

#### Bulletin 8 December 4, 1987

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:

- Porting Common Lisp Files to Lyric
- Compiling Non-Xerox Common Lisp Files in Lyric
- XCL:EXEC Window Property Bug
- XCL:ADD-EXEC Window Property Bug
- Restoring Multiply Advised Functions
- Interpreted and Compiled Macros

Terminology used in this HOTLINE! bulletin:

- UG Users' Guide
- AR Action Request, a Xerox problem tracking number (e.g. AR 8321)
- IRM Interlisp Reference Manual

## Porting Common Lisp Files to Lyric

Helease	Lyric
Keywords	Packages, Files, Portability
Question	How do I port a pure Common Lisp source file into the Xerox environment?
Answer	Loading a textual-pure Common Lisp source file from a non- Xerox environment is supported in Lyric.
	However, since the File Manager did not produce the source file,

ot produce the source file, the user is required to indicate that the file consists of plain Common Lisp source code. The user must also clearly specify the Reader-Environment for interpreting the symbols in the file.

Example A user wants to port a pure Common Lisp source file into the XCL environment, make an edit, and compile the modified code into a DFASL file.

> In the following example, a plain Common Lisp source file F-TO-C.LSP, contains the following text:

Edit Windowfor; {DSK}<LISPFILES>WORK>F-TO-C.LSP;1

```
;;A constant
(DEFCONSTANT *CONVERSION-CONSTANT* (/ 5.0 9))
;;A function to convert Farhenheit to Centigrade
(DEFUN F-TO-C (DEGREES-F)
   (FORMAT T "~3,1F degrees F = ~3,1F degrees C"
           DEGREES-F
           (* *CONVERSION-CONSTANT* (- DEGREES-F 32))))
;;A top-level form
(PRINC "Done Loading File")
```

The commands below are typed in the XCL Exec.

1. Verify that the source file begins with a semicolon. The semicolon signals the LOAD function to interpret the file as a plain Common Lisp source file.

If a semicolon isn't present, it should be added with TEdit, and saved (PUT) with the "Plain-Text" sub-menu option. The semicolon may be preceded by an arbitrary amount of white space. In this example, the text ";;;A constant" provides the semicolon.

2. Load the source file with the following command:

(LOAD 'F-TO-C.LSP : PACKAGE (FIND-PACKAGE "XCL-USER"))

If an argument for the :PACKAGE keyword is not specified, plain Common Lisp source files will be loaded into the USER package. The specified package should contain or import all Common Lisp symbols.

All forms will be executed when the file is loaded.

- 3. After the LOAD has completed, the function and variable will be defined in the XCL-USER package.
  - The function may be executed in interpreted form.
  - The function and variable may be edited with SEdit.
  - IL:FILES? will list both the new function and the variable.
- 4. Call SEdit on the function F-TO-C.

In this example, the user wishes to print the date with the output of the function. Interlisp functions may be conveniently used since all symbols in the IL package are external.

\* SEdit F-TO-C Package: XCL-USER (DEFUN F-TO-C (DEGREES-F) (IL:PRINT (IL:DATE)) (FORMAT T "~3,1F degrees F = ~3,1F degrees C" DEGREES-F (\* \*CONVERSION-CONSTANT\* (- DEGREES-F 32))))

5. Check the File Manager's default Reader-Environment to verify that it matches the appropriate environment for the file. The Reader-Environment consists of a package, a read table and a read base for numbers.

To avoid printing unwanted package prefixes on symbols, the user should set the Reader-Environment package to the package specified in the above LOAD command. In this example, the user would set the read table and package to correspond to the XCL environment.

Evaluating the variable IL:\*DEFAULT-MAKEFILE-ENVIRON-MENT\* will return the default Reader-Environment for files created by the MAKEFILE function. If this isn't the desired environment, it can be changed.

For example, to globally set the MAKEFILE Reader-Environment:

```
(SETQ IL:*DEFAULT-MAKEFILE-ENVIRONMENT*
    '(:READTABLE "XCL" :PACKAGE "XCL-USER" :BASE 10))
```

Or, to locally set the Reader-Environment for the file F-TO-C.LSP:

Note: Changing IL:\*DEFAULT-MAKEFILE-ENVIRONMENT\* will not affect files which already exist. If the user wants to change the Reader-Environment for an existing file, the IL:PUTPROP form must be used.

 Call the function IL:FILES? and assign a file name for the function F-TO-C and the variable \*CONVERSION-CONSTANT\*.

If the IL:MAKEFILE-ENVIRONMENT property has been used to establish the Reader-Environment, then the property list should also be assigned to the same file.

In this example, assume the original source file name is given as the new file name. The user is still prompted to create a "new" file since the File Manager hasn't noticed F-TO-C.LSP. However, the user is not required to use the original source file name. As long as the same file name is given to IL:PUTPROP (if used), IL:FILES? and IL:MAKEFILE (below), the file will be made properly.

7. Call SEdit on the variable IL:F-TO-C.LSPCOMS to add the top-level form.

Since top-level forms in the source file are not collected by IL:FILES?, they should be added using P statements. In SEdit, the expression (P (CL:PRINC "Done Loading File")) is added to the variable IL:F-TO-C.LSPCOMS.

8. Call the function IL:MAKEFILE on the file F-TO-C.LSP to make and compile the file:

(IL:MAKEFILE 'F-TO-C.LSP 'IL:C)

In this example, (since the user has not set the IL:FILETYPE property on F-TO-C.LSP), IL:C points to the compiler specified by the value of the variable IL:\*DEFAULT-CLEANUP-COMPILER\*. Its value should be set to CL:COMPILE-FILE to generate DFASL files.

A new version of F-TO-C.LSP will be created, as well as the object file F-TO-C.DFASL. The original source file comments are not preserved in the new source file generated by IL:MAKEFILE.

9. Load the compiled code. The compiled function may be executed:

(LOAD 'F-TO-C.DFASL)

References Xerox Lisp Release Notes - Integration of Languages: File Package, pp. 23-35. Xerox Lisp Release Notes - Integration of Languages: Compiler, pp. 35-36. Xerox Common Lisp Implementation Notes, Lyric Release, File System Interface, pp. 61. Xerox Common Lisp Implementation Notes, Lyric Release, The Compiler, pp. 89-97. Hotline! No.1, pp. 1-3, 1-4.

## **Compiling Non-Xerox Common Lisp Files in Lyric**

Release	Lyric
Keywords	Packages, Files, Portability
Question	How do I port a pure Common Lisp source file into the Xerox environment without loading the source file?
Discussion	Textual-pure Common Lisp files from non-Xerox environments can be compiled directly (i.e., without loading the source file) under Lyric. However, some care must be taken to ensure that the file definitions will be interned in the desired package when the compiled file is loaded.
Example	The user wants to bring a pure Common Lisp source file into the XCL environment and compile it into a DFASL file. The user doesn't plan on making changes to the source code, and is only interested in generating the compiled file. Assume the same source file as in the previous question (Bulletin 8.1). The commands below are typed in at the XCL Exec.
	1. Verify that the source file begins with a semicolon. Insert a semicolon if it's missing.
	2. Specify the target package for the file definitions with the IN- PACKAGE function.
	In this example, the user should insert the following statement after the initial comment in the source file:
	(IN-PACKAGE "XCL-USER")
	If no package is specified, it will default to the USER package. Then the function and variable definitions will be interned in the USER package when the compiled file is loaded. Likewise, all top-level expressions would be evaluated with *PACKAGE* bound to USER.
	Since the File Manager is being by-passed, the user cannot establish the package with the IL:MAKEFILE-ENVIRONMENT file property or with the variable IL:*DEFAULT-MAKEFILE- ENVIRONMENT*.

3. Compile the file:

٠.

(COMPILE-FILE 'F-TO-C.LSP)

4. Load the compiled code:

(LOAD 'F-TO-C.DFASL)

When loading DFASL files, the :PACKAGE argument cannot be used to override the file's package environment. It is either specified by an IN-PACKAGE function in the source file, or it defaults to the USER package. References Xerox Lisp Release Notes - Integration of Languages: File Package, pp. 23-35. Xerox Lisp Release Notes - Integration of Languages: Compiler, pp. 35-36. Xerox Common Lisp Implementation Notes, Lyric Release, File System Interface, pp. 61. Xerox Common Lisp Implementation Notes, Lyric Release, The Compiler, pp. 89-97.

Common Lisp: The Language, pg. 183.

# EXEC Window Property Bug

	· ·
Release	Lyric
Keywords	EXEC, WINDOW property, PROCESSES
Problem	XCL:EXEC does not use :WINDOW argument
Symptom	If you did something in the IL Exec like:
	(SETQ MYW (CREATEW)) (XCL:EXEC :WINDOW MYW :PROMPT "MYEXEC" :COMMAND-TABLES *EXEC-COMMAND-TABLE* :TITLE "My Test Exec")
	You will see the Title appear on the window you created with CREATEW, but the EXEC process will not be added to that window. Instead, the IL Exec where you called these functions will have the command-table information given to it. In this example, the command prompt changed from 2/40 to 2/41MYEXEC.
	More simply stated: (EXEC :WINDOW (IL:CREATEW)) will prompt you to sweep out a window, but it will not be given the Exec process
Workaround	None
References	AR 9310 Lyric Release Notes, Appendix A "The Exec," p. A-18 IRM, Vol. II, section 23.1, pp. 23.2-23.3

### ADD-EXEC Window Property Bug

Release Lyric

- Keywords ADD-EXEC, WINDOW property
  - Problem XCL:ADD-EXEC does not set WINDOW property of the new exec process. (See related *Hotline!* Bulletin 8.3 on EXEC Window Property bug.)
  - **Example** In Koto and earlier releases, \TopLevelTtyWindow was bound to the executive window. In Lyric it is bound to the window of the very first XCL exec that comes up when Lisp is booted. If a user has multiple exec windows open and desires to programmatically use TTYDISPLAYSTREAM to printout to the various exec windows, how do they go about accessing the windows?
- Symptom The function XCL:ADD-EXEC adds new execs but returns a process instead of the window. Using PROCESSPROP on this process to access the WINDOW property always returns NIL.
- Workaround It is necessary to define a new ADD-EXEC function to work around this problem. Define IL:ADD-EXEC as follows in an Interlisp Exec, in the Interlisp package. When entering this code, note the back-guote forms used after ADD.PROCESS.

#### SEdit ADD-EXEC Package: INTERLISP

```
(CL:DEFUN ADD-EXEC
   (&KEY (XCL::PROFILE XCL:*PROFILE*) XCL::REGION XCL::TTY
         (EXEC 'EXEC) XCL::ID &ALLOW-OTHER-KEYS)
   (LET* ((XCL::WINDOW (XCL::SETUP-EXEC-WINDOW
                               (CREATEW XCL::REGION "Exec")))
          (XCL::HANDLE
             (ADD.PROCESS
                `(PROGN (TTYDISPLAYSTREAM ',XCL::WINDOW)
                        (PROCESSPROP (THIS.PROCESS) 'WINDOW
                                      ',XCL::WINDOW)
                        ,(CASE EXEC
                                (EXEC '(EXEC :TOP-LEVEL-P T
                                              : PROFILE
                                              ',XCL::PROFILE :ID
                                              ',XCL::ID))
                                (T '(XCL::ENTER-EXEC-FUNCTION
                                            ',EXEC
                                            ',XCL::PROFILE
                                            ',XCL::ID))))
                'NAME 'EXEC 'RESTARTABLE T)))
     (AND XCL::TTY (TTY.PROCESS XCL::HANDLE))
     XCL::HANDLE))
  Reference AR 9311, Lyric Release Notes, Appendix A "The Exec", p. A-17
```

### **Restoring Multiply Advised Functions**

Release Lyric

Keywords ADVISE, UNADVISE

Problem Can't unadvise a multiply-advised function.

**Example** When more than one piece of advice is supplied to the same function (by calling ADVISE two or more times) it is not possible to restore the function to its unadvised state by calling UNADVISE, for example:

(ADVISE 'FOO 'BEFORE '(PRINT "Before"))
(ADVISE 'FOO 'AFTER '(PRINT "After"))
(UNADVISE 'FOO)
FOO is not advised

**Workaround** Interlisp functions (defined via DEFINEQ) can be restored with UNSAVEDEF; e.g., (UNSAVEDEF 'FOO)

Common Lisp functions (defined via DEFUN) can only be restored by explicitly reloading the source definition. For example, if MAKEFILE was used to save the definiton in file FOOFILE, then use (LOADFNS 'FOO 'FOOFILE).

Compiled functions can only be restored by explicitly reloading the compiled definition. For example, if the function was compiled with TCOMPL, then use (LOADFNS 'FOO 'FOOFILE.LCOM). Otherwise, reload the source definition and recompile the function.

Reference AR 8687

### Interpreted and Compiled Common Lisp Macros

- Release Lyric
- Keywords Macros
- **Question** Do Common Lisp macros get expanded every time they are called or just the first time they are called?
- **Background** A module which contains a loop that makes several calls to a macro during the loop was converted from Koto Interlisp to a Common Lisp iteration and macro construct. When the loop is run interpreted in Koto it takes only about 15 seconds to complete. When it was converted to XCL the loop takes 1.5 hours to run interpreted. It appears that the macro in the loop is getting expanded during each pass through the loop.
  - Answer Common Lisp macros get expanded using MACROEXPAND-1 everytime they are called when you run interpreted code. Interlisp macros get expanded only during the first call to the macro. When working with Common Lisp macros you should compile the macro in order to avoid performance problems such as this during iteration or other similar situations.