File created: 14-Jun-90 17:33:55 **{PELE:MV:ENVOS}<LISPCORE>SOURCES>CMLPACKAGE.;3**

previous date: 16-May-90 14:12:37 {PELE:MV:ENVOS}<LISPCORE>SOURCES>CMLPACKAGE.;2

Read Table: XCL

Package: XEROX-COMMON-LISP

Format: XCCS


```
(IL:RPAQQ IL:CMLPACKAGECOMS
          (
```

;;; This is the second part of the package system, the first is in LLPACKAGE, which is loaded during the init

```
          (IL:SETFS SYMBOL-PACKAGE)
          (IL:FUNCTIONS IL:DWIM-SYMBOL-PACKAGE ESCAPE-COLONS-PROCEED MAKE-EXTERNAL-PROCEED
                 MAKE-INTERNAL-PROCEED UGLY-SYMBOL-PROCEED)
          (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOCOPY (IL:ADDVARS (IL:DWIMUSERFORMS (IL:DWIM-SYMBOL-PACKAGE))))
```

;; User friendly symbol error resolving functions

```
          (IL:STRUCTURES READ-CONFLICT MISSING-EXTERNAL-SYMBOL MISSING-PACKAGE)
          (IL:VARIABLES *PREFERRED-READING-SYMBOLS*)
          (IL:FUNCTIONS IL:RESOLVE-READER-CONFLICT IL:RESOLVE-MISSING-EXTERNAL-SYMBOL
                 IL:RESOLVE-MISSING-PACKAGE)
          (IL:STRUCTURES PACKAGE-ERROR SYMBOL-CONFLICT USE-CONFLICT EXPORT-CONFLICT EXPORT-MISSING
                 IMPORT-CONFLICT UNINTERN-CONFLICT)
          (IL:FUNCTIONS IL:RESOLVE-USE-PACKAGE-CONFLICT IL:RESOLVE-EXPORT-CONFLICT IL:RESOLVE-EXPORT-MISSING
                 IL:RESOLVE-IMPORT-CONFLICT IL:RESOLVE-UNINTERN-CONFLICT)
          (IL:STRUCTURES SYMBOL-COLON-ERROR)
          (IL:FUNCTIONS IL:\\INVALID.SYMBOL                    ; Also defined (w/o the error condition or proceed case) in
                                                               ; LLREAD.
                 )
```

;; Symbol inspector

```
          (IL:FUNCTIONS IL:SYMBOL-INSPECT-FETCHFN IL:SYMBOL-INSPECT-STOREFN)
          (IL:P (LET ((IL:FORM '((IL:FUNCTION SYMBOLP)
                                 (IL:NAME IL:VALUE IL:PLIST PACKAGE)
                                 IL:SYMBOL-INSPECT-FETCHFN IL:SYMBOL-INSPECT-STOREFN NIL NIL NIL "Symbol
                                 inspector")))
                      (COND ((NOT (IL:MEMBER IL:FORM IL:INSPECTMACROS))
                             (IL:|push| IL:INSPECTMACROS IL:FORM)))))
```

;; Package inspector

```
          (IL:FUNCTIONS IL:PACKAGE-INSPECT-FETCHFN IL:PACKAGE-INSPECT-STOREFN)
          (IL:P (LET ((IL:FORM '((IL:FUNCTION PACKAGEP)
                                 (IL:NAME IL:NICKNAMES IL:USE-LIST IL:INTERNAL-SYMBOLS IL:EXTERNAL-SYMBOLS
                                        IL:SHADOWING-SYMBOLS)
                                 IL:PACKAGE-INSPECT-FETCHFN IL:PACKAGE-INSPECT-STOREFN NIL NIL NIL "Package
                                 inspector")))
                      (COND ((NOT (IL:MEMBER IL:FORM IL:INSPECTMACROS))
                             (IL:|push| IL:INSPECTMACROS IL:FORM)))))
```

;; Package-hashtable inspector

```
          (IL:FUNCTIONS IL:PACKAGE-HASHTABLE-INSPECT-FETCHFN IL:PACKAGE-HASHTABLE-INSPECT-STOREFN)
          (IL:P (LET ((IL:FORM '((IL:FUNCTION CL::PACKAGE-HASHTABLE-P)
                                 (IL:SIZE IL:FREE IL:DELETED IL:CONTENTS)
                                 IL:PACKAGE-HASHTABLE-INSPECT-FETCHFN IL:PACKAGE-HASHTABLE-INSPECT-STOREFN)))
                      (COND ((NOT (IL:MEMBER IL:FORM IL:INSPECTMACROS))
                             (IL:|push| IL:INSPECTMACROS IL:FORM)))))
```

;; Package's Prefix accessor and setfs (Edited by TT 14-June-90 for AR#11112)

```
          (IL:FUNCTIONS PACKAGE-PREFIX SETF-PACKAGE-PREFIX)
          (IL:SETFS PACKAGE-PREFIX)
          (IL:PROP (IL:FILETYPE IL:MAKEFILE-ENVIRONMENT)
                 IL:CMLPACKAGE)
          (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS (IL:ADDVARS (IL:NLAMA)
                                                                                                (IL:NLAML)
                                                                                                (IL:LAMA)))))
```

;;; This is the second part of the package system, the first is in LLPACKAGE, which is loaded during the init

```
(DEFSETF SYMBOL-PACKAGE IL:SETF-SYMBOL-PACKAGE)

(DEFUN IL:DWIM-SYMBOL-PACKAGE ()
   (DECLARE (SPECIAL IL:FAULTX IL:FAULTAPPLYFLG))
   ;; This is placed on DWIMUSERFORMS to attempt corrections where the typed symbol is in the wrong package.
   (LET ((IL:SYM (OR (CAR (IL:LISTP IL:FAULTX))
                     IL:FAULTX))
         IL:OTHERS)
        (COND
           ((AND (IL:LITATOM IL:SYM)
```

```
                (CDR (IL:SETQ IL:OTHERS (FIND-ALL-SYMBOLS (SYMBOL-NAME IL:SYM))))
                (IL:SETQ IL:OTHERS (IL:|for| IL:X IL:|in| IL:OTHERS IL:|collect| IL:X
                                           IL:|when| (AND (IL:NEQ IL:X IL:SYM)
                                                          (NOT (KEYWORDP IL:X))
                                                          (IL:|if| (AND (IL:LITATOM IL:FAULTX)
                                                                        (NOT IL:FAULTAPPLYFLG))
                                                              IL:|then|   ; Error is uba
                                                                    (BOUNDP IL:X)
                                                               IL:|else| (FBOUNDP IL:X))))))
          (IL:|for| IL:CHOICE IL:|in| IL:OTHERS IL:|when| (IL:FIXSPELL1 IL:SYM IL:CHOICE NIL T (AND (CDR IL:OTHERS)
                                                                                                     'IL:MUSTAPPROVE))
             IL:|do|
                  ;; Normally there is only one choice, and we offer it.  If there is more than one choice, probably should do something like a
                  ;; menu.  This is quick and dirty--ask user for each in turn and require approval so that it doesn't choose the first
                  ;; automatically.
                  (RETURN (IL:|if| (IL:LISTP IL:FAULTX)
                                  IL:|then|                    ; SYM = (CAR FAULTX)
                                       (IL:/RPLACA IL:FAULTX IL:CHOICE)
                                  IL:|else| IL:CHOICE)))))))
```

```
(DEFINE-PROCEED-FUNCTION ESCAPE-COLONS-PROCEED :CONDITION SYMBOL-COLON-ERROR
   :REPORT "Treat the extra colon(s) as if they were escaped")

(DEFINE-PROCEED-FUNCTION MAKE-EXTERNAL-PROCEED :CONDITION MISSING-EXTERNAL-SYMBOL
   :REPORT "Return a new external symbol by that name"
   (CONDITION *CURRENT-CONDITION*))

(DEFINE-PROCEED-FUNCTION MAKE-INTERNAL-PROCEED :CONDITION MISSING-EXTERNAL-SYMBOL
   :REPORT "Return a new internal symbol by that name")

(DEFINE-PROCEED-FUNCTION UGLY-SYMBOL-PROCEED :CONDITION MISSING-PACKAGE)

(IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOCOPY

(IL:ADDTOVAR IL:DWIMUSERFORMS (IL:DWIM-SYMBOL-PACKAGE))
)

;; User friendly symbol error resolving functions

(DEFINE-CONDITION READ-CONFLICT (READ-ERROR)
   (NAME PACKAGES)
   (:REPORT (LAMBDA (CONDITION STREAM)
                  '(FORMAT STREAM "Symbols named ~a exist in packages:~{~a ~}" (READ-CONFLICT-NAME CONDITION)
                        (MAPCAR #'PACKAGE-NAME (READ-CONFLICT-PACKAGES CONDITION)))
                  (FORMAT STREAM "Symbols named ~A exists in packages:" (READ-CONFLICT-NAME CONDITION))
                  (DOLIST (PKG (READ-CONFLICT-PACKAGES CONDITION))
                      (PRINC " " STREAM)
                      (PRINC (PACKAGE-NAME PKG)
                             STREAM)))))

(DEFINE-CONDITION MISSING-EXTERNAL-SYMBOL (READ-ERROR)
   (NAME PACKAGE)
   (:REPORT (LAMBDA (CONDITION STREAM)
                  (FORMAT STREAM "External symbol ~a not found in package ~a" (MISSING-EXTERNAL-SYMBOL-NAME
                                                                                CONDITION)
                        (PACKAGE-NAME (MISSING-EXTERNAL-SYMBOL-PACKAGE CONDITION))))))

(DEFINE-CONDITION MISSING-PACKAGE (READ-ERROR)
   (PACKAGE-NAME SYMBOL-NAME EXTERNAL)
   (:REPORT (LAMBDA (CONDITION STREAM)
                  (FORMAT STREAM "Can't find package ~a to look up symbol ~a" (MISSING-PACKAGE-PACKAGE-NAME
                                                                               CONDITION)
                        (MISSING-PACKAGE-SYMBOL-NAME CONDITION)))))

(DEFVAR *PREFERRED-READING-SYMBOLS*
   '(IL:APPEND IL:APPLY IL:APROPOS IL:ARRAY IL:ARRAYP IL:ASSOC IL:ATAN IL:ATOM IL:BLOCK IL:BREAK IL:CHAR
          IL:CHARACTER IL:CLOSE IL:COMMON IL:COMPILE IL:COMPILE-FILE IL:COS IL:COUNT IL:DEFSTRUCT IL:DELETE
          IL:DESCRIBE IL:DIRECTORY IL:DO IL:DOCUMENTATION IL:ELT IL:EQUAL IL:ERROR IL:EVAL IL:EVERY IL:EXP
          IL:EXPT IL:FILL-POINTER IL:FIND IL:FIRST IL:FLOATP IL:FLOOR IL:FORMAT IL:FUNCTION IL:GCD IL:GENSYM
          IL:GETHASH IL:IF IL:INTERSECTION IL:KEYWORD IL:LABELS IL:LAMBDA IL:LDIFF IL:LENGTH IL:LISTP IL:LOAD
          IL:LOCALLY IL:LOG IL:LOOP IL:MAP IL:MAPC IL:MAPCAR IL:MAPCON IL:MAPHASH IL:MAPLIST IL:MEMBER IL:MERGE
          IL:MISMATCH IL:MOD IL:NAMESTRING IL:NOTANY IL:NOTEVERY IL:NTH IL:NUMBER IL:NUMBERP IL:NUMERATOR
          IL:POP IL:POSITION IL:PRIN1 IL:PRINT IL:PUSH IL:PUSHNEW IL:RATIONAL IL:READ IL:READTABLE IL:REMOVE
          IL:REPLACE IL:REST IL:REVERSE IL:SEARCH IL:SECOND IL:SETQ IL:SIGNED-BYTE IL:SIMPLE-STRING IL:SIN
          IL:SOME IL:SORT IL:SQRT IL:STRINGP IL:STRUCTURE IL:SUBLIS IL:SUBSEQ IL:SUBST IL:SYMBOL IL:TAN
          IL:TERPRI TRACE IL:UNION IL:UNLESS IL:VALUES IL:VARIABLE IL:VECTOR IL:WHEN IL:ZEROP IL:* IL:***)
   "List of symbols whose lookup is preferred by the litatom to symbol converter.  Initially it contains a list
   of symbols which are conflicting but are always qualified in old sources.")
```

```
(DEFUN IL:RESOLVE-READER-CONFLICT (IL:ILSYM IL:CLSYM IL:CLSYMWHERE)
   "Reader finds unqualified symbol that exists in both InterLisp and Lisp.  Checks *PREFERRED-READING-SYMBOLS*
   list against names."
   (DECLARE (SPECIAL *PREFERRED-READING-SYMBOLS*))

   ;; CAUTION: Do not attempt to move the namestring check from \NEW.READ.SYMBOL into this function as RESOLVE-READER-CONFLICT has a
   ;; dummy definition in the INIT.  Also, namestring resolutions must be made during the time that packages are turned off in the beginning of the INIT.

   (COND
      ((NOT (EQ IL:CLSYMWHERE :EXTERNAL))                    ; Will not resolve internal (therefore private) symbols from LISP
       IL:ILSYM)
      (T (LET ((IL:ILPREFERRED (MEMBER IL:ILSYM *PREFERRED-READING-SYMBOLS* :TEST 'EQ))
               (IL:CLPREFERRED (MEMBER IL:CLSYM *PREFERRED-READING-SYMBOLS* :TEST 'EQ)))
              (COND
                 ((AND IL:ILPREFERRED (NOT IL:CLPREFERRED))
                  IL:ILSYM)
                 ((AND IL:CLPREFERRED (NOT IL:ILPREFERRED))
                  IL:CLSYM)
                 (T                                          ; Raise the signal
                    (RESTART-CASE (ERROR 'READ-CONFLICT :NAME (SYMBOL-NAME IL:ILSYM)
                                         :PACKAGES
                                         (LIST (FIND-PACKAGE "LISP")
                                               (FIND-PACKAGE "INTERLISP")))
                          (PREFER-CLSYM-PROCEED NIL :CONDITION READ-CONFLICT :REPORT
                                 (LAMBDA (STREAM)
                                         (FORMAT STREAM "Return the LISP symbol ~A; make it preferred" IL:CLSYM)
                                         )
                                 IL:CLSYM)
                          (PREFER-ILSYM-PROCEED NIL :CONDITION READ-CONFLICT :REPORT
                                 (LAMBDA (STREAM)
                                         (FORMAT STREAM "Return the INTERLISP symbol ~A; make it preferred"
                                                 IL:ILSYM))
                                 (SETQ *PREFERRED-READING-SYMBOLS* (REMOVE IL:CLSYM *PREFERRED-READING-SYMBOLS*
                                                                           :TEST #'EQ))
                                 (PUSH IL:ILSYM *PREFERRED-READING-SYMBOLS*)
                                 IL:ILSYM)
                          (RETURN-ILSYM-PROCEED NIL :CONDITION READ-CONFLICT :REPORT
                                 (LAMBDA (STREAM)
                                         (FORMAT STREAM "Just return the INTERLISP symbol ~A" IL:ILSYM))
                                 IL:ILSYM))))))))

(DEFUN IL:RESOLVE-MISSING-EXTERNAL-SYMBOL (IL:NAME PACKAGE)
   "Handle missing external symbols in a package during read."
   (LET ((IL:MY-CONDITION (MAKE-CONDITION 'MISSING-EXTERNAL-SYMBOL :NAME IL:NAME :PACKAGE PACKAGE)))
        (FLET ((IL:FILTER NIL (EQ *CURRENT-CONDITION* IL:MY-CONDITION)))
              (RESTART-CASE (ERROR IL:MY-CONDITION)
                     (MAKE-EXTERNAL-PROCEED NIL :FILTER IL:FILTER :REPORT (LAMBDA (STREAM)
                                                                                  (FORMAT STREAM "Return a new
                                                                                         external symbol in
                                                                                         package ~A named ~A"
                                                                                         (PACKAGE-NAME PACKAGE)
                                                                                         IL:NAME))
                            (LET ((IL:SYMBOL (INTERN IL:NAME PACKAGE)))
                                 (EXPORT IL:SYMBOL PACKAGE)
                                 IL:SYMBOL))
                     (MAKE-INTERNAL-PROCEED NIL :FILTER IL:FILTER :REPORT (LAMBDA (STREAM)
                                                                                  (FORMAT STREAM "Return a new
                                                                                         internal symbol in
                                                                                         package ~A named ~A"
                                                                                         (PACKAGE-NAME PACKAGE)
                                                                                         IL:NAME))
                            (INTERN IL:NAME PACKAGE))))))

(DEFUN IL:RESOLVE-MISSING-PACKAGE (PACKAGE-NAME SYMBOL-NAME EXTERNALP)
   (LET ((IL:MY-CONDITION (MAKE-CONDITION 'MISSING-PACKAGE :PACKAGE-NAME PACKAGE-NAME :SYMBOL-NAME SYMBOL-NAME
                                          :EXTERNAL EXTERNALP)))
        (FLET ((IL:FILTER NIL (EQ *CURRENT-CONDITION* IL:MY-CONDITION)))
              (RESTART-CASE (ERROR IL:MY-CONDITION)
                     (NEW-PACKAGE-PROCEED NIL :FILTER IL:FILTER :REPORT (LAMBDA (STREAM)
                                                                                (FORMAT STREAM "Return new symbol
                                                                                       named ~A made in new
                                                                                       package ~A" SYMBOL-NAME
                                                                                       PACKAGE-NAME))
                            (LET* ((PACKAGE (MAKE-PACKAGE (MISSING-PACKAGE-PACKAGE-NAME IL:MY-CONDITION)))
                                   (SYMBOL (INTERN (MISSING-PACKAGE-SYMBOL-NAME IL:MY-CONDITION)
                                                   PACKAGE)))
                                  (WHEN (MISSING-PACKAGE-EXTERNAL IL:MY-CONDITION)
                                        (EXPORT SYMBOL PACKAGE))
                                  SYMBOL))
                     (UGLY-SYMBOL-PROCEED NIL :FILTER IL:FILTER :REPORT (LAMBDA (STREAM)
                                                                                (FORMAT STREAM "Return new ugly
                                                                                       symbol |~a~a~a| made in
                                                                                       current package ~a"
                                                                                       PACKAGE-NAME
```

```
                                                                                    (IF EXTERNALP
                                                                                        ":"
                                                                                        "::")
                                                                                    SYMBOL-NAME
                                                                                    (PACKAGE-NAME *PACKAGE*)
                                                                                    ))
                                :INTERACTIVE
                                (LAMBDA NIL (LIST *PACKAGE*))
                                (INTERN (IL:CONCAT (MISSING-PACKAGE-PACKAGE-NAME IL:MY-CONDITION)
                                                   (IF (MISSING-PACKAGE-EXTERNAL IL:MY-CONDITION)
                                                       ":"
                                                       "::")
                                                   (MISSING-PACKAGE-SYMBOL-NAME IL:MY-CONDITION))
                                        *PACKAGE*))))))

(DEFINE-CONDITION PACKAGE-ERROR (ERROR)
   (PACKAGE))

(DEFINE-CONDITION SYMBOL-CONFLICT (PACKAGE-ERROR)
   (SYMBOLS))

(DEFINE-CONDITION USE-CONFLICT (SYMBOL-CONFLICT)
   (USED-PACKAGE)
   (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
                    (FORMAT T "Package ~a using ~a results in name conflicts for symbols:~%~{~s ~}"
                           (PACKAGE-NAME (USE-CONFLICT-PACKAGE CONDITION))
                           (PACKAGE-NAME (USE-CONFLICT-USED-PACKAGE CONDITION))
                           (USE-CONFLICT-SYMBOLS CONDITION)))))

(DEFINE-CONDITION EXPORT-CONFLICT (SYMBOL-CONFLICT)
   (EXPORTED-SYMBOLS PACKAGES)
   (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
                    (FORMAT T "Exporting these symbols from the ~a package:~%~{~s ~}~%results in name conflicts
                           with package(s):~%~{~a ~}~%" (PACKAGE-NAME (EXPORT-CONFLICT-PACKAGE CONDITION))
                           (EXPORT-CONFLICT-SYMBOLS CONDITION)
                           (MAPCAR #'PACKAGE-NAME (EXPORT-CONFLICT-PACKAGES CONDITION))))))

(DEFINE-CONDITION EXPORT-MISSING (PACKAGE-ERROR)
   (SYMBOLS)
   (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
                    (FORMAT T "These symbols aren't in package ~a; can't export them from it:~%~{~s ~}"
                           (PACKAGE-NAME (EXPORT-MISSING-PACKAGE CONDITION))
                           (EXPORT-MISSING-SYMBOLS CONDITION)))))

(DEFINE-CONDITION IMPORT-CONFLICT (SYMBOL-CONFLICT)
   NIL
   (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
                    (FORMAT T "Importing these symbols into package ~a causes a name conflict:~%~{~s ~}"
                           (PACKAGE-NAME (IMPORT-CONFLICT-PACKAGE CONDITION))
                           (IMPORT-CONFLICT-SYMBOLS CONDITION)))))

(DEFINE-CONDITION UNINTERN-CONFLICT (SYMBOL-CONFLICT)
   (SYMBOL)
   (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
                    (FORMAT T "Uninterning symbol ~s causes a name conflict among these symbols:~%~{~s ~}"
                           (UNINTERN-CONFLICT-SYMBOL CONDITION)
                           (UNINTERN-CONFLICT-SYMBOLS CONDITION)))))

(DEFUN IL:RESOLVE-USE-PACKAGE-CONFLICT (USED-PACKAGE SYMBOLS PACKAGE)
   "Handle a conflict from use-package."
   (SETQ SYMBOLS (SORT SYMBOLS 'STRING<))
   (LET ((MY-CONDITION (MAKE-CONDITION 'USE-CONFLICT :PACKAGE PACKAGE :SYMBOLS SYMBOLS :USED-PACKAGE
                              USED-PACKAGE)))
        (FLET ((FILTER NIL (EQ *CURRENT-CONDITION* MY-CONDITION)))
              (RESTART-CASE (ERROR MY-CONDITION)
                     (SHADOW-USE-CONFLICTS-PROCEED NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                                                (FORMAT STREAM "Shadow
                                                                                       conflicting symbols
                                                                                       from ~A in ~A"
                                                                                       (PACKAGE-NAME
                                                                                              USED-PACKAGE)
                                                                                       (PACKAGE-NAME
                                                                                              PACKAGE)))
                            (DOLIST (SYMBOL SYMBOLS)
                                (SHADOW SYMBOL PACKAGE)))
                     (UNINTERN-USER-PROCEED NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                                         (FORMAT STREAM "Unintern all
                                                                                conflicting symbols from ~A
                                                                                (DANGEROUS)" (PACKAGE-NAME
```

```
                                                                                                    PACKAGE)))
                                (DOLIST (SYMBOL SYMBOLS)
                                    (IL:MOBY-UNINTERN SYMBOL PACKAGE)))
                            (UNINTERN-USEE-PROCEED NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                                                (FORMAT STREAM "Unintern all
                                                                                       conflicting symbols from ~A
                                                                                       (VERY DANGEROUS)"
                                                                                       (PACKAGE-NAME USED-PACKAGE)
                                                                                       ))
                                (DOLIST (SYMBOL SYMBOLS)
                                    (IL:MOBY-UNINTERN (FIND-SYMBOL (SYMBOL-NAME SYMBOL)
                                                                   USED-PACKAGE)
                                            USED-PACKAGE)))
                            (ABORT NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                                (FORMAT STREAM "Abort making package ~a use ~a"
                                                                       (PACKAGE-NAME PACKAGE)
                                                                       (PACKAGE-NAME USED-PACKAGE)))
                                (IL:RETFROM 'USE-PACKAGE NIL))))))

(DEFUN IL:RESOLVE-EXPORT-CONFLICT (PACKAGE SYMBOLS PACKAGES EXPORTED-SYMBOLS)
   "Handle a conflict raised by export."
   (IL:SETQ SYMBOLS (SORT SYMBOLS 'STRING<))
   (SETQ PACKAGES (SORT PACKAGES #'(LAMBDA (A B)
                                     (STRING< (PACKAGE-NAME A)
                                              (PACKAGE-NAME B)))))
   (LET ((MY-CONDITION (MAKE-CONDITION 'EXPORT-CONFLICT :PACKAGE PACKAGE :SYMBOLS SYMBOLS :EXPORTED-SYMBOLS
                              EXPORTED-SYMBOLS :PACKAGES PACKAGES)))
        (FLET ((FILTER NIL (EQ *CURRENT-CONDITION* MY-CONDITION)))
              (RESTART-CASE (ERROR MY-CONDITION)
                      (UNINTERN-PROCEED NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                                     (FORMAT STREAM "Unintern all
                                                                            conflicting symbols in
                                                                            package~P~{ ~a~} (DANGEROUS)"
                                                                            (IF (NULL (REST PACKAGES))
                                                                                0
                                                                                1)
                                                                            (MAPCAR #'PACKAGE-NAME PACKAGES)
                                                                            ))
                              (DOLIST (PACKAGE PACKAGES EXPORTED-SYMBOLS)
                                  (DOLIST (SYMBOL SYMBOLS)
                                      (IL:MOBY-UNINTERN (FIND-SYMBOL (SYMBOL-NAME SYMBOL)
                                                                     PACKAGE)
                                              PACKAGE))))
                      (ABORT NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                          (FORMAT STREAM "Abort exporting the symbols from
                                                                 package ~a" (PACKAGE-NAME PACKAGE)))
                              (IL:RETFROM 'EXPORT NIL))))))

(DEFUN IL:RESOLVE-EXPORT-MISSING (PACKAGE SYMBOLS)
   "Handle missing symbols needed to export."
   (SETQ SYMBOLS (SORT SYMBOLS 'STRING<))
   (LET ((MY-CONDITION 'EXPORT-MISSING :PACKAGE PACKAGE :SYMBOLS SYMBOLS (MAKE-CONDITION)))
        (FLET ((FILTER NIL (EQ *CURRENT-CONDITION* MY-CONDITION)))
              (RESTART-CASE (ERROR MY-CONDITION)
                      (IMPORT-PROCEED NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                                   (FORMAT STREAM "Import missing symbols
                                                                          into ~A, then export them" PACKAGE
                                                                          ))
                              (IMPORT SYMBOLS PACKAGE))
                      (ABORT NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                          (FORMAT STREAM "Abort export from package ~A"
                                                                 PACKAGE))
                              (IL:RETFROM 'EXPORT NIL))))))

(DEFUN IL:RESOLVE-IMPORT-CONFLICT (PACKAGE SYMBOLS)
   "Handle conflict signalled by import.  Returning from here does shadowing import."
   (SETQ SYMBOLS (SORT SYMBOLS 'STRING<))
   (LET ((MY-CONDITION (MAKE-CONDITION 'IMPORT-CONFLICT :PACKAGE PACKAGE :SYMBOLS SYMBOLS)))
        (FLET ((FILTER NIL (EQ *CURRENT-CONDITION* MY-CONDITION)))
              (RESTART-CASE (ERROR MY-CONDITION)
                      (SHADOWING-IMPORT-PROCEED NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                                             (FORMAT STREAM "Import symbols
                                                                                    into ~S with ~S
                                                                                    instead" (PACKAGE-NAME
                                                                                             PACKAGE)
                                                                                    'SHADOWING-IMPORT))
                              NIL)
                      (ABORT NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                          (FORMAT STREAM "Abort import into package ~S"
                                                                 (PACKAGE-NAME PACKAGE)))
                              (IL:RETFROM 'IMPORT NIL))))))

(DEFUN IL:RESOLVE-UNINTERN-CONFLICT (SYMBOL SYMBOLS PACKAGE)
```

```
   "Handle a conflict noted by unintern."
   (SETQ SYMBOLS (SORT SYMBOLS 'STRING<))
   (LET ((MY-CONDITION (MAKE-CONDITION 'UNINTERN-CONFLICT :SYMBOL SYMBOL :SYMBOLS SYMBOLS :PACKAGE PACKAGE)))
        (FLET ((FILTER NIL (EQ *CURRENT-CONDITION* MY-CONDITION)))
              (RESTART-CASE (ERROR MY-CONDITION)
                     (SHADOWING-IMPORT-PROCEED (SYMBOL-TO-IMPORT)
                            :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                     (FORMAT STREAM "Choose symbol and ~S it to hide
                                                            conflicts in package ~S" 'SHADOWING-IMPORT
                                                            (PACKAGE-NAME PACKAGE)))
                            :INTERACTIVE
                            (LAMBDA NIL
                                   (LOOP (LET ((SYMBOL (IL:MENU (IL:CREATE IL:MENU
                                                                IL:TITLE IL:_ "Choose symbol to
                                                                shadowing-import"
                                                                IL:ITEMS IL:_ SYMBOLS
                                                                IL:CENTERFLG IL:_ T))))
                                               (WHEN (MEMBER SYMBOL SYMBOLS :TEST #'EQ)
                                                   (RETURN (LIST SYMBOL))))))
                            (SHADOWING-IMPORT SYMBOL-TO-IMPORT PACKAGE)
                            (IL:RETFROM 'IL:RESOLVE-UNINTERN-CONFLICT T))
                     (ABORT NIL :FILTER FILTER :REPORT (LAMBDA (STREAM)
                                                         (FORMAT STREAM "Abort unintern of symbol ~s from
                                                                package ~s" SYMBOL (PACKAGE-NAME PACKAGE)))
                            (IL:RETFROM 'UNINTERN NIL))))))


(DEFINE-CONDITION SYMBOL-COLON-ERROR (READ-ERROR)
   (NAME)
   (:REPORT (LAMBDA (CONDITION *STANDARD-OUTPUT*)
                  (FORMAT T "Invalid symbol syntax in \"~A\"" (SYMBOL-COLON-ERROR-NAME CONDITION)))))


(DEFUN IL:\\INVALID.SYMBOL (BASE LEN NCOLONS PACKAGE EXTRASEGMENTS)

;;; Called when scanning a symbol that has more than 2 colons, or more than 1 non-consecutive colon.  If return from here, will read the symbol as
;;; though the extra colons were escaped.

   (DECLARE (SPECIAL IL:\\FATPNAMESTRINGP)                    ; This ain't my fault, honest.
            )
   (LET ((MY-CONDITION (MAKE-CONDITION 'SYMBOL-COLON-ERROR :NAME (IL:CONCAT (IF (AND PACKAGE
                                                                                 (NOT (EQ PACKAGE
                                                                                          IL:*KEYWORD-PACKAGE*
                                                                                          )))
                                                                            (IF (STRINGP PACKAGE)
                                                                                PACKAGE
                                                                                (PACKAGE-NAME PACKAGE))
                                                                            "")
                                                                        (CASE NCOLONS
                                                                            (1 ":")
                                                                            (2 "::")
                                                                            (T ""))
                                                                        (IL:\\GETBASESTRING BASE 0 LEN
                                                                               IL:\\FATPNAMESTRINGP)))))
         (RESTART-CASE (ERROR MY-CONDITION)
                (ESCAPE-COLONS-PROCEED NIL :FILTER (LAMBDA NIL (EQ *CURRENT-CONDITION* MY-CONDITION))
                       :REPORT "Treat the extra colon(s) as if they were escaped" NIL))))
```

;; Symbol inspector

```
(DEFUN IL:SYMBOL-INSPECT-FETCHFN (IL:OBJECT IL:PROPERTY)
   (CASE IL:PROPERTY
       (IL:NAME (SYMBOL-NAME IL:OBJECT))
       (IL:VALUE (IF (BOUNDP IL:OBJECT)
                     (SYMBOL-VALUE IL:OBJECT)
                     'IL:NOBIND))
       (IL:PLIST (SYMBOL-PLIST IL:OBJECT))
       (PACKAGE (SYMBOL-PACKAGE IL:OBJECT))))


(DEFUN IL:SYMBOL-INSPECT-STOREFN (IL:OBJECT IL:PROPERTY IL:VALUE)
   (CASE IL:PROPERTY
       (IL:NAME (IL:PROMPTPRINT "Can't set symbol name"))
       (IL:VALUE (SETF (SYMBOL-VALUE IL:OBJECT)
                       IL:VALUE))
       (IL:PLIST (SETF (SYMBOL-PLIST IL:OBJECT)
                       IL:VALUE))
       (PACKAGE (SETF (SYMBOL-PACKAGE IL:OBJECT)
                      IL:VALUE))))

(LET ((IL:FORM '((IL:FUNCTION SYMBOLP)
                 (IL:NAME IL:VALUE IL:PLIST PACKAGE)
                 IL:SYMBOL-INSPECT-FETCHFN IL:SYMBOL-INSPECT-STOREFN NIL NIL NIL "Symbol inspector")))
     (COND
        ((NOT (IL:MEMBER IL:FORM IL:INSPECTMACROS))
         (IL:|push| IL:INSPECTMACROS IL:FORM))))
```

```
;; Package inspector

(DEFUN IL:PACKAGE-INSPECT-FETCHFN (IL:OBJECT IL:PROPERTY)
   (CASE IL:PROPERTY
       (IL:NAME (CL::%PACKAGE-NAME IL:OBJECT))
       (IL:NICKNAMES (CL::%PACKAGE-NICKNAMES IL:OBJECT))
       (IL:USE-LIST (CL::%PACKAGE-USE-LIST IL:OBJECT))
       (IL:INTERNAL-SYMBOLS (CL::%PACKAGE-INTERNAL-SYMBOLS IL:OBJECT))
       (IL:EXTERNAL-SYMBOLS (CL::%PACKAGE-EXTERNAL-SYMBOLS IL:OBJECT))
       (IL:SHADOWING-SYMBOLS (CL::%PACKAGE-SHADOWING-SYMBOLS IL:OBJECT))))

(DEFUN IL:PACKAGE-INSPECT-STOREFN (IL:OBJECT IL:PROPERTY IL:VALUE)
   (IL:PROMPTPRINT "Can't set the fields of a package"))

(LET ((IL:FORM '((IL:FUNCTION PACKAGEP)
                 (IL:NAME IL:NICKNAMES IL:USE-LIST IL:INTERNAL-SYMBOLS IL:EXTERNAL-SYMBOLS IL:SHADOWING-SYMBOLS)
                 IL:PACKAGE-INSPECT-FETCHFN IL:PACKAGE-INSPECT-STOREFN NIL NIL NIL "Package inspector")))
     (COND
        ((NOT (IL:MEMBER IL:FORM IL:INSPECTMACROS))
         (IL:|push| IL:INSPECTMACROS IL:FORM))))

;; Package-hashtable inspector

(DEFUN IL:PACKAGE-HASHTABLE-INSPECT-FETCHFN (IL:OBJECT IL:PROPERTY)
   (CASE IL:PROPERTY
       (IL:SIZE (CL::PACKAGE-HASHTABLE-SIZE IL:OBJECT))
       (IL:FREE (CL::PACKAGE-HASHTABLE-FREE IL:OBJECT))
       (IL:DELETED (CL::PACKAGE-HASHTABLE-DELETED IL:OBJECT))
       (IL:CONTENTS (CL::PACKAGE-HASHTABLE-TABLE IL:OBJECT))))

(DEFUN IL:PACKAGE-HASHTABLE-INSPECT-STOREFN (IL:OBJECT IL:PROPERTY IL:VALUE)
   (IL:PROMPTPRINT "Can't set the fields of a package-hashtable"))

(LET ((IL:FORM '((IL:FUNCTION CL::PACKAGE-HASHTABLE-P)
                 (IL:SIZE IL:FREE IL:DELETED IL:CONTENTS)
                 IL:PACKAGE-HASHTABLE-INSPECT-FETCHFN IL:PACKAGE-HASHTABLE-INSPECT-STOREFN)))
     (COND
        ((NOT (IL:MEMBER IL:FORM IL:INSPECTMACROS))
         (IL:|push| IL:INSPECTMACROS IL:FORM))))

;; Package's Prefix accessor and setfs (Edited by TT 14-June-90 for AR#11112)

(DEFUN PACKAGE-PREFIX (PACKAGE)                          ; Edited by TT (14-June-90 : for AR#111122)
   (CL::%PACKAGE-NAMESYMBOL (IL:\\PACKAGIFY PACKAGE)))

(DEFUN SETF-PACKAGE-PREFIX (PACKAGE PREFIX)              ; Edited by TT (14-June-90 : for AR#111122)
   (IF (SYMBOLP PREFIX)
       (SETF (CL::%PACKAGE-NAMESYMBOL (IL:\\PACKAGIFY PACKAGE))
             PREFIX)
       (IF (STRINGP PREFIX)
           (SETF (CL::%PACKAGE-NAMESYMBOL (IL:\\PACKAGIFY PACKAGE))
                 (INTERN PREFIX))
           (ERROR "~S must be symbol or string." PREFIX))))

(DEFSETF PACKAGE-PREFIX SETF-PACKAGE-PREFIX)

(IL:PUTPROPS IL:CMLPACKAGE IL:FILETYPE :COMPILE-FILE)

(IL:PUTPROPS IL:CMLPACKAGE IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "XCL"))

(IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS

(IL:ADDTOVAR IL:NLAMA )

(IL:ADDTOVAR IL:NLAML )

(IL:ADDTOVAR IL:LAMA )
)

(IL:PUTPROPS IL:CMLPACKAGE IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1990))
```

FUNCTION INDEX

IL:DWIM-SYMBOL-PACKAGE 1
ESCAPE-COLONS-PROCEED 2
MAKE-EXTERNAL-PROCEED 2
MAKE-INTERNAL-PROCEED 2
IL:PACKAGE-HASHTABLE-INSPECT-FETCHFN 7
IL:PACKAGE-HASHTABLE-INSPECT-STOREFN 7
IL:PACKAGE-INSPECT-FETCHFN 7
IL:PACKAGE-INSPECT-STOREFN 7
PACKAGE-PREFIX 7
IL:RESOLVE-EXPORT-CONFLICT 5
IL:RESOLVE-EXPORT-MISSING 5
IL:RESOLVE-IMPORT-CONFLICT 5
IL:RESOLVE-MISSING-EXTERNAL-SYMBOL 3
IL:RESOLVE-MISSING-PACKAGE 3
IL:RESOLVE-READER-CONFLICT 3
IL:RESOLVE-UNINTERN-CONFLICT 5
IL:RESOLVE-USE-PACKAGE-CONFLICT 4
SETF-PACKAGE-PREFIX 7
IL:SYMBOL-INSPECT-FETCHFN 6
IL:SYMBOL-INSPECT-STOREFN 6
UGLY-SYMBOL-PROCEED 2
IL:\\INVALID.SYMBOL 6

STRUCTURE INDEX

EXPORT-CONFLICT 4
EXPORT-MISSING 4
IMPORT-CONFLICT 4
MISSING-EXTERNAL-SYMBOL .2
MISSING-PACKAGE 2
PACKAGE-ERROR 4
READ-CONFLICT 2
SYMBOL-COLON-ERROR 6
SYMBOL-CONFLICT 4
UNINTERN-CONFLICT 4
USE-CONFLICT 4

SETF INDEX

PACKAGE-PREFIX 7
SYMBOL-PACKAGE 1

VARIABLE INDEX

PREFERRED-READING-SYMBOLS 2
IL:DWIMUSERFORMS 2

PROPERTY INDEX

IL:CMLPACKAGE 7