File created: 15-Aug-88 18:05:13 **{QV}<NOTECARDS>1.3L>LIBRARY>NCLOGGER.;6**

changes to: (ADVICE NC.DeleteNoteCardInternal NC.MakeNoteCard NC.EditNoteCard NC.QuitCard NC.DeactivateCard NC.ActivateCard NC.PutTitle NC.PutPropList NC.PutMainCardData NC.PutLinks NC.CardSaveFn NC.AssignTitle NC.DelToLink NC.MakeLink NC.RelayoutBrowserCard NC.UpdateBrowserCard NC.GetNewCard NC.AddCardType)

previous date: 26-Jul-88 10:18:47 {QV}<NOTECARDS>1.3L>LIBRARY>NCLOGGER.;5

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ NCLOGGERCOMS
       ((DECLARE%: DONTCOPY (PROP MAKEFILE-ENVIRONMENT NCLOGGER))
```

;;; This package instruments NoteCards so that entries are written to a log file whenever certain monitored events take place.

```
        (GLOBALVARS NCLOG.GlobalLogInfo NCLOG.UIDHashArray NCLOG.HashArraySize NCLOG.UIDCtr)
        (GLOBALVARS NC.OrigReadTable NCLOG.SavedCardObject)
        (GLOBALVARS NCLOG.ListOfAllEventTypes NCLOG.ListOfEventTypes NCLOG.NumHashArray)
        (INITVARS (NCLOG.UIDCtr 0)
               (NCLOG.HashArraySize 100)
               (NCLOG.UIDHashArray (NC.CreateUIDHashArray NCLOG.HashArraySize))
               (NCLOG.GlobalLogInfo NIL))
        (INITVARS (NCLOG.NumHashArray (HASHARRAY NCLOG.HashArraySize))
               (NCLOG.ListOfAllEventTypes '(GetNewCard MakeCard.Begin MakeCard.End EditCard.Begin EditCard.End
                                            SaveCard.Begin SaveCard.End QuitCard.Begin QuitCard.End
                                            AssignTitle.Begin AssignTitle.End CacheCard UncacheCard
                                            DelCard MakeLink DelLink PutMainCardData PutLinks PutTitle
                                            PutPropList RecomputeBrowser.Begin RecomputeBrowser.End
                                            RelayoutBrowser.Begin RelayoutBrowser.End AddCardType
                                            StartLogging))
               (NCLOG.ListOfEventTypes NCLOG.ListOfAllEventTypes))
        (RECORDS NCLogInfo)
```

;;; External interface fns.

```
        (FNS NCLOG.StartLogging NCLOG.StopLogging NCLOG.SuspendLogging NCLOG.LoggingOnP)
```

;;; Internal stuff

```
        (FNS NCLOG.FetchLogInfoList NCLOG.SetLogInfoList NCLOG.LogEvent NCLOG.OpenLoggingStream
             NCLOG.WriteEventToLogStream NCLOG.WriteLogFileHeader NCLOG.SetEventTypesToLog NCLOG.UIDFromNum)
```

;;; Stuff to handle the printing of various NC objects.

```
        (FNS NCLOG.CardObjectDEFPRINTFn NCLOG.LinkDEFPRINTFn NCLOG.NoteFileDEFPRINTFn NCLOG.UIDDEFPRINTFn
             NCLOG.SingleArgDEFPRINT NCLOG.NumFromUID NCLOG.StringFromUID)
```

;;; Here's the advice for the functions that we want to monitor.

;;; Operations that happen to an open notefile.

```
        (ADVISE NC.EditNoteCard NC.QuitCard NC.MakeNoteCard NC.DeleteNoteCardInternal)
        (ADVISE NC.ActivateCard NC.DeactivateCard)
        (ADVISE NC.CardSaveFn NC.PutLinks NC.PutMainCardData NC.PutPropList NC.PutTitle)
        (ADVISE NC.AssignTitle)
        (ADVISE NC.MakeLink NC.DelToLink)
        (ADVISE NC.UpdateBrowserCard NC.RelayoutBrowserCard)
        (ADVISE NC.GetNewCard)
```

;;; Other operations.

```
        (ADVISE NC.AddCardType)))

(DECLARE%: DONTCOPY

(PUTPROPS NCLOGGER MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP" :BASE 10))
)
```

;;; This package instruments NoteCards so that entries are written to a log file whenever certain monitored events take place.

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NCLOG.GlobalLogInfo NCLOG.UIDHashArray NCLOG.HashArraySize NCLOG.UIDCtr)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.OrigReadTable NCLOG.SavedCardObject)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NCLOG.ListOfAllEventTypes NCLOG.ListOfEventTypes NCLOG.NumHashArray)
)

(RPAQ? NCLOG.UIDCtr 0)

(RPAQ? NCLOG.HashArraySize 100)

(RPAQ? NCLOG.UIDHashArray (NC.CreateUIDHashArray NCLOG.HashArraySize))

(RPAQ? NCLOG.GlobalLogInfo NIL)

(RPAQ? NCLOG.NumHashArray (HASHARRAY NCLOG.HashArraySize))

(RPAQ? NCLOG.ListOfAllEventTypes '(GetNewCard MakeCard.Begin MakeCard.End EditCard.Begin EditCard.End
                                   SaveCard.Begin SaveCard.End QuitCard.Begin QuitCard.End AssignTitle.Begin
                                   AssignTitle.End CacheCard UncacheCard DelCard MakeLink DelLink
                                   PutMainCardData PutLinks PutTitle PutPropList RecomputeBrowser.Begin
                                   RecomputeBrowser.End RelayoutBrowser.Begin RelayoutBrowser.End
                                   AddCardType StartLogging))

(RPAQ? NCLOG.ListOfEventTypes NCLOG.ListOfAllEventTypes)

(DECLARE%: EVAL@COMPILE

(DATATYPE NCLogInfo (LoggingStream LoggingOnFlg NoticedUIDs EventTypesToLog LoggingFileName))
)

(/DECLAREDATATYPE 'NCLogInfo '(POINTER POINTER POINTER POINTER POINTER)
        ;; ---field descriptor list elided by lister---
        '10)


;;; External interface fns.

(DEFINEQ
```

(NCLOG.StartLogging

```
  [LAMBDA (NoteFile LoggingFileNameOrNCLogInfo EventTypesToLog)  (* pmi%: "29-Oct-87 10:32")

          (* * Turn on logging for NoteFile. If logging was already on, then do nothing and return NIL.
          If LoggingFileName is NIL, then use a {nodircore} file.)

          (* * pmi 10/23/87%: Add current LoggingFileName to NCLogInfo record for this NoteFile)

          (* * pmi 10/27/87%: Made changes to accommodate multiple NCLogInfo records per notefile.
          Also changed so that NIL for NoteFile means turn on global logging.
          Now returns the relevent NCLogInfo record.)

          (* * pmi 10/28/87%: Now takes EventTypesToLog argument which determines which set of event types will be logged.)

    (DECLARE (GLOBALVARS NCLOG.ListOfEventTypes))
    (LET ((NCLogInfoList (NCLOG.FetchLogInfoList NoteFile))
          NCLogInfo LoggingFileName LoggingStream)

          (* * Sort out LoggingFileNameOrNCLogInfo)

         (if (type? NCLogInfo LoggingFileNameOrNCLogInfo)
             then (SETQ NCLogInfo LoggingFileNameOrNCLogInfo)
                  (SETQ LoggingFileName NIL)
           elseif LoggingFileNameOrNCLogInfo
             then (SETQ NCLogInfo NIL)
                  (SETQ LoggingFileName (FULLNAME LoggingFileNameOrNCLogInfo 'NEW))
           else (SETQ LoggingFileName '{NODIRCORE}NCLOGFILE))
         (OR EventTypesToLog (SETQ EventTypesToLog NCLOG.ListOfEventTypes))

          (* * Look for LoggingFileNameOrNCLogInfo as either a NCLogInfo record on this notefile or a filename contained in one of
          the NCLogInfo records.)

         (if [AND NoteFile NCLogInfoList (OR (FMEMB NCLogInfo NCLogInfoList)
                                             (SETQ NCLogInfo (for NCLogInfoRecord in NCLogInfoList
                                                                thereis (EQUAL LoggingFileName (fetch (NCLogInfo
                                                                                                       LoggingFileName
```

```
                                                                                                    )
                                                                                                of
                                                                                                  NCLogInfoRecord
                                                                                                  ]
        then (if (fetch (NCLogInfo LoggingOnFlg) of NCLogInfo)
                 then

(* * Logging is already on. Bail out.)

                      NIL
             elseif (AND (SETQ LoggingStream (fetch (NCLogInfo LoggingStream) of NCLogInfo))
                         (OPENP LoggingStream 'BOTH))
               then

(* * Logging file exists and is open.)

                     (replace (NCLogInfo LoggingOnFlg) of NCLogInfo with T)
                     NCLogInfo
             else

(* * There's no LoggingStream or it's closed, so make one using LoggingFileName.)

                 (NCLOG.OpenLoggingStream NoteFile LoggingFileName LoggingStream NCLogInfo
                       EventTypesToLog))
 else

(* * This notefile has no logging happening; start one op.)

      (NCLOG.OpenLoggingStream NoteFile LoggingFileName NIL NIL EventTypesToLog])
```

(NCLOG.StopLogging

```
[LAMBDA (NoteFile NCLogInfo WriteLegendFlg)                    (* pmi%: "28-Oct-87 12:19")
```

(* * Turn off notefile logging. Return the result of closing the LoggingStream.
If WriteLegendFlg is non-nil, then write down a legend mapping numbers to UIDs.
Write down the location of the start of the legend at the end of the file.)

(* * pmi 10/28/87%: Changed to handle multiple NCLogInfo records per notefile.
If a NCLogInfo is specified, then stop logging on it. If not, stop logging on all NCLogInfo records associated with this notefile.)

```
(DECLARE (GLOBALVARS NC.OrigReadTable))
(if (NCLOG.LoggingOnP NoteFile NCLogInfo)
    then (LET (NCLogInfoList LoggingStream)
              (NCLOG.SuspendLogging NoteFile NCLogInfo)
              [SETQ NCLogInfoList (MKLIST (OR NCLogInfo (NCLOG.FetchLogInfoList NoteFile]
              (for NCLogInfoRecord in NCLogInfoList
                 collect (SETQ LoggingStream (fetch (NCLogInfo LoggingStream) of NCLogInfoRecord))
                         (if (AND WriteLegendFlg LoggingStream (OPENP LoggingStream))
                             then (LET ((StartLegendLoc (GETFILEPTR LoggingStream)))
                                       (for UID in (fetch (NCLogInfo NoticedUIDs) of NCLogInfoRecord)
                                          do (PRINT (LIST (NCLOG.NumFromUID UID)
                                                          UID)
                                                    LoggingStream NC.OrigReadTable))
                                       (PRINT StartLegendLoc LoggingStream NC.OrigReadTable)))
                         (PROG1 (AND LoggingStream (OPENP LoggingStream)
                                     (CLOSEF LoggingStream))
                             (replace (NCLogInfo LoggingStream) of NCLogInfoRecord with NIL)
                             (replace (NCLogInfo NoticedUIDs) of NCLogInfoRecord with NIL))])
```

(NCLOG.SuspendLogging

```
[LAMBDA (NoteFile NCLogInfo)                                   ; Edited 13-Jul-88 16:30 by pmi
```

;; Temporarily turn off the logging on NoteFile, but don't close LoggingStream.

;; pmi 10/27/87: Changed to handle multiple NCLogInfo records per notefile. If NCLogInfo is specified, suspend its logging. If not, suspend logging
;; on all NCLogInfo records associated with this notefile. Also changed so that a NIL NoteFile means suspend global logging.

;; pmi 7/13/88: Fixed LET variable initialization (added parentheses).

```
(DECLARE (GLOBALVARS NCLOG.GlobalLogInfo))
(LET ((NCLogInfoList (NCLOG.FetchLogInfoList NoteFile)))
     (if NoteFile
         then (if NCLogInfo
                  then (AND (MEMBER NCLogInfo NCLogInfoList)
                            (replace (NCLogInfo LoggingOnFlg) of NCLogInfo with NIL))
                  else (for NCLogInfoRecord in NCLogInfoList do (replace (NCLogInfo LoggingOnFlg) of
                                                                         NCLogInfoRecord
                                                                    with NIL)))
          else (replace (NCLogInfo LoggingOnFlg) of NCLOG.GlobalLogInfo with NIL])
```

(NCLOG.LoggingOnP

```
[LAMBDA (NoteFile NCLogInfo)                                   (* pmi%: " 3-Nov-87 11:55")
```

(* * Return non-nil if logging is turned on for NoteFile.)

(* * pmi 10/27/87%: Changed to handle multiple NCLogInfo records per notefile.
If no NCLogInfo is specified and there is at least one NCLogInfo record on this notefile with a non-NIL LoggingOnFlg, then T

```
                is returned. Also, now returns the LoggingOnFlg for NCLOG.GlobalLogInfo if NoteFile is NIL.)

    (DECLARE (GLOBALVARS NCLOG.GlobalLogInfo))
    (LET ((NCLogInfoList (NCLOG.FetchLogInfoList NoteFile)))
         (if NoteFile
             then (if NCLogInfo
                      then (AND (MEMBER NCLogInfo NCLogInfoList)
                                (fetch (NCLogInfo LoggingOnFlg) of NCLogInfo))
                    else (for NCLogInfoRecord in NCLogInfoList thereis (fetch (NCLogInfo LoggingOnFlg) of
                                                                                              NCLogInfoRecord
                                                                                        )))
           elseif (type? NCLogInfo NCLOG.GlobalLogInfo)
             then (fetch (NCLogInfo LoggingOnFlg) of NCLOG.GlobalLogInfo)
           else NIL])
)
```

;;; Internal stuff

```
(DEFINEQ

(NCLOG.FetchLogInfoList
  [LAMBDA (NoteFile)                                                (* pmi%: "23-Oct-87 15:03")

                (* * Fetch the NCLogInfo records from the NoteFile if any. If NoteFile is nil, then return the global one.)

                (* * pmi 10/23/87%: Was NCLOG.FetchLogInfo, changed to handle a list of NCLogInfos on each notefile.)

    (DECLARE (GLOBALVARS NCLOG.GlobalLogInfo))
    (COND
       ((type? NoteFile NoteFile)
        (NCP.NoteFileProp NoteFile 'NCLogInfoList))
       ((NULL NoteFile)
        NCLOG.GlobalLogInfo)
       (T NIL])

(NCLOG.SetLogInfoList
  [LAMBDA (NoteFile NewLogInfoRecord)                               (* pmi%: "28-Oct-87 16:06")

                (* * Add this NewLogInfoRecord to the list of the NCLogInfo records for NoteFile.
                If NoteFile is nil, then replace the global one.)

                (* * pmi 10/28/87%: Was NCLOG.SetLogInfo, changed to keep a list of NCLogInfos on a notefile, instead of just one.
                Also now reset list of NCLogInfos to NIL if NewLogInfoRecord is NIL.)

    (DECLARE (GLOBALVARS NCLOG.GlobalLogInfo))
    (if (type? NoteFile NoteFile)
        then (if NewLogInfoRecord
                 then (NCP.NoteFileAddProp NoteFile 'NCLogInfoList NewLogInfoRecord)
               else (NCP.NoteFileProp NoteFile 'NCLogInfoList NIL))
             NewLogInfoRecord
      elseif (NULL NoteFile)
        then (SETQ NCLOG.GlobalLogInfo NewLogInfoRecord)
             NewLogInfoRecord
      else NIL])

(NCLOG.LogEvent
  [LAMBDA (EventType NoteFile EventArgs)                            ; Edited 11-Jul-88 17:09 by pmi

    ;; Log an event of type EventType happening to NoteFile with args EventArgs.  NoteFile may be NIL if there's no notefile object.  Can log either on
    ;; global logging stream or on Logging stream for given notefile or both.

    ;; pmi 10/27/87: Changed to handle multiple NCLogInfo records for a notefile.

    ;; pmi 7/11/88: Changed call to NCLOG.FetchLogInfo to NCLOG.FetchLogInfoList.

    (LET ((EventTime (IDATE))
          GlobalLogInfo NoteFileLogInfo LoggingStream)              ; Log on global logging stream if turned on.
         (if (AND (NCLOG.LoggingOnP NIL)
                  (SETQ GlobalLogInfo (NCLOG.FetchLogInfoList NIL))
                  (SETQ LoggingStream (fetch (NCLogInfo LoggingStream) of GlobalLogInfo))
                  (OPENP LoggingStream 'OUTPUT))
             then (NCLOG.WriteEventToLogStream LoggingStream GlobalLogInfo EventType NoteFile EventTime
                         (MKLIST EventArgs)))                       ; Log on local logging stream if turned on.
         (if (type? NoteFile NoteFile)
             then (for NoteFileLogInfo in (NCLOG.FetchLogInfoList NoteFile)
                     when (AND (NCLOG.LoggingOnP NoteFile NoteFileLogInfo)
                               (SETQ LoggingStream (fetch (NCLogInfo LoggingStream) of NoteFileLogInfo))
                               (OPENP LoggingStream 'OUTPUT)
                               (FMEMB EventType (fetch (NCLogInfo EventTypesToLog) of NoteFileLogInfo)))
                     do (NCLOG.WriteEventToLogStream LoggingStream NoteFileLogInfo EventType NoteFile EventTime
                               (MKLIST EventArgs])

(NCLOG.OpenLoggingStream
  [LAMBDA (NoteFile LoggingFileName LoggingStream NCLogInfo EventTypesToLog)
```

(* pmi%: "28-Oct-87 17:57")

(* * Creates a new NCLogInfo record for this notefile and logging filename.
Opens a stream to the logging file and sets the appropriate fields of the NCLogInfo record.
Returns the NCLogInfo record, if successful.)

(* * pmi 10/28/87%: Now takes EventTypesToLog argument which determines which set of event types will be logged.)

```
(if NCLogInfo
  else (SETQ NCLogInfo (create NCLogInfo))
       (NCLOG.SetLogInfoList NoteFile NCLogInfo))
(if (SETQ LoggingStream (OPENSTREAM (OR LoggingStream LoggingFileName)
                                    'BOTH))
    then (replace (NCLogInfo LoggingStream) of NCLogInfo with LoggingStream)
         (replace (NCLogInfo LoggingOnFlg) of NCLogInfo with T)
         (replace (NCLogInfo LoggingFileName) of NCLogInfo with (FULLNAME LoggingStream))
         (replace (NCLogInfo EventTypesToLog) of NCLogInfo with EventTypesToLog)
         (NCLOG.WriteLogFileHeader LoggingStream NoteFile)
         NCLogInfo])
```

(NCLOG.WriteEventToLogStream

```
[LAMBDA (Stream NCLogInfo EventType NoteFile EventTime EventArgs)
```

; Edited 5-Dec-87 20:02 by rht:

(* * Write given event, time, NoteFile and args to stream. Temporarily change the UID print fn so that UIDs and other objects will get printed to the stream in a readable manner.)

```
(DECLARE (GLOBALVARS NC.OrigReadTable))
(RESETLST
    [RESETSAVE (NCLOG.SingleArgDEFPRINT (LIST 'CardObject (FUNCTION NCLOG.CardObjectDEFPRINTFn]
    [RESETSAVE (NCLOG.SingleArgDEFPRINT (LIST 'Link (FUNCTION NCLOG.LinkDEFPRINTFn]
    [RESETSAVE (NCLOG.SingleArgDEFPRINT (LIST 'NoteFile (FUNCTION NCLOG.NoteFileDEFPRINTFn]
    [RESETSAVE (NCLOG.SingleArgDEFPRINT (LIST 'UID (FUNCTION NCLOG.UIDDEFPRINTFn]
    (PRINT `(,EventType ,EventTime ,NoteFile ,@EventArgs)
           Stream NC.OrigReadTable))])
```

(NCLOG.WriteLogFileHeader

```
[LAMBDA (LoggingStream NoteFile)
```

(* rht%: "18-Mar-87 16:00")

(* * Write header info to the logging file for this notefile. Currently, this is just an event having type StartLogging.)

```
(NCLOG.LogEvent 'StartLogging NoteFile (AND (type? NoteFile NoteFile)
                                            (fetch (NoteFile FullFileName) of NoteFile])
```

(NCLOG.SetEventTypesToLog

```
[LAMBDA (NoteFile NCLogInfo EventTypesToLog)
```

(* pmi%: "28-Oct-87 11:20")

(* * pmi 10/28/87%: Created to change the EventTypesToLog field in the NCLogInfo record of NoteFile.
If EventTypesToLog is NIL, then change the EventTypesToLog fields of all NCLogInfo records associated with NoteFile.
If NoteFile is NIL, change the EventTypesToLog field of the global NCLogInfo record.
If EventTypesToLog is NIL, then use NCLOG.ListOfEventTypes.)

```
(DECLARE (GLOBALVARS NCLOG.GlobalLogInfo NCLOG.ListOfEventTypes))
(LET ((NCLogInfoList (NCLOG.FetchLogInfoList NoteFile)))
     (OR EventTypesToLog (SETQ EventTypesToLog NCLOG.ListOfEventTypes))
     (if NoteFile
         then (if NCLogInfo
                  then (for NCLogInfoRecord in NCLogInfoList when (EQ NCLogInfoRecord NCLogInfo)
                          do (replace (NCLogInfo EventTypesToLog) of NCLogInfo with EventTypesToLog))
              else (for NCLogInfoRecord in NCLogInfoList do (replace (NCLogInfo EventTypesToLog) of NCLogInfo
                                                               with EventTypesToLog)))
          else (replace (NCLogInfo EventTypesToLog) of NCLOG.GlobalLogInfo with EventTypesToLog])
```

(NCLOG.UIDFromNum

```
[LAMBDA (Num)
```

(* pmi%: "23-Nov-87 12:06")

(* * Translate an integer into a UID using a global hash array.)

```
(DECLARE (GLOBALVARS NCLOG.NumHashArray))
(GETHASH Num NCLOG.NumHashArray])
```

)

;;; Stuff to handle the printing of various NC objects.

```
(DEFINEQ
```

(NCLOG.CardObjectDEFPRINTFn

```
[LAMBDA (Card Stream)
```

(* rht%: "11-Mar-87 22:27")

(* * This is called when PRINT, PRIN1, etc. try to print instance of Card datatype.
Only meant to be used under NCLOG.WriteEventToLogStream.)

```
    (if (AND Stream (NEQ Stream 'T)
             (NOT (IMAGESTREAMP Stream)))
        then (PRINT (fetch (Card UID) of Card)
                    Stream NC.OrigReadTable)
             T
      else NIL])
```

(NCLOG.LinkDEFPRINTFn

```
[LAMBDA (Link Stream)                                          (* rht%: "11-Mar-87 22:28")
```

(* * This is called when PRINT, PRIN1, etc. try to print instance of Link datatype.
Only meant to be used under NCLOG.WriteEventToLogStream.)

```
    (if (AND Stream (NEQ Stream 'T)
             (NOT (IMAGESTREAMP Stream)))
        then (PRINT (fetch (Link UID) of Link)
                    Stream NC.OrigReadTable)
             T
      else NIL])
```

(NCLOG.NoteFileDEFPRINTFn

```
[LAMBDA (NoteFile Stream)                                      (* rht%: "11-Mar-87 22:28")
```

(* * This is called when PRINT, PRIN1, etc. try to print instance of NoteFile datatype.
Only meant to be used under NCLOG.WriteEventToLogStream.)

```
    (if (AND Stream (NEQ Stream 'T)
             (NOT (IMAGESTREAMP Stream)))
        then (PRINT (fetch (NoteFile UID) of NoteFile)
                    Stream NC.OrigReadTable)
             T
      else NIL])
```

(NCLOG.UIDDEFPRINTFn

```
[LAMBDA (UID Stream)                                           (* rht%: "18-Mar-87 17:25")
```

(* * This is called when PRINT, PRIN1, etc. try to print instance of UID datatype.
Only meant to be used under NCLOG.WriteEventToLogStream.)

```
    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (if (AND Stream (NEQ Stream 'T)
             (NOT (IMAGESTREAMP Stream)))
        then (LET [(NCLogInfo (CAR (NLSETQ (STKARG 'NCLogInfo 'NCLOG.WriteEventToLogStream]
                   (if NCLogInfo
                       then (pushnew (fetch (NCLogInfo NoticedUIDs) of NCLogInfo)
                                     UID)))
             (PRINT (NCLOG.NumFromUID UID)
                    Stream NC.OrigReadTable)
             T
      else NIL])
```

(NCLOG.SingleArgDEFPRINT

```
[LAMBDA (TypeAndFn)                                            (* rht%: "11-Mar-87 17:03")
```

(* * An incredibly dorky hack to get around the brain-damaged way RESETLST works.
Need this because though DEFPRINT returns the old value, it takes two args thus is unsuitable for RESETSAVE.
UGH!)

```
    (LIST (CAR TypeAndFn)
          (DEFPRINT (CAR TypeAndFn)
                 (CADR TypeAndFn])
```

(NCLOG.NumFromUID

```
[LAMBDA (UID)                                                  (* pmi%: "14-Dec-87 19:15")
```

(* * Translate the UID into an integer using a global hash array.
This is to save space on the logging file.)

(* * pmi 11/24/87%: Added NCLOG.NumHashArray for getting UID's back from numbers later.)

```
    (DECLARE (GLOBALVARS NCLOG.UIDHashArray NCLOG.NumHashArray NCLOG.UIDCtr))
    (if (GETHASH UID NCLOG.UIDHashArray)
      else (PUTHASH UID (add NCLOG.UIDCtr 1)
                    NCLOG.UIDHashArray)
           (PUTHASH NCLOG.UIDCtr UID NCLOG.NumHashArray])
```

(NCLOG.StringFromUID

```
[LAMBDA (UID)                                                  (* rht%: "11-Mar-87 15:59")
```

(* * Return an atom made from the numbers making up UID separated by commas.)

```
    (CONCAT 'U (ffetch (UID UID0) of UID)
            '%,
            (ffetch (UID UID1) of UID)
            '%,
            (ffetch (UID UID2) of UID)
            '%,
            (ffetch (UID UID3) of UID)
            '%,
            (ffetch (UID UID4) of UID)
            '%,
            (ffetch (UID UID5) of UID)
            '%,
            (ffetch (UID UID6) of UID])

)
```

;;; Here's the advice for the functions that we want to monitor.

;;; Operations that happen to an open notefile.

```
[XCL:REINSTALL-ADVICE 'NC.EditNoteCard :BEFORE '((:LAST (NCLOG.LogEvent 'EditCard.Begin (fetch (Card NoteFile)
                                                                                          of Card)
                                                                Card)))
       :AFTER
       '((:LAST (NCLOG.LogEvent 'EditCard.End (fetch (Card NoteFile) of Card)
                      Card]

[XCL:REINSTALL-ADVICE 'NC.QuitCard :AROUND '((:LAST (LET* ((ClosingCard (NC.CoerceToCard CardIdentifier))
                                                           (ClosingCardNoteFile (fetch (Card NoteFile)
                                                                                   of ClosingCard)))
                                                          (NCLOG.LogEvent 'QuitCard.Begin ClosingCardNoteFile
                                                                 (LIST ClosingCard Don'tDeactivateFlg))
                                                          *
                                                          (NCLOG.LogEvent 'QuitCard.End ClosingCardNoteFile
                                                                 (LIST ClosingCard Don'tDeactivateFlg]

[XCL:REINSTALL-ADVICE 'NC.MakeNoteCard :BEFORE '[(:LAST (NCLOG.LogEvent 'MakeCard.Begin NoteFile
                                                                (LIST NoteCardType Title NoDisplayFlg]
       :AFTER
       '((:LAST (NCLOG.LogEvent 'MakeCard.End NoteFile (LIST NoteCardType Title (NC.CoerceToCard !VALUE]

[XCL:REINSTALL-ADVICE 'NC.DeleteNoteCardInternal :BEFORE '((:LAST (NCLOG.LogEvent 'DelCard (fetch (Card NoteFile)
                                                                                             of Card)
                                                                                   (LIST Card (NC.FetchTitle Card]

(READVISE NC.EditNoteCard NC.QuitCard NC.MakeNoteCard NC.DeleteNoteCardInternal)

[XCL:REINSTALL-ADVICE 'NC.ActivateCard :BEFORE '((:LAST (NCLOG.LogEvent 'CacheCard (fetch (Card NoteFile)
                                                                                       of Card)
                                                                Card]

[XCL:REINSTALL-ADVICE 'NC.DeactivateCard :BEFORE '((:LAST (NCLOG.LogEvent 'UncacheCard (fetch (Card NoteFile)
                                                                                         of Card)
                                                                  Card]

(READVISE NC.ActivateCard NC.DeactivateCard)

[XCL:REINSTALL-ADVICE 'NC.CardSaveFn :BEFORE '[(:LAST (LET ((Card (NC.CoerceToCard WindowOrID)))
                                                           (NCLOG.LogEvent 'SaveCard.Begin (fetch (Card NoteFile)
                                                                                              of Card)
                                                                 Card]
       :AFTER
       '((:LAST (LET ((Card (NC.CoerceToCard WindowOrID)))
                     (NCLOG.LogEvent 'SaveCard.End (fetch (Card NoteFile) of Card)
                            Card]

[XCL:REINSTALL-ADVICE 'NC.PutLinks :BEFORE '((:LAST (NCLOG.LogEvent 'PutLinks (fetch (Card NoteFile) of Card)
                                                           Card]

[XCL:REINSTALL-ADVICE 'NC.PutMainCardData :BEFORE '((:LAST (NCLOG.LogEvent 'PutMainCardData (fetch (Card NoteFile
                                                                                                    )
                                                                                             of Card)
                                                                  Card]

[XCL:REINSTALL-ADVICE 'NC.PutPropList :BEFORE '((:LAST (NCLOG.LogEvent 'PutPropList (fetch (Card NoteFile)
                                                                                       of Card)
                                                             Card]

[XCL:REINSTALL-ADVICE 'NC.PutTitle :BEFORE '((:LAST (NCLOG.LogEvent 'PutTitle (fetch (Card NoteFile) of Card)
                                                           Card]

(READVISE NC.CardSaveFn NC.PutLinks NC.PutMainCardData NC.PutPropList NC.PutTitle)

[XCL:REINSTALL-ADVICE 'NC.AssignTitle :BEFORE '[(:LAST (LET ((Card (NC.CoerceToCard CardIdentifier)))
                                                            (NCLOG.LogEvent 'AssignTitle.Begin
```

```
                                                                    (fetch (Card NoteFile) of Card)
                                                                    (LIST Card (NC.FetchTitle Card]
       :AFTER
       '((:LAST (LET ((Card (NC.CoerceToCard CardIdentifier)))
                     (NCLOG.LogEvent 'AssignTitle.End (fetch (Card NoteFile) of Card)
                            (LIST Card (NC.FetchTitle Card]

(READVISE NC.AssignTitle)

[XCL:REINSTALL-ADVICE 'NC.MakeLink :AFTER '((:LAST (LET [(NoteFile (fetch (Card NoteFile)
                                                                       of (OR (NC.CoerceToCard SourceCard)
                                                                              (NC.CoerceToCard Window]
                                                                (if (type? Link !VALUE)
                                                                    then (NCLOG.LogEvent 'MakeLink NoteFile
                                                                                (LIST !VALUE (fetch (Link SourceCard)
                                                                                                    of !VALUE)
                                                                                             (fetch (Link DestinationCard)
                                                                                                of !VALUE)
                                                                                             (fetch (Link Label) of !VALUE)
                                                                                             (fetch (Link AnchorMode)
                                                                                                of !VALUE)
                                                                                             (fetch (Link DisplayMode)
                                                                                                of !VALUE)))
                                                                  else (NCLOG.LogEvent 'MakeLink NoteFile !VALUE]

[XCL:REINSTALL-ADVICE 'NC.DelToLink :BEFORE
       '((:LAST (LET [(NoteFile (fetch (Card NoteFile) of (fetch (Link SourceCard) of Link]
                     (if (type? Link Link)
                         then (NCLOG.LogEvent 'DelLink NoteFile (LIST Link (fetch (Link SourceCard) of Link)
                                                                           (fetch (Link DestinationCard) of Link)
                                                                           (fetch (Link Label) of Link)
                                                                           (fetch (Link AnchorMode) of Link)
                                                                           (fetch (Link DisplayMode) of Link)))
                       else (NCLOG.LogEvent 'DelLink NoteFile Link]

(READVISE NC.MakeLink NC.DelToLink)

[XCL:REINSTALL-ADVICE 'NC.UpdateBrowserCard :BEFORE '[(:LAST (LET ((Card (NC.CoerceToCard Window)))
                                                                 (NCLOG.LogEvent 'RecomputeBrowser.Begin
                                                                        (fetch (Card NoteFile) of Card)
                                                                        Card]
       :AFTER
       '((:LAST (LET ((Card (NC.CoerceToCard Window)))
                     (NCLOG.LogEvent 'RecomputeBrowser.End (fetch (Card NoteFile) of Card)
                            Card]

[XCL:REINSTALL-ADVICE 'NC.RelayoutBrowserCard :BEFORE '[(:LAST (LET ((Card (NC.CoerceToCard Window)))
                                                                   (NCLOG.LogEvent 'RelayoutBrowser.Begin
                                                                          (fetch (Card NoteFile) of Card)
                                                                          Card]
       :AFTER
       '((:LAST (LET ((Card (NC.CoerceToCard Window)))
                     (NCLOG.LogEvent 'RelayoutBrowser.End (fetch (Card NoteFile) of Card)
                            Card]

(READVISE NC.UpdateBrowserCard NC.RelayoutBrowserCard)

[XCL:REINSTALL-ADVICE 'NC.GetNewCard :AFTER '((:LAST (NCLOG.LogEvent 'GetNewCard NoteFile (LIST !VALUE Type]

(READVISE NC.GetNewCard)
```

;;; Other operations.

```
[XCL:REINSTALL-ADVICE 'NC.AddCardType :BEFORE '((:LAST (NCLOG.LogEvent 'AddCardType NIL TypeName]

(READVISE NC.AddCardType)

(PUTPROPS NCLOGGER COPYRIGHT ("Xerox Corporation" 1987 1988))
```

FUNCTION INDEX

ADVICE INDEX

VARIABLE INDEX

RECORD INDEX

PROPERTY INDEX