What's New with X11R6 for IRIX 6.0 and 5.3

based on

X Window System, Version 11, Release 6

Release Notes

Stephen Gildea
X Consortium

SGI Annotations: *Mark J. Kilgard*
Silicon Graphics, Inc.

X Consortium version date: May 16, 1994
SGI version date: July 11, 1994
$Revision: 1.2 $

This document is based on the releases notes accompanying the X11R6 release from the X Consortium. The original text is annotated with text in italics (like this) explaining how X11R6 is supported for IRIX 6.0 and 5.3.

The main difference between the IRIX 6.0 and 5.3 versions of X11R6 is the existence of 64-bit X shared libraries in /usr/lib64 in IRIX 6.0. Otherwise, the two operating system releases will be largely identical in their X11R6 support.

Because IRIX 5.3 will release after IRIX 6.0, it should be expected that IRIX 5.3 will have further bug fixes. As an X Consortium member, Silicon Graphics is privy to official bug fixes from the X Consortium. Reported bugs in the public X11R6 release are likely to be resolved in the IRIX 6.0 and 5.3 releases.

All the X programs supplied with IRIX 6.0 and 5.3 will be 32-bit. This of course includes the X server.

1. What Is New in Release 6

This section describes changes in the X Consortium distribution since Release 5. Release 6 contains much new functionality in many areas. In addition, many bugs have been fixed. However, in the effort to develop the new technology in this release, some bugs, particularly in client programs, did not get fixed.

Except where noted, all libraries, protocols, and servers are upward compatible with Release 5. That is, R5 clients and applications should continue to work with R6 libraries and servers.

1.1. New Standards

The following are new X Consortium standards in Release 6. Each is described in its own section below.

- X Image Extension
- Inter-Client Communications Conventions Manual (update)
- Inter-Client Exchange Protocol
- Inter-Client Exchange Library
- X Session Management Protocol
- X Session Management Library
- Input Method Protocol
- X Logical Font Descriptions (update)
- SYNC extension
- XTEST extension
- PEX 5.1 Protocol (released after R5)
- PEXlib (released after R5)
- BIG-REQUESTS extension
- XC-MISC extension

IRIX 6.0 and 5.3 do not provide any support for the X Image Extension. The XC-MISC, BIG-REQUESTS, and SYNC extensions are supported in the client-side extension library (libXext) but are not supported by the Silicon Graphics X server. Support for the X Image Extension is not currently planned. Server support for the XC-MISC, BIG-REQUESTS, and SYNC extensions will likely be supported when the X server is upgraded to be X11R6 based.

Client-side support for the XC-MISC, BIG-REQUESTS, and SYNC extensions means IRIX 6.0 and 5.3 programs will be able to use these extensions if they connect to an X11R6 X server that supports these extensions.

1.2. XIE (X Image Extension)

IRIX 6.0 and 5.3 have no client or server-side support for XIE. Support for the XIE is not currently planned.

The sample implementation in Release 6 is a complete implementation of full XIE 5.0 protocol, except for the following techniques that are excluded from the SI:

ColorAlloc:	Match, Requantize
Convolve:	Replicate
Decode:	JPEG lossless
Encode:	JPEG lossless
Geometry:	AntialiasByArea, AntialiasByLowpass

xieperf exercises the server functionality; it provides unit testing and a reasonable measure of multi-element photoflo testing.

A draft standard of the XIElib specification is included in this release and is open for Public Review. The XIElib code matches the 5.0 protocol.

The JPEG compression and decompression code is based on the Independent JPEG Group's (IJG) JPEG software, Release 4. This software provides baseline Huffman DCT encoding as defined by ISO/IEC DIS 10918-1, ''Digital Compression and Coding of Continuous-tone Still Images, Part 1: Requirements and guidelines'', and was chosen as a basis for our implementation of JPEG compression and decompression primarily because the IJG's design goals matched ours for the implementation of the XIE SI: achieve portability and flexibility without sacrificing performance. Less than half of the files distributed by the IJG have been incorporated into the XIE SI. The IJG's software is made available with restrictions; see **xc/programs/Xserver/XIE/mixie/jpeg/README**.

1.3. Inter-Client Communications Conventions Manual

The Inter-Client Communications Conventions Manual are essentially a set of clarifications. IRIX 5.3 and 6.0 do nothing special to support version 2.0 of ICCCM other than changes to libXt (see section 1.18).

Release 6 includes version 2.0 of the ICCCM. This version contains a large number of changes and clarifications in the areas of window management, selections, session management, and resource sharing.

1.3.1. Window Management

The circumstances under which the window manager is required to send synthetic ConfigureNotify events have been clarified to ensure that any ConfigureWindow request issued by the client will result in a ConfigureNotify event, either from the server or from the window manager. We have also added advice about how a client should inspect events so as to minimize the number of situations where it is necessary to use the TranslateCoordinates request.

The window_gravity field of WM_NORMAL_HINTS has a new value, StaticGravity, which specifies that the window manager should not shift the client window's location when reparenting the window.

The base size in the WM_NORMAL_HINTS property is now to be included in the aspect ratio calculation.

The WM_STATE property now has a formal definition (it was previously only suggested).

1.3.2. Selections

We have clarified the CLIENT_WINDOW, LENGTH, and MULTIPLE targets. We have also added a number of new targets for Encapsulated PostScript and for the Apple Macintosh PICT structured graphics format. We have also defined a new selection property type C_STRING, which is a string of non-zero bytes. (This is in contrast to the STRING type, which excludes many control characters.)

A selection requester can now pass parameters in with the request.

Another new facility is manager selections. This use of the selection mechanism is not to transfer data, but to allow clients known as *managers* to provide services to other clients. Version 2.0 also specifies that window managers should hold a manager selection. At present, the only service defined for window managers is to report the ICCCM version number to which the window manager complies. Now that this facility is in place, additional services can be added in the future.

1.3.3. Resource Sharing

A prominent new addition in version 2.0 is the ability of clients to take control of colormap installation under certain circumstances. Earlier versions of the ICCCM specified that the window manager had exclusive control over colormap installation. This proves to be inconvenient for certain situations, such as when a client has the server grabbed. Version 2.0 allows clients to install colormaps themselves after having informed the window manager. Clients must hold a pointer grab for the entire time they are doing their own colormap installation.

Version 2.0 also clarifies a number of rules about how clients can exchange resources. These rules are important when a client places a resource ID into a hints property or passes a resource ID through the selection mechanism.

1.3.4. Session Management

Some of the properties in section 5 of ICCCM 1.1 are now obsolete, and new properties for session management have been defined.

1.4. ICE (Inter-Client Exchange)

IRIX 6.0 and 5.3 have ICE support rolled into libXt. There is no separate libICE library to link with. A symbolic link makes /usr/lib/libICE.so and /usr/lib/libICE.a point to /usr/lib/libXt.so and /usr/lib/libXt.a respectively.

IRIX 6.0 and 5.3 do supply the iceauth program.

ICE provides a common framework to build protocols on. It supplies authentication, byte order negotiation, version negotiation, and error reporting conventions. It supports multiplexing multiple protocols over a single transport connection. ICElib provides a common interface to these mechanisms so that protocol implementors need not reinvent them.

An *iceauth* program was written to manipulate an ICE authority file; it is very similar to the *xauth* program.

1.5. SM (Session Management)

IRIX 6.0 and 5.3 have SM support (like ICE support) rolled into libXt. There is no seperate libSM library to link with. A symbolic link makes /usr/lib/libSM.so and /usr/lib/libSM.a point to /usr/lib/libSM.so and /usr/lib/libSM.a respectively. The Indigo Magic desktop does not support SM-style session management.

IRIX 6.0 and 5.3 do not supply the xsm program.

The X Session Management Protocol (XSMP) provides a uniform mechanism for users to save and restore their sessions using the services of a network-based session manager. It is built on ICE. SMlib is the C interface to the protocol. There is also support for XSMP in Xt.

A simple session manager, *xsm* is included in **xc/workInProgress/xsm**.

A new protocol, rstart, greatly simplifies the task of starting applications on remote machines. It is built upon already existing remote execution protocols such as *rsh*. The most important feature that it adds is the ability to pass environment variables and authentication data to the applications being started.

1.6. Input Method Protocol

IRIX 6.0 and 5.3 do support various input methods. A Kanji (Japanese) input method server is available in the japanese_eoe images.

The format of Xlib's X Locale database has been changed for X11R6. The R6 version of Xlib only understands the new X Locale database. Since Xlib (libX11) was only shipped as a dynamic shared object (.so) in previous releases of IRIX, the R5 X Locale database is no longer being shipped. The old R5 X Locale database resided in /usr/lib/X11/nls. The new location for the R6 X Locale database is /usr/lib/X11/locale.

Some languages need complex pre-editing input methods, and such an input method may be implemented separately from applications in a process called an Input Method (IM) Server. The IM Server handles the display of pre-edit text and the user's input operation. The Input Method (IM) Protocol standardizes the communication between the IM Server and the IM library linked with the application.

The IM Protocol is a completely new protocol, based on experience with R5's sample implementations. The following new features are added, beyond the mechanisms in the R5 sample implementations:

- The IM Server can support any of several transports for connection with the IM library.
- Both the IM Server and clients can authenticate each other for security.
- A client can connect to an IM Server without restarting even if it starts up before the IM Server.
- A client can initiate string conversion to the IM Server for re-conversion of text.
- A client can specify some keys as hot keys, which can be used to escape from the normal input method processing regardless of the input method state.

The R6 sample implementation for the internationalization support in Xlib has a new pluggable framework, with the capability of loading and switching locale object modules dynamically. For backward compatibility, the R6 sample implementation can support the R5 protocols by switching to IM modules supporting those protocols. In addition, the framework provides the following new functions and mechanisms:

X Locale database format:

An X Locale database format is defined, and the subset of a user's environment dependent on language is provided as a plain ASCII text file. You can customize the behavior of Xlib without changing Xlib itself.

ANSI C and non-ANSI C bindings

The common set of methods and structures are defined, which bind the X locale to the system locales within libc, and a framework for implementing this common set under non-ANSI C base system is provided.

Converters

The sample implementation has a mechanism to support various encodings by pluggable converters, and provides the following converters:

- Light weight converter for C and ISO 8859
- Generic converter (relatively slow) for other encoding
- High performance converter for Shift-JIS and EUC
- Converter for UCS-2 defined in ISO/IEC 10646-1

You can add your converter using this mechanism for your specific performance requirement.

Locale modules

The library is implemented such that input methods and output methods are separated and are independent of each other. Therefore, an output-only client does not link with the IM code, and an input-only client does not link with the OM code. Locale modules can be loaded on demand if the platform supports dynamic loading.

Transport Layer

There are several kinds of transports for connection between the IM library and the IM Server. The IM Protocol is independent of a specific transport layer protocol, and the sample implementation has a mechanism to permit an IM Server to define the transports which the IM Server is willing to use. The sample implementation supports transport over the X protocol, TCP/IP and DECnet.

There are IM Servers for Japanese and for Korean, internationalized clients using IM services, and an IM Server developer's kit in contrib. The IM Server developer's kit hides the details of the IM Protocol and the transport layer protocols, and hides the differences between the R5 and R6 protocols from the IM Server developer, so that an IM developer has an easier task in developing new IM Servers.

1.7. X Logical Font Description

The Silicon Graphics X server and font server supplied with IRIX 6.0 and 5.3 do not support the new enhanced XLFD features.

The X Logical Font Description has been enhanced to include general 2D linear transformations, character set subsets, and support for polymorphic fonts. See **xc/doc/specs/XLFD/xlfd.tbl.ms** for details.

1.8. SYNC extension

IRIX 6.0 and 5.3 support the client-side API in the X extension library (libXext) for the SYNC extension but the Silicon Graphics X server does not support the extension itself.

The Synchronization extension lets clients synchronize via the X server. This eliminates the network delays and the differences in synchronization primitives between operating systems. The extension provides a general Counter resource; clients can alter the value of a Counter, and can block their execution until a Counter reaches a specific threshold. Thus, for example, two clients can share a Counter initialized to zero, one client can draw some graphics and then increment the Counter, and the other client can block until the Counter reaches a value of one and then draw some additional graphics.

1.9. BIG-REQUESTS extension

IRIX 6.0 and 5.3 has client-side support in Xlib (libX11) for the BIG-REQUESTS extension but the

Silicon Graphics X server does not support the extension itself. OpenGL has its own solution for handling very large requests and does not need the BIG-REQUESTS extension.

The standard X protocol only allows requests up to 2^{18} bytes long. A new protocol extension, BIG-REQUESTS, has been added that allows a client to extend the length field in protocol requests to be a 32-bit value. This useful for PEX and other extensions that transmit complex information to the server.

1.10. XC-MISC extension

IRIX 6.0 and 5.3 has client-side support in Xlib (libX11) for the XC-MISC extension but the Silicon Graphics X server does not support the extension itself.

A new extension, XC-MISC, allows clients to get back ID ranges from the server. Xlib handles this automatically under the covers. This is useful for long-running applications that use many IDs over their lifetime.

1.11. XTEST extension

IRIX 6.0 and 5.3 both support the XTEST extension in both the client-side X extension library (libXext) and in the Silicon Graphics X server. The XTEST extension was already supported in IRIX 5.2.

The XTEST extension, which first shipped as a patch to Release 5, is included.

1.12. Tree Reorganization

The X11R6 tree reorganization only affects the source code organization; it is not visible to a user of IRIX 6.0 or 5.3.

Many of the directories under **xc/** (renamed from **mit/**) have been moved. See the section **The XC Tree** for the new layout. The reorganization has simplified dependencies in the build process. Once you get used to the new layout, things will be easier to find.

Various filenames have been changed to minimize name conflicts on systems that limit file names to eight characters, a period, and three more characters. Conflicts remain for various header (.h) files.

1.13. Configuration Files

IRIX 6.0 and 5.3 users of Imakefiles should be aware the configuration files have changed a bit between X11R5 and X11R6. This may mean users will have problems with some Imakefiles. Most problems with Makefiles can probably be resolved by doing a "make Makefile ; make Makefiles" to rebuild the Makefiles from the current configuration files. Using xmkmf is the best way to regenerate a new Makefile from scratch. The -64 and -32 options will be supported by xmkmf to control if a Makefile for building 64-bit (IRIX 6.0 only) or 32-bit binaries will be generated.

The configuration files have changed quite a bit, we hope in a mostly compatible fashion. The main config files are now in **xc/config/cf**, imake sources are in **xc/config/imake**, and makedepend sources are in **xc/config/makedepend**. The *lndir* program (for creating link trees) is in **xc/config/util**; there is a **Makefile.ini** in that directory that may be useful to get *lndir* built the first time (before you build the rest of the tree).

The rules for building libraries have changed a lot; it is now much easier to add a new library to the system.

The selection of *vendor***.cf** file has moved from **Imake.tmpl** to a new **Imake.cf**.

The config variable that was called ServerOSDefines in R5 has been renamed to ServerExtraDefines, and applies globally to all X server sources. The variable ServerOSDefines now applies just to the os directory of the server.

There are a number of new config variables dealing with C++, all of which have ‘‘Cplusplus’’ in their names.

‘‘#’’ should no longer be thought of as a valid comment character in Imakefiles; use ‘‘XCOMM’’ instead.

There are new variables (e.g., HasPoll, HasBSD44Sockets, ThreadedX) and rules (SpecialCObjectRule). Read **xc/config/cf/README** for details.

The way libraries get built has changed: the unshared library .o’s are now placed in a subdirectory rather than the shared library .o’s.

Multi-threaded programs can often just include **Threads.tmpl** in their **Imakefile** to get the correct compile-time defines and libraries.

1.14. Kerberos

IRIX 6.0 and 5.3 will not have any support for Kerberos. This situation may change in future releases due to customer demand for Kerberos.

There is a new authorization scheme for X clients, MIT-KERBEROS-5. It implements MIT’s Kerberos Version 5 user-to-user authentication. See the *Xsecurity* manual page for details on how Kerberos works in X. As with any other authentication protocol, *xdm* sets it up at login time, and Xlib uses it to authenticate the client to the X server.

If you have Kerberos 5 on your system, set the HasKrb5 config variable in **site.def** to YES to enable Kerberos support.

1.15. X Transport Library (xtrans)

IRIX 6.0 and 5.3 have the X Transport Library built into the libraries listed below (excepting the X server) but it does not change the operation of any of the libraries listed below.

IRIX 6.0 and 5.3 continue to not support DECnet transports.

The X Transport Library is intended to combine all system and transport specific code into a single place in the source tree. This API should be used by all libraries, clients and servers of the X Window System. Note that this API is *not* an X Consortium standard; it is merely in internal part of our implementation. Use of this API should allow the addition of new types of transports and support for new platforms without making any changes to the source except in the X Transport Interface code.

The following areas have been updated to use xtrans:

lib/X11 (including the Input Method code)
lib/ICE
lib/font/fc
lib/FS
XServer/os
xfs/os

The XDMCP code in xdm and the X server has not been modified to use xtrans.

No testing has been done for DECnet.

1.16. Xlib

IRIX 6.0 and IRIX 5.3 do NOT support multi-threaded access to a single display connection. Silicon Graphics customers desiring multi-threaded use of X should be directed to use multiple open X connections. Even this is not entirely safe for some Xlib (libX11) operations.

Programs that need -DXLIB_ILLEGAL_ACCESS should not be expected to run correctly from release to release. The offending code should definitely be fixed to guarantee compatibility.

The XInternAtoms and XGetAtomNames routines were already supported in IRIX 5.2.

Support for Kerberos does not exist in SGI's Xlib (libX11) implementation (see section 1.14).

Xlib now supports multi-threaded access to a single display connection. Xlib functions lock the display structure, causing other threads calling Xlib functions to be suspended until the first thread unlocks. Threads inside Xlib waiting to read to or write from the X server do not keep the display locked, so for example a thread hanging on XNextEvent will not prevent other threads from doing output to the server.

Multi-threaded Xlib runs on SunOS 5.3, DEC OSF/1 1.3, Mach 2.5 Vers 2.00.1, AIX 2.3, and Microsoft Windows NT 3.1. Locking for Xcms and I18N support has not been reviewed. A version of ico that can be compiled to use threads is in **contrib/programs/ico**.

The Display and GC structures have been made opaque to normal application code; references to private fields will get compiler errors. You can work around some of these by compiling with −DXLIB_ILLEGAL_ACCESS, but better to fix the offending code.

The Xlib implementation has been changed to support a form of asynchronous replies, meaning that a request can be sent off to the server, and then other requests can be generated without waiting for the first reply to come back. This is used to advantage in two new functions, XInternAtoms and XGetAtomNames, which reduce what would otherwise require multiple round trips to the server down to a single round trip. It is also used in some existing functions, such as XGetWindowAttributes, to reduce two round trips to just one.

Lots of Xlib source files were renamed to fit better on systems with short filenames. The ‘‘X’’ prefix was dropped from most file names, and ‘‘CIE’’ and ‘‘TekHVC’’ prefixes were dropped.

Support for using poll() rather than select() is implemented, selected by the HasPoll config option.

The BIG-REQUESTS extension is supported.

The following Xlib functions are new in Release 6:

XInternAtoms, XGetAtomNames
XExtendedMaxRequestSize
XInitImage
XReadBitmapFileData
IsPrivateKeypadKey
XConvertCase
XAddConnectionWatch, XRemoveConnectionWatch, XProcessInternalConnection
XInternalConnectionNumbers
XInitThreads, XLockDisplay, XUnlockDisplay

XOpenOM, XCloseOM
XSetOMValues, XGetOMValues
XDisplayOfOM, XLocaleOfOM

XCreateOC, XDestroyOC
XOMOfOC
XSetOCValues, XGetOCValues
XDirectionalDependentDrawing, XContextualDrawing
XRegisterIMInstantiateCallback, XUnregisterIMInstantiateCallback
XSetIMValues

XAllocIDs
XESetBeforeFlush
_XAllocTemp, _XFreeTemp

Support for MIT-KERBEROS-5 has been added.

1.17. Internationalization

As discussed earlier (in section 1.6), the X Locale database format used in X11R5's Xlib (libX11) has been changed. Since SGI only supplied Xlib as a dynamic shared object (.so), IRIX 6.0 and 5.3 will not supply the old X Locale database found in /usr/lib/X11/nls. Instead, only the new X11R6 X Locale database found in /usr/lib/X11/locale is supplied.

Internationalization (also known as I18N, there being 18 letters between the *i* and *n*) of the X Window System, which was originally introduced in Release 5, has been significantly improved in R6. The R6 I18N architecture follows that in R5, being based on the locale model used in ANSI C and POSIX, with most of the I18N capability provided by Xlib. R5 introduced a fundamental framework for internationalized input and output. It could enable basic localization for left-to-right, non-context sensitive, 8-bit or multi-byte codeset languages and cultural conventions. However, it did not deal with all possible languages and cultural conventions. R6 also does not cover all possible languages and cultural conventions, but R6 contains substantial new Xlib interfaces to support I18N enhancements, in order to enable additional language support and more practical localization.

The additional support is mainly in the area of text display. In order to support multi-byte encodings, the concept of a FontSet was introduced in R5. In R6, Xlib enhances this concept to a more generalized notion of output methods and output contexts. Just as input methods and input contexts support complex text input, output methods and output contexts support complex and more intelligent text display, dealing not only with multiple fonts but also with context dependencies. The result is a general framework to enable bi-directional text and context sensitive text display.

1.18. Xt

As discussed earlier (in sections 1.4 and 1.5), ICE and SM routines are actually included in Xt (libXt) for IRIX 6.0 and 5.3.

IRIX 6.0 and IRIX 5.3 do NOT support multi-threaded use of Xt (libXt) Silicon Graphics customers desiring multi-threaded use of X should be directed to use multiple open X connections. Even this is not entirely safe for some Xt and Xlib (libX11) operations.

The Silicon Graphics Xt implementation continues to support ''schemes'' which allow better customization of colors and fonts. The IRIX 6.0 schemes support is the same as what was supported in IRIX 5.2; IRIX 5.3 has improved schemes functionality.

The IRIX 6.0 and 5.3 version of Xt are compiled to support the R5 buggy behavior of GetValues (discussed below) and supports Motif backward compatibility hacks.

Support has been added for participation in session management, with callbacks to application functionality in response to messages from the session manager.

The entire library is now thread-safe, allowing one thread at a time to enter the library and protecting global data as necessary from concurrent use.

Support is provided for registering event handlers for events generated by X protocol extensions, and for dispatching those events to the appropriate widget.

A mechanism has also been added for dispatching events for non-widget drawables (such as pixmaps used within a widget) to a widget.

Two new widget methods for instance allocation and deallocation allow widgets to be treated as C++ objects in a C++ environment.

A new interface allows bundled changes to the managed set of children of a Composite, reducing the visual disruption of multiple changes to geometry layout.

Several new resources have been added to Shell widgets, making the library compliant with the Release 6 ICCCM. Parameterized targets of selections (new in Release 6) and the MULTIPLE target are supported with new APIs.

Safe handling of POSIX signals and other asynchronous notifications is now provided.

A hook has been added to give notification of blocking in the event manager.

The client will be able to register callbacks on a per-display basis for notification of a large variety of operations in the X Toolkit. This feature is useful to external agents such as screen readers.

New String resource converters: XtStringToGravity and XtCvtStringToRestartStyle.

The file search path syntax has a new %D substitution that inserts the default search path, making it easy to prepend and append to the default search path.

The Xt implementation allows a configuration choice of poll or select for I/O multiplexing, selectable at compile time by the HasPoll config option.

The Release 6 Xt implementation requires Release 6 Xlib. Specifically, it uses the following new Xlib features: XInternAtoms instead of multiple XInternAtom calls where possible, input method support (Xlib internal connections), and tests for the XVisibleHint in the flags of XWMHints.

When linking with Xt, you now need to also link with SMlib and ICElib. This is automatic if you use the XTOOLLIB make variable or XawClientLibs *imake* variable in your **Imakefiles**.

This implementation no longer allows NULL to be passed as the value in the name/value pair in a request to XtGetValues. The default behavior is to print the error message ‘‘NULL ArgVal In XtGetValues’’ and exit. To restore the R5 behavior, set the config variable **GetValuesBC** in **site.def**. The old behavior was never part of the Xt specification, but some applications erroneously rely on it.

Motif 1.2 defines the types XtTypedArg and XtTypedArgList in VaSimpleP.h. These types are now defined in IntrinsicP.h. To work around the conflict, in Motif VaSimple.c, if IntrinsicP.h is not already included before VaSimpleP.h, do so. In VaSimpleP.h, fence off the type declarations with #if (XT_REVISION < 6) and #endif.

See Chapter 13 of the Xt specification for more details.

1.19. Xaw

The R6 version of the Athena widget library (libXaw) cannot be made fully compatible with the R5 version (the widget structures have been expanded in a way that would make programs which derive widgets from existing Athena widgets broken). The old R5 libXaw.so is shipped as /usr/lib/libXaw.so.1 and the new R6 libXaw.so is shipped as /usr/lib/libXaw.so.2 with a symbolic link from /usr/lib/libXaw.so pointing to libXaw.so.2.

The Clock, Logo, and Mailbox widgets continue to be supported in the IRIX 6.0 and IRIX 5.3 version of *libXaw.*

Some minor bugs have been fixed. Please note that the Athena Widgets have been and continue to be low on our priority list; therefore many bugs remain and many requests for enhancements have not been implemented.

Text and Panner widget translations have been augmented to include keypad cursor keysyms in addition to the normal cursor keysyms.

The Clock, Logo, and Mailbox widgets have moved to their respective applications.

Internationalization support is now included. Xaw uses native widechar support when available, otherwise it uses the Xlib widechar routines. Per system specifics are set in XawI18n.h.

The shared library major version number on SunOS 4 has been incremented because of these changes.

1.19.1. AsciiText

The name AsciiText is now a misnomer, but has been retained for backward compatibility. A new resource, XtNinternational, has been added. If the value of the XtNinternational resource is False (the default) AsciiSrc and AsciiSink source and sink widgets are created, and the widget behaves as it did for R5. If the value is True, MultiSrc and MultiSink source and sink widgets are created. The MultiSrc widget will connect to an Input Method Server if one is available, or if one isn't available, it will use an Xlib internal pseudo input method that, at a minimum, does compose processing. Application programmers who wish to use this feature will need to add a call to XtSetLanguageProc to their programs.

The symbolic constant FMT8BIT has been changed to XawFmt8Bit to be consistent with the new symbolic constant XawFmtWide. FMT8BIT remains for backwards compatibility, however its use is discouraged as it will eventually be removed from the implementation. See the Xaw manual for details.

1.19.2. Command, Label, List, MenuButton, Repeater, SmeBSB, and Toggle

Two new resources have been added, XtNinternational and XtNfontSet. If XtNinternational is set to True the widget displays its text using the specified fontset. See the Xaw manual for details.

1.20. PEX

IRIX 6.0 and 5.3 support PEX 5.1. A client-side implementation of PEXlib (libPEX5) is provided. And the Silicon Graphics X server supports a dynamically loadable PEX 5.1 extension that renders via OpenGL (no installed by default). Silicon Graphics does not use the X Consortium PEX sample implementation to implement its PEX implementation. No support is provided for the PHIGS API and the PHIGS workstation subset of PEX is not supported by SGI's PEX implementation. No PEX demo programs are supplied.

In discussing PEX it is important to understand the nature of 3D graphics and the purpose of the existence of the PEX SI. The type of graphics for which PEX provides support, while capable of being done in software, is most commonly found in high performance hardware. Creation and maintenance of software rendering code is costly and resource consumptive. The original Sample Implementation for the PEX Protocol 5.0 was primarily intended for consumption by vendors of the X Consortium who intended to provide PEX products for sale. This implementation was intended to be fairly complete however it was understood that vendors who intended to commercialize it would dispose of portions of it, often fairly substantial ones. It was therefore understood that functionality most likely to be disposed of by them might be neglected in the development of a Sample Implementation. As PEX is now a fairly mature standard distributed by most if not all major vendors, and the standard itself has evolved from the 5.0 protocol level to the 5.1 protocol level, the X Consortium and its supporting vendors have recognized a need to focus on certain portions of the PEX technology while deemphasizing others.

This release incorporates PEX functionality based upon the PEX 5.1 level protocol. The PEX Sample Implementation (SI) is composed of several parts. The major components are the extension to the X Server, which implements the PEX 5.1 protocol, and the client side API, which provides a mechanism by which clients can generate PEX protocol.

The API now provided with the PEX-SI is called PEXlib. This is a change from R5 which shipped an API based upon the ISO IS PHIGS and PHIGS PLUS Bindings. That API has been moved to contrib in favor of the PEXlib API based upon the PEXlib 5.1 binding, which itself is an X Consortium standard. The PEXlib binding is a lower-level interface than the previous PHIGS binding was and maps more closely to the PEX protocol itself. It supports immediate mode rendering functionality as well as the previous PHIGS workstation modes and is therefore suited to a wider range of applications. It is also suited for the development of higher level APIs. There are in fact commercial implementations of the PHIGS API which utilize the PEXlib API.

The PHIGS API based verification tool called InsPEX is moved to contrib. A prototype of a possible new tool called suspex is in the directory **contrib/test/suspex**. Suspex is PEXlib based.

Demo programs are no longer supported and have moved to contrib.

1.20.1. PEX Standards and Functionality

This release conforms to the PEX Protocol Specification 5.1 though it does not implement all the functionality specified therein.

The release comes with 2 fonts, Roman and Roman_M (see the *User's Guide* for more details).

As discussed briefly above certain functionality is not implemented in this Sample Implementation. Most notably Hidden Line, Hidden Surface Removal is not implemented. This is a result of both architectural decisions and the fact that it surely would have been replaced by vendors with proprietary code. A contributed implementation which supports some of the HLHSR functionality utilizing a Z buffer based technique is available for ftp from ftp.x.org in the directory contrib/PEX_HLHSR.

This release does not support monochrome displays, though it does support 8 bit and 24 bit color.

Other functionality not complete in this release is:

Backface Attributes and Distinguish Flag
Font sharing between clients
Patterns, Hatches and associated attributes
Transparency
Depth Cueing for Markers

Double Buffering is available for the PHIGS Workstation subsets directly through the workstation. The buffer mode should be set on when creating the workstation. For immediate mode users double buffering is achieved via the Multi Buffering Extension (aka MBX) found in the directory **xc/lib/Xext**.

PEX 5.1 protocol adds certain functionality to the Server extension, accessible directly via the PEXlib API. This functionality includes Picking via the Immediate Mode Renderer (Render Elements and Accumulate State commands in Chapter 6, all of Chapter 7); new Escape requests to allow vendors to support optional functionality; a Match Rendering Targets request to return information about visuals, depth and drawables the server can support; a noop Output command; Hierarchical HLHSR control (i.e., during traversals); and renderer clearing controls are the most important features.

1.21. Header Files

Two new macros are defined in **Xos.h**: X_GETTIMEOFDAY and strerror. X_GETTIMEOFDAY is like gettimeofday() but takes one argument on all systems. strerror is defined only on systems that don't already have it.

A new header file **Xthreads.h** provides a platform-independent interface to threads functions on various systems. Include it instead of the system threads header file. Use the macros defined in it instead of the system threads functions.

1.22. Fonts

IRIX 6.0 and IRIX 5.3 ship the same fonts as shipped in IRIX 5.2.

There are three new Chinese bdf fonts in **xc/fonts/bdf/misc** (**gb16fs.bdf**, **gb16st.bdf**, **gb24st.bdf**).

Bitmap Charter fonts that are identical to the output generated from the outline font have been moved to **xc/fonts/bdf/unnec_**{**75**,**100**}**dpi**.

The Type 1 fonts contributed by Bitstream, IBM, and Adobe that shipped in contrib in Release 5 have been moved into the core.

Some of the **misc** fonts, mostly in the *Clean* family, have only the ASCII characters, but were incorrectly labeled ''ISO8859-1''. These fonts have been renamed to be ''ISO646.1991-IRV''. Aliases have been provided for the Release 5 names.

The **9x15** font has new shapes for some characters. The **6x10** font has the entire ISO 8859-1 character set.

1.23. Font library

IRIX 6.0 and 5.3 use the new font library for the font server and the font tools. But the Silicon Graphics X server continues to use the R5 font library since the server has not yet been updated for R6.

IRIX 5.2 (and therefore IRIX 6.0 and 5.3 too) support the Type1 rasterizer. See the type1xfonts man page for more details.

The Type1 rasterizer that shipped in contrib in Release 5 is now part of the core.

There is an option to have the X server request glyphs only as it needs them. The X server then caches the glyphs for future use.

Aliases in a **fonts.alias** file can allow one scalable alias name to match all instances of another font. The ''!'' character introduces a comment line in **fonts.alias** files.

A sample font authorization protocol, ''hp-hostname-1'' has been added. It is based on host names and is non-authenticating. The client requesting a font from a font server provides (or passes through from its client) the host name of the ultimate client of the font. There is no check that this host name is accurate, as this is a sample protocol only.

The Speedo rasterizer can now read fonts with retail encryption. This means that fonts bought over-the-counter at a computer store can be used by the font server and X server.

Many, many bugs have been fixed.

1.24. Font server

IRIX 6.0 and 5.3 have not yet renamed the X font server to be called xfs instead of fs. But the X font server in IRIX 6.0 and 5.3 is based on the R6 implementation.

The font server has been renamed from *fs* to *xfs* to avoid confusion with an AFS program. The default port has changed from 7000 (used by AFS) to 7100 and has been registered with the Internet Assigned Numbers Authority.

The font server now implements a new major protocol version, version 2. This change was made only to correct errors in the implementation of version 1. Version 1 is still accepted by *xfs*.

You can now connect to *xfs* using the **local/** transport.

Many, many bugs have been fixed.

1.25. X server

IRIX 6.0 and IRIX 5.3 continue to ship the Silicon Graphics X server based on the R5 version of the X server shipped with IRIX 5.2. Most of the X server changes for R6 were not very significant to X users. A later release will upgrade the X server to be based on X11R6 sources.

Specifically, R6 extensions like XIE, SYNC, and XC-MISC are not supported by the Silicon Graphics X server in IRIX 6.0 or 5.3.

The X Keyboard extension (XKB) is supported by IRIX 6.0 and 5.3.

The server sources have moved to **xc/programs/Xserver**. Server-side extension code exists as subdirectories. The **ddx** directory is gone; **mi**, **cfb**, and **mfb** are at the top level, and a **hw** (hardware) subdirectory now exists for holding vendor-specific ddx code. Note: the absence of a ddx directory does not imply that the conceptual split between dix and ddx is gone.

Function prototypes have been added to header files in **xc/programs/Xserver/include**, **cfb**, **mfb**, **mi**, and **os**.

Support for pixmap privates has been added. It is turned off by default, but can be activated by putting −DPIXPRIV in the ServerExtraDefines parameter in your *vendor***.cf** file. See the porting layer document for details.

New screen functions, called primarily by code in window.c, have been added to make life easier for vendors with multi-layered framebuffers. Several functions and some pieces of functions have moved from window.c to miwindow.c. See the porting layer document for details. Also, the contents of union _Validate (validate.h) are now device dependent; mivalidate.h contains a sample definition.

An implementation of the SYNC extension is in **xc/programs/Xserver/Xext/sync.c**. As part of this work, client priorities have also been implemented; see the tail end of WaitForSomething() in WaitFor.c. The priority scheme is *strict* in that the client(s) with the highest priority always runs. *twm* has been modified to provide simple facilities for setting client priorities.

The server can now fetch font glyphs on demand instead of loading them all at once. See **xc/programs/Xserver/dix/dixfonts.c**, **xc/lib/font/fc/fserve.c**, and **xc/lib/font/fc/fsconvert.c**. A new X server command line option, **−deferglyphs**, controls which types of fonts (8 vs. 16 bit) to demand load; see the X manual page for details.

The os layer now uses sigaction on POSIX systems; a new function OsSignal was added for convenience, which you should use in your ddx code.

A new timer interface has been added to the os layer; see the functions in os/WaitFor.c. This interface is used by XKB, but we haven't tried to use it anywhere else (such as Xext/sleepuntil.c) yet.

Redundant code for GC funcs was moved from cfbgc.c and mfbgc.c to migc.c. This file also contains a few utility functions such as miComputeCompositeClip, which replaces the chunk of code that used to appear near the top of most versions of ValidateGC.

The cfb code can now be compiled multiple times to provide support for multiple depths in the same server, e.g., 8, 12, and 24. See **Imakefile** and **cfb/cfbmskbits.h** under the **xc/programs/Xserver/** directory for starters.

The cfb and mfb code have been modified to perform 64 bit reads and writes of the framebuffer on the Alpha AXP. These modifications should be usable on other 64 bit architectures as well, though we have not tested it on any others. There are a few hacks in dix, notably ProcPutImage and ProcGetImage, to work around the fact that the protocol doesn't allow you to specify 64 bit padding. Note that the server will still not run on a machine such as a Cray that does not have a 32 bit data type.

For performance, all region operations are now invoked via macros which by default make direct calls to the appropriate mi functions. You can conditionally compile them to continue calling through the screen structure. The following change was made throughout the server:

> ‘‘(*pScreen->RegionOp)(...)’’ changes to ‘‘REGION_OP(pScreen, ...)’’

Some of the trivial region ops have been inlined in the macros. For compatibility, the region function pointers remain in the screen structure even if the server is compiled to make direct calls to mi. See include/regionstr.h.

A generic callback manager is included and can be used to add notification-style hooks anywhere in the server. See dixutils.c. The callback manager is now being used to provide notification of when the server is grabbed/ungrabbed, when a client's state changes, and when an event is sent to a client. The latter two are used by the RECORD extension.

A new option has been added, **–config** *filename*. This lets you put server options in a file. See **os/utils.c**.

Xtrans has been installed into the os layer. See os/connection.c, io.c, and transport.c. As a result, the server now supports the many flavors of SVR4 local connections.

The client structure now has privates like windows, pixmaps, and GCs. See include/dixstruct.h, dix/privates.c, and dispatch.c.

Thin line pixelization is now consistent across cfb, mfb, and mi. It is also reversible, meaning the same pixels are touched when drawing from point A to point B as are touched when drawing from point B to point A. A new header file, miline.h, consolidates some miscellaneous line drawing utilities that had previously been duplicated in a number of places.

1.25.1. Xnest

Xnest is not supported by IRIX 6.0 or IRIX 5.3. There is not plan to ever support Xnest.

A new server, Xnest, uses Xlib to implement ddx rendering. See xc/programs/Xserver/hw/xnest. Xnest lets you run an X server in a window on another X server. Uses include testing dix and extensions, debugging client protocol errors, debugging grabs, and testing interactive programs in a hardware-starved environment.

1.25.2. Xvfb

Xvfb is not supported by IRIX 6.0 or IRIX 5.3. There is not plan to ever support Xvfb.

Another new server, Xvfb, uses cfb or mfb code to render into a framebuffer that is allocated in virtual memory. See **xc/programs/Xserver/hw/vfb**. The framebuffer can be allocated in normal memory, shared memory, or as a memory mapped file. Xvfb's screen is normally not visible; however, when allocated as a memory mapped file, *xwd* can display the screen by specifying the framebuffer file as its input.

1.25.3. ddx

Sun ddx
Expanded device probe table finds multiple frame buffers of the same type. Expanded keymap tables provide support for European and Asian keyboards. Added per-key autorepeat support. Considerable cleanup and duplicate code eliminated. Deletion of SunView support. GX source code now included.

HP ddx
cfb-based sources included as **xc/programs/Xserver/hw/hp**.

svga ddx
new svga ddx for SVR4 included as **xc/programs/Xserver/hw/svga**.

xfree86 ddx
ddxen from XFree86, Inc. included as **xc/programs/Xserver/hw/xfree86**.

Amoeba ddx
ddx for Sun server on the Amoeba operating system included as **xc/programs/Xserver/hw/sunAmoeba**. The server will require additional patches for this to be usable.

1.26. New Programs

IRIX 6.0 and 5.3 continue to supply lndir, but do not supply the new mkshadow. Users are invited to read the tlink man page for a similiar but more powerful command available in IRIX.

xc/config/util/mkshadow/, a replacement for *lndir*.

1.27. Old Software

Programs below that were shipped in IRIX 5.2 continue to be shipped in IRIX 6.0 and 5.3.

We have dropped support for the following libraries and programs and have moved them to **contrib**: CLX library, PHIGS library, *MacFS*, *auto_box*, *beach_ball*, *gpc*, *ico*, *listres*, *maze*, *puzzle*, *showfont*, *viewres*, *xbiff*, *xcalc*, *xditview*, *xedit*, *xev*, *xeyes*, *xfontsel*, *xgas*, *xgc*, *xload*, *xman*, and *xpr*.

1.28. xhost

Kerberos support for xhost is not compiled for IRIX 6.0 and 5.3 (see section 1.14).

Two new families have been registered: LocalHost, for connections over a non-network transport, and Krb5Principal, for Kerberos V5 principals.

To distinguish between different host families, a new xhost syntax ‘‘family:name’’ has been introduced. Names are as before; families are as follows:

inet:	Internet host
dnet:	DECnet host
nis:	Secure RPC network name

krb:	Kerberos V5 principal
local:	contains only one name, ‘‘’’

The old-style syntax for names is still supported when the name does not contain a colon.

1.29. xrdb

Many new symbols are defined to tell you what extensions and visual classes are available.

1.30. twm

IRIX 6.0 and 5.3 have an R6 version of twm that has been further enhanced with internationalization features from the IRIX 5.2 version of twm.

An interface for setting client priorities with the Sync extension has been added.

Many bugs have not been fixed yet.

1.31. xdm

IRIX 6.0 and 5.3 do not have new features of the R6 xdm. These features will probably be available in the next release. The IRIX 6.0 and 5.3 supplied xdm is based on the SGI-enhanced X11R5-based xdm supplied in IRIX 5.2. The IRIX 5.3 xdm has further enhancements for trusted OS support (and are not present in IRIX 6.0).

There is a new resource, **choiceTimeout**, that controls how long to wait for a display to respond after the user has selected a host from the chooser.

Support has been added for a modular, dynamically-loaded greeter library. This feature allows different dynamic libraries to by loaded by *xdm* at run-time to provide different login window interfaces without access to the *xdm* sources. It works on DEC OSF/1 and SVR4. The name of the greeter library is controlled by another new resource, **greeterLib**.

When you log in via *xdm*, *xdm* will use your password to obtain the initial Kerberos tickets and store them in a local credentials cache file. The credentials cache is destroyed when the session ends.

1.32. xterm

Now supports a few escape sequences from HP terminals, such as memory locking. See **xc/doc/specs/xterm/ctlseqs.ms** for details.

The **termcap** and **terminfo** files have been updated.

ctlseqs.ms has moved out of the xterm source directory into **xc/doc/specs/xterm**.

The logging mis-feature of xterm is removed. This change first appeared as a public patch to Release 5.

Many bugs have not been fixed yet.

1.33. xset

The screen saver control option has two new sub-options to immediately activate or deactivate the screen saver: **xset s activate** and **xset s reset**.

1.34. X Test Suite

The UniSoft X Test Suite 1.2 is used to test SGI's X implementation but the current version of the test suite that we compile and use is based on the pre-R6 release of the test suite. This test suite is used internally and is not shipped to customers.

The X Test Suite, shipped separately from R5, is now part of the core distribution in R6.

The code has been fixed to work on Alpha AXP. The Xi tests contributed by HP and XIM tests contributed by Sun are integrated.

1.35. Work in Progress

None of the ''Work in Progress'' supplied with R6 will be shipped in either IRIX 5.3 or 6.0 with the exception of the X Keyboard extension (XKB) which actually shipped with IRIX 5.2.

Everything under **xc/workInProgress** represents a work in progress of the X Consortium.

Fresco, Low Bandwidth X (LBX), the Record extension, and the X Keyboard extension (Xkb, which logically belongs here but was too tightly coupled into Xlib and the server to extract) are neither standards nor draft standards, are known to need design and/or implementation work, are still evolving, and will not be compatible with any final standard should such a standard eventually be agreed upon. We are making them available in early form in order to gather broader experimentation and feedback from those willing to invest the time and energy to help us produce better standards.

Any use of these interfaces in commercial products runs the risk of later source and binary incompatibilities.

1.35.1. Fresco

Silicon Graphics has no current plans to support Fresco. Mark Linton of Silicon Graphics is the architect of Fresco. He can be contacted for more information about Fresco for Silicon Graphics machines.

R6 includes the first sample implementation of Fresco, a user interface system specified using CORBA IDL and implemented in C++. Fresco is not yet a Consortium standard or draft standard, but is being distributed as a work in progress to demonstrate our current directions and to gather feedback on requirements for a Fresco standard.

The Fresco Sample Implementation has been integrated into the X11R6 build process, and will be built automatically if you have a C++ compiler available. Documentation on Fresco can be found in **xc/doc/specs/Fresco**. The Fresco and Xtf libraries are found in **xc/workInProgress/Fresco** and **xc/workInProgress/Xtf**, respectively. There are some simple Fresco example programs in **contrib/examples/Fresco**, and a number of related programs in **contrib/programs**, including:

ixx An IDL to C++ translator

i2mif A program to generate FrameMaker MIF documents from comments in an IDL specification

fdraw A simple Fresco drawing editor

dish A TCL interpreter with hooks to Fresco

Working Imakefiles are provided for all of the utilities and examples.

A demo program (dish) is included that shows how a scripting language (Tcl) can rather easily be bound to Fresco through the CORBA dynamic invocation mechanism. A copy of Tcl is included in **contrib/lib/tcl**.

To build Fresco you must define HasCplusplus in **site.def**; in addition, you may have to set CplusplusCmd and/or CplusplusDependIncludes to invoke the appropriate C++ compiler and find the required header files during make depend. Finally, you should check the *vendor***.cf** to see if there are any other configuration variables you should set to provide information about your C++ compiler.

Fresco requires a C++ compiler that implements version 3 of the C++ language (as approximately defined by USL cfront version 3). While Fresco does not currently use templates or exceptions, it does make extensive use of nested types, which were inadequately supported in earlier versions of the language.

Fresco has been built with the following platforms and C++ compilers:

SPARCstation	SunOS 4.1.3	CenterLine C++
SPARCstation	Solaris 2.3	CenterLine C++ (requires v2.0.6)
SPARCstation	Solaris 2.3	SPARCCompiler C++ v4.0
HP 9000/700	HPUX 9.0.1	CenterLine C++
SGI Indy	IRIX 5.2	SGI C++
IBM RS/6000	AIX 3.2.5	IBM xlC
Sony NEWS	NEWSOS 6.0	Sony C++

Fresco has also been compiled on the DEC Alpha under OSF/1 version 2.0 using a beta test version of DEC C++ 1.3. Fresco cannot be built with the Gnu C++ compiler (version 2.5.8 or earlier) due to bugs and limitations in g++.

Building Fresco with CenterLine C++ requires that you pass the **−Xa** flag to the C++ compiler. Place the following lines in your site.def:

```
#define HasCenterLineCplusplus YES
#define CplusplusOptions  −Xa
```

If CC is not in your default search path, add this line to **site.def**:

#define CplusplusCmd */path/to/your/CC*

If you are building under Solaris 2, you must use ObjectCenter version 2.0.6 or later; the C++ compiler in ObjectCenter 2.0.4 will produce Fresco applications that dump core on startup.

Fresco does not yet build under Microsoft Windows/NT.

1.35.2. XKB (X Keyboard Extension)

Support for XKB is not compiled in to Xlib by default. It is compiled in the X server by default only on Sun and Omron Luna machines. You can compile it in by setting

```
#define BuildXKB YES              /* for support in the X server */
#define BuildXKBLib YES           /* for support in the X library */
```

in the file **xc/config/cf/site.def**. Note that enabling XKB in the X server is a pervasive change; you need to clean the server and rebuild everything if you change this option.

Turning on XKB in the X server usually requires changes to the vendor ddx keyboard handling. There is currently support only in the Sun and Omron ddx.

If you turn on **BuildXKBLib**, additional functions are added to Xlib. Since the resulting library is non-standard, it is given a different name: **libX11kb** instead of **libX11**. All Makefiles produced by *imake* will use **−lX11kb** to link Xlib.

The library changes for XKB are known not to work on the Cray; many other systems have been tested, including the Alpha AXP.

There are some XKB test programs in **contrib/test/Xkb**.

The XKB support in Xlib is still at an early stage of formal review and could change. We expect some additions in an eventual standard, but few changes to the interfaces provided in this implementation. A working draft of the protocol is in **/xc/doc/specs/Xkb/**.

1.35.3. LBX (Low Bandwidth X)

The X Consortium is working to define a standard for running X applications over serial lines, wide area networks, and other slow links. This effort, called Low Bandwidth X (LBX), aims to improve the startup time, performance, and interactive feel of X applications run over low bandwidth transports.

LBX does this by interposing a *pseudo-server* (called the *proxy*) between the X clients and the X server. The proxy caches data flowing between the server and the clients, merges the X protocol streams, and compresses the data that is sent over the low bandwidth wire. The X server at the other end uncompresses the data and splits it back out into separate request streams. The target is to make many X applications transparently usable over 9600 bps modems.

A snapshot of the code for this effort is included in **xc/workInProgress/lbx/** for people to examine and begin experimenting with. It contains the following features:

- LZW compression of the binary data stream. Since commercial use of LZW requires licensing patented technology, we are also looking for an unencumbered algorithm and implementation to provide as well.
- Delta compression of X packets (representing packets as differences from previously sent packets).
- Re-encoding of some graphics requests (points, lines, segments, rectangles, and arcs).
- Motion event throttling (to keep from flooding the wire).
- Caching of data in the proxy for large data objects that otherwise would be transmitted over the wire multiple times (e.g., properties, font metrics, keyboard mappings, connection startup data, etc.).
- Short-circuiting of requests for constant data (e.g., atoms, colorname/rgb mappings, and read-only color cells).

However, the following items have yet to be implemented (which is why it isn't a standard yet):

- Re-encoding of a number of requests (e.g., QueryFont), events, etc.
- Support for BIG-REQUESTS extension.
- A non-networked serial protocol for environments which cannot support os-level networking over serial lines.
- A full specification needs to be written describing the network protocol used between the proxy and the server.

The X Consortium is continuing to work on both the implementation of the remaining items and the full specification. The goal is to have all of the pieces ready for public review later this year. Since the specification for LBX *will* change, we strongly recommend against anyone incorporating LBX into a product based on this prototype. But, they are encouraged to start looking at the code, examining the concepts, and providing feedback on its design.

1.35.4. RECORD extension

RECORD is an X protocol extension that supports the recording of all core X protocol and arbitrary X extension protocol.

A version of the extension is included in **xc/workInProgress/record**. The implementation does not quite match the version 1.2 draft specification, but the spec is going to change anyway; the version 1.3 draft is in **xc/doc/specs/Xext/record.ms**. The GetConfig request is not fully implemented. A test program is in **contrib/test/record**.

1.35.5. Simple Session Manager

It is possible that some future version of the Indigo Magic desktop will include SM compatible session management, but it is not yet planned.

A simple session manager has been developed to test the new Session Management protocol. At the moment, it does not exercise the complete XSMP protocol and the user interface is rather simple. While it does have enough functionality to make it useful, it needs more work before we would want people to depend on it or use it as a good example of how to implement the session protocol.

- Handles accepting connections from clients
- Handles graceful or unexpected termination of clients
- Maintains database of all properties set by clients
- User interface provides a way to issue checkpoint and shutdown messages to clients
- Manages client interaction with the user
- Can restart clients. Clients running on remote machines are handled using the new *rstart* protocol.
- Requires MIT-MAGIC-COOKIE-1 authentication from clients.

We have not yet written a proxy for connecting ICCCM 1.0 clients to the session manager.

A sample client, *xsmclient*, has been written to demonstrate the session support in Xt.

1.35.6. Multi-Threaded X Server

Silicon Graphics has no plans to implement a multi-threaded version of the X server.

An attempt has been made to merge the multi-threaded server source with the single-threaded source. The result is in the **xc/workInProgress/MTXserver** directory. The sources here include only files that were changed from the single-threaded server. The multi-threaded server may not compile. Unfortunately, the single-threaded server sources have continued to evolve since this snapshot of the MTXserver was produced, so there is work to be done to get the MTXserver sources back into a state where they can be compiled.

1.36. ANSIfication

We've changed our sources to stop using the BSD function names index, rindex, bcopy, bcmp; we now use strchr, strrchr, memcpy/memmove, and memcmp. We still use the name bzero (because there is no BSD equivalent for the general case of memset) but it is translated to memset via a #define in <X11/Xfuncs.h>. The BSD function names are still supported in <X11/Xos.h> and <X11/Xfuncs.h>.

Most client-side uses of caddr_t should now be gone from our sources.

Explicit declarations of errno are now only used on non-ANSI systems.

The libraries use more standard POSIX *_t types.

1.37. Miscellaneous

IRIX 6.0 and 5.3 continue to ship the R5 version of patch.

A new version of the *patch* program is in **xc/util/patch**; it understands the unified diff format produced by GNU *diff*.