NeWS™ 1.1 Release Notes

1

NeWS 1.1 Release Notes

This document describes the notable changes, enhancements and fixes in release 1.1 of NeWS.

NeWS 1.1 is primarily a bug fix release, plus:

- it works on SPARC (Sun-4) machines
- it has some changes for better font support
- it supports the full POSTSCRIPT language font model (including **definefont**)
- dashed lines work
- POSTSCRIPT language previewing is much better, plus there is a new version of `psview`
- typing keys repeat
- `psterm` is faster and has other enhancements
- a Kanji font is now provided
- the communication package with the server has been replaced
- it supports journalling (the recording and replaying of events), with a cassette player-like control panel to record/playback/stop journalling
- its root menu has been reorganized
- `psload` (a load average routine)
- it can create windows with scrollbars and menus in many styles
- it allows selections to be transferred between SunView1 and NeWS
- new demos

1.1. Changes and Enhancements

Communication Package with Server Replaced

NeWS used to use standard I/O to communicate between CPS and the server. Now we are using a new package called `psio`. If you are referencing the global variables `PostScript` or `PostScriptInput` you will have to use this new package. This was done to enhance portability between different environments. Generally, `psio_` will replace the `f` prefix on calls such as `feof` or `ferror`.

For calls such as `fileno` simply prepend the `psio_` prefix.

CAUTION **Failure to make this change will result in compile-time errors (see the *NeWS Manual*, Chapter 9, *C Client Interface* and the `psio(3)` manual page).**

File Name Changes

In the interests of portability, some font, icon, file, and image filenames have been changed to fit the System V 14-character filename limitation. The following files in `$NEWSHOME/smi` are the only ones that might need to be changed in your programs:

Table 1-1 *Changed Filenames*

Old Name	*New Name*
`Add-On_Hardware`	`Catalyst/Hardware`
`Architectural_and_Civil_Engineering`	`Catalyst/Arch_Civil_Eng`
`Artificial_Intelligence`	`Catalyst/AI`
`Biological_and_Physical_Sciences`	`Catalyst/Sciences`
`Data_Base_Management_Systems`	`Catalyst/Data_Base`
`Earth_Resource_Engineering`	`Catalyst/Earth_Resource`
`Electrical_Engineering`	`Catalyst/Electrical_Eng`
`Mathematics_and_Statistics`	`Catalyst/Math_Stat`
`Mechanical_Engineering`	`Catalyst/Mech_Eng`
`Office_Automation`	`Catalyst/Office_Auto`
`Project_Management`	`Catalyst/Proj_Manage`
`Software_Engineering`	`Catalyst/Software_Eng`
`mona-smile-hires.im8`	`mona-hires.im8`
`virginofrocks.im8`	`virgino.im8`
`washing-hires.im8`	`wash-hires.im8`

Font Library Management Changes

There is a new font library management scheme that allows all font filenames to be short. NeWS is now completely insensitive to font file names. The changes are:

New **findfilefont** Primitive

findfilefont

string **findfilefont** font

Reads in a family definition from the file named by *string*, and returns a unit high font that refers to it.

enumeratefontdicts Primitive Changed

enumeratefontdicts – **enumeratefontdicts** names

Now pushes the font *family* filenames onto the stack, rather than particular font names.

NOTE *Only* `init.ps` *should ever call this, anyway*

enumeratefontdicts is supposedly only used as an internal (not public) interface to the NeWS-specific disk font libraries. Anyone wanting to enumerate the set of fonts should be using the POSTSCRIPT language mechanism of scanning **FontDirectory**.

The change in font family filenames brings increased portability, but also means that it is no longer possible to deduce the family name of the font from the name of the file: all that can be deduced is that it is a font-family file. The old semantics could not have been carried over in a useful way.

- **FontDictionary** is now initialized and available to the C world, rather than just being accessible in POSTSCRIPT language.
- **findfont** now looks in **FontDictionary**, as per the *PostScript Language Reference Manual*, rather than its own private database.
- `init.ps` now uses **enumeratefontdicts** and **findfilefont** to initialize **FontDictionary**.

Full Font Model Now Supported

definefont, setcachedevice, and **setcharwidth** now all work. **definefont** now works; this makes NeWS capable of previewing many more kinds of POSTSCRIPT language output without modification.

NOTE **charpath** *does not work: if you read the Adobe specification carefully you will see that as it defines the font mechanism,* **charpath** *cannot work on user-defined fonts, and all fonts in NeWS are user-defined.*

Character Mapping

All fonts now use the Adobe collating sequence. This can be overridden by using **definefont** after changing the **Encoding**.

Other Changes Related to Fonts

`bldfamily` and `dumpfont` have been upgraded to:

- generate the new shorter font filenames
- regularize the naming conventions and shorten the extensions.
- fix a bug having to do with building metric files

Bitmap font transforms are now done a character at a time, not a font at a time. This substantially improves performance when drawing single characters at strange orientations.

A Kanji Font

A Kanji font is now provided.

Default Line Width Changed

The default value for line width was changed to 0 (from 1) and the line quality was changed to 1 (from 0). This means that if you don't specify these values (with **setlinewidth** and **setlinequality**, respectively), then you will get 1/72" wide lines, even if you scale your coordinate space. If you just specify the line width, then you will get high-quality wide lines. If you want to have fast skinny lines (regardless of what is specified for the width), then set the line quality to 0.

This fixes a problem where NeWS would not display wide lines if you changed the line width and didn't change the line quality.

Dashed Lines Work

NeWS 1.1 fully supports dashed lines using the standard **setdash** and **currentdash** POSTSCRIPT language primitives.

dashpath is an extension that lets you apply the current dash pattern to the current path, replacing the current path.

Emulation of statusdict

We have added a **statusdict** to **systemdict** for users needing extreme printer compatibility. The file `statusdict.ps` implements the **statusdict** dictionary and its printer-specific operators such as **printername** and **setsccbatch**, as specified in Section D.6 of the *PostScript Language Reference Manual*. Many of these operators are pseudo-implemented, since they have no meaning in a window system. `statusdict.ps` is loaded automatically by `init.ps` at start-up.

New Version of `psview`

The changes described above permit NeWS to do a much better job of previewing existing POSTSCRIPT language files. Accompanying them is a new version of `psview`(1) POSTSCRIPT language page previewer. The new version looks for the POSTSCRIPT language `%%EndProlog`, `%%Page`, and `%%Trailer` conventions to determine where pages start and provides a slider to move to any page, and a menu to go to the first, previous, next or last page.

This preprocessing occurs on the client side, so the new `psview` serves as a good example of an application that divides processing between the C client side and the server. Its source is in `$NEWSHOME/clientsrc/client`

Journalling

A new package has been added which supports journalling, the ability to record and play back NeWS user input events. The file `$NEWSHOME/lib/NeWS/journal.ps` implements the following three procedures: Begin replaying from the journalling file. The default filename is `/tmp/NeWS.journal`. Start a journalling session by opening the journalling file and logging user actions to it. Ends a journalling session started by **journalrecord** and closes the journalling file.

The replay is at a very low level, so the system should be in exactly the same state at the beginning of the replay as it was at the start of the journalling session — exactly the same windows in the same positions on the screen, the same user running the system from the same directory, etc. **journalplay** does take care of repositioning the mouse for you.

Using Journalling

When you select 'Applications ⇒ Journal' from the root menu, a new pull-right menu is added to the root menu. From this you can start recording user input events, stop recording, play them back, or remove journalling. You can also bring up a nifty control panel with buttons for controlling journalling, the speed of playback, auto-repeat, the journalling file to use, and so on (see the `journalling(1)` manual page).

NeWS Libraries

NeWS 1.1 is built with SunOS Release 3.4 libraries. It will still run on 3.2.

Repeating Keys in NeWS

By default, the standard typing array, but not any function keys or shift keys, repeat 20 times per second, after a .5 second threshold. Repeating keys are implemented by a standalone repeat-keys package, `$NEWSHOME/lib/NeWS/repeat.ps`, loaded as part of the Extended Input System started by `init.ps`. When multiple keys are depressed, only the last key down is repeated. When the last key pressed is lifted, all repetition stops.

The threshold and repeat rate can be adjusted to your preference by modifying two values in the **UserProfile** dictionary. You can put something like the following in your `user.ps` file to change them:

```
UserProfile begin
  /KeyRepeatThresh   1 60 div 5 div def
  /KeyRepeatTime     1 60 div 12 div def
end
```

Problems with Function Keys

The repeater process works at input distribution priority 2, to get ahead of keyboard focus distribution. SunView1 function keys are at priority 3, so they don't repeat. As a side effect of the changes that permit repeating keys, you may have to modify any code in your `user.ps` or `startup.ps` which defines keys, so that it also has input redistribution priority 3. NeWS now has a built-in procedure for defining function keys; see *Assigning Function Keys* below.

Assigning Function Keys

The **bindkey** utility lets you assign a procedure to a function key.

bindkey

key arg **bindkey** –

Causes the given procedure to be executed whenever the named key goes down. **bindkey** inspects *arg* and reacts as follows:

- If the *arg* is a string, '{ (*arg*) forkunix }' is executed when *key* goes down.
- If *arg* is executable, it is executed when *key* goes down.

(If neither, it's a no-op.)

Example:

```
/FunctionF7    { dup begin
                 /Name /InsertValue  def
                 /Action (!make\n)   def
                end redistributeevent
               } bindkey
/FunctionF8    (sv2news_put) bindkey
/FunctionF9    (news2sv_put) bindkey
```

binds the string `!make` to key F7, and assigns the NeWS←→SunView1 selection converters to F8 and F9.

unbindkey

key **unbindkey** –

Removes the binding for a given key. There is no need to call **unbindkey** before rebinding a key to a new value using **bindkey** — the new value will replace the old.

Example:

```
/FunctionF9  unbindkey
```

will undo the effect of the **bindkey** command for key F9.

New Version of `psterm`

The new version of `psterm` is faster and has extra features, including:

- **`-li`** #— specifies the number of lines
- **`-co`** #— specifies the number of columns
- **`-xy`** *x y*— specify origin (use with the **`-f`** fixed-size option)
- Rows & columns are properly extracted from `termcap`(5), not the `psterm`'s parent process.
- The editing characters are determined by first checking in `WINDOW_TTYPARMS`, then looking at controlling terminal (if any), otherwise defaulting to a standard set.
- The pattern matcher has been rewritten for better performance.
- Pseudo-ttys are initialized and handled better.
- `/etc/utmp` is handled properly.
- Page mode added.
- Automatic margin option added.
- There is a menu to turn page mode and automatic margin on and off.
- Visual bell added.

New *Go* Game

The *go* demo program has been rewritten to be interactive, and hence serve as a more useful example of client-side programming. It now:

- uses the *litewin* window package
- permits client-side input (LEFT mouse button sets black, MIDDLE sets white, and RIGHT removes stones)
- handles damage on the client side
- permits the board to be scaled
- updates its icon every time it is closed
- uses color

New Backgammon Game

A backgammon game has been added to the games menu; it resides in `$NEWSHOME/demo/gammon`. Each counter and point on the board is a separate canvas. The computer moves its counters along pleasing curved paths.

New Images

Some new images have been added to the 'Preview' menu, including a multi-page overview of NeWS.

Coexistence with SunView1

The following facilities make it easier to use SunView1 binaries and NeWS at the same time.

Ensuring that the Selection Service is Running

You can run SunView1 binaries while running NeWS, but the `selection_svc` program must be running for SunView1 programs to be able use SunView1's Selection Service to cut and paste between their windows. `$NEWSHOME/bin/ensure_sel_svc` is a small program which looks to see if there is a (SunView1) Selection Service available, and start one if there isn't. The demo menu code (in `$NEWSHOME/lib/NeWS/demomenu.ps`) calls `ensure_sel_svc` before running any of the SunView1 applications that need the Selection Service, and if the Selection Service is not available, it starts `selection_svc`. If you start up SunView1 programs yourself, you should use `ensure_sel_svc` in a similar fashion.

Pasting Selections Between NeWS and SunView1

The utility shell scripts `news2sv_put` and `sv2news_put` copy the NeWS selection to the SunView1 shelf and vice-versa. They are available from the 'SunView1 ⇒ Selection Transfer' menu as 'NeWS to SunView1 Shelf' and 'SunView1 to NeWS Shelf'.

Both use the program `news_selection` to get the NeWS selection and set the NeWS shelf. `news2sv_put` uses the program `set_selection` to set the SunView1 shelf. `sv2news_put` uses the standard `get_selection`(1) utility to get the SunView1 selection.

The NeWS Socket

In Release 1.0 of NeWS, the socket number to listen on for connections from clients was defined to be `2000` in `init.ps`. In NeWS 1.1, the official socket assigned to NeWS (allocated by DARPA) is `144`. However, the NeWS server now looks at the `NEWSSOCKET` environment variable (if defined) for the name of the socket on which it will listen for connections. The format of this name is `%socket1`*nnn* where *nnn* is the socket number. For example, to get NeWS to listen on socket `2001`, type

```
paper% setenv NEWSSOCKET '%socket12001'
paper% news_server
```

The socket on which NeWS listens can also be set in your `user.ps` file with a line of the form:

```
/NeWS_socket (%socket12001) def
```

This will override any socket specified in the `NEWSSOCKET` environment variable. If there is no socket specified in the `NEWSSOCKET` environment variable or in `user.ps`, NeWS will try to listen on socket `144`. Since this is a privileged socket, unless `news_server` is running as root, the attempt to listen on `144` will fail, and the NeWS server will then try to listen on socket `2000`.

New Security Feature

There is now a dictionary called **RemoteHostRegistry** maintained in the server whose keys are the names of hosts which are allowed to connect to the NeWS server. When NeWS starts up, this just contains the name of the local host. Whenever a connection is attempted, the name of the remote host is checked to see if it is in this dictionary, and if it isn't then a message is issued to the user and the connection is closed.

This is exactly the same security that the X window system has, although the NeWS implementation is significantly shorter than the above paragraph.

The shell script `newshost(1)` allows you to manage the registry of permitted host names from the command line. `newhost` manipulates the registry on the NeWS server specified by the `NEWSSERVER` environment variable. If you want to disable this security feature, set the POSTSCRIPT language variable **NetSecurityWanted** to false.

New Dictionary of Common Colors

ColorDict is a new dictionary which contains named colors. It is implemented by `colors.ps` which is loaded by `init.ps`. The color names are taken from the lib/rgb values in X.10V4. Here are some examples:

```
/Aquamarine          112 219 147 RGBcolor def
/MediumAquamarine     50 204 153 RGBcolor def
/Black                0 0 0 RGBcolor def
/Blue                 0 0 255 RGBcolor def
/CadetBlue           95 159 159 RGBcolor def
/CornflowerBlue       66 66 111 RGBcolor def
/DarkSlateBlue       107 35 142 RGBcolor def
```

... where **RGBcolor** simply converts 0 – 255 color values into NeWS colors:

```
/RGBcolor {                          % R G B => color
                                     % (Takes traditional 0 – 255 arguments for R G B)
   3 {255 div 3 1 roll} repeat rgbcolor
} def
```

New 'Colornames' Demo Program

The 'Color Demo' program (in `$NEWSHOME/demo/colornames`) displays the various colors in **ColorDict** in a scrolling window. It also demonstrates the new scrolling window package, and the use of new menu parameters to create a very different style of menu with a menu bar and row-column menu pop-ups.

Changes to the Class Mechanism

/new (in class **Object**) now can be sent to instances as well as classes. It works by simply sending **/new** to the class of the instance, producing identical results.

Changes to the Menu Package

Polymorphic Menu Keys

Menu keys may be strings, icon names, procedures, or class instances. The string and icon name simply display the corresponding object. In addition, the menu keys may be wrapped in an array. This allows for font and color changes, and slight adjustments in the x,y location of the key relative to its default position. It also allows passing additional arguments to the user's procedure or class instance. Thus

```
[/MyIcon 1 0 0 rgbcolor .5 .5]
```

is a key that shows 'MyIcon' in red with a slight (.5 .5) offset.

The colornames demo has examples of advanced menu key usage.

Vertical/Horizontal/Table Menu Layout

The **LitePullRightMenu** subclass of class **LiteMenu** now allows general item layout. This is controlled by the **/LayoutStyle** class variable which may be **/Vertical**, **/Horizontal**, or an array of rows and columns; the default is **/Vertical**.

The colornames demo has examples of advanced menu layout usage.

/show Replaced by **/showat**

The **/show** method has been replaced by **/showat** which takes either the x,y location to be used for the top left of the menu or an event whose x,y location will be used. Any menu started by an action procedure for **forkeventmgr** which used to look like:

```
/MenuButton {/show MyMenu send} DownTransition ...
```

can simply replace **/show** with **/showat**:

```
/MenuButton {/showat MyMenu send} DownTransition ...
```

because the action procedure is called with the **MenuButton** event on the stack, from which **/showat** can figure out where to display the menu.

NOTE *There is a direct replacement for* **/show**, *but you are encouraged not to use it since using the current mouse location to display menus causes problems with journalling. See "In* **LiteMenu**" *below for more information.*

Searching for Keys in Menus

The new **/searchkey** method searches for the given key's position in the menu and returns true and its location if found, false otherwise. This is generally used in conjunction with **/insertitem**, **/deleteitem**, or **/changeitem** to alter an existing menu.

Searching for Keys in Menus

The new **/searchaction** method searches for the given action's position in the menu, returning true and its location if found, false otherwise. This is generally used in conjunction with **/insertitem**, **/deleteitem**, or **/changeitem** to alter an existing menu.

Other Name Changes

For consistency with other parts of NeWS, the user preferences **StrokeSelection** and **CenterItems** become **StrokeSelection?** and **CenterItems?**, respectively. To avoid conflicts with `bind` and `setautobind`, **/fork** becomes **/activate**. **ShowAtMouse?** was removed.

Internal Changes in the Menu Package

The main change that affect clients using the internals of menus is that each menu item is now a separate dictionary containing its key, action, position and width data, and so on. The keys can also be things other than strings, such as icons, procedures, and the like.

The **MenuKeys** and **MenuActions** arrays have been replaced by a single array, **MenuItems**, whose elements are dictionaries. The initial contents of the dictionary for class **LiteMenu** are:

Key Menu Action w h

where **Key** is the menu key, **Menu** is a nested menu (if the action to **/new** was a dictionary), **Action** is the menu callback (if the action to **/new** was not a dictionary), and **w** and **h** are the size of the key. Class LitePullRightMenu adds the fields:

x y X Y W H

where **x**, **y**, **w**, and **h** are the bounding box of the key itself, and , **X**, **Y**, **W** and **H** are the bounding box of the table entry of the menu.

Changes to the Window Package

There is a new window class defined in `$NEWSHOME/lib/litewin.ps`, **ScrollingWindow**. This has two simple scrollbars (see **SimpleScrollbar** in *Changes to Class LiteItem* below) in the frame margin. The two scrollbars are initialized to return values between 0 and 1. When viewing a typical document, this corresponds to a position within the document, where 0 indicates the beginning of the document, 1 is the end, and a fraction is somewhere in between.

The colornames demo uses the new scrollbars.

Internal Changes

PaintClient, FrameLabel, IconLabel, IconImage, and **ClientMenu** were changed to be class variables rather than instance variables. This makes subclassing windows easier.

ClientWidth and **ClientHeight** have been added to the class variables. These are the current size of the **ClientCanvas. FramePath, IconPath,** and **ClientPath** all take the bounding box (x y w h) on the stack and make the current path be the desired path. **FramePath** defaults to rectangular, while **IconPath** and **ClientPath** default to **FramePath.** Thus changing **FramePath** to be '{ovalpath}' changes all three paths to be ovals.

The **/destroy** method now simply calls **FrameDestroy** and **ClientDestroy;** the default **ClientDestroy** executes 'currentprocess killprocessgroup'. This used to be done by **/destroy** itself.

Changes to Class *LiteItem*

Two new classes, **ScrollbarItem** and **SimpleScrollbar,** have been added. The former is an abstract superclass which reflects the structure of scrollbars, but does not entirely implement one. **SimpleScrollbar** implements a simple, one button scrollbar.

New **/reshape** Method

/reshape is a new method which moves and resizes items. You should use it instead of passing **/new** the width and height followed by a **move. /new** now can omit height and width parameters and use the new **/reshape** message. This was done to bring items more in sync with windows and canvases. Thus:

```
/foo (Foo:) ... mycanvas 200 20 /new CycleItem send def
10 100 /move foo send
```

should be replaced by:

```
/foo (Foo:) ... mycanvas /new CycleItem send def
10 100 200 20 /reshape foo send
```

/map and **/unmap**

Two methods, **/map** and **/unmap,** have been added to map and unmap the item's canvas. The default is mapped, thus these are not called by casual clients.

/getvalue and **/setvalue**

Two methods, **/getvalue** and **/setvalue,** have been added to get and set the item's **ItemValue.**

Internal Changes

In **MessageItem, /print** becomes **/printstring.** Items now handle **/Damage.** Thus, if an item is created with **/Transparent** set to false, it will repaint by getting damage events which in turn call the **/paint** method for the item.

Event Manager Changes

EventMgrFore and **EventMgrAft** now default to being empty. They used to be **gsave** and **grestore**, respectively.

Executable Interests for forkeventmgr

If any of the interests in the dictionary or array of interests passed as an argument to **forkeventmgr** is executable, it will be executed. If so, it is assumed that it will create an interest and **expressinterest** in it. This interest should be sure to install a dictionary in the interest's **/ClientData**, and to install the callback procedure in the **ClientData /CallBack** field. The executable can simply perform some form of initialization, if required, although this is not the intent.

New Key in the Canvas Dictionary

A new key, **Interests,** has been added to the Canvas dictionary:

Interests

– **Interests** array

The interest list for the canvas is returned as an *array* of events. The order of events in the *array* is the priority order of the interests, highest first.

Writing Canvas Contents to a File

writecanvas

filearg **writecanvas** –

There are four new primitives: **writecanvas, eowritecanvas, writescreen,** and **eowritescreen.** The prefix **eo-** indicates even/odd winding rule as opposed to the nonzero winding rule. Each primitive takes one argument, *filearg*, which is either a filename string or a file object.

These primitives write canvas images to files as Sun format rasterfiles (see `rasterfile(5)`. If you want to send an image to a UNIX process communicating with the NeWS server, 'currentfile writecanvas' should do the trick.

All these primitives write the region outlined by the current path in the current canvas. The rasterfile will contain the smallest rectangle that can enclose the region. Pixels in the rasterfile but outside of the region will be 0. **writecanvas** writes only pixels belonging to the current canvas; it will take the pixels from both the screen and visible and invisible parts of the canvas. If the canvas is non-retained, its covered non-retained pixels will be written as zero. **writescreen** writes the same region, but it only writes pixels from the screen, and it includes pixels from canvases that overlap the current canvas.

You would use **writescreen** to make a conventional screen dump. You would use **writecanvas** if you built a bitmap painting program and wanted to save the image in a file. If the current path is empty, the whole canvas is written, thus

```
framebuffer setcanvas (/tmp/snap) writescreen
```

snapshots the entire screen in a file named `/tmp/snap`.

New sleep procedure

A new **sleep** procedure has been added to `util.ps`:

sleep

interval **sleep** –
sleep sends itself an event timestamped *interval* in the future, and returns when that event is delivered. *interval* is in minutes, with 16 bits of fraction. The usable resolution is about 10 milliseconds.

New errored Procedure

errored acts just like the **stopped** primitive, but for errors. Because this is generally what **stopped** has been used for, **errored** is recommended

Using **errored** also allows the debugger to work properly. Thus, if you are currently using **stopped** as a way to detect errors, simply replace it with errored .

New **StandardErrorNames** Array

StandardErrorNames is an array of the names of the standard errors. It is used by **errored** and the debugger, and is available for other programs' use.

IsQueued Field Added to Events

Another read-only field has been added to the event "dictionary", **IsQueued**. **IsQueued** is true when the event has been put in the input queue (by **sendevent**) and has not yet been delivered.

Event Logging

The file `eventlog.ps` defines a procedure to turn logging of event distribution on and off, and a dictionary of events which should be excluded from the log. "Logging" means that a copy of each event is printed as it is taken out of the event queue for distribution. This is useful for debugging the server and clients using events heavily. It adds **eventlog** and **UnloggedEvents** to **systemdict**.

eventlog

bool **eventlog** –
Starts or stops event logging according to whether or not *bool* is true or false. The fields of the event which are printed are **Serial#**, **TimeStamp**, **Location**, **Name**, **Action**, **Canvas**, **Process**, **KeyState**, and **ClientData**. Here's a sample log message:

```
#300 1.582 [166 161] EnterEvent 1 canvas(512x512,root,parent) null [] null
```

UnloggedEvents

This is a dictionary of event names which are considered uninteresting to the event logger; an event whose **Name** is found in this dictionary will not be logged. The default definition of **UnloggedEvents** is

```
/UnloggedEvents 20 dict dup begin
  /Damaged dup def
  /CaretTimeOut dup def
% /EnterEvent dup def
% /ExitEvent dup def
  /MouseDragged dup def
end def
```

Support for Multiple Process Communication

CPS has support which allows clients to establish private communication "channels" with multiple processes running within the NeWS server, talking to each through a *ClientID* channel. Refer to the section entitled *The CID Utilities* in Chapter 9, *C Client Interface* in the *NeWS Manual.*

New extenddamage Operators

extenddamage and **eoextenddamage** add the current path to the damage shape for the current canvas. A **/Damaged** event will be sent to those processes that have expressed interest. **extenddamage** uses the nonzero winding number rule, while **eoextenddamage** uses the even/odd winding number rule.

1.2. Changes in Semantics

Some minor changes have been made to the semantics of some NeWS procedures. Most are side-effects of enhancements described in the *Changes and Enhancements* section; here are the others.

autobind Semantics in Conformance with bind

Keywords being autobound are now looked up in the whole dictionary stack, not just in **systemdict**.

Avoiding Overlap of Method Names and Operator Names

Some method names have been changed to avoid colliding with system operator names. If you are working with menu and *liteitem* internals, you will have to modify your code, otherwise you will have problems using **bind**. The changes are as follows.

In **LiteMenu**

The **/show** method has been renamed **/popup**. The **/fork** method has been renamed **/activate**.

NOTE *You should probably replace your invocations of* **/show** *with* **/showat** *to avoid dependency on the current mouse position; see "***/show** *Replaced by* **/showat***" above for an explanation of* **/showat**.

In **MessageItem**

/print has been renamed **/printstring**.

Checking for Class–Operator Name Conflicts

If you are working with classes, here is a way to see if you have system operator names in your own code. Simply load in these two procedures using `psh`(1):

```
/methcheck {          % dict => - (print out sys name clashes)
{
   pop systemdict 1 index known {
      systemdict exch get dup type /operatortype eq {
         (Name Clash: %\n) printf
      } {pop} ifelse
   } {pop} ifelse
} forall
} def
/checkall {           % - => - (print out name clashes for all systemdict classes)
systemdict {
   dup type /dicttype eq {
      dup /ClassName known {
         1 index (Checking: %\n) printf
         methcheck pop
      } {pop pop} ifelse
   } {pop pop} ifelse
} forall
} def
```

. . . then execute 'checkall' after loading your classes.

NOTE *This relies on your classes being installed in* **systemdict**.

cvs Semantics in Accordance with Specification

In release 1.0 of NeWS the **cvs** primitive prepended a / to strings if the object being converted was a nametype object. The Adobe definition is to not include the / in the resulting string, so NeWS 1.1 has been changed to match the specification. For example, '/x ( ) cvs' now returns '(x)' rather than '(/x)'.

The change to no longer include the / means that the trick of changing a string to an executable object and executing it to put the values contained in it on the stack no longer works. If you have been using this technique, you must change your code to check the type of an object *before* converting it to a string with **cvs**, and prepend a / if the object is a nametype. You will also notice this change if you have been using **printf**.

FontBBox Adheres to Specification

The **FontBBox** field of a font is now agrees with the Adobe specification: it is in the character coordinate system and must be transformed according to the **FontMatrix**.

fontheight, **fontascent**, and **fontdescent** (which were implemented in NeWS 1.0) should be used in places where **FontBBox** used to be — they essentially do the transformation.

setkeyboardtranslation and getkeyboardtranslation

setkeyboardtranslation now takes, and **getkeyboardtranslation** returns, a boolean, rather than a small integer. **true** means the kernel is interpreting the keyboard; **false** means keyboard interpretation is being left to PostScript code, as in **liteUI**.

currentcursorlocation Semantics Changed

The behavior of **currentcursorlocation** is subtly different in NeWS 1.1. You should probably not be affected by this change and can skip the following explanation.

The old behavior was to return the coordinates of the cursor on the screen at the time it was called; now **currentcursorlocation** returns the last mouse position to have been taken from the input queue for delivery to a client. Another way of saying this is that it used to return the location current at the tail (insertion end) of the queue; now it returns the position at the head (delivery end) of the queue. If there is a difference, it will be accounted for in mouse motion events currently in the queue waiting to be delivered.

The value of **currentcursorlocation** is only updated by events with their **Name** field set to **/MouseDragged**, **/EnterEvent**, and **/ExitEvent**. **setcursorlocation** does *not* update **currentcursorlocation**; it merely changes the values that will be inserted in new events as they are generated. Thus, it is possible for a client to call **setcursorlocation**, and then call **currentcursorlocation**, and have it seem that the first call didn't work — its effect won't have propagated through the queue yet.

1.3. Bug Fixes

Core Trash

Several causes of core trashes (where the NeWS server inadvertently modifies its own data in memory, usually resulting in a core dump) have been fixed. We believe these fix most if not all of the "mystery" trashes seen using NeWS 1.0.

Root Canvas Repaint

If you set the background color to a value other than the default in your `user.ps` file, the root canvas isn't repainted after the initialization procedure completes (thus failing to pick up the new color). Instead, place this override in your `startup.ps` file:

```
systemdict /DefaultRootGrayOrColor .75 put
```

It was a bug in NeWS 1.0 that this extra repaint occurred.

(Non)Retained Canvases

A major bug in the code that makes canvases retained has been fixed. It was a complex interaction between when the canvas was made retained, whether or not it was transparent, what else the client had done with the canvas, and the pixrect/clip cacheing mechanism.

Other Bugs Fixed

Here are some of the other bugs fixed in NeWS 1.1. If you have any questions regarding these bug fixes, you can, if you have the necessary Software Support Contract, contact Sun's technical support staff for more information; please mention the relevant *Bug ID*.

Bug ID	*Synopsis*
1004333	clean up screen better on exit
1004544	pie.ps does not appear to be needed in $NEWSHOME/lib/NeWS
1004601	Damage event generated for bottom canvas when covered
1004617	psterm should handle wrapped line selections correctly

Table 1-1 *Changed Filenames— Continued*

Bug ID	*Synopsis*
1004620	input focus not set when new window (psterm) under cursor
1004625	calculator answers not correct
1004628	anthromorphism of the start up message "I" is inappropriate
1004629	multi processes prompting for rect get same rect
1004663	make psterm support "sun"
1004678	support variable rows and columns in psterm
1004682	improve fixed-startup feature in psterm
1004685	port number 2000 conflicts with other applications
1004686	auto starting of SunView1 selection svc when in NeWS
1004695	errors in user.ps hang server
1004696	retained transcript that crashes the server
1004698	dumpfont problem with .metrics files
1004700	rotated text wierd results
1004704	want to change the header in psterm dynamically
1004705	psterm dumps core if no $TERM
1004738	test case puts server into hard loop
1004740	storing into event name field crashes server
1004785	font scaling takes a LONG time
1004786	changing /etc/termcap doesn't affect psterm in all cases
1004788	large calculator has characters in wrong place
1004831	using getanimated to define a line causes server to crash.
1004881	server not prepared for greater then 32 fds
1004976	setcanvascursor takes 3 args (not 4); omit canvas
1004983	NeWS cannot compare strings to names.
1004987	transform & itransform doesn't transform the x coordinate
1004988	hsbcolor operator fails with args > 4 decimal places
1005060	doc of ps_open_PostScript return value wrong
1005179	psterm does not deal with "tc" from termcap file correctly
1005180	psterm's larger than 80 coloumns report a warning
1005184	clip bug when child canvas is clipped by parent
1005201	lite{menu,window} shouldn't call currentcursorlocation
1005384	psterm: performance problem when REDISPLAY
1005411	psterm can leave screen dirt
1005517	Canvas as current dictionary can crash server
1005533	save & restore don't for graphics state
1005551	comparing a neg number to a float >= 10000.0 is wrong
1005565	NeWS keyboard handler occasionally misses shifts
1005620	de-reference of a NULL pointer in cs_stroke()
1005652	a memory freeing bug in ComputeMatchingVersion().
1005766	idenmatrix values are wrong
1005792	using undef in a forall loop fails intermittently
1005807	clientsrc Makefile doesn't use ld correctly
1005808	clientsrc makefiles could use some improvement
1005886	arraydelete return value documented wrong
1005887	PaintClient shouldn't be an instance variable
1005989	printf function in news does not work reliably

Table 1-1 *Changed Filenames— Continued*

Bug ID	*Synopsis*
1005995	server hangs-infinite msg (usually) "select: Bad file number"
1005996	lost precision in ascii rep. of #'s whose last digit is 5
1006008	Stop key does not halt autorepeat
1006089	currentpoint confusion between show and "path-builders"
1006096	need better message for unexpected signals
1006097	bad FONTPATH environment value => server crashes
1006098	mouse jumps to corner during startup
1006104	quitting NeWS server with rsh running causes a logout
1006109	psterm doesn't execute pause often enough
1006110	getanimated loses input events
1006129	StopText in litetext was changed to stoptext, but not in EOL
1006138	"No NeWS is Bad News" crashes server
1006139	Catalyst demo crashes when changing the displayed image
1006149	putenv fails for strings of length greater than 200
1006175	setdash should be implemented
1006177	getrect doesn't return an array, it returns a process.
1006199	psterm gets continuous input stream
1006353	problem with 24 -> 8 scanned image conversion
1006478	setcursorlocation doesn't transform from canvas coordinates
1006519	Playback of long journal script dies with stack overflow
1006558	creating and reshaping 154 windows crashes news server
1006589	imagemask operator does not behave correctly
1006611	psterm has problems with ^Z and with file completion
1006771	canvas's interest-lists should be accessible
1006777	popup menu leaves a zombie process on the stack
1006822	failure after return from file exec
1006825	setfileinputtoken references aren't reclaimed properly
1006837	rcheck/wcheck/status work strangely after closefile
1006872	/usr/NeWS/clientsrc/sc is an empty directory
1006873	newshost: usage line incorrect