




X11/NeWS™ Release 1.0 Notes

NeWS™, X11/NeWS™, SunView™, XView™, and OpenWindows™ are trademarks of Sun Microsystems, Inc. **Sun Workstation®**, **Sun Microsystems®**, and the **Sun logo**  are registered trademarks of Sun Microsystems Inc.

POSTSCRIPT® is a registered trademark of Adobe Systems Inc. Adobe owns copyrights related to the POSTSCRIPT language and the POSTSCRIPT interpreter. The trademark POSTSCRIPT is used herein to refer to the material supplied by Adobe or to programs written in the POSTSCRIPT language as defined by Adobe.

The X Window System is a trademark of Massachusetts Institute of Technology.

UNIX® is a registered trademark of AT&T.

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations.

Copyright © 1989 Sun Microsystems, Inc. – Printed in U.S.A.

All rights reserved. No part of this work covered by copyright hereon may be reproduced in any form or by any means – graphic, electronic, or mechanical – including photocopying, recording, taping, or storage in an information retrieval system, without the prior written permission of the copyright owner.

Restricted rights legend: use, duplication, or disclosure by the U.S. government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 and in similar clauses in the FAR and NASA FAR Supplement.

The Sun Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees.

Contents

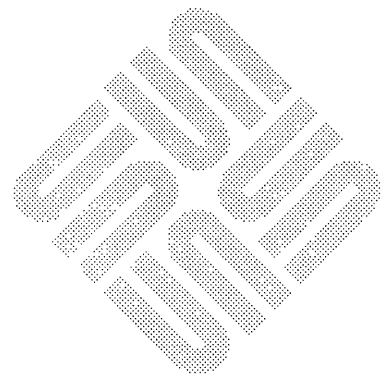
Chapter 1 Introduction	3
Chapter 2 Changes to NeWS	7
2.1. Changes in File Location	7
2.2. Changes in File Access	8
2.3. New Environment Variables	8
2.4. Revised *.ps Files	8
Root Menus	8
<i>LiteWindow</i> Changes	8
Making a Debugging Server	9
Verbose Loading	9
2.5. Changes in NeWS Class Implementation	9
2.6. Look and Feel Changes	9
2.7. New and Modified NeWS Operators	10
New Operators	10
Renamed Operators	10
Polymorphic Operators	10
Raster-Op Code not Reset After setrafteropcode	11
2.8. Improved Support of the POSTSCRIPT Language	11
Image Operators Implemented	11
2.9. New Types	11
2.10. Changes to Existing Types	11
2.11. Paths	11
2.12. Canvases	12

Change in the Behavior of Transparent Canvases	12
Overlays	12
2.13. Events	12
Input Handling Changes	12
Internal Implementation of <code>eventmgrinterest</code> and forkeventmgr	13
Changed Action on Canvas Entry/Exit	13
Change in Focus Event Naming	13
Time Value Changes	13
Shared Interests	13
Executable Matches	13
Special Events	13
2.14. Processes	14
Changes to Process Objects	14
2.15. Changes in Error Handling	14
2.16. Changes to Magic Dictionaries	15
2.17. NeWS Fonts	15
Adobe Bitmap Font Format	16
Scalable Fonts	16
2.18. Changes to CPS	16
Changed CPS routines	16
New CPS routines	16
2.19. Memory Management	17
Chapter 3 Changes to X11	21



Tables

Table 2-1 New Operators	10
-------------------------------	----



Preface

These release notes accompany the X11/NeWS[™] server provided with release 1.0 of OpenWindows[™]. The notes discuss changes that have been made to NeWS[™] since the release of NeWS 1.1; they also discuss differences from release 3 of X11, from MIT. This document should be read by customers who have been using either NeWS 1.1 or public domain versions of the X11 server available from MIT.

After reading these release notes, reread the *Read This First* document provided with this release of OpenWindows.

The XView toolkit, the NeWS toolkit, and the X11/NeWS window manager are based on an early version of the OPEN LOOK[™] Graphical User Interface Specification, and they therefore do not necessarily implement every element in the most recent revision of that specification.¹ OPEN LOOK references thus point to software that has not yet been validated by AT&T as fully OPEN LOOK compliant, although full validation is expected shortly.

¹ OPEN LOOK[™] is a trademark of AT&T.

Introduction

Introduction 3

[Faint, illegible text from the reverse side of the page, appearing as bleed-through.]

Introduction

This document describes the notable changes and enhancements in release 1.0 (Revision A) of X11/NeWS, compared with both NeWS 1.1 and the X11 Release 3 window system available from MIT.

NeWS is now supported as part of the X11/NeWS server, which itself forms a part of the OpenWindows distributed window system. The X11/NeWS server runs windowing applications written with either the X11 or the NeWS protocol. Though the two protocols are entirely different, X11/NeWS provides an integrated environment in which both are supported. Thus, when X11 and NeWS applications simultaneously display windows, the windows coexist on the screen and can be made to overlap in any way. *Cut and paste* selections can be made between X11 and NeWS windows. All the windows are manipulated by a single window manager.

The X11/NeWS server, which forms the window system platform of the OpenWindows environment, can be used to run applications of the following kinds:

- All correctly-written X11 applications, such as those built with the XView user interface
- NeWS applications that conform to NeWS 1.1 documented interfaces (including *Lite toolkit*-based applications)
- SunView- and SunWindows-based applications

The server also supports the running of *networked* applications: thus, you can run client programs on remote machines, using your local machine to display the corresponding windows.

A set of XView applications is provided with this release of OpenWindows; the set includes a shell command tool, a file manager, a text editor, a mail tool, and other tools.

Changes to NeWS

Changes to NeWS	7
2.1. Changes in File Location	7
2.2. Changes in File Access	8
2.3. New Environment Variables	8
2.4. Revised *.ps Files	8
Root Menu	8
<i>LiteWindow</i> Changes	8
Making a Debugging Server	9
Verbose Loading	9
2.5. Changes in NeWS Class Implementation	9
2.6. Look and Feel Changes	9
2.7. New and Modified NeWS Operators	10
New Operators	10
Renamed Operators	10
Polymorphic Operators	10
Raster-Op Code not Reset After setrasteropcode	11
2.8. Improved Support of the POSTSCRIPT Language	11
Image Operators Implemented	11
2.9. New Types	11
2.10. Changes to Existing Types	11
2.11. Paths	11
2.12. Canvases	12
Change in the Behavior of Transparent Canvases	12

Overlays	12
2.13. Events	12
Input Handling Changes	12
Internal Implementation of eventmgrinterest and forkeventmgr	13
Changed Action on Canvas Entry/Exit	13
Change in Focus Event Naming	13
Time Value Changes	13
Shared Interests	13
Executable Matches	13
Special Events	13
2.14. Processes	14
Changes to Process Objects	14
2.15. Changes in Error Handling	14
2.16. Changes to Magic Dictionaries	15
2.17. NeWS Fonts	15
Adobe Bitmap Font Format	16
Scalable Fonts	16
2.18. Changes to CPS	16
Changed CPS routines	16
New CPS routines	16
2.19. Memory Management	17

Changes to NeWS

Numerous features have been added to the NeWS language, including extensible NeWS objects, packed arrays, classes for input handling, and better support of the POSTSCRIPT language as defined in the *PostScript Language Reference Manual*.

In addition, the following changes have occurred:

- The NeWS toolkit (formally known as the *NeWS Development Environment* or *NDE*) has become the preferred NeWS-based toolkit. The old *Lite* toolkit has been retained in its original form but will no longer be enhanced.
- Windows provided by XView and the NeWS toolkit correspond to the OPEN LOOK User Interface Specification. OPEN LOOK is an easy-to-use graphical interface.
- Minor differences between the X11 and NeWS window systems have necessitated changes to the NeWS language.
- Adjustments have been made to facilitate future development of the product and ensure continued quality.

2.1. Changes in File Location

The default installation-directory for OpenWindows is now, for all machines except the 386i, `/home/openwin`. For a 386i, the default installation-directory is `/files/local/sun386/openwin`. You must then set the environment variable `OPENWINHOME` to the directory in which the product has been installed. The `NEWSHOME` environment variable is now ignored. See the *OpenWindows Installation and Start-Up Guide* for full information.

The following changes have also occurred:

- Startup files — The user `.ps` and `startup.ps` files, which are read from your home directory when X11/NeWS is started, have now been renamed `.user.ps` and `.startup.ps`.
- Images are stored in `$OPENWINHOME/demo/images` instead of `$NEWSHOME/smi`.
- The client source is in `$OPENWINHOME/share/src/xnews` instead of `$NEWSHOME/clientsrc`.
- The server is now `$OPENWINHOME/bin/xnews`, not `$NEWSHOME/bin/news_server`.

- POSTSCRIPT language files read in by the server at start-up are stored in `$OPENWINHOME/etc/NEWS` instead of `$NEWSHOME/lib/NEWS`.
- Font files read in by the server are stored in `$OPENWINHOME/lib/fonts`.

2.2. Changes in File Access

With the change in the location of the start-up POSTSCRIPT language files, the file and run POSTSCRIPT language operators (together with all the NeWS utilities that use them) now search the following directories in turn:

1. The directory `.` / (the directory in which the server has been started)
2. The directory `~` / (the user's home directory)
3. The directory `$OPENWINHOME/etc/`

2.3. New Environment Variables

The X11/NEWS server offers a number of new environment variables. Among these is `NEWSONLY`. When no X11 clients will be run, `NEWSONLY` can be set to reduce memory consumption. See the manual page for `xnews` in the *X11/NEWS Server Guide* for a description of all the environment variables.

2.4. Revised *.ps Files

Much of the behavior and functionality of the X11/NEWS server is determined by files containing POSTSCRIPT language code read in at startup. These files are different from the `*.ps` files in NeWS 1.0 and 1.1; thus, NeWS users may find some incompatibilities (these are described in this document). The changes have been made to allow the server to be started up in various modes (for example, as a filter, or as an output-only device).

The following sections discuss the incompatibilities in greater detail. Refer to the *NeWS Programmer's Guide* for a fuller description of the new `*.ps` files.

Root Menus

The rootmenu is now implemented using the NeWS toolkit instead of *Lite*. The programming interface is different. The preferred method of customizing the root menu involves modifying files in the SunView rootmenu format and running `buildmenu`. See the *X11/NEWS Server Guide* for instructions.

LiteWindow Changes

In the past, some client programs used `get` to retrieve the `FrameMenu` of their window in order to change it:

```
win /FrameMenu get
```

Using `get` this way is improper, and it was a coincidence of implementation that it ever worked at all. In X11/NEWS, this construct no longer works. Instead, you should use `send` to get the menu:

```
/FrameMenu win send
```

Note that in X11/NEWS, the frame menu is shared by all instances of `LiteWindow`. If you modify the frame menu in a particular application, all other applications ultimately use the modified menu. Thus, instead of modifying the frame menu,

you should define a `ClientMenu` for your window. You can include the `FrameMenu` as a submenu of the `ClientMenu` if you wish. You should define the `ClientMenu` during window initialization:

```

/win framebuffer /new DefaultWindow send def
{
  /CreateFrameMenu win send
  /ClientMenu [
    (First Function) { first }
    (Second Function) { second }
    (Frame =>) /FrameMenu win send
  ] /new DefaultMenu send def
} win send

```

Making a Debugging Server

The procedures `'go!'` and `'DebuggingServer?'` have been removed. To make the server start up an executive session on the terminal from which you start it, you should change, within your `.user.ps` file, the statement

```

/DebuggingServer? true def

```

to the following:

```

/&main /&main load {executive} append cvx def

```

This redefines the “main routine” of the server to start an executive session.

Verbose Loading

The old `'verbose?'` flag, which made the server print out the files as it loaded them, has been removed from `init.ps`. To achieve the same effect, simply replace `LoadFile` in your `.startup.ps` file so that it provides some feedback (note that if you redefine `LoadFile` in your `.user.ps` file, the change takes place after most files have been loaded.)

2.5. Changes in NeWS Class Implementation

Many new operators and methods are now available for use with classes. These operators and methods are described in detail in Chapter 4, *Classes*, of the *NeWS Programmer's Guide*. The class system now supports multiple inheritance. See the *NeWS Programmer's Guide* for details.

Also, there is a new class-based event manager named `ClassEventManager`. This is simply a class-based use of the existing `forkeventmgr`–`eventmgrinterest` pair.

2.6. Look and Feel Changes

The default typing style in OPEN LOOK is *click-to-type*, meaning that you must click the left mouse button in a window before it can receive your keystrokes. If you are familiar with the *focus-follows-cursor* style, which was previously used by NeWS, the new style may take some getting used to.

It is possible to set the typing style back to *focus-follows-cursor* by adding the following line to your `~/Xdefaults` file:

```
OpenWindows.SetInput: followmouse
```

You can also use the `Properties` program on the rootmenu to change back to focus-follows-cursor style.

2.7. New and Modified NeWS Operators

This section discusses NeWS operators that have been either added or modified since NeWS 1.1. See the *NeWS Programmer's Guide* for information on these operators.

New Operators

The following operators are new:

Table 2-1 *New Operators*

<code>canvasesunderpath</code>	<code>getcard32</code>	<code>putcard32</code>
<code>canvasesunderpoint</code>	<code>getcolor</code>	<code>refcnt</code>
<code>countfileinputtoken</code>	<code>getfileinputtoken</code>	<code>reffinder</code>
<code>createcolormap</code>	<code>getprocesses</code>	<code>setbackcolor</code>
<code>createcolorsegment</code>	<code>getprocessgroup</code>	<code>setbackpixel</code>
<code>currentbackcolor</code>	<code>grabcursor</code>	<code>setcursorlocation</code>
<code>currentbackpixel</code>	<code>harden</code>	<code>setpacking</code>
<code>currentpacking</code>	<code>lasteventkeystate</code>	<code>setpixel</code>
<code>currentpixel</code>	<code>lasteventtime</code>	<code>setplanemask</code>
<code>currentplanemask</code>	<code>lasteventx</code>	<code>setshared</code>
<code>currentshared</code>	<code>lasteventy</code>	<code>soft</code>
<code>currentstate</code>	<code>newcursor</code>	<code>soften</code>
<code>encodefont</code>	<code>objectdump</code>	<code>truetype</code>
<code>eoextenddamageall</code>	<code>packedarray</code>	<code>writeobject</code>
<code>extenddamageall</code>	<code>postcrossings</code>	

Renamed Operators

The following operators have been renamed:

- `^C`
This operator is now named `shutdownserver`.
- `forkunix`
This operator is now named `runprogram`.

Polymorphic Operators

The following operators now have two possible syntactic forms:

createcolorsegment
encodefont
expressinterest
getfileinputtoken
movecanvas
newcanvas
newcursor
reshapecanvas
revokeinterest
setfileinputtoken

Raster-Op Code not Reset After `setrasteropcode`

In NeWS 1.1, a bug existed whereby certain operations, if performed after the raster-op code was set, caused the raster-op code to be reset to another value. In X11/NeWS 1.0, you must explicitly reset the raster-op code. Applications that rely on the raster-op code being non-explicitly reset may now paint their canvases strangely.

2.8. Improved Support of the POSTSCRIPT Language

OpenWindows includes additional support for the POSTSCRIPT language as defined by Adobe Systems in the *PostScript Language Reference Manual*.

Image Operators Implemented

The `settransfer` and `currenttransfer` functions are implemented.

2.9. New Types

The following types are new:

/colormaptype
/colormapentrytype
/cursortype
/environmenttype
/packedarraytype
/visualtype

See the *NeWS Programmer's Guide* for information on these types.

2.10. Changes to Existing Types

OpenWindows contains all the types that exist in NeWS 1.1. However, some of the types now contain additional keys and have new functionality. The changes are listed in the sections below; see the *NeWS Programmer's Guide* for more detailed information.

2.11. Paths

The type formerly named `shapetype` is now named `pathtype`.

2.12. Canvases

The following keys are new:

Cursor
Colormap
Grabbed
GrabToken
Visual
VisualList
OverrideRedirect
BorderWidth
UserProps
XID
SharedFile
RowBytes

Change in the Behavior of Transparent Canvases

In NeWS 1.1, making a transparent child canvas retained would change its parent to be retained; however, the converse, making a transparent retained child non-retained had no effect.

In X11/NeWS, making a transparent child retained or non-retained does not affect the transparent canvas or its parent.

Overlays

NeWS 1.1 handled only one overlay at a time. If more than one existed, you would generally only get correct behavior from the most recently created overlay. X11/NeWS can handle multiple simultaneous active overlays.

2.13. Events

The following keys are new:

IsPreChild
Coordinates

Input Handling Changes

As part of the merge with X11, there have been a number of changes in NeWS' input handling. One major change is that *global interests* have been replaced with *pre-child interests*.

Attendant incompatibilities include:

- If the **Canvas** field of an interest contains null, the interest is placed on the root canvas' pre-child interest list, but the **Canvas** field remains null.
- In event delivery, an event with an explicit destination canvas is tested for a match against pre-child interests on each of its ancestors (in root-to-leaf order) before being tested against the destination canvas.
- In event delivery, the **Canvas** field of the delivered event is no longer modified to indicate the canvas of the interest that matched the event; that information is available from the **Interest** field of the delivered event, as it has always been.

Internal Implementation of eventmgrinterest and forkeventmgr	eventmgrinterest returns an interest that you pass to forkeventmgr . The interface to these operators is unchanged from NeWS 1.1 in X11/NeWS, however, their implementation is different. Programs should not assume any particulars about the internal format of the interests used by eventmgrinterest/forkeventmgr .
Changed Action on Canvas Entry/Exit	The value of the Action field in the EnterEvent or ExitEvent (generated when the cursor crosses a canvas) is different in X11/NeWS than in NeWS 1.1.
Change in Focus Event Naming	<p>In NeWS 1.1, when a canvas lost the keyboard focus, a /DeSelect event with action /InputFocus was generated. This was ambiguous, however, because /DeSelect also signals the loss of ownership of the selection named in the Action field, and /InputFocus could be the name of a particular selection.</p> <p>So, in X11/NeWS, when a canvas loses the focus, an event named /LoseFocus is generated instead of a special kind of /DeSelect.</p> <p>Focus events are now generated like Enter and Exit events: thus, ancestors of the canvas that has the focus receive events, but with different Action values.</p>
Time Value Changes	Time stamps in the X11/NeWS server have been changed, for convenience in merging with X11. This change affects the value of timestamps in NeWS events (the TimeStamp field in the event), and the values returned by the currenttime and lasteventtime primitives. In NeWS 1.1, '1.0' in a time value was one minute. It now is 2^{16} milliseconds, or 65.536 seconds. This means that the NeWS clock in X11/NeWS "ticks" about 9% slower than before. To convert the new time values to minutes, multiply by 1.092267. One second is now about 0.01526, where it used to be 0.016667.
Shared Interests	Interests can now be shared between canvases. The Canvas field of an event can thus contain either a single canvas, or a dictionary or array that contains multiple canvases. See the <i>NeWS Programmer's Guide</i> for information.
Executable Matches	Executable matches are now permitted by executable Canvas , Name , and Action dictionary values in interests. These provide a highly efficient way of executing code according to the canvas on which an interest has been matched. Using this procedure, POSTSCRIPT language constructs such as case , which are normally used to vector a matched event to the correct handler, are made unnecessary.
Special Events	<p>NeWS now generates the following special events:</p> <ul style="list-style-type: none"> □ /Damaged: <i>Damage events</i> are generated for a canvas whenever it is graphically <i>damaged</i> (a definition of <i>damage</i> is provided in the chapter <i>Canvases</i> of the <i>NeWS Programmer's Guide</i>). The server will not send another damage event until the damage has been cleared by use of the damagepath operator. The Action key value for the event is null; the Canvas key value specifies the affected canvas. □ /Obsolete: The server generates <i>obsolescence</i> events to inform clients that an object is obsolete and that processes should destroy their references to it. The Action key of the event is a soft reference to the obsolete object.

- **/ProcessDied**: the server generates a **ProcessDied** event if a process dies (quits) when it is still referenced. The **Action** key of the event specifies the process that died. When a process is dead but is still referenced, its **State** key is set to **zombie**.

2.14. Processes

The following keys are new:

- \$error**
- errordict**
- Priority**
- Stdout**
- Stderr**
- SendContexts**
- SendStack**

Changes to Process Objects

X11/NEWS adds a new key to process objects, **Priority**. All processes now have a priority that they inherit from their parent. Currently this priority is not used to order their execution or favor certain lightweight processes (as it is in the UNIX kernel).

However, there is a minimum priority that a process must exceed if it is to run. If a process's priority is lower than this system-wide minimum, then it is queued until the system-wide priority drops. Currently NeWS processes can set their priority but cannot influence the system-wide priority.

The motivation for this change is to support the various flavors of X11 *Grab-Server* requests. During a grab, process and system-wide priorities are adjusted so that only important system processes (including the repeat key handler, the global input distribution process, and the connection managers) and the X11 connection requesting the grab execute. The change should not affect NeWS programmers, and Sun advises against developers using the mechanism for their own purposes: it is not a general purpose mechanism and is subject to change.

2.15. Changes in Error Handling

This section describes the differences in error handling between NeWS 1.1 and X11/NEWS. Note that error handling has changed since preliminary versions of X11/NEWS. Error handling is now more compatible with the *PostScript Language Reference Manual*.

Upon detecting an error, NeWS 1.1 would search for an error handler by looking for dictionaries called **errordict** on the dictionary stack. If it found an **errordict** containing the error name as a key, it would execute the associated value. The *PostScript Language Reference Manual* indicates that **errordict** is a special dictionary that can be found in **systemdict**. The error behavior described in the *PostScript Language Reference Manual* mentions no such dictionary stack search. In X11/NEWS, each process can have a private **errordict**. This **errordict** is found in **systemdict** and as a magic key of processes. (See the description of the **errordict** key in the *NeWS Programmer's Guide* for a description of how this dictionary is shared.) In X11/NEWS, this is the only dictionary that is searched for the name of the error; when initialized, it contains an entry for each error that the server can generate.

NeWS 1.1 would override the action listed in `errordict` if the error was `stackoverflow` or `execstackoverflow`. In X11/NeWS, the listed error handler is only overridden if the operand stack has less than 10 slots left before it overflows, or if the execution stack has less than 3 slots left before it overflows. Since operations such as `2000 array aload` can cause operand stack overflows while the operand stack is still fairly small, the new behavior is therefore somewhat less restrictive.

If no error handler was found in the dictionary search, the NeWS 1.1 default error action would compose the `$error` dictionary as mentioned in the *PostScript Language Reference Manual*, and place that dictionary in the `userdict`. If the process had no `userdict`, no `$error` dictionary would be generated. The conditional behavior of the NeWS 1.1 default error action caused problems in finding the `$error` dictionary after an error. In X11/NeWS, the `$error` dictionary, which is private to each process, can be found in `systemdict` and as a magic key of the process. (See the description of the `$error` key in the *NeWS Programmer's Guide* for a description of how this dictionary is shared.) Until a process has an error, the value is null; thereafter it is a dictionary containing the information mentioned in the *PostScript Language Reference Manual*.

2.16. Changes to Magic Dictionaries

“Magic dictionaries” are used by some of the NeWS types, including canvases, events, colors, and processes. These had a fixed set of keys in NeWS 1.1. X11/NeWS allows you to add your own keys beyond the fixed set. The fixed set of keys are still there when the object is created and still cannot be removed.

Note that if you misspell one of the existing keys, you will get no complaint, but will have created a new key *for that one instance of the object only*. In NeWS 1.1 the documentation warned programmers not to put arbitrary keys in magic dictionaries. However, if you did so, the implementation meant that the key would be added to all instances of the dictionary, becoming, in effect, global. For example, we found many programs that misspelled `/TimeStamp` as `/Timestamp` when setting the time in events. Other programs would `def` some key while the uppermost dictionary on the stack was an event. It is possible that your programs rely on the NeWS 1.1 behavior in such cases.

2.17. NeWS Fonts

The source format for NeWS fonts has not changed. However, the format of the NeWS native bitmap fonts (in the `$OPENWINHOME/lib/fonts` directory, usually with the extension `.fb`) has changed in X11/NeWS and is thus incompatible with the NeWS 1.1 format. The new format is used by both NeWS and X11 code; thus, the two protocols can share the same bitmap fonts. Note that the X11/NeWS bitmap format is different from the X11 format used by versions of the X11 server released by MIT for Suns.

The format of the NeWS font family files (in the `$OPENWINHOME/lib/fonts` directory, usually with the extension `.ff`) also changes with X11/NeWS.

These two changes may not affect you, since the X11/NeWS server comes with a full set of fonts in the new format, but if you have developed your own fonts, you must proceed as follows:

1. Change directory to your own font directory.
2. Run `convertfont` to convert your font to the new NeWS format.
3. Run `bldfamily` to rebuild the font family files in their new formats.

Adobe Bitmap Font Format

The NeWS ASCII bitmap font format generated by `convertfont(1)` was changed to be compatible with the very latest (version 2.1) Adobe bitmap font format specification. This is also what X11 uses, with a slightly different interpretation. NeWS used version 1.4. However, the `convertfont` program understands the old and new versions of the Adobe bitmap font format, as well as the X11 interpretation of version 2.1. The differences among these versions are very small. `convertfont` also understands old (NeWS 1.1) format font files. To convert a font, run `convertfont` on it. The new and old formats have new magic numbers, so there is no chance of NeWS or `convertfont` getting confused.

For more information on font support in X11/NeWS, see the *X11/NeWS Server Guide*.

Scalable Fonts

A new font format, OpenFont, is supported. This new format allows a single font file to be scaled to any point size and rotated to any angle. The suffix for these fonts is `.f3b`. Typing `ls *.f3b` in the directory `$OPENWINHOME/lib/fonts` shows which fonts are in this format. A list of these fonts is provided in the *X11/NeWS Server Guide*.

2.18. Changes to CPS

If your program includes a header file generated by `cps`, you no longer need the following in your C client source:

```
#include psio.h
```

It is included for you when `cps` creates the `.h` file.

Changed CPS routines

`ps_flush_PostScript()` used to call `psio_error()` itself and would `exit(0)` if there was an error. It now does not do this — it is the responsibility of the client to check for errors.

`ps_close_PostScript()` now closes `PostScriptInput` as well as `PostScript`.

New CPS routines

There are many new functions for examining the tokens in the client-side input queue.

```
ps_check_PostScript_event()  
ps_query_PostScript_event(tag)  
ps_read_PostScript_event(ptag)  
ps_peek_PostScript_event(ptag)  
ps_skip_PostScript_event()
```

The above routines are supported by new `psio` routines:

```
ps_checkfor(PostScriptInput,PSIO_CHECK_INPUT,0)
ps_checkfor(PostScriptInput,PSIO_FIND_TAG,tag)
ps_currenttag(PostScriptInput,PSIO_GET_TAG,ptag)
ps_currenttag(PostScriptInput,PSIO_PEEK_TAG,ptag)
ps_skip(PostScriptInput)
```

The internal routine `next_user_token` was changed to `ps_next_user_token()` to avoid name clashes.

2.19. Memory Management

NeWS provides a facility of *reference counting* that allows objects to survive as long as references to them exist; *references* are created by the system whenever one object becomes associated with another.

In NeWS 1.1 there was one kind of reference to an object. There are now two kinds of reference: one controls how long an object is in use, and the other is used to track an object in the system. When all of the first kind of reference is destroyed, the object is said to be obsolete. Processes with references to obsolete objects should destroy those references to allow garbage collection to occur.

When no references of any kind exist to an object, its storage is automatically reclaimed. See the *NeWS Programmer's Guide* for details.

Changes to X11

Changes to X11 21

Changes to X11

The X11/NeWS server is a different piece of software from the MIT X11 distribution. No changes have been made to the X11 protocol. X11/NeWS is compatible with X11 Revision 3. However, there are some differences between X11/NeWS and X11 Revision 3 that should be noted:

- X11/NeWS typically supports multiple color visuals and chooses to make a static color visual the default. This promotes sharing of limited color slots. This is different from MIT's X11.
- Color applications should not assume that the default visual is dynamic (like the MIT distribution). They need to look at the visuals returned from the X11/NeWS server and locate the dynamic one. Better still, they should use the static one so that colors are shared.
- The X11-related files in the distribution are limited to the server, Xlib, some demos, and some utilities. The X11 Revision 3 version of `xterm` is provided in `$OPENWINHOME/demo`: it retains the status of *pass through* or demo software. Other libraries and applications should be gleaned from the MIT X11 distribution tape.
- The retained values of closedown mode are unimplemented.
- Although the X11/NeWS font utilities accept the same font source format as is found on the MIT distribution, the binary format is incompatible. The X11/NeWS server supplies the same fonts as MIT, plus many more. If you have your own fonts, you must run `convertfont` and `bldfamily` on them. See the *X11/NeWS Server Guide* for information on fonts and font naming conventions.
- The default window manager `pswm` is ICCCM compliant, as all release 4 window managers will be. The ICCCM specifies the split between applications and the window manager. See the *X11/NeWS Server Guide* for a discussion of the window manager. See the manual page for `xnews` in the *X11/NeWS Server Guide* for a description of how to use the environment variable `X11ONLY` to prevent the window manager from starting.