HOTLINE!

Bulletin 11
February 5, 1988

HOTLINE! is published periodically by the Customer Support group of Xerox Artificial Intelligence Systems to assist its customers in using the Xerox Lisp environment. Topics covered include answers to questions that are most frequently asked of Customer Support, suggestions to help you work in the Xerox Artificial Intelligence Environment (XAIE) as well as announcements of known problems that may be encountered.

Feel free to make copies of individual bulletin pages and insert them in the appropriate place(s) in your Interlisp Reference Manual, Lisp Library Modules manual or other relevant manual. The documentation reference at the end of each topic can be used as a filing guide.

For more information on the questions or problems addressed in this or other bulletins please call us toll-free in the Continental United States 1-800-228-5325 (or in California 1-800-824-6449). Customer Support can also be reached via the ArpaNet by sending mail to AISUPPORT.PASA@Xerox.com, or by writing to:

Xerox AIS Customer Support
250 North Halstead Street
P.O. Box7018
Pasadena, CA 91109-7018
M/S 5910-432

In this issue

In response to user requests we have decided to have *HOTLINE!* cover all supported releases of XAIE, instead of Lyric only. Supported releases include Koto and Lyric. Each item now contains a "Release" field for any item that is release specific. The following topics are covered in this issue:

- How to recover from internal garbage collection table overflow
- Koto-Lyric readtable inconsistency
- Problems loading Sketch on 1186
- Is SEdit opening windows randomly?
- Using DEFSTRUCT
- Hard disk error after deleting file
- "Hard disk error – page not found"
- How to change a window's title font

Terminology

Terminology used in this *HOTLINE!* bulletin:

UG – User's Guide

AR – Action Request, a Xerox problem tracking number (e.g. AR 8321)

IRM – Interlisp Reference Manual

How to recover from internal garbage collection table overflow

Topic Internal garbage collection table overflow

Note: This bulletin corrects an error in Issue #6.5 of *HOTLINE!* Thanks to Bill van Melle for bringing it to our attention.

Release Koto and Lyric

Question What do I do when the system prompts me with: "Internal garbage collector tables overflowed. Too many pointers with reference count greater than 1" "The garbage collector is now disabled" "Save your work and reload as soon as possible"?

Answer Once the internal garbage collection is full, you will get the "GC disabled warning" message. After the message appears, the garbage collector is disabled. (There is no way to enable it). If you ignored the "GC disabled warning" message, you would be heading for a fatal loss of work.

When you receive the "GC disabled warning" message, we recommend that you save your work as soon as you can. Since GC is disabled, running GAINSPACE will not give any free pages back.

References IRM 22.11-22.12

Koto-Lyric readtable inconsistency

Please remove section #2.4 of *HOTLINE!* The problem and solution are incorrectly stated. The root of this problem was non-standard Koto FILERDTBL used to generate the bad file. Thanks to Bill van Melle for bringing it to our attention.

Problems loading Sketch on 1186

Release Lyric

Keywords Sketch, 1186 User's Guide

Problem The 1186 User's Guide does not properly document how a user can successfully load Sketch from floppies to an 1186.

Symptom After completing the loading sequence in the 1186 User's Guide, the user brings up a Sketch window and will notice that the graphics options are missing from the Sketch menu. In addition, attempts to draw a line using the mouse will cause Sketch to break.

Workaround After the following instructions in the 1186 User's Guide concerning the loading of Sketch, proceed in the following manner:

a. Insert the Lyric Library #3 floppy into the drive and type in the Interlisp Exec:

```
DIR {FLOPPY}
```

b. After the directory is displayed, type in the Interlisp Exec window:

```
(LOAD 'SKETCH.LCOM T)
```

(Note that by specifying T in the second parameter, the loading will not be as verbose.)

References AR# 9078
1186 User's Guide, page 74

Is SEdit opening windows randomly?

Release Lyric

Question Does SEdit open edit windows randomly?

Background The last time I invoked SEdit, it opened a tiny window right on top of the TTY window, so that it was immediately buried by the TTY window right after being created and before any editing could be done.

Answer SEdit saves a stack of previously used regions. The default behavior is to add a region to the stack only when an SEdit window is closed. When the SEdit region manager needs to know where to place a window it is about to open, it pops a window region off the stack of previously used regions. If the stack is empty, SEdit prompts for a new region.

If the variable SEDIT.KEEP.WINDOW.REGION is set to NIL, a window's region is also added to the regions stack when an SEdit window is shrunk. Then when an icon is expanded, the window will be reshaped to the next available region. Initially this variable is set to T, so that shrinking an SEdit window will not give up the window's region.

On a multi-user system, SEdit may appear as though it creates new windows at random locations, but it actually remembers previously used windows that other users may have defined.

One of the side-effects of calling the function SEDIT.RESET is it clears the region's stack. All SEdit windows should be closed before calling this function.

References Lyric Release Notes, Appendix B-10 and B-11

Using DEFSTRUCT

Release Lyric

Keywords DEFSTRUCT

Topic DEFSTRUCT automatically creates functions which may implicitly redefine existing functions.

Discussion Using DEFSTRUCT to define a structure automatically creates access, constructor and copier functions, as well as a type checking predicate. If a function has been previously defined with the same name as one of these new functions, the old definition will be replaced without warning (this includes system functions).

Example Under the XCL Exec, a user defines a structure as follows:

```
(DEFSTRUCT FILE NAME AUTHOR LENGTH)
```

Functions with the following names are automatically created:

```
Slot access funtions: FILE-NAME, FILE-AUTHOR, and FILE-
LENGTH.

A constructor function: MAKE-FILE.

A copier function: COPY-FILE.

A predicate for the new data type FILE: FILE-P
```

The Common Lisp functions FILE-AUTHOR and FILE-LENGTH are redefined by this DEFSTRUCT.

If the user really wanted to use FILE as the structure name, DEFSTRUCT has options for explicitly specifying the names of the resulting functions. For example:

```
(DEFSTRUCT (FILE (:CONSTRUCTOR CREATE-FILE)
    (:COPIER CREATE-FILE-COPY)
    (:PREDICATE IS-A-FILE?)
    (:CONC-NAME LOOKUP-FILE-))
       NAME AUTHOR LENGTH)
```

The access function names will now be LOOKUP-FILE-NAME, LOOKUP-FILE-AUTHOR and LOOKUP-FILE-LENGTH.

Reference AR# 8952
CLtL, Chapter 19, pages 305-320.
Xerox Common Lisp Implementation Notes, pages 53-55.

Hard disk error after deleting file

Release Koto on 1186

Keywords VERIFYERROR, READERROR, Hard disk error

Problem A "Hard disk error VERIFYERROR" or "Hard disk error READERROR" occurs when a file is being written to local disk, after a large file (>4000 pages) had been deleted. This error occurs because the freed disk pages are not updated properly. When the disk page is allocated to a new file, a verify error occurs. The error does not necessarily occur on the first disk write operation after the file is deleted.

This problem exists only on 1186 running Koto; it does not affect the 1108. The problem is fixed in Lyric.

Example User deletes a file, {DSK}<LISPFILES>SYSOUTS>MYLISP.SYSOUT, which occupied 11,500 disk pages. At some future point, MAKEFILE to local disk breaks with the error message "hard disk error VERIFYERROR." Logical volume scavenge of the Lispfiles volume from the system tool reports the following error:

```
[401B, 0] type=10049; missing pages [0...4300)
```

Workaround The files on the rigid disk must be backed up and the volume erased using the systemtools Erase! command. The error recovery procedure described in Hotline bulletin 11.7 will not work in this case.

A patch file, DOVELFPATCH.DCOM, is currently being prepared for distribution. If you need the patch file immediately, please call Hotline or contact AISupport.pasa@Xerox.com.

Reference AR# 8087
Koto 1186 UG pages 53 -56

"Can't find file page" error recovery

Release Koto and Lyric

Keywords SCAVENGEDSKDIRECTORY

Problem "Can't find file page" break during SCAVENGEDSKDIRECTORY

Background The example assumes that the user has reason to believe (e.g. a HARD DISK ERROR) that the local file system needed scavenging, and that he has logged out and first performed Scavenge! from the System Tools and then attempted SCAVENGEDSKDIRECTORY from Lisp. System Tools reported that page 0 of a file is missing. Lisp Breaks with "Can't find file page."

If the user did not first perform the System Tool scavenge, but rather began by performing the SCAVENGEDSKDIRECTORY, and page 0 of some file was missing, the Directory deletion would proceed, a new Directory would be created, but not all files would be added to the Directory.

This procedure will allow for the recreation of the Directory for all files except the one which was missing page 0; that file will be deleted.

Example Again, always Scavenge! Lispfiles from the System Tools first. It reconstructs the file system in a way that SCAVENGEDSKDIRECTORY cannot.

The Break "Can't find file page" will occur if the user attempts a SCAVENGEDSKDIRECTORY when page 0 of a file is missing. It is this page which contains the file name information necessary to rebuild the Lisp directory.

The old directory will be deleted, and the new directory created. Files will be added to the new directory until the attempt is made to add the file missing page 0.

In Lyric, the break will appear as a SIMPLE-DEVICE-ERROR with the message "In \PFFindPageAddr: Device error: Can't find file page."

In Koto, the break message will read 'HARD DISK ERROR "Can't find file page."'

Workaround The recovery procedure is the same in both Koto and Lyric.

From the break menu, select BT! to bring up a backtrace window, Bring up the stack frame for the function \LFOpenOldFile. (Select the function name with the middle mouse button in Koto, and the left button in Lyric to display the local arguments to the function.)

Interpret (\PFTrimHelper (\PFGetVol n) fileDesc 0), where n is the number of the volume you are trying to scavenge starting from 0 – check the number shown in the DSKDISPLAY window if you are uncertain with (DSKDISPLAY 'ON) – and fileDesc is the first argument to \LFOpenOldFile. The fileDesc argument appears in

the following form: {FileDescriptor}#nn, nnnnn. Shift select this argument into the \PFTrimHelper expression. It will appear something like (\VAG2 66Q 46670Q).

After evaluating this expression, you should expect to get another break (says "Page group not found"). In the break window interpret (RESET) and evaluate SCAVENGEDSKDIRECTORY again from the executive. It should run to completion this time (you will have deleted the damaged file).

Caution Do not call the undocumented functions \LFOpenOldFile, \PFTrimHelper, or \PFGetVol from any Exec or attempt to use them in any other way unless explicitly advised to do so by Xerox Customer Support.

Note: also that if you called SCAVENGEDSKDIRECTORY from outside the IL Readtable, the backslash functions will print and must be read with the additional backslash, and the upper and lower case will have to be preserved with escape characters; e.g. |\\PFTrimHelper|.

Reference AR# 9327
1186 UG Lyric Release, page 44
1108 UG Lyric Release, page 38
1186 UG Koto Release, page 54
1108 UG Koto Release, page 36

How to change a window's title font

Topic Using WindowTitleDisplayStream to change the default font for a window's title

Release Koto and Lyric

Keywords WindowTitleDisplayStream, DSPFONT

Question How do I change the default font for a window's title?

Answer To change the default to one of your own choosing, for example, TimesRoman 24:

```
←(SETQ OLDFONT (DSPFONT NIL WindowTitleDisplayStream))
←(DSPFONT (FONTCREATE 'TIMESROMAN 24) WindowTitleDisplay
    Stream)
←(CREATEW '(200 200 200 200) "My Window"]
```

To change the font back to the default:

```
←(DSPFONT OLDFONT WindowTitleDisplayStream)
```

Or define a function which will do it all for you:

```
←(DEFINEQ (MyWindowFontFn
    (REGION TITLE BORDERSIZE NOOPENFLG FONTDESCRIPTOR)
    (RESETLST  (RESETSAVE
        (DSPFONT FONTDESCRIPTOR
            WindowTitleDisplayStream)
        '(PROGN (DSPFONT OLDVALUE
            WindowTitleDisplayStream)))
        (CREATEW REGION TITLE BORDERSIZE NOOPENFLG))]
```

Or in Lyric, use the Common Lisp construct:

```
←(DEFUN MYWINDOWFONTFN
    (REGION TITLE BORDERSIZE NOOPENFLG FONTDESCRIPTOR)
    (LET ((OLDFONT
            (IL:DSPFONT NIL IL:|WindowTitleDisplayStream|)))
        (UNWIND-PROTECT
        (PROGN
            (IL:DSPFONT FONTDESCRIPTOR
                IL:|WindowTitleDisplayStream|)
            (IL:CREATEW REGION
                TITLE BORDERSIZE NOOPENFLG))
            (IL:DSPFONT OLDFONT
                IL:|WindowTitleDisplayStream|))))
```

Reference IRM vol. III, page 28.14, 27.11