of the printer instances listed in the print server configuration files. The use of XPRINTER and its syntax is an application convention only; there is nothing in the supplied libraries that uses (or parses) this environment variable.

3.8. Proxy Management Protocol

The Proxy Management Protocol is an ICE based protocol that provides a way for application servers to easily locate proxy services such as the LBX proxy and the X firewall proxy.

Typically, a service called a ‘‘proxy manager’’ is responsible for resolving requests for proxy services, starting new proxies when appropriate, and keeping track of all of the available proxy services. The proxy manager strives to reuse existing proxy processes whenever possible.

The Proxy Management Protocol is described in **xc/doc/specs/PM/PM_spec**.

3.9. Configuration

As in R6.1, the top-level Makefile is no longer over-ridden by the first build. Instead a new file *xmakefile* is created. Thus is it not necessary to take any additional steps to reset the builds.

The file **xc/config/cf/README** provides more guidance on how to write an Imakefile, including a list of variables that may be set in an Imakefile. This file is strongly recommended reading for Imakefile authors.

The LaTeX text processor is supported as of R6.1. If you have LaTeX on your system, turn on HasLatex to have the MakeLatexDoc rule use it.

Also since R6.1, with System V Release 4 (SVR4) compilers we now use the −Xa (ANSI C with native extensions) compiler flag rather than −Xc (limit environment to that specified in the standard). This provides access to the full richness of the platform. Unfortunately, it also defines the preprocessor symbol __STDC__ to 0, instead of 1 as specified by the standard. Therefore we use ‘‘#ifdef __STDC__’’ in our sources rather than ‘‘#if __STDC__’’. On HP-UX systems we use the −Ae compiler option instead of

−Aa, also to access the full environment offered by the platform.

As in R6.1, the imake variables **InstallXdmConfig**, **InstallXinitConfig**, and **InstallAppDefFiles** suppress overwriting existing files; if the files didn't previously exist, the files are always installed. This interpretation makes bootstrapping a new system easier than in R6 and earlier releases.

A new configuration build option, GzipFontCompression, has been added to use *gzip* rather than *compress* for font compression. It defaults to NO.

The build creates a new directory **xc/exports** into which the header files, libraries, and certain build utility binaries are symlinked. This greatly simplifies Imakefile construction and supports multiple development projects (such as X, Motif, and CDE) on a single system.

Imake rules and template files for building Motif and CDE were contributed by the OSF CDE/Motif project and are included in R6.3.

3.10. Documentation

Additional X server internals documentation is provided in the **/xc/doc/specs/Xserver/** directory for the XC-APPGROUP and SECURITY extensions. An analysis and rationale for the SECURITY extension will also be found in that directory. Specifications for the other new standards are in **/xc/doc/specs/RX/**, **/xc/doc/specs/XPRINT/**, and **/xc/doc/specs/Xext/**.

3.11. Header Files

xc/include/Xos_r.h is a new header file to promote portable source code using thread-safe implementations of getpwnam, getpwuid, gethostbyname, gethostbyaddr, and getservbyname. It is not required by any X Consortium standard.

3.12. X Server

The security, LBX, printing, and AppGroup extensions are all new. In R6.3 only MIT-MAGIC-COOKIE-1 is supported in the security extension. Parts of the security policy are configured at run-time from the file **/usr/X11R6.3/lib/X11/xserver/SecurityPolicy**. Site-defined policy strings used by *xfwp* and rules for property access by untrusted clients are defined there. See the *Xserver* man page for full details.

3.12.1. New Device Support

Support has been added for the Sun TCX frame buffer as a dumb 8-bit frame buffer on Solaris 2.5.

New XFree86 servers based on XFree86 3.2 are included.

3.12.2. Internal Changes

The security extension provides new internal resource ID lookup interfaces that incorporate the access control lookup. In order to be declared secure and therefore be made available to untrusted clients, other extensions should, at a minimum, be changed to use these interfaces. Depending on what the extension does, more may need to be done in its implementation before it can appropriately be labeled ‘‘secure’’.

Refer to the documents **xc/doc/specs/Xserver/appgroup.ms** and **xc/doc/specs/Xserver/secint.tex** for implementation details of the application group and security extensions, respectively.

3.13. ICE Library Addition

To support proxy managers and firewall proxies using ICE on well-known TCP ports, an additional interface has been added to the ICE library. This new interface, IceListenForWellKnownConnections, has equivalent calling parameters to IceListenForConnections plus an ICE network id parameter.

3.14. Xlib Vertical Writing and User-Defined Characters

The Xlib output method implementation has been enhanced to support the XOM value drawing direction XOMOrientation_TTB_RTL. Vertical writing information and other locale specific information is read from the file <XLocaleDir>/%L/XLC_LOCALE where the XLocaleDir configuration option defaults to **/usr/X11R6.3/lib/X11/locale**.

The X[mb|wc]TextEscapement functions now return the text escapement in pixels for the vertical or horizontal direction depending on the XNOrientation XOCValue.

The X[mb|wc]DrawString functions will now render a character string in the vertical or horizontal direction depending on the XNOrientation XOCValue.

The Xlib NLS database implementation has been enhanced to support extended segments used for interchanging non-standard code sets. Support has been added for control sequences and encoding names used in extended segments and conversion of glyph indexes when interchanging data in extended segments.

3.15. Xt Geometry Management Debugger

Daniel Dardailler's "GeoTattler" code has been merged into the Xt Intrinsics library implementation. This is not a standard. If libXt is compiled with the XT_GEO_TATTLER symbol defined (currently there is no build configuration support to do this) then a "geoTattler" resource may be specified for any widget in an application. If the geoTattler resource for a widget instance is True then libXt will generate debugging information to stdout when the widget makes geometry change requests.

For example, if the resources specify:

```
myapp*draw.XmScale.geoTattler: ON
*XmScrollBar.geoTattler:ON
*XmRowColumn.exit_button.geoTattler:ON
```

then geometry management debugging information will be generated for all the XmScale children of the widget named draw, all the XmScrollBars, and the widget named exit_button in any XmRowColumn.

3.16. New Programs

There are new core programs *lbxproxy*, *proxymngr*, *xfindproxy*, *xfwp*, *Xprt*, and *xrx*.

lbxproxy — The *lbxproxy* program is used to ''translate'' X protocol to LBX protocol. It should be executed on the same host as the client application or on a host connected to the client host by a fast network. *lbxproxy* appears to the clients using it as another X server; that is, the clients connect through it using the conventional DISPLAY syntax, specifying the proxy host in place of the server. *lbxproxy* can be used stand-alone or in conjunction with *proxymngr* and *xfindproxy*. See the *lbxproxy* man page for further details.

proxymngr — *proxymngr* is a process that runs continuously to control other proxy applications, such as *lbxproxy* and *xfwp*. It maintains a list of active proxy processes and responds to queries from *xfindproxy*. See the *proxymngr* man pages for further details.

xfindproxy — *xfindproxy* is used to locate a running proxy process for a given network service, such as *lbxproxy* or *xfwp*, or to request that a proxy be started if one is not already running. *xfindproxy* communicates with *proxymngr* to perform the actual work.

xfwp — *xfwp* is the X firewall application proxy. It is designed to run on a network firewall host and relay X protocol between applications (typically outside the firewall) and the X server (inside the firewall). *xfwp* appears to the clients using it as another X server; that is, clients connect through it using the conventional DISPLAY syntax. *xfwp* will not do anything useful without *proxymngr* and *xfindproxy* or *xrx*. See the *xfwp* man page for further details.

Xprt — *Xprt* is the print server, built as part of the Xserver build if the **XprtServer** config option is YES. The print server supports printing to PostScript and PCL devices, as well as raster output to an *xwd* format file (and thence to any printer that *xpr* supports). The print extension was designed to be integrated with the ''video'' server in a single process but the R6.3 implementation does not support a combined video and print server. Details of configuration for *Xprt* are in **xc/doc/specs/Xserver/Xprt.mif** (FrameMaker interchange source) and **xc/doc/hardcopy/Xserver/Xprt.PS.Z** (compressed PostScript).

xrx, libxrx — *xrx* is the Web browser helper application that interprets documents in the RX MIME type to remotely launch applications via the Web. Its companion *libxrx* is a plug-in for Netscape Navigator 3.0 that supports in addition the capability to visually embed the remote applications in the associated browser Web page window. See the *xrx* man page for further details.

3.16.1. Using The LBX Proxy

The implementation of *lbxproxy* provided here will support an arbitrary number of clients connecting to the same X server. A separate *lbxproxy* process is required for each separate X server process. A typical command line to invoke *lbxproxy* is

```
lbxproxy :22 -display myhost:0
```

This command runs a proxy with the X server ''myhost:0'' as the target. Clients must connect to the proxy using ''proxyhost:22'' as the DISPLAY. The .Xauthority file for these clients must contain an entry for server ''proxyhost:22'' with the same MIT-MAGIC-COOKIE as ''myhost:0'', or the X server must be configured to permit connections from any host on the network.

Here is an example showing how to setup the appropriate .Xauthority entries:

```
% lbxproxy :22 -display myws:0
% xauth list
myws:0  MIT-MAGIC-COOKIE-1  7fd231ccdce2
myws/unix:0  MIT-MAGIC-COOKIE-1  7fd231ccdce2
% xauth -f $HOME/proxyauth add proxyhost:22 .  7fd231ccdce2
xauth:  creating new authority file /usr/myself/proxyauth
% xauth -f $HOME/proxyauth  add proxyhost/unix:22 .  7fd231ccdce2
% setenv XAUTHORITY $HOME/proxyauth
```

In this example, the authorization token for display 0 is copied into a new file ‘‘proxyauth’’ and associated with the LBX proxy server display number (22). The new authority file may then be copied to another host and used as the value of the XAUTHORITY environment variable.

The *proxymngr* daemon is usually configured to invoke *lbxproxy* automatically when a user or a CGI script runs *xfindproxy -name LBX*.

See the *lbxproxy* man page for further details.

3.17. Major Additions to Existing Programs

The *generate* option of *xauth* is used to obtain additional authorization tokens for client connections. These authorization tokens may specify that the client using them is to be restricted in the operations that may be performed in the X server. The authorization tokens may be independently revoked. Refer to the SECURITY extension for further details on authorizations.

The *xauth* man page gives full details on the new generate command. Here is an example use:

```
xauth -f untrusted-auth-file g :0 . timeout 0
setenv XAUTHORITY untrusted-auth-file
```

This will cause *xauth* to contact server ‘‘:0’’ to get a long-lasting untrusted cookie which it then stores in untrusted-auth-file. By setting XAUTHORITY to point to untrusted-auth-file, subsequent applications run from this shell to server :0 will be untrusted. The ‘‘g’’ is short for ‘‘generate’’, and the ‘‘.’’ is short for ‘‘MIT-MAGIC-COOKIE-1’’. If you omit the -f argument, *xauth* will use $XAUTHORITY (or ˜/.Xauthority), which may not be what you want, especially if you are creating an untrusted auth. This is because *xauth* will replace the trusted auth in ˜/.Xauthority (put there by xdm) with the untrusted one, preventing you from making any further trusted connections to the server.

The *xterm* terminal emulator now supports the active icon mode that was in X version 10 Release 4. See the *xterm* man page for further details. There is support in the *xterm* source to build xterm without the active icon mode for those who may care for some reason to not provide it.

3.18. ANSIfication

As noted previously under ‘‘Configuration Files’’, for pragmatic reasons we changed the way we use __STDC__ to test for standard C compilers. R6.1 was officially the last release that supported traditional K&R C. R6.3 assumes a standard C compiler and environment. We have not intentionally removed any K&R C support from old code; most of the release will continue to build on older platforms.

4. Known Bugs

There are no examples in this release showing how to use the print extension. CDE 2.1 has several such applications.

lbxproxy fails to start on SCO Open Server.

x11perf running through *lbxproxy* will tickle a drawing bug in cfb-based X servers that causes some lines and curves to be drawn to the wrong coordinates and outside the window boundaries. Use the *-nogfx* option to *lbxproxy* as a workaround on affected servers.

If *proxymngr* exits abnormally all managed proxies die.

Documentation is missing on how to use the vertical writing and user-defined character support.

Documentation is sparse on how to configure *Xprt*.

There are no example fonts in the release with vertical text escapement (“vertical writing fonts”).

5. Filing Bug Reports

If you find a reproducible bug in software in the **xc/** directory, or find bugs in the **xc** documentation, please send a bug report to The Open Group using the form in the file **xc/bug-report** and this destination address:

xbugs@x.org

Please try to provide all of the information requested on the form if it is applicable; the little extra time you spend on the report will make it much easier for someone to reproduce, find, and fix the bug.

Bugs in the contributed software that is available on the net are not handled on any official basis. Consult the documentation for the individual software to see where (if anywhere) to report the bug. Many authors of contributed software subscribe to the mailing list “contrib-bugs” hosted at x.org, so this might be a useful place to report bugs. (To subscribe to contrib-bugs yourself, send email to contrib-bugs-request@x.org.)

6. Acknowledgements

Release 6.3 of X Version 11 was brought to you by the X staff at the X Consortium, Inc.: Donna Converse (emeritus), Jim Fournier, Stephen Gildea (emeritus), Kaleb Keithley, Matt Landau (emeritus), Arnaud Le Hors, Ralph Mor (emeritus), Bob Scheifler, Ralph Swick, Ray Tice, Mark Welch (emeritus), and Dave Wiggins (emeritus). Kevin Samborn and George Tsang (emeritus) of the CDE staff at X Consortium, Inc. worked hard on the print extension, including the PostScript driver; David Kaelbling of the CDE staff converged the X, Motif, and CDE imake/config support and helped with Xos_r.h; and Daniel Dardailler (emeritus) of the CDE staff contributed the libXt geometry tracing code. Also, contractors Reed Augliere, Roger Helmendach (Liberty Systems), and Ann Pichey each worked on critical components.

Several companies and individuals have cooperated and worked extremely hard to make this release a reality, and our thanks go out to them. You will find many of them listed in the acknowledgements in the individual specifications.

Ken Raeburn of XFree86 and Cygnus Support contributed the gzip font compression support.

The Common Desktop Environment sponsors Digital Equipment Corp, Fujitsu, Hewlett-Packard, Hitachi, IBM, Novell, and SunSoft jointly contributed the print extension and the Xlib vertical writing and user-defined character support. Axel Deininger, Harry Phinney, Tom Gilg, Charles Prince, and Jim Miller all from Hewlett-Packard did the print extension and PCL and raster drivers. Fujitsu did the Xlib vertical writing and user-defined character support.