File created: 5-Nov-2020 20:01:49 **{DSK}<users>arunwelch>skydrive>documents>unix>lisp>lde>notecards>system>NCREPAIR.;5**

previous date: 9-Jan-94 18:58:25 {DSK}<users>arunwelch>skydrive>documents>unix>lisp>lde>notecards>system>NCREPAIR.;4

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ NCREPAIRCOMS
       [
```

;;; Scavenger mechanisms

```
        (GLOBALVARS NC.TitlesIdentifier NC.PropsIdentifier NC.LinksIdentifier NC.ItemIdentifier
               NC.CardInspectorAttachedMenuFont NC.IdentifierAtoms NC.LightShade NC.IndexArrayOffsets
               NC.ScavengerInteractionWinRegion NC.ScavengerCriticalOffsets NC.OffScreenPosition
               NC.CardsPerMenuLimit NC.ScavengerAttachedMenuFont NC.NumCharsOfTitleToShow)
        (INITVARS (NC.IndexArrayOffsets '(3 1 2 4))
               (NC.ScavengerCriticalOffsets '(1 2))
               (NC.CardInspectorAttachedMenuFont (FONTCREATE 'HELVETICA 12 'BOLD))
               (NC.LightShade 4096)
               (NC.OffScreenPosition (CONS 1500 1500))
               (NC.ScavengerInteractionWinRegion (CREATEREGION (fetch (POSITION XCOORD)
                                                                  of NC.OffScreenPosition)
                                                        (fetch (POSITION YCOORD)
                                                               of NC.OffScreenPosition)
                                                        400 250))
               (NC.CardsPerMenuLimit 200)
               (NC.ScavengerAttachedMenuFont (FONTCREATE 'HELVETICA 12 'BOLD))
               (NC.NumCharsOfTitleToShow 10))
```

;;; Phase 3 scavenger stuff.

```
        (FNS NC.ScavengeDatabaseFile)
```

;;; Top level functions.

```
        (FNS NC.ScavengerPhase1 NC.ScavengerCleanup NC.CheckUnknownCardTypes NC.RepositionWindowIfNeeded
             NC.MessageWinAttachedMenuWhenSelectedFn NC.MessageWinCloseFn)
```

;;; Functions for getting the scavenger info for all valid card parts and reading things in the data area robustly.

```
        (FNS NC.GetScavengerInfo NC.RobustReadCardPart NC.CheckForValidSubstance NC.AtEndOfItemP
             NC.RobustReadString NC.RobustReadList NC.RobustReadLinks NC.RobustReadAtom NC.RobustReadRegion
             NC.RobustGetSubstance)
```

;;; Functions for accessing scavenger info for a card.

```
        (FNS NC.SetScavengerInfo NC.SetScavengerTitleInfo NC.SetScavengerMainDataInfo NC.SetScavengerLinksInfo
             NC.SetScavengerPropListInfo NC.FetchScavengerInfo NC.FetchScavengerTitleInfo
             NC.FetchScavengerMainDataInfo NC.FetchScavengerLinksInfo NC.FetchScavengerPropListInfo
             NC.FetchTypeFromScavengerInfo NC.FetchTitleFromScavengerInfo)
```

;;; Functions for building bad cards list based on the index array and scavenger array.

```
        (FNS NC.BuildBadCardsList NC.WorthlessCardP NC.CheckIndexLocs NC.CheckIndexLoc
             NC.CheckForLocBeyondCheckptPtr NC.EncodeCardProblems)
```

;;; Functions for interacting with the user.

```
        (FNS NC.BuildCardInspectorMenu NC.CardInspectorCloseFn NC.CardInspectorRepaintFn
             NC.CardInspectorMenuWhenSelectedFn NC.CardInspectorAttachedMenuWhenSelectedFn
             NC.BuildCardPartsInspector NC.CardPartsAttachedMenuWhenSelectedFn NC.BuildTitlesInspectorMenu
             NC.BuildSubstancesInspectorMenu NC.BuildLinksInspectorMenu NC.BuildPropListsInspectorMenu
             NC.CardPartsMenusWhenSelectedFn NC.CardTitleVersionInspector NC.CardSubstanceVersionInspector
             NC.CardLinksVersionInspector NC.CardPropListVersionInspector NC.CloseInspectorWindows)
```

;;; Miscellaneous.

```
        (FNS NC.CheckForBadLinksAndTitlesAndPropLists NC.UndeletableCardP NC.ShaveTitleString
             NC.SortCardsBySlotNums)
```

;;; Possible reasons for bad card parts.

```
        (INITPROPS (BADMAINDATA (ReasonString "improper main card data."))
               (BADLINKS (ReasonString "improper links data."))
               (BADTITLE (ReasonString "improper title data."))
               (BADPROPLIST (ReasonString "improper prop list data."))
               (MAINDATAPASTCHKPT (ReasonString "main card data beyond chkpt pointer."))
               (LINKSPASTCHKPT (ReasonString "links beyond chkpt pointer."))
               (TITLEPASTCHKPT (ReasonString "title beyond chkpt pointer."))
               (PROPLISTPASTCHKPT (ReasonString "prop list beyond chkpt pointer."))
               (UNKNOWNCARDTYPE (ReasonString "card type definition not loaded.")))
        (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              NCREPAIR)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA])
```

;;; Scavenger mechanisms

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.TitlesIdentifier NC.PropsIdentifier NC.LinksIdentifier NC.ItemIdentifier
       NC.CardInspectorAttachedMenuFont NC.IdentifierAtoms NC.LightShade NC.IndexArrayOffsets
       NC.ScavengerInteractionWinRegion NC.ScavengerCriticalOffsets NC.OffScreenPosition NC.CardsPerMenuLimit
       NC.ScavengerAttachedMenuFont NC.NumCharsOfTitleToShow)
)

(RPAQ? NC.IndexArrayOffsets '(3 1 2 4))

(RPAQ? NC.ScavengerCriticalOffsets '(1 2))

(RPAQ? NC.CardInspectorAttachedMenuFont (FONTCREATE 'HELVETICA 12 'BOLD))

(RPAQ? NC.LightShade 4096)

(RPAQ? NC.OffScreenPosition (CONS 1500 1500))

(RPAQ? NC.ScavengerInteractionWinRegion (CREATEREGION (fetch (POSITION XCOORD)
                                                             of NC.OffScreenPosition)
                                                (fetch (POSITION YCOORD)
                                                       of NC.OffScreenPosition)
                                                400 250))

(RPAQ? NC.CardsPerMenuLimit 200)

(RPAQ? NC.ScavengerAttachedMenuFont (FONTCREATE 'HELVETICA 12 'BOLD))

(RPAQ? NC.NumCharsOfTitleToShow 10)
```

;;; Phase 3 scavenger stuff.

```
(DEFINEQ
```

(NC.ScavengeDatabaseFile

```
  [LAMBDA (NoteFileOrFileName BadLinkLabelsFlg ListOfBoxesToReconstruct
                 ListOfCardsNeedingGlobalLinksReconstructed InterestedWindow)
                                                                        ; Edited  3-Dec-87 18:59 by rht:
```

(* Scavenge the database FileName. Essentially throw away all of the information about From and ToLinks and recreate them by retrieving the link information from the substance of each card and from the list of global links from the card.)

(* * rht 8/9/84%: Now calls NC.OpenDatabaseFile to do the file open.)

(* * rht 7/17/85%: Changed so can take a stream argument. Also handles link labels.
If BadLinkLabelsFlg is non-nil, then don't try to read current link labels.
Just rebuild them from what's out there. Otherwise, only rebuild if find new any new ones.)

(* * fgh 22-Jul-85 Takes a list of bad file box cards and reconstructs the file boxes from the From pointer lists of all the cards in the NoteFile.)

(* * fgh 30-Jul-85 Takes a list of cards with bad global links and reconstructs the global links list from the From pointer lists of all the cards in the NoteFile.)

(* * rht 11/23/85%: Updated to handle new notefile and card object formats.)

(* * rht 12/1/85%: Now calls NC.GetMainCardData and NC.GetLinks instead of NC.GetNoteCard.)

(* * rht 12/19/85%: Massive overhaul for sake of speed. Should be wizzier now.)

(* * fgh |2/4/86| Now works on open NFs. No need to error check since this function should always be called from earlier phases of the inspect & repaier.)

(* * fgh |5/21/86| Fixed bug in handling of global links.)

```
(* * rht 7/16/86%: Added InterestedWindow arg.)

(* * rht 7/16/86%: Now calls NC.PutLinks passing UseOldDatesFlg.)

(* * rht 9/5/86%: Now checks that link is valid before passing it to NC.DelReferencesToCard.)

(* * rht 10/29/86%: Now closes prompt win at end if no filing info to tell.)

(* * rht 12/10/86%: No longer calls NC.GetLinks for cards appearing on the ListOfCardsNeedingGlobalLinksReconstructed
list.)

(* * rht 1/19/87%: Now uses NC.CardNeedsFilingP to check for unfiled cards at the end.)

(* * rht 1/22/87%: Added yet another BLOCK call.)

(* * rht 2/18/87%: Added SPAWN.MOUSE call.)

(* * rht 3/17/87%: Changed call to OPENP to NC.OpenDatabaseFile to recover lost change from KIRKPATCH033.
Added RESETLST to close notefile in case we bomb out.)

(* * rht 4/3/87%: Call to NC.OpenNoteFile had extra NIL in it.)

(* * pmi 5/20/87%: Removed HashArray argument in calls to NC.OpenNoteFile.)

(* * rht 6/9/87%: Now calls NC.PutMainCardData if did a delete of bad link icons.)

(* * pmi 7/1/87%: Replaced (SPAWN.MOUSE) with (ALLOW.BUTTON.EVENTS)%.
Added call to NC.ForceDatabaseClose at end of function -
the RESETSAVE isn't working properly when compiled -
very strange!)

(* * rg&pmi 7/15/87%: removed extra NC.ForceDatabaseClose from previous fix.)

(PROG (NoteFile FileName)
      (ALLOW.BUTTON.EVENTS)

      (* * First, take care of checking stream's validity, etc.)

      (SETQ FileName (if (type? NoteFile NoteFileOrFileName)
                         then (SETQ NoteFile NoteFileOrFileName)
                              (fetch (NoteFile FullFileName) of NoteFileOrFileName)
                       else NoteFileOrFileName))              (* Try to open notefile.)
      (if (NULL (NC.NoteFileOpenP FileName))
          then (if (NULL (SETQ NoteFile (NC.OpenNoteFile FileName T NIL NIL NIL NIL T T InterestedWindow)))
                   then (NC.PrintMsg InterestedWindow NIL "Couldn't open " FileName "." (CHARACTER 13)
                              "Repair aborted.")
                        (RETURN NIL)))
      (LET
       (CardTotal NoteCardNumber OldLinkLabels DiscoveredLinkLabels ReconstructLinks ReconstructGlobalLinks
              ToBeFiledCards)
       (RESETLST
           [RESETSAVE NIL `(NC.ForceDatabaseClose ,NoteFile]

      (* * If link labels aren't screwed up, then read them in.)

           (OR BadLinkLabelsFlg (SETQ OldLinkLabels (NC.RetrieveLinkLabels NoteFile T)))

      (* * Mark every card that needs its global links or substance reconstructed so we don't have to search the lists so much.)

           (for Card in ListOfCardsNeedingGlobalLinksReconstructed do (NC.SetUserDataProp Card
                                                                                          '
                                                                                           NeedsGlobalLinksReconstructedFlg
                                                                                          T))
           (for Box in ListOfBoxesToReconstruct do (NC.SetUserDataProp Box 'NeedsReconstructingFlg T))

      (* Read through all NoteCard substances to find actual pointers.
      Use this to create the To Links list. The list collection function checks to make sure each link is valid.)

           (SETQ CardTotal (SUB1 (fetch (NoteFile NextIndexNum)
                                        NoteFile)))
           (NC.PrintMsg InterestedWindow T "Rebuilding notefile links." (CHARACTER 13)
                  "Collecting Links for item " 1 " out of " CardTotal ".")
           (SETQ NoteCardNumber 0)
           [NC.MapCards
            NoteFile
            (FUNCTION (LAMBDA (Card)
                        (BLOCK)
                        (SETQ NoteCardNumber (ADD1 NoteCardNumber))
                        (AND (ZEROP (REMAINDER NoteCardNumber 10))
                             (NC.PrintMsg InterestedWindow T "Rebuilding notefile links." (CHARACTER 13)
                                    "Collecting Links for item " NoteCardNumber " out of " CardTotal "."))
                                                          (* Get global links unless links are unreadable.)
                        (if (NOT (NC.FetchUserDataProp Card 'NeedsGlobalLinksReconstructedFlg))
                            then (NC.GetLinks Card))
                        (if (NC.FetchUserDataProp Card 'NeedsReconstructingFlg)
                            then
```

```
(* Card substance and links will be reconstructed so no need to try to read substance.)

                    (if (NOT (NC.FetchUserDataProp Card 'NeedsGlobalLinksReconstructedFlg))
                        then (NC.SetUserDataProp Card 'ScavengerToLinks (NC.FetchGlobalLinks
                                                                          Card))
                             (NC.SetUserDataProp Card 'ScavengerGlobalLinks (NC.FetchGlobalLinks
                                                                              Card)))
                    (NC.DeactivateCard Card T)
               else
               (NC.GetMainCardData Card)
               (NC.ActivateCard Card)
               (if (EQ (NC.FetchStatus Card)
                       'ACTIVE)
                   then                                  (* Collect links having active destinations.
                                                         Delete the others.)
                        [NC.SetUserDataProp
                         Card
                         'ScavengerToLinks
                         (NCONC (for Link in (CAR (NC.CollectReferences Card)) eachtime (BLOCK)
                                   when (if (EQ (NC.FetchStatus (fetch (Link DestinationCard)
                                                                   of Link))
                                                'ACTIVE)
                                            else (AND (type? Link Link)
                                                      (NC.DelReferencesToCard Card Link))
                                                NIL)
                                   collect Link)
                                (if (NC.FetchUserDataProp Card 'NeedsGlobalLinksReconstructedFlg)
                                  else (NC.SetUserDataProp Card 'ScavengerGlobalLinks (
                                                                                  NC.FetchGlobalLinks
                                                                                    Card))
                                       (NC.FetchGlobalLinks Card]
                        (if (NC.FetchUserDataProp Card 'NeedsGlobalLinksReconstructedFlg)
                          else (NC.SetUserDataProp Card 'ScavengerGlobalLinks (NC.FetchGlobalLinks
                                                                                Card)))

(* If there are file boxes to be reconstructed, then look thru the From links to see if this card was filed in one of the
to-be-reconstructed boxes)

                    (AND ListOfBoxesToReconstruct
                         (for Link in (NC.FetchFromLinks Card) eachtime (BLOCK)
                            when (AND (NC.ChildLinkP Link)
                                      (NC.FetchUserDataProp (fetch (Link SourceCard)
                                                               of Link)
                                            'NeedsReconstructingFlg))
                            do (push ReconstructLinks Link)))

(* If there are global links to be reconstructed, then look thru the From links to see if this card had a global link from a card
whose global links need reconstructing.)

                    (AND ListOfCardsNeedingGlobalLinksReconstructed
                         (for Link in (NC.FetchFromLinks Card) eachtime (BLOCK)
                            when (AND (NC.GlobalLinkP Link)
                                      (NC.FetchUserDataProp (fetch (Link SourceCard)
                                                               of Link)
                                            'NeedsGlobalLinksReconstructedFlg))
                            do (push ReconstructGlobalLinks Link)))
                                                         (* If we deleted some bad link icons above, then need to write
                                                         down substance.)
                    (if (NC.CardDirtyP Card)
                        then (NC.PutMainCardData Card))
                    (NC.DeactivateCard Card T]

(* * Reconstruct any cards as requested)

     (for BoxToReconstruct in ListOfBoxesToReconstruct eachtime (BLOCK)
        do                                               (* Make a new file box using the given card.)
           (NC.MakeNoteCard 'FileBox NoteFile "Untitled: Reconstructed during repair" T NIL
                  BoxToReconstruct)

(* File cards whose from links indicate that they used to be filed in this file box.
Also add these new links to collected ToLinks.)

          [NC.SetUserDataProp BoxToReconstruct 'ScavengerToLinks
                 (APPEND (NC.FetchUserDataProp BoxToReconstruct 'ScavengerToLinks)
                         (for Link in ReconstructLinks eachtime (BLOCK)
                            when (NC.SameCardP BoxToReconstruct (fetch (Link SourceCard) of Link))
                            collect (NC.MakeChildLink (fetch (Link DestinationCard) of Link)
                                          BoxToReconstruct NIL]
                                                         (* Put the card away)
          (NC.PutMainCardData BoxToReconstruct)
          (NC.DeactivateCard BoxToReconstruct T))

(* * Reconstruct any global link lists as required)

     [for Link in ReconstructGlobalLinks bind ThisCardsToLinks ThisCardsGlobalLinks SourceCard
        eachtime (BLOCK) do (SETQ SourceCard (fetch (Link SourceCard) of Link))
                                                         (* Add it to the GlobalLinks list for its source card unless it's
                                                         already there.)
```

```
                                    (if (for GlobalLink in (SETQ ThisCardsGlobalLinks (NC.FetchUserDataProp
                                                                                       SourceCard
                                                                                       'ScavengerGlobalLinks))
                                           eachtime (BLOCK) never (NC.SameLinkP Link GlobalLink))
                                        then (NC.SetUserDataProp SourceCard 'ScavengerGlobalLinks
                                                    (CONS Link ThisCardsGlobalLinks)))
                                                    (* Add it to the source card's ToLinks list unless it's already
                                                    there)
                                    (if (for ToLink in (SETQ ThisCardsToLinks (NC.FetchUserDataProp SourceCard
                                                                                       'ScavengerToLinks))
                                           eachtime (BLOCK) never (NC.SameLinkP Link ToLink))
                                        then (NC.SetUserDataProp SourceCard 'ScavengerToLinks (CONS Link
                                                                                                ThisCardsToLinks
                                                                                                    ]
```

(* * Compute the From Links list by "inverting" the To Links list)

```
    (NC.PrintMsg InterestedWindow T "Repairing NoteFile." (CHARACTER 13)
           "Inverting links for item " 1 " out of " CardTotal ".")
    (SETQ NoteCardNumber 0)
    [NC.MapCards NoteFile
           (FUNCTION (LAMBDA (Card)
                        (SETQ NoteCardNumber (ADD1 NoteCardNumber))
                        (AND (ZEROP (REMAINDER NoteCardNumber 100))
                             (NC.PrintMsg InterestedWindow T "Repairing NoteFile." (CHARACTER 13)
                                    "Inverting links for item " NoteCardNumber " out of " CardTotal
                                    "."))
                        (if (EQ (NC.FetchStatus Card)
                                'ACTIVE)
                            then (for Link in (NC.FetchUserDataProp Card 'ScavengerToLinks)
                                    bind DestinationCard LinkLabel eachtime (BLOCK)
                                    do                  (* Add this ToLink as a FromLink for the link's destination card.)
                                       [NC.SetUserDataProp (SETQ DestinationCard (fetch (Link
                                                                                          DestinationCard
                                                                                          )
                                                                                    of Link))
                                             'ScavengerFromLinks
                                             (CONS Link (NC.FetchUserDataProp DestinationCard
                                                                'ScavengerFromLinks]
                                                    (* Accumulate the link labels into a list.)
                                       (if (NOT (FMEMB (SETQ LinkLabel (fetch (Link Label) of Link))
                                                       DiscoveredLinkLabels))
                                           then (push DiscoveredLinkLabels LinkLabel]
```

(* * Reset all of the To and From Links lists in the database)

```
    (NC.PrintMsg InterestedWindow T "Repairing NoteFile." (CHARACTER 13)
           "Rewriting links for item " 1 " out of " CardTotal ".")
    (SETQ NoteCardNumber 0)
    [NC.MapCards NoteFile (FUNCTION (LAMBDA (Card)
                                       (BLOCK)
                                       (SETQ NoteCardNumber (ADD1 NoteCardNumber))
                                       (AND (ZEROP (REMAINDER NoteCardNumber 10))
                                            (NC.PrintMsg InterestedWindow T "Repairing NoteFile."
                                                   (CHARACTER 13)
                                                   "Rewriting links for item " NoteCardNumber " out of
" CardTotal "."))
                                       (if (EQ (NC.FetchStatus Card)
                                               'ACTIVE)
                                           then (NC.SetGlobalLinks Card (NC.FetchUserDataProp
                                                                            Card
                                                                            'ScavengerGlobalLinks))
                                                (NC.SetToLinks Card (NC.FetchUserDataProp Card
                                                                      'ScavengerToLinks))
                                                (NC.SetFromLinks Card (NC.FetchUserDataProp
                                                                        Card
                                                                        'ScavengerFromLinks))
                                                    (* Check whether this card isn't filed anywhere.)
                                                (if (NC.CardNeedsFilingP Card)
                                                    then (push ToBeFiledCards Card))
                                                (NC.PutLinks Card T))
                                                    (* Clean any junk off the card.)
                                       (NC.DeactivateCard Card T)
                                       (NC.SetUserDataPropList Card NIL]
```

(* * File any unfiled cards in the ToBeFiled box.)

```
    (if ToBeFiledCards
        then (NC.PrintMsg InterestedWindow T "Filing " (LENGTH ToBeFiledCards)
                     " cards in ToBeFiled box ...")
             (NCP.FileCards ToBeFiledCards (fetch (NoteFile ToBeFiledCard) of NoteFile)))
                                                    (* Rewrite link labels if we've found any new ones.)
    (if (LDIFFERENCE DiscoveredLinkLabels OldLinkLabels)
        then (NC.StoreLinkLabels NoteFile (UNION DiscoveredLinkLabels OldLinkLabels)))
                                                    (* Clean up and get out.)
    (NC.CheckpointDatabase NoteFile T)
    (NC.PrintMsg InterestedWindow T "Repair Completed for " (FULLNAME FileName)
```

```
                    ".")
               (if ToBeFiledCards
                   then (NC.PrintMsg InterestedWindow NIL "Filed " (LENGTH ToBeFiledCards)
                                " cards in ToBeFiled box.")
                 else (NC.ClearMsg InterestedWindow T)))])

)
```

;;; Top level functions.

```
(DEFINEQ
```

(NC.ScavengerPhase1

```
  [LAMBDA (FileNameOrNoteFile ReadSubstancesFlg ScavengerInteractionWin RecheckBadCardsFlg InterestedWindow)
                                                                          ; Edited  8-Dec-88 16:52 by krivacic
```

;;; This is the first phase of the scavenger. Runs over entire data portion of the notefile, accumulating pointers to healthy parts of cards. Then runs over index array asking user what to do with bad or outdated pointers. If ReadSubstancesFlg is non-nil then it'll do robust gets of the substances. This slows things down, but makes checking more comprehensive.

;;; rht 12/7/85: Updated to handle new notefile and card object formats.

;;; rht 3/22/86: No longer hangs bad cards off proplist of Reason atoms. Uses local var ReasonsHashArray instead. NC.ScavengerPhase1 no longer hanging on completion of phase 3

;;; rht 7/7/86: Now passes non-nil Don'tCheckOperationInProgress flg to NC.OpenDatabaseFile.

;;; rht 7/16/86: Added InterestedWindow arg.

;;; rht 9/16/86: Changed call to NC.OpenDatabaseFile so that it won't try to get special cards.

;;; rht 10/31/86: Changed call from NC.OpenDatabaseFile to NC.OpenNoteFile. Now returns non-nil if successful.

;;; rht 11/13/86: Now passes non-nil Don'tCreateFlg to NC.OpenNoteFile.

;;; rht 2/18/87: Added SPAWN.MOUSE call.

;;; rht 3/14/87: Now calls new function NC.TotalCardsInNoteFile to compute the total number of cards.

;;; pmi 5/20/87: Removed HashArray argument in calls to NC.OpenNoteFile.

;;; pmi 7/7/87: Replaced (SPAWN.MOUSE) with (ALLOW.BUTTON.EVENTS). Added InspectAndRepairFinishedEvent argument to be used to signal the end of Scavenging. Added Protection wrapper around the body of this function so that this notefile can not be accessed while I&R is in progress.

;;; rg&pmi 7/15/87 removed protection wrapper

;;; rg 8/11/87 protection wrapper moved back down to this level, where we know we have a real NoteFile to protect

;;; pmi&rg 8/19/87: fixes problem where this was not returning a notefile if successful.

;;; rg 9/16/87 now checks FirstTimeFlg (yuck!) before doing AWAIT.EVENT

```
    (DECLARE (GLOBALVARS NC.ScavengerAttachedMenuFont NC.ScavengerInteractionWinRegion))
    (PROG (FileName NoteFile NoteFileStream)
```

;;; First, take care of opening notefile if needed.

```
          (ALLOW.BUTTON.EVENTS)
          (if (AND (type? NoteFile FileNameOrNoteFile)
                   (SETQ NoteFileStream (fetch (NoteFile Stream) of FileNameOrNoteFile))
                   (OPENP NoteFileStream))
              then                                          ; This notefile is already open For when we do recursive call.
                   (SETQ NoteFile FileNameOrNoteFile)
            else                                            ; Get file name and open the file if conditions are okay.
                 (SETQ FileName FileNameOrNoteFile)
                 (AND (NULL FileName)
                      (NULL (SETQ FileName (NC.DatabaseFileName "What is the name of the NoteFile to
                                                Inspect&Repair? " NIL T NIL NIL InterestedWindow)))
                      (RETURN NIL))
                 (AND (NULL (SETQ NoteFile
                                  (NC.OpenNoteFile FileName T T NIL NIL NIL T T InterestedWindow NIL NIL NIL NIL T)))
                      (NC.PrintMsg InterestedWindow NIL "Couldn't open " FileName "." (CHARACTER 13)
                             "Repair aborted.")
                      (RETURN NIL))
                 (NC.ClearMsg InterestedWindow T))
          (RETURN
           (NC.ProtectedNoteFileOperation
            NoteFile "Inspect&Repair" InterestedWindow
            (PROG (UnknownCardTypesList ReasonsList ReasonsHashArray CardsToDelete Menu MenuItems LinkLabelsNews
                        CardTotal BadNewsList BadBoxes ExtraBadNews FirstTimeFlg InspectorPendingEvent
                        NoteFileMenu NoteFileOpsMenuItem CanDoPhase3Flg InspectAndRepairFinishedEvent)
                  (SETQ CardTotal (NC.TotalCardsInNoteFile NoteFile))
                                                            ; Build a window for talking to the user if one wasn't passed in.
                  (if (WINDOWP ScavengerInteractionWin)
                      then (CLEARW ScavengerInteractionWin)
                    else (SETQ ScavengerInteractionWin (CREATEW NC.ScavengerInteractionWinRegion
```

```
                                                    "Inspect&Repair Interaction Window" NIL T))
                                                        ; This flg indicates that we're in the first call to the scavenger.
            (SETQ FirstTimeFlg T)
            (WINDOWADDPROP ScavengerInteractionWin 'CLOSEFN (FUNCTION NC.MessageWinCloseFn)
                   T))                                  ; Stash InterestedWindow for calls to phase 3 under menu
                                                        ; whenselected fn.
      [if FirstTimeFlg
          then (SETQ InspectAndRepairFinishedEvent (CREATE.EVENT 'InspectAndRepairInProgress]
      (OR (WINDOWPROP ScavengerInteractionWin 'INTERESTEDWINDOW)
          (WINDOWPROP ScavengerInteractionWin 'INTERESTEDWINDOW InterestedWindow))
                                                        ; Get all relevant info about the data area of the notefile onto the
                                                        ; cards' prop lists.
      (if (OR (NOT FirstTimeFlg)
              (EQ (NC.GetScavengerInfo NoteFile ReadSubstancesFlg ScavengerInteractionWin
                         InterestedWindow)
                  'SUCCESS))
          then (WINDOWPROP ScavengerInteractionWin 'NOTEFILE NoteFile)
               (WINDOWPROP ScavengerInteractionWin 'CARDTOTAL CardTotal)
        else                                            (* Something's wrong. Couldn't get scavenger info.
                                                        Bail out.)
               (NC.ScavengerCleanup ScavengerInteractionWin InterestedWindow)
               (CLOSEW ScavengerInteractionWin)
               (RETURN NIL))

;; Check the list of card types that are undefined to see if user has loaded a definition since the last time we checked.  If he has, then
;; go try to read the substance card parts for those newly defined card types.

      (NC.CheckUnknownCardTypes NoteFile ReadSubstancesFlg ScavengerInteractionWin)

;; Next step is to run down the in-core index and find those cards having pointers to bad items in the data area.  We also need to look
;; for undefined card types and for pointers past the checkpoint pointer.  However, we can reuse old bad news list if nothing has
;; changed.

      [if (OR FirstTimeFlg RecheckBadCardsFlg (WINDOWPROP ScavengerInteractionWin 'NEEDCHECKPOINT))
          then (WINDOWPROP ScavengerInteractionWin 'ORIGINALBADNEWSLIST (SETQ BadNewsList
                                                                                (NC.BuildBadCardsList NoteFile

                                                                                      ScavengerInteractionWin
                                                                                        FirstTimeFlg
                                                                                        InterestedWindow)))
        else (SETQ BadNewsList (WINDOWPROP ScavengerInteractionWin 'ORIGINALBADNEWSLIST]

;; Okay, now all the troublesome IDs and the reasons for their troubles are recorded in BadNewsList.  We next need to get directives
;; from the user as to what to do for each problem card.

      (NC.RepositionWindowIfNeeded ScavengerInteractionWin)

;; If there's bad news for link labels then take off list and store in a local var.

      (if (SETQ LinkLabelsNews (for BadCardEntry in BadNewsList
                                  bind (LinkLabelsCard _ (fetch (NoteFile LinkLabelsCard) of NoteFile))
                                  eachtime (BLOCK) when (NC.SameCardP LinkLabelsCard (CAR
                                                                                           BadCardEntry
                                                                                           ))

                                  do (RETURN BadCardEntry)))
          then (SETQ BadNewsList (DREMOVE LinkLabelsNews BadNewsList)))

;; Accumulate general statistics on the problems.

      (SETQ ReasonsHashArray (HASHARRAY 100))
      [for BadNews in BadNewsList bind Card Type eachtime (BLOCK)
         unless (FMEMB (CADR BadNews)
                       '(DELETED FREE))
         do (SETQ Card (CAR BadNews))
            (for Reason in (CDDDR BadNews) eachtime (BLOCK)
               do (PUTHASH Reason (CONS Card (GETHASH Reason ReasonsHashArray))
                           ReasonsHashArray)
                  (if (NOT (FMEMB Reason ReasonsList))
                      then (SETQ ReasonsList (CONS Reason ReasonsList)))
                  (if (EQ Reason 'UNKNOWNCARDTYPE)
                      then                              ; Accumulate the list of unknown card types for nondeleted
                                                        ; cards.
                           (if (NOT (FMEMB (SETQ Type (NC.FetchTypeFromScavengerInfo Card))
                                           UnknownCardTypesList))
                               then (push UnknownCardTypesList Type]

;; Build the menu entries that we know will be present regardless of which cards are bad.

      [SETQ MenuItems '((Abort 'Abort "Quit this Inspect&Repair operation.")
                        (|Recheck Bad Cards| '|Recheck Bad Cards| "Recompute bad cards list.  Useful
                               if you've just loaded some card type definitions.")
                        (Inspect% Cards 'Inspect% Cards "Bring up the cards inspector menu."
                               (SUBITEMS (|Include Deleted Cards| '|Include Deleted Cards| "Throw in
                                                 deleted cards as well."]

;; Print a message if news on link labels is worse than just past checkpoint.

      (if [AND LinkLabelsNews (NOT (EQUAL (CDDDR LinkLabelsNews)
                                          '(MAINDATAPASTCHKPT]
          then (push ExtraBadNews LinkLabelsNews)
               (NC.PrintMsg ScavengerInteractionWin NIL "The link types are bad." (CHARACTER 13)
                      "If you don't back them up to a previous version, then phase 3 of
                      Inspect&Repair will rebuild them."
                      (CHARACTER 13))
```

```
                    (WINDOWPROP ScavengerInteractionWin 'NEEDLINKSCAVENGE T))
```

;; Collect any fileboxes that have bad substances.

```
      (if (SETQ BadBoxes (LET (Boxes)
                              (SETQ BadNewsList
                               (for News in BadNewsList bind Box eachtime (BLOCK)
                                  unless (if (AND (EQ (NC.FetchTypeFromScavengerInfo (SETQ Box
                                                                                      (CAR News)))
                                                      'FileBox)
                                                  (FMEMB 'BADMAINDATA (CDDDR News)))
                                             then (push Boxes Box)
                                                  ; If nothing else is wrong with those boxes, then take off bad
                                                  ; news list.
                                                  (EQ (LENGTH (CDDDR News))
                                                      1)
                                           else NIL)
                                  collect News))
                            Boxes))
          then (NC.PrintMsg ScavengerInteractionWin NIL "Fileboxes " (for Box in BadBoxes
                                                                        collect (
                                                                          NC.FetchTitleFromScavengerInfo
                                                                            Box))
                            " have bad substance(s)."
                            (CHARACTER 13)
                            "If you don't delete them or back up to a previous version, then phase 3 of
                            Inspect&Repair will rebuild their contents."
                            (CHARACTER 13))
                    (WINDOWPROP ScavengerInteractionWin 'NEEDLINKSCAVENGE T))
```

;; Print out totals of active and deleted cards.

```
      (LET ((ActivesTotal 0)
            (DeletedsTotal 0))
           [NC.MapCards NoteFile (FUNCTION (LAMBDA (Card)
                                           (SELECTQ (NC.FetchStatus Card)
                                                (ACTIVE (SETQ ActivesTotal (ADD1 ActivesTotal)))
                                                (DELETED (SETQ DeletedsTotal (ADD1 DeletedsTotal))
)
                                                NIL]
           (NC.PrintMsg ScavengerInteractionWin NIL "Out of " CardTotal " cards:" (CHARACTER 13)
                  "there are " ActivesTotal " active cards and " DeletedsTotal " deleted cards."
                  (CHARACTER 13)))
      (if ReasonsList
          then                                          ; Print out messages for bad cards.
               (NC.PrintMsg ScavengerInteractionWin NIL "Of the non-deleted ones," (CHARACTER 13))
               [for Reason in ReasonsList eachtime (BLOCK)
                  do (NC.PrintMsg ScavengerInteractionWin NIL (LENGTH (GETHASH Reason
                                                                              ReasonsHashArray))
                                  " have "
                                  (GETPROP Reason 'ReasonString)
                                  (CHARACTER 13))
                     (if (EQ Reason 'UNKNOWNCARDTYPE)
                         then (NC.PrintMsg ScavengerInteractionWin NIL "The unknown types are: "
                                           UnknownCardTypesList "." (CHARACTER 13]
        else (NC.PrintMsg ScavengerInteractionWin NIL "All non-deleted cards look okay." (CHARACTER
                                                                                            13)))
```

;; Only allow continuation to phase 3 of repair, links rebuilding, if there's no bad news that can't be fixed. We can fix bad proplist, titles
;; or links. We can also fix even bad substances if they're for fileboxes.

```
      (if [for News in BadNewsList eachtime (BLOCK) unless (FMEMB (CADR News)
                                                                  '(DELETED FREE))
            unless (EQ (NC.FetchTypeFromScavengerInfo (CAR News))
                       'FileBox)
            never (INTERSECTION (CDDDR News)
                                '(BADMAINDATA UNKNOWNCARDTYPE]
          then                                          ; Add the appropriate menu items.
               (if (NOT (WINDOWPROP ScavengerInteractionWin 'NEEDLINKSCAVENGE))
                   then (SETQ MenuItems (CONS '(End% Inspect&Repair 'End% Inspect&Repair "This exits
                                                                     Inspect&Repair normally, closing the
                                                                     notefile.")
                                              MenuItems)))
               (SETQ CanDoPhase3Flg T)
               (SETQ MenuItems (CONS '(Continue% Repair 'Continue% Repair "Complete Inspect&Repair
                                                                          by rebuilding the links.")
                                     MenuItems)))
```

;; Make sure a checkpoint will happen before continuing to phase 3 if there are any card parts beyond the checkpt pointer.

```
      (if (INTERSECTION ReasonsList '(MAINDATAPASTCHKPT LINKSPASTCHKPT TITLEPASTCHKPT
                                                        PROPLISTPASTCHKPT))
          then (AND CanDoPhase3Flg (NC.PrintMsg ScavengerInteractionWin NIL "'Continue Repair' will
                                                integrate any card part versions beyond chkpt pointer."
                                                (CHARACTER 13)))
               (WINDOWPROP ScavengerInteractionWin 'NEEDCHECKPOINT T))
```

;; Ugliness! Have to cache all these vars on window so that attached menu's whenselectedfn will be able to grab them.

```
      (WINDOWPROP ScavengerInteractionWin 'BADNEWSLIST BadNewsList)
      (WINDOWPROP ScavengerInteractionWin 'EXTRABADNEWS ExtraBadNews)
      (WINDOWPROP ScavengerInteractionWin 'LINKSLABELSNEWS LinkLabelsNews)
```

```
                (WINDOWPROP ScavengerInteractionWin 'BADBOXES BadBoxes)
                (if FirstTimeFlg
                    then (WINDOWPROP ScavengerInteractionWin 'FINISHEDEVENT InspectAndRepairFinishedEvent)
                         (WINDOWPROP ScavengerInteractionWin 'LockProcess (THIS.PROCESS)))
                (ATTACHMENU (create MENU
                                    ITEMS _ MenuItems
                                    WHENSELECTEDFN _ (FUNCTION NC.MessageWinAttachedMenuWhenSelectedFn)
                                    MENUFONT _ NC.ScavengerAttachedMenuFont)
                      ScavengerInteractionWin
                      'RIGHT
                      'TOP)
                (if FirstTimeFlg
                    then (AWAIT.EVENT InspectAndRepairFinishedEvent)))
          NoteFile])
```

(NC.ScavengerCleanup

```
[LAMBDA (MessageWin InterestedWindow)                                   (* rht%: "16-Jul-86 18:19")
```

(* * Abort the scavenger by closing the notefile without checkpointing.)

(* * rht 12/8/85%: Updated to handle new notefile and card object formats.)

(* * rht 3/22/86%: Took out NOTIFY.EVENT stuff since NC.ScavengerPhase1 is no longer hanging on completion of phase 3.0)

(* * rht 7/16/86%: Added InterestedWindow arg.)

```
  (LET [(NoteFile (WINDOWPROP MessageWin 'NOTEFILE]
       (if (OPENP (fetch (NoteFile Stream) of NoteFile))
           then (NC.ForceDatabaseClose NoteFile)
                (NC.PrintMsg InterestedWindow T "Repair of " (fetch (NoteFile FullFileName) of NoteFile)
                       " aborted."])
```

(NC.CheckUnknownCardTypes

```
[LAMBDA (NoteFile ReadSubstancesFlg MessageWin)                         (* rht%: "22-Mar-86 18:13")
```

(* * Check whether any of the card types on UnknownCardTypes have suddenly become defined.
If so, then go try to read the substances of cards of that type and update ScavengerArray.
Return the new UnknownCardTypesList.)

(* * rht 10/18/85%: Now fixes the windowprop of MessageWin here.
Also returns the list of newly found types, or NIL if none.)

(* * rht 12/7/85%: Modified to reflect new card and notefile object formats.)

(* * rht 3/22/86%: substance ptrs for bad card types are no longer put on card type's prop list.
Now they're stored in an assoc list hung off a WINDOWPROP.)

```
  (LET ((UnknownCardTypes (WINDOWPROP MessageWin 'UNKNOWNCARDTYPESLIST))
        (UnknownCardTypesSubstancePtrs (WINDOWPROP MessageWin 'UNKNOWNCARDTYPESSUBSTANCEPTRS))
        (Stream (fetch (NoteFile Stream) of NoteFile))
        OkayTypes)
       (for Type in UnknownCardTypes bind (EndPtr _ (GETEOFPTR Stream)) eachtime (BLOCK)
          when (NCP.ValidCardType Type) do (push OkayTypes Type)
                                           (for SubstancePtr in (LISTGET UnknownCardTypesSubstancePtrs Type)
                                              do (SETFILEPTR Stream SubstancePtr)
                                                 (NC.RobustReadCardPart NoteFile ReadSubstancesFlg MessageWin
                                                        EndPtr))
                                           (LISTPUT UnknownCardTypesSubstancePtrs Type NIL))
       (if OkayTypes
           then (WINDOWPROP MessageWin 'UNKNOWNCARDTYPESLIST (LDIFFERENCE UnknownCardTypes OkayTypes)))
      OkayTypes])
```

(NC.RepositionWindowIfNeeded

```
[LAMBDA (Win)                                                           (* rht%: "17-Jul-85 17:04")
```

(* * If window is at NC.OffScreenPosition, then let user move to new location.)

```
  (if (EQUAL (WINDOWPOSITION Win)
             NC.OffScreenPosition)
      then (LET ((Region (WINDOWREGION Win)))
                (MOVEW Win (GETBOXPOSITION (fetch (REGION WIDTH) of Region)
                                           (fetch (REGION HEIGHT) of Region)
                                           NIL NIL NIL "Please position the repair interaction window."])
```

(NC.MessageWinAttachedMenuWhenSelectedFn

```
[LAMBDA (Item Menu MouseKey)                                            (* pmi%: " 7-Jul-87 15:32")
```

(* * Called when selecting from the attached menu to the main message window.)

(* * rht 9/17/85%: Now gets MaxIDNum from WINDOWPROP of MessageWin and uses it for looping through cards.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object fomats.)

(* * rht 3/22/86%: Took out NOTIFY.EVENT stuff since NC.ScavengerPhase1 is no longer hanging on completion of phase 3.0)

(* * rht 7/16/86%: Added NC.AttachPromptWindow calls.)

(* * pmi 7/7/87%: Added InspectAndRepairFinishedEvent as another WINDOWPROP on the MessageWin to use in signalling the end of Scavenging. Also calls NC.SwitchNoteFileLock to take on the process id of the calling process (NC.ScavengerPhase1.))

```
(LET ((MenuWin (WFROMMENU Menu))
      (Operation (CAR Item))
      MessageWin NoteFile InspectorWins BadNewsList ExtraBadNews InspectAndRepairFinishedEvent)
     (SETQ MessageWin (MAINWINDOW MenuWin))
     (SETQ NoteFile (WINDOWPROP MessageWin 'NOTEFILE))
     (SETQ BadNewsList (WINDOWPROP MessageWin 'BADNEWSLIST))
     (SETQ ExtraBadNews (WINDOWPROP MessageWin 'EXTRABADNEWS))
     (SETQ InspectAndRepairFinishedEvent (WINDOWPROP MessageWin 'FINISHEDEVENT))
     [WINDOWPROP MessageWin 'LockProcess (NC.SwitchNoteFileLock NoteFile (WINDOWPROP MessageWin
                                                                          'LockProcess]
     (SELECTQ Operation
          (Abort (if (NC.AskYesOrNo "Do you really want to abort the Inspect & Repair process? " NIL
                          'Yes T (NC.AttachPromptWindow MessageWin)
                          NIL NIL)
                     then                                          (* Bail out.)
                          (WINDOWDELPROP MessageWin 'CLOSEFN (FUNCTION NC.MessageWinCloseFn))
                          (CLOSEW MessageWin)
                          (NC.ScavengerCleanup MessageWin)
                          (NOTIFY.EVENT InspectAndRepairFinishedEvent)))
          (|Recheck Bad Cards|                                     (* Rerun the end of phase1.)
               (DETACHWINDOW MenuWin)
               (CLOSEW MenuWin)
               (NC.ScavengerPhase1 NoteFile NIL MessageWin T (NC.AttachPromptWindow MessageWin)))
          (Inspect% Cards                                          (* Bring up the big card inspector menu.)
               (DETACHWINDOW MenuWin)
               (CLOSEW MenuWin)
               (NC.BuildCardInspectorMenu [NC.MapCards NoteFile [FUNCTION (LAMBDA (Card)
                                                                          Card]
                                                  (FUNCTION (LAMBDA (Card)
                                                            (AND (FMEMB (NC.FetchStatus Card)
                                                                        '(ACTIVE BADPOINTER NIL))
                                                                 (NOT (NC.WorthlessCardP Card]
                          (APPEND ExtraBadNews BadNewsList)
                          NoteFile MessageWin))
          (|Include Deleted Cards|                                 (* Like above except include also the deleted cards.)
               (DETACHWINDOW MenuWin)
               (CLOSEW MenuWin)
               (NC.BuildCardInspectorMenu [NC.MapCards NoteFile [FUNCTION (LAMBDA (Card)
                                                                          Card]
                                                  (FUNCTION (LAMBDA (Card)
                                                            (AND (FMEMB (NC.FetchStatus Card)
                                                                        '(ACTIVE BADPOINTER NIL DELETED))
                                                                 (NOT (NC.WorthlessCardP Card]
                          (APPEND ExtraBadNews BadNewsList)
                          NoteFile MessageWin))
          (Continue% Repair                                        (* If changes were made, then checkpoint the notefile.)
               [if (NEQ (NC.CheckForBadLinksAndTitlesAndPropLists NoteFile MessageWin BadNewsList)
                        'ABORT)
                   then (if (WINDOWPROP MessageWin 'NEEDCHECKPOINT)
                            then (NC.CheckpointDatabase NoteFile NIL NIL (NC.AttachPromptWindow MessageWin)))
                        (WINDOWDELPROP MessageWin 'CLOSEFN (FUNCTION NC.MessageWinCloseFn))
                        (CLOSEW MessageWin)                        (* Rebuild the links.)
                        (NC.ScavengeDatabaseFile NoteFile (WINDOWPROP MessageWin 'LINKSLABELSNEWS)
                               (WINDOWPROP MessageWin 'BADBOXES)
                               (WINDOWPROP MessageWin 'CARDSWITHLINKSRESET)
                               (WINDOWPROP MessageWin 'INTERESTEDWINDOW)
                               (WINDOWPROP MessageWin 'LockProcess]
               (NOTIFY.EVENT InspectAndRepairFinishedEvent))
          (End% Inspect&Repair                                     (* Don't have to scavenge links. Close up gracefully.)
               (NC.CheckForBadLinksAndTitlesAndPropLists NoteFile MessageWin BadNewsList)
               (WINDOWDELPROP MessageWin 'CLOSEFN (FUNCTION NC.MessageWinCloseFn))
               (CLOSEW MessageWin)
               (NC.CloseDatabaseFile NoteFile)
               (NOTIFY.EVENT InspectAndRepairFinishedEvent))
          NIL])
```

(NC.MessageWinCloseFn

```
[LAMBDA (MainWin)                                                  (* rht%: "24-Mar-86 18:29")
```

(* * Runs when main message win is closed. Checks whether user means to abort and closes each card inspector window.)

(* * rht 12/8/85%: Modified to reflect new notefile format.)

(* * kirk 23Jan86 Changed to use NC.AskYesOrNo)

(* * rht 3/24/86%: Added call to NC.CloseInspectorWindows.)

```
    (if (NC.AskYesOrNo "Do you really want to abort the Inspect & Repair process? " NIL 'Yes T (
                                                                                    NC.AttachPromptWindow
                                                                                        MainWin)
                T NIL)
        then (NC.CloseInspectorWindows MainWin)
             (for MenuWin in (WINDOWPROP MainWin 'MENUWINDOWS) do (WINDOWDELPROP MenuWin 'CLOSEFN
                                                                          (FUNCTION NC.CardInspectorCloseFn))
                                                                  (DETACHWINDOW MenuWin)
                                                                  (CLOSEW MenuWin))
             (MOVEW MainWin NC.OffScreenPosition)
             (NC.ScavengerCleanup MainWin)
      else 'DON'T])

)

;;; Functions for getting the scavenger info for all valid card parts and reading things in the data area robustly.

(DEFINEQ
```

(NC.GetScavengerInfo

```
  [LAMBDA (NoteFile ReadSubstancesFlg MessageWin InterestedWindow)
                                                                  (* rht%: "17-Jul-86 12:07")
```

(* * Return an array containing pointers to all valid versions of notecard parts, i.e. substances, titles, links, and proplists. If ReadSubstancesFlg is non-nil, then read substances when checking rather than just checking that start and end substance pointers make sense.)

(* * rht 9/17/85%: Now keeps track of largest ID seen and caches on MessageWin's WINDOWPROP.)

(* * rht 12/1/85%: Updated to handle new notefile and card format. Ripped out MaxIDNum stuff.)

(* * rht 7/16/86%: Added InterestedWindow arg.)

```
    (LET ((Stream (fetch (NoteFile Stream) of NoteFile))
          EndPtr CurPtr BadCardTypesList)
         (SETQ EndPtr (GETEOFPTR Stream))
```

(* Initialize bad card types list to nil. NC.RobustReadCardPart will add to it as it finds unknown card types.)

```
         (WINDOWPROP MessageWin 'BADCARDTYPESLIST NIL)
         (NC.PrintMsg InterestedWindow T "Processing data area of notefile ..." (CHARACTER 13)
                "Searching for start of data area ...")          (* Find start of data area)
         (SETQ CurPtr (NC.SearchFor### Stream 0))                (* Walk, don't run, through the data area trying to find good card
        parts.)
         (if (for bind (LastPtrHighBits _ 0)
                    CurPtrHighBits Card
               eachtime (BLOCK) while (AND CurPtr (LESSP CurPtr EndPtr))
               do                                                 (* Print a message every 8K or so bytes.)
                   (if (GREATERP (SETQ CurPtrHighBits (LRSH CurPtr 13))
                                 LastPtrHighBits)
                       then (NC.PrintMsg InterestedWindow T "Processing data area of notefile ..." (CHARACTER
                                                                                                        13)
                                  "Byte number: "
                                  (QUOTIENT CurPtr 1000)
                                  "K out of "
                                  (QUOTIENT EndPtr 1000)
                                  "K.")
                            (SETQ LastPtrHighBits CurPtrHighBits))
```

(* Try to read a card part. If returns nil, then search for next %# marker and try again.)

```
               [if (SETQ Card (NC.RobustReadCardPart NoteFile ReadSubstancesFlg MessageWin EndPtr))
                   then (SETQ CurPtr (GETFILEPTR Stream))
                 else (SETQ CurPtr (NC.SearchFor### Stream (PLUS 4 CurPtr]
             finally (RETURN 'SUCCESS))
          then (NC.ClearMsg InterestedWindow T)
               'SUCCESS])
```

(NC.RobustReadCardPart

```
  [LAMBDA (NoteFile ReadSubstancesFlg MessageWin EndPtr)          (* rht%: "22-Jan-87 17:41")
```

(* * Assume stream is positioned at start of a card part. Try to read one and modify scavenger array according to what we find.)

(* * rht 9/17/85%: Now returns ID rather than litatom SUCCESS is wins.)

(* * rht 12/1/85%: Updated to handle new notefile and card object formats.)

(* * rht 3/22/86%: substance ptrs for bad card types are no longer put on card type's prop list. Now they're stored in an assoc list hung off a WINDOWPROP.)

(* * fgh |5/1/86| Fixed bug in previous fix. Code was bombing the first time it tried to put something on a nonexistent assoc list. Put in a COND to create the assoc list if it isn't already created.)

(* * fgh |7/30/86| Replaced call to NCP.ValidCardType with call to NCP.CardTypeP)

(* * rht 1/21/87%: If card part belongs to a card not in the notefile index, force the new card created to have the given UID.)

```
(DECLARE (GLOBALVARS NC.TitlesIdentifier NC.PropsIdentifier NC.LinksIdentifier NC.ItemIdentifier))
(LET ((Stream (fetch (NoteFile Stream) of NoteFile))
      IdentifierAndVersionNum IdentifierAtom Date UID Card Title ToLinks FromLinks GlobalLinks PropList Type
      SubstanceLength CardPartLength CurPtr UnknownCardTypesSubstancePtrs)
     (OR EndPtr (SETQ EndPtr (GETEOFPTR Stream)))
     (SETQ CurPtr (GETFILEPTR Stream))
     (if [AND (SETQ CardPartLength (NC.ReadPtr Stream 3))
              (LEQ (PLUS CurPtr CardPartLength)
                   EndPtr)
              (SETQ IdentifierAndVersionNum (NC.RobustReadItemIdentifier Stream))
              (if (GEQ (CDR IdentifierAndVersionNum)
                       1)
                  then (SETQ Date (NC.RobustReadDate Stream))
                else T)
              (SETQ UID (NC.RobustReadUID Stream))
              (SETQ Card (OR (NC.CardFromUID UID NoteFile)
                             (NC.GetNewCard NoteFile NIL UID]
         then (SETQ Date (CAR Date))
              (SETQ IdentifierAtom (CAR IdentifierAndVersionNum))
              (if [COND
                    [(EQ IdentifierAtom NC.TitlesIdentifier) (* Hoping to get a healthy title.)
                     (if (AND (SETQ Title (NC.RobustReadString Stream))
                              (NC.AtEndOfItemP Stream EndPtr))
                         then (NC.SetScavengerTitleInfo Card (CONS (LIST CurPtr Date Title)
                                                                   (NC.FetchScavengerTitleInfo Card]
                    [(EQ IdentifierAtom NC.PropsIdentifier)  (* Hoping to get a healthy prop list.)
                     (if (AND (SETQ PropList (NC.RobustReadList Stream))
                              (NC.AtEndOfItemP Stream))
                         then (NC.SetScavengerPropListInfo Card (CONS (LIST CurPtr Date (LENGTH (CAR PropList)))
                                                                      (NC.FetchScavengerPropListInfo Card]
                    [(EQ IdentifierAtom NC.LinksIdentifier)

(* Hoping to get healthy links. Try to read three lists%: ToLinks, FromLinks, and Global links.)

                     (if (AND (SETQ ToLinks (NC.RobustReadLinks Stream NoteFile))
                              (SETQ FromLinks (NC.RobustReadLinks Stream NoteFile))
                              (SETQ GlobalLinks (NC.RobustReadLinks Stream NoteFile))
                              (NC.AtEndOfItemP Stream EndPtr))
                         then (NC.SetScavengerLinksInfo Card (CONS [LIST CurPtr Date (LIST (LENGTH (CAR ToLinks)
                                                                                                   )
                                                                                           (LENGTH (CAR
                                                                                                       FromLinks
                                                                                                       ))
                                                                                           (LENGTH (CAR
                                                                                                     GlobalLinks
                                                                                                     ]
                                                                                    (NC.FetchScavengerLinksInfo Card]
                    ((EQ IdentifierAtom NC.ItemIdentifier)   (* Hoping to get healthy substance.)
                     (if (AND (SETQ Type (CAR (NC.RobustReadAtom Stream)))
                              (NC.RobustReadRegion Stream EndPtr)
                              (SETQ SubstanceLength (NC.CheckForValidSubstance Stream EndPtr Card Type
                                                          ReadSubstancesFlg))
                              (NC.AtEndOfItemP Stream))
                         then [if (NOT (NCP.CardTypeP Type))
                                  then                          (* Code for this card part type has not been loaded.)
                                       (WINDOWADDPROP MessageWin 'UNKNOWNCARDTYPESLIST Type)
                                                                (* Save the pointer to the substance card part on an assoc list)
                                       (COND
                                          [(SETQ UnknownCardTypesSubstancePtrs (WINDOWPROP MessageWin
                                                                                           '
                                                                                           UNKNOWNCARDTYPESSUBSTANCEPTRS
                                                                                           ))
                                           (LISTPUT UnknownCardTypesSubstancePtrs Type
                                                  (CONS CurPtr (LISTGET UnknownCardTypesSubstancePtrs Type]
                                          (T (WINDOWPROP MessageWin 'UNKNOWNCARDTYPESSUBSTANCEPTRS
                                                         (LIST Type (LIST CurPtr]
                              (NC.SetScavengerMainDataInfo Card (CONS (LIST CurPtr Date Type (OR SubstanceLength
                                                                                                '
                                                                                                UNKNOWNCARDTYPE
                                                                                                ))
                                                                      (NC.FetchScavengerMainDataInfo Card]
              then Card])
```

(NC.CheckForValidSubstance

```
[LAMBDA (Stream EofPtr Card CardType ReadSubstanceFlg)          (* rht%: " 9-May-87 18:23")
```

(* * Check whether we've got a valid substance at current pos in Stream.
Check for a valid StartPtr and EndPtr. If ReadSubstanceFlg is non-nil, then do robust get of the substance.
Return length of substance or nil.)

(* * rht 12/1/85%: Updated to handle new notefile format in MainCardData card part.)

(* * rht 3/22/86%: Now handles substance version numbers.)

(* * rht 5/9/87%: Now sets card type in card object because the call to NC.ApplyFn was calling NC.GetType and screwing up our file ptr.)

```
(LET ((OldPtr (GETFILEPTR Stream))
      Length EndPtr)
     (OR EofPtr (SETQ EofPtr (GETEOFPTR Stream)))
     (if (AND (LESSP OldPtr (DIFFERENCE EofPtr 3))
              (SETQ Length (NC.ReadPtr Stream 3))
              (LEQ (SETQ EndPtr (PLUS OldPtr 4 Length))
                   EofPtr)
              [OR (NOT ReadSubstanceFlg)
                  (NOT (NCP.ValidCardType CardType))
                  (PROGN (NC.SetType Card CardType)
                         (NC.RobustGetSubstance Card Length Stream (NC.ReadPtr Stream 1]
              (SETFILEPTR Stream EndPtr)
              Length)
         else (SETFILEPTR Stream OldPtr)
              NIL])
```

(NC.AtEndOfItemP

```
[LAMBDA (Stream EofPtr)                                    (* rht%: "22-Jan-87 15:49")
```

(* * Return T if at the end of an item. That means either we're at the end of the file, or there's a %# at the current location.)

(* * rht 12/1/85%: Updated to handle new notefile and card object formats.
Ripped out the kludgy check for NOBIND litatom. Also now handles the 3 byte length info at start of every card part header.)

(* * rht 1/22/87%: Now never SETFILEPTR's beyond end of file.)

```
(OR EofPtr (SETQ EofPtr (GETEOFPTR Stream)))
(LET ((OldPtr (GETFILEPTR Stream))
      HashPtr ResultFlg)
     (if (GEQ (SETQ HashPtr (PLUS 3 OldPtr))
              EofPtr)
         then T
         else                                              (* Skip past the card part length.)
              (SETFILEPTR Stream HashPtr)
              (SETQ ResultFlg (EQ (CHARACTER (BIN Stream))
                                  '%#))
              (SETFILEPTR Stream OldPtr)
              ResultFlg])
```

(NC.RobustReadString

```
[LAMBDA (Stream)                                           (* rht%: " 1-Dec-85 22:57")
```

(* * Try to read a string from Stream. Return NIL if not.)

```
(LET (Val (OldPtr (GETFILEPTR Stream)))
     (if [STRINGP (SETQ Val (CAR (NC.RobustRead Stream]
         then                                              (* Skip CR.)
              (NC.RobustReadChar Stream)
              Val
         else (SETFILEPTR Stream OldPtr)
              NIL])
```

(NC.RobustReadList

```
[LAMBDA (Stream)                                           (* rht%: " 1-Dec-85 22:57")
```

(* * Try to read a list from Stream. Returns a list of one element, the list read, if successful.
NIL otherwise.)

```
(LET (Val (OldPtr (GETFILEPTR Stream)))
     (if [AND (SETQ Val (NC.RobustRead Stream))
              (OR (NULL (CAR Val))
                  (LISTP (CAR Val]
         then                                              (* Skip CR at end.)
              (NC.RobustReadChar Stream)
              Val
         else (SETFILEPTR Stream OldPtr)
              NIL])
```

(NC.RobustReadLinks

```
[LAMBDA (Stream NoteFile)                                  (* rht%: " 8-Dec-85 19:01")
```

(* * Try to read a list of links from Stream. Return list of one element, the list read, if successful, NIL otherwise.)

```
(LET [(OldPtr (GETFILEPTR Stream))
      (ListOfThingRead (RESETVAR HELPFLAG NIL (NLSETQ (NC.ReadListOfLinks NoteFile]
```

```
         (if ListOfThingRead
           else (SETFILEPTR Stream OldPtr)
                NIL])
```

(NC.RobustReadAtom

```
  [LAMBDA (Stream)                                          (* rht%: " 1-Dec-85 23:05")

          (* * Returns a list of one element, the atom found. If unsuccessful read, then return nil.)

    (LET ((OldPtr (GETFILEPTR Stream))
          (Val (NC.RobustRead Stream)))
         (if (AND Val (ATOM (CAR Val)))
             then                                           (* Skip CR.)
                  (NC.RobustReadChar Stream)
                  Val
           else (SETFILEPTR Stream OldPtr)
                NIL])
```

(NC.RobustReadRegion

```
  [LAMBDA (Stream EofPtr)                                   (* rht%: " 1-Dec-85 23:07")

          (* * Try to read a notecards region. Return region or nil.)

    (LET ((OldPtr (GETFILEPTR Stream)))
         (OR EofPtr (SETQ EofPtr (GETEOFPTR Stream)))
         (if [AND (LESSP OldPtr (DIFFERENCE EofPtr 8))
                  (for i from 1 to 4 bind Num collect (if (GEQ (SETQ Num (NC.GetPtr Stream 2))
                                                               0)
                                                          then Num
                                                        else (RETURN NIL]
           else (SETFILEPTR Stream OldPtr)
                NIL])
```

(NC.RobustGetSubstance

```
  [LAMBDA (Card Length Stream SubstanceVersion)             (* rht%: " 8-May-87 14:20")

          (* * Try to get substance robustly. RESETVAR prevents breaks.
          Returns either substance or nil if unsuccessful.)

          (* * rht 12/1/85%: Updated to use card and notefile format.)

          (* * fgh |2/5/86| Added call to NC.ApplyFn)

          (* * rht 3/22/86%: Now passes substance version %#s and Stream to GetFn.)

          (* * rht 5/8/87%: Changed to use LET rather than RESETVAR to temporarily nullify HELPFLAG value.)

    (LET ((HELPFLAG NIL))
         (NLSETQ (NC.ApplyFn GetFn Card Length Stream SubstanceVersion])

)


;;; Functions for accessing scavenger info for a card.

(DEFINEQ
```

(NC.SetScavengerInfo

```
  [LAMBDA (Card NewScavengerInfo)                           (* rht%: " 1-Dec-85 21:25")

          (* * Set the scavenger info record for this card.)

    (NC.SetUserDataProp Card 'SCAVENGERINFO NewScavengerInfo])
```

(NC.SetScavengerTitleInfo

```
  [LAMBDA (Card TitleInfo)                                  (* rht%: " 1-Dec-85 21:41")

          (* * Set the title part of the scavenger info record for Card.)

    (LET ((ScavengerInfo (NC.FetchScavengerInfo Card)))
         [if (NOT (type? ScavengerInfo ScavengerInfo))
             then (NC.SetScavengerInfo Card (SETQ ScavengerInfo (create ScavengerInfo]
         (replace (ScavengerInfo TitleInfo) of ScavengerInfo with TitleInfo])
```

(NC.SetScavengerMainDataInfo

```
  [LAMBDA (Card MainDataInfo)                               (* rht%: " 1-Dec-85 21:41")

          (* * Set the main data part of the scavenger info record for Card.)

    (LET ((ScavengerInfo (NC.FetchScavengerInfo Card)))
         [if (NOT (type? ScavengerInfo ScavengerInfo))
             then (NC.SetScavengerInfo Card (SETQ ScavengerInfo (create ScavengerInfo]
```

```
        (replace (ScavengerInfo MainDataInfo) of ScavengerInfo with MainDataInfo])
```

(NC.SetScavengerLinksInfo

```
[LAMBDA (Card LinksInfo)                                    (* rht%: " 1-Dec-85 21:41")

          (* * Set the links part of the scavenger info record for Card.)

   (LET ((ScavengerInfo (NC.FetchScavengerInfo Card)))
        [if (NOT (type? ScavengerInfo ScavengerInfo))
            then (NC.SetScavengerInfo Card (SETQ ScavengerInfo (create ScavengerInfo]
        (replace (ScavengerInfo LinksInfo) of ScavengerInfo with LinksInfo])
```

(NC.SetScavengerPropListInfo

```
[LAMBDA (Card PropListInfo)                                 (* rht%: " 1-Dec-85 21:41")

          (* * Set the prop list part of the scavenger info record for Card.)

   (LET ((ScavengerInfo (NC.FetchScavengerInfo Card)))
        [if (NOT (type? ScavengerInfo ScavengerInfo))
            then (NC.SetScavengerInfo Card (SETQ ScavengerInfo (create ScavengerInfo]
        (replace (ScavengerInfo PropListInfo) of ScavengerInfo with PropListInfo])
```

(NC.FetchScavengerInfo

```
[LAMBDA (Card)                                              (* rht%: " 1-Dec-85 21:24")

          (* * Get the scavenger info record for this card.)

   (NC.FetchUserDataProp Card 'SCAVENGERINFO])
```

(NC.FetchScavengerTitleInfo

```
[LAMBDA (Card)                                              (* rht%: " 1-Dec-85 21:41")

          (* * Get the title part of the scavenger info record for Card.)

   (LET ((ScavengerInfo (NC.FetchScavengerInfo Card)))
        (if (type? ScavengerInfo ScavengerInfo)
            then (fetch (ScavengerInfo TitleInfo) of ScavengerInfo)
          else NIL])
```

(NC.FetchScavengerMainDataInfo

```
[LAMBDA (Card)                                              (* rht%: " 1-Dec-85 21:48")

          (* * Get the main data part of the scavenger info record for Card.)

   (LET ((ScavengerInfo (NC.FetchScavengerInfo Card)))
        (if (type? ScavengerInfo ScavengerInfo)
            then (fetch (ScavengerInfo MainDataInfo) of ScavengerInfo)
          else NIL])
```

(NC.FetchScavengerLinksInfo

```
[LAMBDA (Card)                                              (* rht%: " 1-Dec-85 21:48")

          (* * Get the links part of the scavenger info record for Card.)

   (LET ((ScavengerInfo (NC.FetchScavengerInfo Card)))
        (if (type? ScavengerInfo ScavengerInfo)
            then (fetch (ScavengerInfo LinksInfo) of ScavengerInfo)
          else NIL])
```

(NC.FetchScavengerPropListInfo

```
[LAMBDA (Card)                                              (* rht%: " 1-Dec-85 21:49")

          (* * Get the prop list part of the scavenger info record for Card.)

   (LET ((ScavengerInfo (NC.FetchScavengerInfo Card)))
        (if (type? ScavengerInfo ScavengerInfo)
            then (fetch (ScavengerInfo PropListInfo) of ScavengerInfo)
          else NIL])
```

(NC.FetchTypeFromScavengerInfo

```
[LAMBDA (Card)                                              (* rht%: " 7-Dec-85 20:12")

          (* * Return the type of the card by fetching from the MainDataInfo of ScavengerInfo.)

   (LET ((MainDataInfo (NC.FetchScavengerMainDataInfo Card)))
        (AND MainDataInfo (CADDAR MainDataInfo])
```

(NC.FetchTitleFromScavengerInfo

```
  [LAMBDA (Card)                                                    (* rht%: " 7-Dec-85 20:13")
```

(* * Return the title of the card by fetching from the TitleInfo of ScavengerInfo.)

```
    (LET ((TitleInfo (NC.FetchScavengerTitleInfo Card)))
         (AND TitleInfo (CADDAR TitleInfo])

)
```

;;; Functions for building bad cards list based on the index array and scavenger array.

```
(DEFINEQ
```

(NC.BuildBadCardsList

```
  [LAMBDA (NoteFile MessageWin FirstTimeFlg InterestedWindow)    ; Edited 3-Dec-87 18:59 by rht:
```

(* * Returns a list of all IDs with illegal index pointers, i.e. pointers not to valid data areas recorded in ScavengerArray. Also record those IDs with pointers beyond checkpoint ptr.)

(* * rht 9/17/85%: Now takes MessageWin argument so can extract the MaxIDNum off its props.)

(* * rht 12/7/85%: Modified to reflect new card and notefile object formats.)

(* * fgh |2/4/86| Fixed minor bug where UNKNOWNCARDTYPE apeeared in a singleton list.)

(* * rht 7/16/86%: Added InterestedWindow arg.)

(* * fgh |7/30/86| Replaced call to NCP.ValidCardType with call to NCP.CardTypeP)

```
    (LET ((CardTotal (SUB1 (fetch (NoteFile NextIndexNum) of NoteFile)))
          (CheckptPtr (fetch (NoteFile CheckptPtr) of NoteFile))
          (Num 0)
          Results)
         (NC.PrintMsg InterestedWindow T "Building bad cards list ..." (CHARACTER 13)
                "Processing item number " 1 " out of " CardTotal ".")
         [NC.MapCards NoteFile (FUNCTION (LAMBDA (Card)
                                         (LET
                                          (Problems)
                                          (SETQ Num (ADD1 Num))
                                          (if (ZEROP (IREMAINDER Num 100))
                                              then (NC.PrintMsg InterestedWindow T "Building bad cards list
                                                                ..." (CHARACTER 13)
                                                                "Processing item number " Num " out of " CardTotal
                                                                "."))
```

(* If card is not worthless, has reasonable status, and at least one problem, then make a bad card entry for it.)

```
                                          (if [AND (NOT (NC.WorthlessCardP Card))
                                                   (FMEMB (NC.FetchStatus Card)
                                                          '(ACTIVE DELETED SPECIAL BADPOINTER NIL))
                                                   (SETQ Problems
                                                    (LET ((IndexLocsProblems (NC.CheckIndexLocs Card MessageWin
                                                                                                CheckptPtr FirstTimeFlg))
                                                          (Type (NC.FetchTypeFromScavengerInfo Card)))
                                                         (if (AND Type (NOT (NCP.CardTypeP Type)))
                                                             then (CONS 'UNKNOWNCARDTYPE IndexLocsProblems)
                                                           else IndexLocsProblems]
                                              then (push Results `(,Card ,(NC.FetchStatus Card)
                                                                         ,(NC.EncodeCardProblems Problems)
                                                                         ,@Problems]
         (NC.ClearMsg InterestedWindow T)
         Results])
```

(NC.WorthlessCardP

```
  [LAMBDA (Card)                                                    (* rht%: " 8-Dec-85 16:03")
```

(* * Return non-nil if card is totally worthless, i.e. is not a top level card, has no valid versions of any of its parts in scavenger array, has zeroes in all the fields of the index array, and has status NIL or DELETED.)

(* * rht 12/7/85%: Modified to reflect new card and notefile object formats.)

```
    (LET ((NoteFile (fetch (Card NoteFile) of Card)))
         (AND (FMEMB (fetch (Card Status) of Card)
                     '(NIL DELETED))
              (ZEROP (fetch (Card MainLoc) of Card))
              (ZEROP (fetch (Card LinksLoc) of Card))
              (ZEROP (fetch (Card TitleLoc) of Card))
              (ZEROP (fetch (Card PropListLoc) of Card))
              (NOT (NC.UndeletableCardP Card))
              (NULL (NC.FetchScavengerMainDataInfo Card))
              (NULL (NC.FetchScavengerTitleInfo Card))
              (NULL (NC.FetchScavengerLinksInfo Card))
              (NULL (NC.FetchScavengerPropListInfo Card])
```

(NC.CheckIndexLocs

```
[LAMBDA (Card MessageWin CheckptPtr CanModifyFlg)                    (* rht%: " 8-Dec-85 12:18")

        (* * Check if Card's index locs agree with the scavenger info. Also may possibly modify index locs if latest version of a card
        part is beyond checkpoint ptr. Return list of atoms indicating the problems we found or NIL if none.)

   (LET ((MainDataInfo (NC.FetchScavengerMainDataInfo Card))
         (LinksInfo (NC.FetchScavengerLinksInfo Card))
         (TitleInfo (NC.FetchScavengerTitleInfo Card))
         (PropListInfo (NC.FetchScavengerPropListInfo Card)))
        (if CanModifyFlg
            then (NC.CheckForLocBeyondCheckptPtr Card MainDataInfo (FUNCTION NC.SetMainLoc)
                        CheckptPtr MessageWin)
                 (NC.CheckForLocBeyondCheckptPtr Card LinksInfo (FUNCTION NC.SetLinksLoc)
                        CheckptPtr MessageWin)
                 (NC.CheckForLocBeyondCheckptPtr Card TitleInfo (FUNCTION NC.SetTitleLoc)
                        CheckptPtr MessageWin)
                 (NC.CheckForLocBeyondCheckptPtr Card PropListInfo (FUNCTION NC.SetPropListLoc)
                        CheckptPtr MessageWin))
        (LSUBST NIL NIL (LIST (NC.CheckIndexLoc Card MainDataInfo (fetch (Card MainLoc) of Card)
                                    'MAINDATA CheckptPtr)
                              (NC.CheckIndexLoc Card LinksInfo (fetch (Card LinksLoc) of Card)
                                    'LINKS CheckptPtr)
                              (NC.CheckIndexLoc Card TitleInfo (fetch (Card TitleLoc) of Card)
                                    'TITLE CheckptPtr)
                              (NC.CheckIndexLoc Card PropListInfo (fetch (Card PropListLoc) of Card)
                                    'PROPLIST CheckptPtr])
```

(NC.CheckIndexLoc

```
[LAMBDA (Card CardPartInfo Loc CardPartAtomIdentifier CheckptPtr)
                                                                     (* rht%: " 8-Dec-85 18:07")

        (* * Check if Loc is a valid ptr in the scavenger array. If not, return an atom indicating the problem.)

   (if (SASSOC Loc CardPartInfo)
       then

        (* Found an occurrence of the index ptr in the scavenger array.
        We return NIL signifying no error or a "past checkpoint" indicator.)

          (if (GEQ Loc CheckptPtr)
              then (PACK* CardPartAtomIdentifier 'PASTCHKPT)
            else NIL)
     else                                                            (* The index points to a bad item.)
        (PACK* 'BAD CardPartAtomIdentifier])
```

(NC.CheckForLocBeyondCheckptPtr

```
[LAMBDA (Card CardPartInfo SetLocFn CheckptPtr MessageWin)           (* rht%: " 8-Dec-85 11:42")

        (* * See if there's a ptr from the data area for this card part that's beyond the checkpoint ptr.
        If so, update loc to point to it and update status if needed.)

   (LET [(MaxBeyondCheckptPtr (CAR (for Elt in CardPartInfo eachtime (BLOCK) when (GEQ (CAR Elt)
                                                                                       CheckptPtr)
                                      largest (CAR Elt]
        (if MaxBeyondCheckptPtr
            then (OR (NC.FetchStatus Card)
                     (NC.SetStatus Card 'ACTIVE))
                 (APPLY* SetLocFn Card MaxBeyondCheckptPtr)
                 (WINDOWPROP MessageWin 'NEEDLINKSCAVENGE T])
```

(NC.EncodeCardProblems

```
[LAMBDA (ProblemIndicators)                                          (* fgh%: " 4-Feb-86 19:54")

        (* * Return a string of up to 4 characters encoding the problems with Card.
        It's null if card is okay. Otherwise contains the chars L, S, P, and/or T representing bum links, substance, prop list or title.
        Lowercase letters l, s, p, t represent fact that latest links, say, were written out beyond the checkpoint pointer.
        There's also the letter U representing unknown card type.)

        (* * rht 12/8/85%: Took out the problem identifier atoms like BADLINKSPTR that are no longer kept.)

        (* * fgh |2/4/86| Removed parens around many of the problem identifier atoms which were not supposed to be there.)

   (CONCAT (COND
              ((FMEMB 'BADLINKS ProblemIndicators)
               'L)
              ((FMEMB 'LINKSPASTCHKPT ProblemIndicators)
               'l)
              (T ""))
           (COND
              ((FMEMB 'BADMAINDATA ProblemIndicators)
               'S)
              ((FMEMB 'MAINDATAPASTCHKPT ProblemIndicators)
               's)
```

```
            (T ""))
         (COND
            ((FMEMB 'BADPROPLIST ProblemIndicators)
             'P)
            ((FMEMB 'PROPLISTPASTCHKPT ProblemIndicators)
             'p)
            (T ""))
         (COND
            ((FMEMB 'BADTITLE ProblemIndicators)
             'T)
            ((FMEMB 'TITLEPASTCHKPT ProblemIndicators)
             't)
            (T ""))
         (COND
            ((FMEMB 'UNKNOWNCARDTYPE ProblemIndicators)
             'U)
            (T ""])

)
```

;;; Functions for interacting with the user.

```
(DEFINEQ
```

(NC.BuildCardInspectorMenu

```
  [LAMBDA (Cards BadNewsList NoteFile MessageWin)                ; Edited  3-Dec-87 18:59 by rht:
```

(* * Build a list of menu containing all valid cards giving ID %#s and indicators as to problems, if any. ID %#s can be invoked causing a window/menu containing info on the card to be brought up.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

(* * rht 6/12/86%: Fixed loop that does SHADEITEMS.)

```
    (LET (MenuWindows TotalMenus FirstWindow FirstWindowSize AttachedMenu)
         (if (NOT (WINDOWP MessageWin))
             then (NC.ReportError "NC.BuildCardInspectorMenu" (CONCAT "Arg not window: " MessageWin)))
         (NC.SortCardsBySlotNums Cards)
         (SETQ TotalMenus (ADD1 (QUOTIENT (SUB1 (LENGTH Cards))
                                          NC.CardsPerMenuLimit)))
```

(* Build the attached menu for global menu operations including moving to next or previous page if necessary.)

```
         (SETQ AttachedMenu (create MENU
                                    ITEMS _ '[(Abort 'Abort "Abort the repair process, throwing away all
                                                     changes.")
                                              (Done 'Done "Integrate changes (i.e. checkpoint notefile) and
                                                    continue repair.")
                                              (Search 'Search "Find IDs for cards with titles containing search
                                                      string.")
                                              ,@(if (GEQ TotalMenus 2)
                                                    then '((Previous% Page 'Previous% Page "Move to menu for
                                                                  previous page of cards.")
                                                           (Next% Page 'Next% Page "Move to menu for previous page
                                                                  of cards.")
                                                           (First% Page 'First% Page "Move to menu for first page
                                                                  of cards."]
                                    CENTERFLG _ T
                                    MENUCOLUMNS _ 1
                                    WHENSELECTEDFN _ (FUNCTION NC.CardInspectorAttachedMenuWhenSelectedFn)
                                    MENUFONT _ NC.ScavengerAttachedMenuFont))
                                                                  (* Build the menus and attachments and store on message
                                                                  window prop.)
         (WINDOWPROP MessageWin 'MENUWINDOWS
                (SETQ MenuWindows
                 (for MenuNum from 1 to TotalMenus bind (RestOfCards _ Cards)
                                                        Menu Window Items AttachedWindow
                    collect [SETQ Menu
                                  (create MENU
                                          ITEMS _
                                          (SETQ Items
                                           (for old RestOfCards on RestOfCards as i from 1 to NC.CardsPerMenuLimit
                                              bind Card ProblemIdentifierAtom TitleString
                                              collect
```

(* * Build string to display in menu entry by concat'ing together the card's slot num with the first few characters of card's title with problem identification chars.)

```
                                                  (SETQ Card (CAR RestOfCards))
                                                  (SETQ ProblemIdentifierAtom (CADDR (FASSOC Card BadNewsList)))
                                                  [SETQ TitleString (CONCAT (NC.FetchSlotNum Card)
                                                                            ": "
                                                                            (NC.ShaveTitleString (
                                                                                                  NC.FetchTitleFromScavengerInfo
                                                                                                    Card]
                                                  (LIST (if ProblemIdentifierAtom
```

```
                                                    then (CONCAT TitleString "|" ProblemIdentifierAtom)
                                                  else TitleString)
                                                Card)))
                                    TITLE _ (CONCAT "Card inspector: Page " MenuNum " of " TotalMenus)
                                    WHENSELECTEDFN _ (FUNCTION NC.CardInspectorMenuWhenSelectedFn)
                                    MENUROWS _ (MIN 30 (LENGTH Items]
                                (SETQ Window (ADDMENU Menu NIL NC.OffScreenPosition))
                                                                    (* Shade the troublesome items in this menu.)
                                [for News in BadNewsList do (LET* [(BadCard (CAR News))
                                                                   (Item (for Item in Items
                                                                            when (NC.SameCardP BadCard (CADR Item))
                                                                            do (RETURN Item]
                                                                  (if Item
                                                                      then (SHADEITEM Item Menu NC.LightShade Window]
                                (WINDOWPROP Window 'NOTEFILE NoteFile)
                                                                    (* Rig so that lines through deleted entries will be drawn
                                                                    whenever menu is redisplayed.)
                                (WINDOWADDPROP Window 'REPAINTFN (FUNCTION NC.CardInspectorRepaintFn))
```

(* This makes closing the cards inspector menu same as selecting "abort" from the attached menu.
That is, abort the repair process.)

```
                                (WINDOWADDPROP Window 'CLOSEFN (FUNCTION NC.CardInspectorCloseFn)
                                       'FIRST)
                                (ATTACHWINDOW (ADDMENU AttachedMenu NIL NC.OffScreenPosition)
                                       Window
                                       'RIGHT
                                       'TOP)
                               Window)))
```

(* Attach the first menu with its attached operations menu to message window's lower left edge.)

```
        (SETQ FirstWindow (CAR MenuWindows))
        (ATTACHWINDOW FirstWindow MessageWin 'BOTTOM 'LEFT)
        (REDISPLAYW FirstWindow])
```

(NC.CardInspectorCloseFn

```
  [LAMBDA (Win Don'tAskFlg)                                         (* rht%: "16-Jul-86 22:02")
```

(* * Closing inspector window is just like selecting "abort" from attached menu, i.e.
abort the repair process.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

(* * kirk 23Jan86 Changed to use NC.AskYesOrNo)

(* * rht 3/24/86%: Added call to NC.CloseInspectorWindows.)

(* * rht 7/16/86%: Added NC.AttachPromptWindow call.)

```
    (LET ((MainWin (MAINWINDOW Win)))
         (if (OR Don'tAskFlg (NC.AskUser "Do you really want to abort the Inspect & Repair process? " NIL
                                  'Yes T (NC.AttachPromptWindow MainWin)
                                  NIL NIL))
             then (DETACHWINDOW Win)
                  (NC.CloseInspectorWindows MainWin)
                  (for MenuWin in (WINDOWPROP MainWin 'MENUWINDOWS) unless (EQ MenuWin Win)
                     do (WINDOWDELPROP MenuWin 'CLOSEFN (FUNCTION NC.CardInspectorCloseFn))
                        (DETACHWINDOW MenuWin)
                        (CLOSEW MenuWin))
                  (WINDOWDELPROP MainWin 'CLOSEFN (FUNCTION NC.MessageWinCloseFn))
                  (CLOSEW MainWin)
                  (MOVEW Win NC.OffScreenPosition)
                  (NC.ScavengerCleanup MainWin)
           else 'DON'T])
```

(NC.CardInspectorRepaintFn

```
  [LAMBDA (Win Region)                                              (* rht%: " 8-Dec-85 17:26")
```

(* * Checks the list of cards and draws lines through all the newly deleted or undeleted ones.
Draws in invert mode.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

```
    (LET ([Menu (CAR (WINDOWPROP Win 'MENU]
          (MessageWin (MAINWINDOW Win))
          DeletedCards)
         [SETQ DeletedCards (COPY (WINDOWPROP MessageWin 'DELETEDCARDS]
         [for Item in (fetch (MENU ITEMS) of Menu) bind Status OnDeletedListFlg Card ItemRegion
            when (NC.CardP (SETQ Card (CADR Item)))
            do (SETQ ItemRegion (MENUITEMREGION Item Menu))
               (SETQ Status (NC.FetchStatus Card))
               (SETQ OnDeletedListFlg (FMEMB Card DeletedCards))
                                                                    (* Either take off newly undeleted card or add newly deleted
                                                                    one.)
```

```
            (COND
               ((AND OnDeletedListFlg (EQ Status 'ACTIVE))
                (SETQ DeletedCards (DREMOVE Card DeletedCards))
                (SETQ OnDeletedListFlg NIL))
               ((AND (NOT OnDeletedListFlg)
                     (EQ Status 'DELETED))
                (SETQ DeletedCards (CONS Card DeletedCards))
                (SETQ OnDeletedListFlg T)))
            (if OnDeletedListFlg
                then                                          (* Draw line in invert mode through menu item.)
                     (LET ((YPoint (PLUS (LRSH (fetch (REGION HEIGHT) of ItemRegion)
                                               1)
                                         (fetch (REGION BOTTOM) of ItemRegion)))
                           (XLeft (fetch (REGION LEFT) of ItemRegion)))
                          (DRAWLINE XLeft YPoint (PLUS XLeft (fetch (REGION WIDTH) of ItemRegion))
                                    YPoint 1 'INVERT Win]
        (WINDOWPROP MessageWin 'DELETEDCARDS DeletedCards])
```

(NC.CardInspectorMenuWhenSelectedFn

```
[LAMBDA (CardPair Menu MouseKey)                              (* rht%: "22-Jan-87 18:23")
```

(* * Called when a card is selected from main card inspector menu.
Pop up menu offering choice of "Inspect" or "Delete" %.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

(* * fgh |2/4/86| Fixed bug where delete was no marking the index dirty.)

(* * rht 1/22/87%: Fixed typo in loop that checks open inspector windows.)

```
   (AND CardPair
        (LET ((Win (WFROMMENU Menu))
              [NormalItems '((Inspect 'Inspect "Bring up description of card.")
                             (Delete 'Delete "Mark this card as deleted."]
              [ItemsWithUndelete '((Inspect 'Inspect "Bring up description of card.")
                                   (Undelete 'Undelete "Undelete this card."]
              [ItemsWithoutDelete '((Inspect 'Inspect "Bring up description of card."]
              (Card (CADR CardPair))
              DeletedCards NoteFile ExistingWin MessageWin)
             (SETQ MessageWin (MAINWINDOW Win))
             (SETQ DeletedCards (WINDOWPROP MessageWin 'DELETEDCARDS))
             (SETQ NoteFile (WINDOWPROP Win 'NOTEFILE))
             (if (EQ MouseKey 'MIDDLE)
                 then [MENU (create MENU
                                    ITEMS _ (LIST (NC.FetchTitleFromScavengerInfo Card]
                 else (SELECTQ [MENU (create MENU
                                             ITEMS _ (COND
                                                        ((NC.UndeletableCardP Card)
                                                         ItemsWithoutDelete)
                                                        ((FMEMB Card DeletedCards)
                                                         ItemsWithUndelete)
                                                        (T NormalItems]
                               (Inspect (if (SETQ ExistingWin (for InspectorWindow in (WINDOWPROP MessageWin
                                                                                                  'INSPECTORWINDOWS)
                                                                 when [AND (OPENWP InspectorWindow)
                                                                           (NC.SameCardP Card (WINDOWPROP
                                                                                                InspectorWindow
                                                                                                'CARD]
                                                                 do (RETURN InspectorWindow)))
                                            then (FLASHW ExistingWin)
                                            else (WINDOWADDPROP MessageWin 'INSPECTORWINDOWS (NC.BuildCardPartsInspector
                                                                                               Card NoteFile Win))))
                               (Delete (NC.SetStatus Card 'DELETED)
                                       (WINDOWPROP Win 'MADECHANGES T)
                                       (WINDOWPROP Win 'NEEDLINKSCAVENGE T)
                                       (REDISPLAYW Win))
                               (Undelete                      (* I wonder if allowing undeletion is dangerous.
                                                              We shall see.)
                                         (NC.SetStatus Card 'ACTIVE)   (* Indicate that we made a real change to some card.)
                                         (WINDOWPROP Win 'MADECHANGES T)
                                         (WINDOWPROP Win 'NEEDLINKSCAVENGE T)
                                         (REDISPLAYW Win))
                               NIL])
```

(NC.CardInspectorAttachedMenuWhenSelectedFn

```
[LAMBDA (Item Menu MouseKey)                                  (* pmi%: "18-Aug-87 14:46")
```

(* * Called when leaving the main cards inspector menu. User is either aborting the scavenge or absorbing changes and continuing. In latter case, we need to recompute scavenger array to check for if she has fixed whatever problems there were. If so, then continue with link scavenge.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

(* * kirk 23Jan86 Changed to use NC.AskYesOrNo)

(* * rht 3/24/86%: Added call to NC.CloseInspectorWindows.)

(* * rht 7/16/86%: Added NC.AttachPromptWindow calls. String search is now case INsensitive.)

(* * rht 1/22/87%: Now calls NC.AskYesOrNo instead of NC.AskUser.)

(* * rht 3/14/87%: Fixed minor attachwindow bug in call to NC.ScavengerPhase1.)

(* * pmi 8/18/87%: Now gets InspectAndRepairFinishedEvent from the property list of MessageWin.
Calls NOTIFY.EVENT on it to signal the end of Scavenging.)

```
(DECLARE (GLOBALVARS NC.OffScreenPosition))
(LET ((MainWin (MAINWINDOW (WFROMMENU Menu)))
      (Operation (CAR Item))
      MessageWin NoteFile InspectorWins NextWin MenuWindows SearchString InspectAndRepairFinishedEvent)
     (SETQ MessageWin (MAINWINDOW MainWin))
     (SETQ MenuWindows (WINDOWPROP MessageWin 'MENUWINDOWS))
     (SETQ InspectAndRepairFinishedEvent (WINDOWPROP MessageWin 'FINISHEDEVENT))
     (if [AND (OR (NEQ Operation 'Abort)
                  (NC.AskYesOrNo "Do you really want to abort the Inspect & Repair process? " NIL
                         'Yes T (NC.AttachPromptWindow MessageWin)
                         NIL NIL))
              (OR [NOT (FMEMB Operation '(Abort Done]
                  (for Win in (SETQ InspectorWins (WINDOWPROP MessageWin 'INSPECTORWINDOWS))
                     never (OPENWP Win))
                  (PROG2 (NC.PrintMsg (NC.AttachPromptWindow MessageWin)
                                 T "There are open card inspector windows.")
                         (if (NC.AskYesOrNo "Want to close them? " NIL "Yes" NIL (NC.AttachPromptWindow
                                                                                    MessageWin))
                             then (for Win in InspectorWins when (OPENWP Win) do (CLOSEW Win))
                           T]
         then (SETQ NoteFile (WINDOWPROP MainWin 'NOTEFILE))
              (SELECTQ Operation
                   (Abort (WINDOWDELPROP MainWin 'CLOSEFN (FUNCTION NC.CardInspectorCloseFn))
                          (NC.CardInspectorCloseFn MainWin T)
                          (CLOSEW MainWin)
                          (NOTIFY.EVENT InspectAndRepairFinishedEvent))
                   (Done (DETACHWINDOW MainWin)
                         (WINDOWDELPROP MainWin 'CLOSEFN (FUNCTION NC.CardInspectorCloseFn))
                         (NC.CloseInspectorWindows MessageWin)
                         (for MenuWin in (WINDOWPROP MessageWin 'MENUWINDOWS) unless (EQ MenuWin MainWin)
                            do (WINDOWDELPROP MenuWin 'CLOSEFN (FUNCTION NC.CardInspectorCloseFn))
                               (CLOSEW MenuWin))
                         (if (WINDOWPROP MainWin 'MADECHANGES)
                             then (WINDOWPROP MessageWin 'NEEDCHECKPOINT T))
                         [WINDOWPROP MessageWin 'NEEDLINKSCAVENGE (OR (WINDOWPROP MessageWin
                                                                             'NEEDLINKSCAVENGE)
                                                                      (WINDOWPROP MainWin 'NEEDLINKSCAVENGE]
                         (CLOSEW MainWin)
```

(* This "recursive" call will check to make sure everything is okay before invoking link scavenge.
I'm calling with ReadSubstancesFlg off, even if this call had it on.
That's because we don't want it to be so slow on the second go-around.)

```
                         (NC.ScavengerPhase1 NoteFile NIL MessageWin NIL))
                   (Search
```

(* Simple-minded search for titles of cards containing the given search string.
User is asked for string in the prompt window, but answer appears in MessageWin for easy access.)

```
                           (SETQ SearchString (NC.AskUser "Search string? " NIL NIL T (NC.AttachPromptWindow
                                                                                        MessageWin)
                                                      T T NIL))
                           (NC.PrintMsg (NC.AttachPromptWindow MessageWin)
                                  T "Searching ... ")
                           (NC.PrintMsg MessageWin NIL "Card titles containing string '" SearchString "': "
                                  [NC.MapCards NoteFile [FUNCTION (LAMBDA (Card PredicateResult)
                                                                    PredicateResult]
                                                (FUNCTION (LAMBDA (Card)
                                                            (LET ((Title (NC.FetchTitleFromScavengerInfo Card)))
                                                                 (if (STRPOS (U-CASE SearchString)
                                                                             (U-CASE Title))
                                                                     then (CONCAT (NC.FetchSlotNum Card)
                                                                                  ": " Title)
                                                                   else NIL]
                                  (CHARACTER 13))
                           (NC.ClearMsg (NC.AttachPromptWindow MessageWin)
                                  T))
                   (Previous% Page                          (* Get previous menu window by accessing list of windows
                                                            cached on MessageWin.)
                        (DETACHWINDOW MainWin)
                        (MOVEW MainWin NC.OffScreenPosition)
                        (ATTACHWINDOW [SETQ NextWin (if (EQ MainWin (CAR MenuWindows))
                                                        then (CAR (LAST MenuWindows))
                                                      else (CAR (NLEFT MenuWindows 1 (FMEMB MainWin
                                                                                            MenuWindows]
                              MessageWin
```

```
                              'BOTTOM
                              'LEFT))
                    (Next% Page                                    (* Get next menu window by accessing list of windows cached
                                                                      on MessageWin.)
                               (DETACHWINDOW MainWin)
                               (MOVEW MainWin NC.OffScreenPosition)
                               (ATTACHWINDOW [SETQ NextWin (if (EQ MainWin (CAR (LAST MenuWindows)))
                                                               then (CAR MenuWindows)
                                                              else (CADR (FMEMB MainWin MenuWindows]
                                     MessageWin
                                     'BOTTOM
                                     'LEFT))
                    (First% Page                                   (* Get first menu window by accessing list of windows cached
                                                                      on MessageWin.)
                                (if (EQ MainWin (SETQ NextWin (CAR MenuWindows)))
                                    then (FLASHW MainWin)
                                  else (DETACHWINDOW MainWin)
                                       (MOVEW MainWin NC.OffScreenPosition)
                                       (ATTACHWINDOW NextWin MessageWin 'BOTTOM 'LEFT)))
                    NIL])
```

(NC.BuildCardPartsInspector

```
  [LAMBDA (Card NoteFile CardsMenuWindow)                          ; Edited 3-Dec-87 18:59 by rht:
```

(* * Build an attached group of menus for the 4 card parts%: substance, links, title, and proplist.
These contain items for each version of that part of Card found on Stream.
The highlighted item is the one currently pointed to. Return a list of user's changes.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

(* * rht 7/16/86%: Fixed a bug in menu entry naming.)

```
    (LET ((CheckptPtr (fetch (NoteFile CheckptPtr) of NoteFile))
          (CardType (NC.FetchTypeFromScavengerInfo Card))
          (CardStatus (NC.FetchStatus Card))
          MenusAndItemNums MainWindow Menus OldItemNums)
         (SETQ MenusAndItemNums (LIST (NC.BuildTitlesInspectorMenu Card CardType CheckptPtr)
                                      (NC.BuildSubstancesInspectorMenu Card CardType CheckptPtr CardsMenuWindow)
                                      (NC.BuildLinksInspectorMenu Card CheckptPtr)
                                      (NC.BuildPropListsInspectorMenu Card CheckptPtr)))
         (SETQ Menus (for MenuAndItemNum in MenusAndItemNums collect (CAR MenuAndItemNum)))
         (SETQ OldItemNums (for MenuAndItemNum in MenusAndItemNums collect (CDR MenuAndItemNum)))
                                                                   (* Titles menu occupies main window.
                                                                   Rest of menus are attached in order below.)
         (SETQ MainWindow (ADDMENU (CAR Menus)
                                   NIL
                                   (GETBOXPOSITION (fetch (MENU IMAGEWIDTH) of (CAR Menus))
                                          (fetch (MENU IMAGEHEIGHT) of (CAR Menus))
                                          NIL NIL NIL "Position the card parts menu.")))
         (for Menu in (CDR Menus) do (ATTACHWINDOW (ADDMENU Menu)
                                                   MainWindow
                                                   'BOTTOM
                                                   'LEFT))         (* Save menus and item numbers of original selections.)
         (WINDOWPROP MainWindow 'CARDPARTSMENUS Menus)
         (WINDOWPROP MainWindow 'CARDPARTSMENUOLDITEMNUMS OldItemNums)
         (WINDOWPROP MainWindow 'NOTEFILE NoteFile)
         (WINDOWPROP MainWindow 'CARD Card)
         (WINDOWPROP MainWindow 'CARDSMENUWINDOW CardsMenuWindow)
                                                                   (* Shade the original selections.)
         (for Menu in Menus as ItemNum in OldItemNums do (SHADEITEM (CAR (FNTH (fetch (MENU ITEMS) of Menu)
                                                                               ItemNum))
                                                                    Menu NC.LightShade)
                                                          (PUTMENUPROP Menu 'CURITEMNUM ItemNum))
         (ATTACHWINDOW (ADDMENU
                        (create MENU
                                ITEMS _
                                `[(CANCEL 'CANCEL "Quit this card parts inspector, throwing away changes.")
                                  (UPDATE 'UPDATE "Change to indicated versions for this card.")
                                  (RESET 'RESET "Go back to original version of card parts for this card.")
                                  ,@(if (NOT (NC.UndeletableCardP Card))
                                        then (LIST (if (EQ CardStatus 'DELETED)
                                                       then '(UNDELETE 'UNDELETE "Undelete this card, restoring to
                                                                                  indicated or most recent versions of card
                                                                                  parts.")
                                                     else '(DELETE 'DELETE "Mark this card as deleted."]
                                MENUFONT _ NC.CardInspectorAttachedMenuFont
                                MENUBORDERSIZE _ 1
                                MENUOUTLINESIZE _ 1
                                CENTERFLG _ T
                                MENUROWS _ 2
                                WHENSELECTEDFN _ (FUNCTION NC.CardPartsAttachedMenuWhenSelectedFn)))
                MainWindow
                'TOP
                'LEFT)
               MainWindow])
```

(NC.CardPartsAttachedMenuWhenSelectedFn

```
[LAMBDA (Item Menu MouseKey)                                    (* rht%: "22-Mar-86 18:22")

          (* * Called from the upper attached menu of the card parts menu.
          Contains options on what to do with changes to versions of this card.
          Can RESET to original versions, UPDATE by returning the new versions selected, ABORT, or DELETE the card.)

          (* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

    (LET ((MainWin (MAINWINDOW (WFROMMENU Menu)))
          Menus OldItemNums Card CardsMenuWindow)
         (SETQ Menus (WINDOWPROP MainWin 'CARDPARTSMENUS))
         (SETQ OldItemNums (WINDOWPROP MainWin 'CARDPARTSMENUOLDITEMNUMS))
         (SETQ Card (WINDOWPROP MainWin 'CARD))
         (SETQ CardsMenuWindow (WINDOWPROP MainWin 'CARDSMENUWINDOW))
         (SELECTQ (CAR Item)
              (RESET (for Menu in Menus as OldItemNum in OldItemNums bind Items
                        do                                      (* First unshade currently shaded item, then shade the original one.)
                           (SETQ Items (fetch (MENU ITEMS) of Menu))
                           (SHADEITEM [CAR (FNTH Items (GETMENUPROP Menu 'CURITEMNUM]
                                  Menu)
                           (SHADEITEM (CAR (FNTH Items OldItemNum))
                                  Menu NC.LightShade)
                           (PUTMENUPROP Menu 'CURITEMNUM OldItemNum)))
              (CANCEL (CLOSEW MainWin))
              (DELETE (NC.SetStatus Card 'DELETED)
                      (CLOSEW MainWin)                          (* Indicate that we made a real change to some card.)
                      (WINDOWPROP CardsMenuWindow 'MADECHANGES T)
                      (WINDOWPROP CardsMenuWindow 'NEEDLINKSCAVENGE T)
                      (REDISPLAYW CardsMenuWindow))
              (UNDELETE                                         (* I wonder if allowing undeletion is dangerous.
                                                                We shall see.)
                        (NC.SetStatus Card 'ACTIVE)
                        (CLOSEW MainWin)                        (* Indicate that we made a real change to some card.)
                        (WINDOWPROP CardsMenuWindow 'MADECHANGES T)
                        (WINDOWPROP CardsMenuWindow 'NEEDLINKSCAVENGE T)
                        (REDISPLAYW CardsMenuWindow))
              (UPDATE                                           (* Change the card locs for each card part that user has decided to change.)
                      (for Menu in Menus as OldItemNum in OldItemNums as SetLocFn
                         in '(NC.SetTitleLoc NC.SetMainLoc NC.SetLinksLoc NC.SetPropListLoc) bind CurItemNum
                         when [NEQ OldItemNum (SETQ CurItemNum (GETMENUPROP Menu 'CURITEMNUM]
                         do (APPLY* SetLocFn Card (CADAR (FNTH (fetch (MENU ITEMS) of Menu)
                                                               CurItemNum)))
                                                                (* Indicate that we made a real change to some card.)
                            (WINDOWPROP CardsMenuWindow 'MADECHANGES T)
                                                                (* Changes to links or substance require a link scavenge.)
                            (if (FMEMB SetLocFn '(NC.SetMainLoc NC.SetLinksLoc))
                                then (WINDOWPROP CardsMenuWindow 'NEEDLINKSCAVENGE T)))
                      (CLOSEW MainWin))
              NIL])
```

(NC.BuildTitlesInspectorMenu

```
[LAMBDA (Card CardType CheckptPtr)                              (* rht%: "16-Jul-86 23:18")

          (* * Make a menu containing items for each title version on Stream for Card.
          Menu item contains date and title. There is no further detail obtainable for titles.
          Return cons of menu and current selection.)

          (* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

    (LET ((Versions (NC.FetchScavengerTitleInfo Card))
          (CurTitleLoc (fetch (Card TitleLoc) of Card))
          MenuItems SelectionNum Menu)
         [SETQ MenuItems (for TitleInfo in Versions as ItemNum from 1
                            collect (LET ((Loc (CAR TitleInfo))
                                          (Date (CADR TitleInfo))
                                          (Title (CADDR TitleInfo))
                                          Item)
                                         (PROG1 (SETQ Item (LIST (CONCAT (if (GEQ Loc CheckptPtr)
                                                                             then "*"
                                                                           else "")
                                                                         "["
                                                                         (SUBATOM Title 1 (MIN 10 (NCHARS Title)))
                                                                         "] "
                                                                         (OR Date "NO DATE AVAILABLE"))
                                                                 Loc))
                                                (if (EQUAL Loc CurTitleLoc)
                                                    then (SETQ SelectionNum ItemNum)))]
         (if (NULL SelectionNum)
             then (SETQ MenuItems (CONS (LIST 'BADTITLE CurTitleLoc)
                                        MenuItems))
                  (SETQ SelectionNum 1))
         (PROG1 (CONS (SETQ Menu (create MENU
```

```
                                    ITEMS _ MenuItems
                                    TITLE _ (CONCAT (if (EQ (NC.FetchStatus Card)
                                                            'DELETED)
                                                        then "DELETED "
                                                    else "")
                                                    CardType " | Title Versions")
                                    WHENSELECTEDFN _ (FUNCTION NC.CardPartsMenusWhenSelectedFn)))
                SelectionNum)                                   (* Seems like the only way to communicate to the
                                                                   whenselectedfn is through menuprops.)
        (PUTMENUPROP Menu 'VERSIONS Versions)
        (PUTMENUPROP Menu 'INSPECTORFN (FUNCTION NC.CardTitleVersionInspector)))])
```

(NC.BuildSubstancesInspectorMenu

```
[LAMBDA (Card CardType CheckptPtr CardsMenuWindow)              (* rht%: " 8-Dec-85 16:35")
```

(* * Make a menu containing items for each title version on Stream for Card.
Menu item contains date and title. There is no further detail obtainable for titles.
Return cons of menu and current selection.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

```
  (LET ((Versions (NC.FetchScavengerMainDataInfo Card))
        (CurMainDataLoc (fetch (Card MainLoc) of Card))
        [BadTypeFlg (FMEMB CardType (WINDOWPROP (MAINWINDOW CardsMenuWindow)
                                                'UNKNOWNCARDTYPESLIST]
        SelectionNum MenuItems Menu)
       [OR BadTypeFlg (SETQ MenuItems (for SubstanceInfo in Versions as ItemNum from 1
                                         collect (LET ((Loc (CAR SubstanceInfo))
                                                       (Date (CADR SubstanceInfo))
                                                       (Type (CADDR SubstanceInfo))
                                                       (SubstanceLength (CADDDR SubstanceInfo))
                                                       Item)
                                                      (PROG1 (SETQ Item
                                                                   (LIST (CONCAT (if (GEQ Loc CheckptPtr)
                                                                                     then "*"
                                                                                 else "")
                                                                                 "[" SubstanceLength "] "
                                                                                 (OR Date "NO DATE AVAILABLE"))
                                                                         Loc))
                                                             (if (EQUAL Loc CurMainDataLoc)
                                                                 then (SETQ SelectionNum ItemNum)))]
       (if (NULL SelectionNum)
           then (SETQ MenuItems (CONS (LIST (if BadTypeFlg
                                                then 'UNKNOWNCARDTYPE
                                              else 'BADMAINDATA)
                                            CurMainDataLoc)
                                      MenuItems))
                (SETQ SelectionNum 1))
       (PROG1 (CONS (SETQ Menu (create MENU
                                       ITEMS _ MenuItems
                                       TITLE _ "Substance Versions"
                                       WHENSELECTEDFN _ (FUNCTION NC.CardPartsMenusWhenSelectedFn)))
                    SelectionNum)                               (* Seems like the only way to communicate to the
                                                                   whenselectedfn is through menuprops.)
        (PUTMENUPROP Menu 'VERSIONS Versions)
        (PUTMENUPROP Menu 'INSPECTORFN (FUNCTION NC.CardSubstanceVersionInspector)))])
```

(NC.BuildLinksInspectorMenu

```
[LAMBDA (Card CheckptPtr)                                       (* rht%: " 8-Dec-85 16:37")
```

(* * Make a menu containing items for each title version on Stream for Card.
Menu item contains date and title. There is no further detail obtainable for titles.
Return cons of menu and current selection.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

```
  (LET ((Versions (NC.FetchScavengerLinksInfo Card))
        (CurLinksLoc (fetch (Card LinksLoc) of Card))
        MenuItems SelectionNum Menu)
       [SETQ MenuItems (for LinksInfo in Versions as ItemNum from 1
                          collect (LET ((Loc (CAR LinksInfo))
                                        (Date (CADR LinksInfo))
                                        (LinkListsLengths (CADDR LinksInfo))
                                        Item)
                                       (PROG1 (SETQ Item (LIST (CONCAT (if (GEQ Loc CheckptPtr)
                                                                           then "*"
                                                                       else "")
                                                                       "["
                                                                       (CAR LinkListsLengths)
                                                                       "|"
                                                                       (CADR LinkListsLengths)
                                                                       "|"
                                                                       (CADDR LinkListsLengths)
                                                                       "] "
                                                                       (OR Date "NO DATE AVAILABLE"))
```

```
                                                    Loc))
                                          (if (EQUAL Loc CurLinksLoc)
                                              then (SETQ SelectionNum ItemNum)))]
        (if (NULL SelectionNum)
            then (SETQ MenuItems (CONS (LIST 'BADLINKS CurLinksLoc)
                                       MenuItems))
                 (SETQ SelectionNum 1))
        (PROG1 (CONS (SETQ Menu (create MENU
                                        ITEMS _ MenuItems
                                        TITLE _ "Links Versions"
                                        WHENSELECTEDFN _ (FUNCTION NC.CardPartsMenusWhenSelectedFn)))
                     SelectionNum)                          (* Seems like the only way to communicate to the
                                                            whenselectedfn is through menuprops.)
            (PUTMENUPROP Menu 'VERSIONS Versions)
            (PUTMENUPROP Menu 'INSPECTORFN (FUNCTION NC.CardLinksVersionInspector)))])
```

(NC.BuildPropListsInspectorMenu

```
[LAMBDA (Card CheckptPtr)                                   (* rht%: " 8-Dec-85 19:04")
```

(* * Make a menu containing items for each title version on Stream for Card.
Menu item contains date and title. There is no further detail obtainable for titles.
Return cons of menu and current selection.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

```
   (LET ((Versions (NC.FetchScavengerPropListInfo Card))
         (CurPropListLoc (fetch (Card PropListLoc) of Card))
         MenuItems SelectionNum Menu)
        [SETQ MenuItems (for PropListInfo in Versions as ItemNum from 1
                           collect (LET ((Loc (CAR PropListInfo))
                                         (Date (CADR PropListInfo))
                                         (PropListLength (CADDR PropListInfo))
                                         Item)
                                        (PROG1 (SETQ Item (LIST (CONCAT (if (GEQ Loc CheckptPtr)
                                                                            then "*"
                                                                          else "")
                                                                        "[" PropListLength "] " (OR Date "NO DATE
                                                                                                      AVAILABLE"))
                                                                Loc))
                                               (if (EQUAL Loc CurPropListLoc)
                                                   then (SETQ SelectionNum ItemNum)))]
        (if (NULL SelectionNum)
            then (SETQ MenuItems (CONS (LIST 'BADPROPLIST CurPropListLoc)
                                       MenuItems))
                 (SETQ SelectionNum 1))
        (PROG1 (CONS (SETQ Menu (create MENU
                                        ITEMS _ MenuItems
                                        TITLE _ "PropList Versions"
                                        WHENSELECTEDFN _ (FUNCTION NC.CardPartsMenusWhenSelectedFn)))
                     SelectionNum)                          (* Seems like the only way to communicate to the
                                                            whenselectedfn is through menuprops.)
            (PUTMENUPROP Menu 'VERSIONS Versions)
            (PUTMENUPROP Menu 'INSPECTORFN (FUNCTION NC.CardPropListVersionInspector)))])
```

(NC.CardPartsMenusWhenSelectedFn

```
[LAMBDA (SelectedItem Menu MouseKey)                        ; Edited  3-Dec-87 18:59 by rht:
```

(* * Called when a version is selected from card stylesheet. Pop up menu offering choice of Inspect or ChangeSelection.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

```
   (LET ((MainWin (MAINWINDOW (WFROMMENU Menu)))
         (Items (fetch (MENU ITEMS) of Menu))
         (InspectItem '(Inspect 'Inspect "Bring up display of this version."))
         (ChangeSelectionItem '(ChangeSelection 'ChangeSelection "Change to this version."))
         OldShadedItem OperationItems)
        [SETQ OldShadedItem (CAR (FNTH Items (GETMENUPROP Menu 'CURITEMNUM]
        [SETQ OperationItems `(,@(if [NOT (FMEMB (CAR SelectedItem)
                                                 '(UNKNOWNCARDTYPE BADMAINDATA BADTITLE BADPROPLIST BADLINKS]
                                     then (LIST InspectItem))
                                 ,@(if (NEQ OldShadedItem SelectedItem)
                                     then (LIST ChangeSelectionItem]
        (SELECTQ (AND OperationItems (MENU (create MENU
                                                   ITEMS _ OperationItems)))
            (Inspect [APPLY* (GETMENUPROP Menu 'INSPECTORFN)
                             (WINDOWPROP MainWin 'CARD)
                             (SASSOC (CADR SelectedItem)
                                     (GETMENUPROP Menu 'VERSIONS]
                     'ABORT)
            (ChangeSelection                                (* First unshade currently shaded item, then shade this one.)
                (SHADEITEM OldShadedItem Menu)
                (SHADEITEM SelectedItem Menu NC.LightShade)
                (PUTMENUPROP Menu 'CURITEMNUM (for Item in Items as i from 1 when (EQ SelectedItem Item)
                                                 do (RETURN i))))
            'ABORT])
```

(NC.CardTitleVersionInspector

```
[LAMBDA (Card TitleInfo)                                    (* rht%: " 8-Dec-85 16:43")
```

(* * Fill in a TITLEDATA record and bring up an inspector on it.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

```
    (INSPECT (create TITLEDATA
                    CARD _ Card
                    VERSIONDATE _ (CADR TitleInfo)
                    TITLE _ (CADDR TitleInfo))
             'TITLEDATA])
```

(NC.CardSubstanceVersionInspector

```
[LAMBDA (Card SubstanceInfo)                                (* rht%: "26-Mar-86 12:26")
```

(* * Fill in a SUBSTANCEDATA record and bring up an inspector on it.)

(* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

(* * rht 3/22/86%: Fixed to handle substance version nums. Now calls NC.ApplyFn and handles Hash card types. Also changed the way text, graph, and sketch cards are handled.)

```
    (LET ((NoteFile (fetch (Card NoteFile) of Card))
          Stream Type WindowTitle Sketch SketchObjDatum Length SubstanceVersionNum ListMenu TempStream StartLoc)
```

(* Position file at main card data. Hopefully no need to check validity of what we're reading.)

```
         (SETFILEPTR (SETQ Stream (fetch (NoteFile Stream) of NoteFile))
                (CAR SubstanceInfo))
         (NC.ReadCardPartHeader Card NC.ItemIdentifier)
         [NC.SetType Card (SETQ Type (CAR (NC.RobustReadAtom Stream]
         (NC.RobustReadRegion Stream (GETEOFPTR Stream))
         (SETQ Length (NC.ReadPtr Stream 3))
         (SETQ SubstanceVersionNum (NC.ReadPtr Stream 1))
         (SETQ WindowTitle (CONCAT Type ": " (NC.ShaveTitleString (NC.FetchTitleFromScavengerInfo Card))
                                   " | Date: "
                                   (CADR SubstanceInfo)))        (* Unfortunately, can only inspect system-defined substances now.)
         (COND
            ((NC.IsSubTypeOfP Type 'Text)
```

(* This is so that user edits a copy of the text and can't affect original.)

```
             (COPYBYTES Stream (SETQ TempStream (OPENSTREAM '{NODIRCORE} 'BOTH 'NEW))
                    (SETQ StartLoc (GETFILEPTR Stream))
                    (PLUS StartLoc Length))
             (OPENTEXTSTREAM (COPYTEXTSTREAM (OPENTEXTSTREAM TempStream)
                                             T)
                    (CREATEW (PROGN (NC.PrintMsg NIL T "Choose region for Text card display.")
                                    (GETREGION))
                             WindowTitle)))
            ((NC.IsSubTypeOfP Type 'Graph)
             (SHOWGRAPH (NC.MakeExternalGraphCopy (NC.ApplyFn GetFn Card Length Stream SubstanceVersionNum))
                    (CREATEW (PROGN (NC.PrintMsg NIL T "Choose region for Graph card display.")
                                    (GETREGION))
                             WindowTitle)
                    NIL NIL NIL T))
            ((NC.IsSubTypeOfP Type 'Sketch)
             (SETQ Sketch (NC.ApplyFn GetFn Card Length Stream SubstanceVersionNum))
             (SKETCHW.CREATE (NC.ExternalizeLinkIconsInSketch Sketch)
                    (NC.FetchRegionViewed Card)
                    (PROGN (NC.PrintMsg NIL T "Choose region for Sketch card display.")
                           (GETREGION))
                    WindowTitle
                    (NC.FetchScale Card)))
            ((NC.IsSubTypeOfP Type 'List)
             (SETQ ListMenu (create MENU
                                    ITEMS _ (NC.ApplyFn GetFn Card Length Stream SubstanceVersionNum)
                                    TITLE _ WindowTitle))
             (ADDMENU ListMenu NIL (GETBOXPOSITION (fetch (MENU IMAGEWIDTH) of ListMenu)
                                                   (fetch (MENU IMAGEHEIGHT) of ListMenu)
                                                   NIL NIL NIL "Choose region for List card display.")))
            ((NC.IsSubTypeOfP Type 'Hash)
             (WINDOWPROP (INSPECT (NC.ApplyFn GetFn Card Length Stream SubstanceVersionNum))
                    'TITLE WindowTitle))
            (T (NC.PrintMsg NIL T "Sorry, can only inspect card types inheriting from Text, Graph, Sketch, List
                      or Hash."])
```

(NC.CardLinksVersionInspector

```
[LAMBDA (Card LinksInfo)                                    (* rht%: " 8-Dec-85 16:55")
```

(* * Fill in a LINKSDATA record and bring up an inspector on it.)

```
          (* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

    (LET ((NoteFile (fetch (Card NoteFile) of Card))
          Stream)

          (* Position file at links lists. Hopefully no need to check validity of what we're reading.)

         (SETFILEPTR (SETQ Stream (fetch (NoteFile Stream) of NoteFile))
                (CAR LinksInfo))
         (NC.ReadCardPartHeader Card NC.LinksIdentifier)
         (INSPECT (create LINKSDATA
                          CARD _ Card
                          VERSIONDATE _ (CADR LinksInfo)
                          TOLINKS _ (CAR (NC.RobustReadLinks Stream NoteFile))
                          FROMLINKS _ (CAR (NC.RobustReadLinks Stream NoteFile))
                          GLOBALLINKS _ (CAR (NC.RobustReadLinks Stream NoteFile)))
                'LINKSDATA)
         (NC.PrintMsg NIL T "Inspect individual links as datatype Link."])
```

(NC.CardPropListVersionInspector

```
  [LAMBDA (Card PropListInfo)                                   (* rht%: " 8-Dec-85 16:59")

          (* * Fill in a PROPLISTDATA record and bring up an inspector on it.)

          (* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

    (LET ((NoteFile (fetch (Card NoteFile) of Card))
          Stream IdentifierAndVersionNum)

          (* Position file at prop list. Hopefully no need to check validity of what we're reading.)

         (SETFILEPTR (SETQ Stream (fetch (NoteFile Stream) of NoteFile))
                (CAR PropListInfo))
         (NC.ReadCardPartHeader Card NC.PropsIdentifier)
         (INSPECT (create PROPLISTDATA
                          CARD _ Card
                          VERSIONDATE _ (CADR PropListInfo)
                          PROPLIST _ (CAR (NC.RobustReadList Stream)))
                'PROPLISTDATA])
```

(NC.CloseInspectorWindows

```
  [LAMBDA (MessageWin)                                          (* rht%: "24-Mar-86 18:25")

          (* * Close any open inspector windows.)

    (for InspectorWin in (WINDOWPROP MessageWin 'INSPECTORWINDOWS) when (OPENWP InspectorWin)
       do (CLOSEW InspectorWin))
    (WINDOWPROP MessageWin 'INSPECTORWINDOWS NIL])

)


;;; Miscellaneous.

(DEFINEQ
```

(NC.CheckForBadLinksAndTitlesAndPropLists

```
  [LAMBDA (NoteFile MessageWin BadNewsList)                     (* kirk%: "23-Jan-86 21:35")

          (* * For any cards with bad links, set links to NIL if user agrees.
          This will only really destroy global links, since Tolinks and Fromlinks will get rebuilt by link scavenger later.
          Look around for bad titles and see if user wants to set them all to "Untitled" Untitled.
          Then check for cards with bad prop lists and reset to NIL if user wants.
          Set a windowprop on the MessageWin to the cards with bad links, if they got reset.
          Then the 2nd phase of the scavenger can rebuild those cards' global links.)

          (* * rht 12/8/85%: Modified to reflect new card and notefile object formats.)

          (* * kirk 23Jan86 Changed to use NC.AskYesOrNo)

    (LET (ActiveCardsWithBadNews CardsWithBadLinks CardsWithBadTitles CardsWithBadPropLists)
                                                                (* Collect active cards on the bad news list.)
         (SETQ ActiveCardsWithBadNews (for BadNews in BadNewsList bind Card
                                         when (EQ (NC.FetchStatus (SETQ Card (CAR BadNews)))
                                                  'ACTIVE)
                                         collect Card))         (* Collect active cards that still have bad links.)
         (SETQ CardsWithBadLinks (for Card in ActiveCardsWithBadNews unless (SASSOC (fetch (Card LinksLoc)
                                                                                       of Card)
                                                                                    (NC.FetchScavengerLinksInfo Card))
                                    collect Card))              (* Collect active cards that still have bad titles.)
         (SETQ CardsWithBadTitles (for Card in ActiveCardsWithBadNews unless (SASSOC (fetch (Card TitleLoc)
                                                                                       of Card)
                                                                                    (NC.FetchScavengerTitleInfo Card))
                                     collect Card))             (* Collect active cards that still have bad prop lists.)
```

```
        (SETQ CardsWithBadPropLists (for Card in ActiveCardsWithBadNews unless (SASSOC (fetch (Card PropListLoc)
                                                                                          of Card)
                                                                                   (NC.FetchScavengerPropListInfo
                                                                                    Card))
                                           collect Card))
        (if (OR CardsWithBadLinks CardsWithBadTitles CardsWithBadPropLists)
            then                                                    (* Print out the bad news.)
                 (if CardsWithBadLinks
                     then (NC.PrintMsg MessageWin NIL "Cards: " CardsWithBadLinks " still have bad links.  Links
                                rebuilder will rebuild them." (CHARACTER 13)))
                 (if CardsWithBadTitles
                     then (NC.PrintMsg MessageWin NIL "Cards: " CardsWithBadTitles " still have bad titles.
                                They will be set to 'Untitled'." (CHARACTER 13)))
                 (if CardsWithBadPropLists
                     then (NC.PrintMsg MessageWin NIL "Cards: " CardsWithBadPropLists " still have bad prop
                                lists.  They will be set to NIL." (CHARACTER 13)))

         (* If it's okay with user, then reset links, titles and prop lists for those cards.)

                 (if (NC.AskYesOrNo "Is this okay?" NIL 'Yes NIL MessageWin (NC.AttachPromptWindow MessageWin)
                           NIL NIL)
                     then (WINDOWPROP MessageWin 'CARDSWITHLINKSRESET (if CardsWithBadLinks
                                                                          then (NC.PrintMsg MessageWin NIL
                                                                                    "Resetting links ...")
                                                                               (for Card in CardsWithBadLinks
                                                                                  do (NC.SetToLinks Card NIL)
                                                                                     (NC.SetFromLinks Card NIL)
                                                                                     (NC.SetGlobalLinks Card NIL
                                                                                           )
                                                                                     (NC.PutLinks Card))
                                                                               (NC.PrintMsg MessageWin NIL
                                                                                    "Done.")
                                                                               CardsWithBadLinks))
                          (if CardsWithBadTitles
                              then (NC.PrintMsg MessageWin NIL "Resetting titles ...")
                                   (for Card in CardsWithBadTitles do (NC.SetTitle Card "Untitled")
                                                                      (NC.PutTitle Card)
                                                                      (NC.SetTitle Card NIL))
                                   (NC.PrintMsg MessageWin NIL "Done."))
                          (if CardsWithBadPropLists
                              then (NC.PrintMsg MessageWin NIL "Resetting prop lists ...")
                                   (for Card in CardsWithBadPropLists do (NC.SetPropList Card NIL)
                                                                         (NC.PutPropList Card))
                                   (NC.PrintMsg MessageWin NIL "Done."))
                    else 'ABORT])
```

(NC.UndeletableCardP

```
  [LAMBDA (Card)                                                    (* rht%: " 2-Mar-86 15:09")

          (* * Returns non-nil if this card is either a top level card or the link labels card.)

          (* * rht 3/2/86%: Changed to use all special cards not just top level boxes.)

    (for SpecialCard in (NC.FetchSpecialCards (fetch (Card NoteFile) of Card)) thereis (NC.SameCardP Card
                                                                                              SpecialCard])
```

(NC.ShaveTitleString

```
  [LAMBDA (TitleString)                                             (* rht%: " 8-Dec-85 19:36")

          (* * Cut off the end of TitleString.)

    (SUBSTRING TitleString 1 (MIN (NCHARS TitleString)
                                  NC.NumCharsOfTitleToShow])
```

(NC.SortCardsBySlotNums

```
  [LAMBDA (Cards)                                                   (* rht%: " 9-Dec-85 19:14")

          (* * Call SORT with a CompareFn that compares cards on basis of slot num)

    (SORT Cards (FUNCTION (LAMBDA (Card1 Card2)
                            (LESSP (NC.FetchSlotNum Card1)
                                   (NC.FetchSlotNum Card2])

)


;;; Possible reasons for bad card parts.

(LOADINITPROPS (BADMAINDATA (ReasonString "improper main card data."))
       (BADLINKS (ReasonString "improper links data."))
       (BADTITLE (ReasonString "improper title data."))
       (BADPROPLIST (ReasonString "improper prop list data."))
       (MAINDATAPASTCHKPT (ReasonString "main card data beyond chkpt pointer."))
       (LINKSPASTCHKPT (ReasonString "links beyond chkpt pointer."))
```

```
       (TITLEPASTCHKPT (ReasonString "title beyond chkpt pointer."))
       (PROPLISTPASTCHKPT (ReasonString "prop list beyond chkpt pointer."))
       (UNKNOWNCARDTYPE (ReasonString "card type definition not loaded.")))

(PUTPROPS NCREPAIR FILETYPE :FAKE-COMPILE-FILE)

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

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)

(PUTPROPS NCREPAIR COPYRIGHT ("Venue & Xerox Corporation" 1985 1986 1987 1988 1989 1990 1993 1994 2020))
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX