*File created:* 25-Mar-2024 12:53:03 **{DSK}<home>frank>il>notecards>system>NCCARDS.;16**

*changes to:* (FNS NC.EditPropList)

*previous date:* 10-Mar-2024 23:36:16 {DSK}<home>frank>il>notecards>system>NCCARDS.;15

*Read Table:* INTERLISP

*Package:* INTERLISP

*Format:* XCCS

```
(RPAQQ NCCARDSCOMS
       (
```

;;; Internal variables

```
        (GLOBALVARS NC.SubBoxMarkerLabel NC.FiledCardMarkerLabel NC.PlaceMarkerDisplayFont USERNAME
               NC.SavePropEditMenu TEDIT.SELPENDING NC.SystemLinkLabels NC.SourceLinkLabel NC.FiledCardLinkLabel
               NC.CloseCardsOffScreenFlg NC.BringUpCardAtOldPosFlg NC.DefaultLinkIconAttachedBitMapHeights
               NC.SystemCardPropNames NC.UnknownLinkIconAttachedBitMap NC.UnknownLinkIconAttachedBitMaps)
        (BITMAPS NC.UnknownLinkIconAttachedBitMap)
        (INITVARS (NC.SubBoxMarkerLabel "File Boxes")
               (NC.FiledCardMarkerLabel "Note Cards")
               (NC.PlaceMarkerDisplayFont (FONTCREATE 'HELVETICA 12 'ITALIC))
               (NC.SystemCardPropNames '(Updates))
               (NC.DefaultLinkIconAttachedBitMapHeights '(11 13 15 17 18 19 20 21 23 25 27 29 31 33 35 37 39))
               (NC.UnknownLinkIconAttachedBitMaps NC.UnknownLinkIconAttachedBitMap))
```

;;; Setup the Card ICON info

```
        (GLOBALVARS NC.CardIconMaster NC.CardIconMasterMask)
        (BITMAPS NC.CardIconMaster NC.CardIconMasterMask)
        [INITVARS (NC.CardIconTileReg '(26 3 141 18]
        [INITVARS (NC.IconFont (FONTCREATE 'HELVETICA 10 '(BOLD REGULAR REGULAR)
                                    NIL
                                    'DISPLAY]
        (INITVARS (NC.IconProps NIL))
        (FNS NC.AddCardIconToIconProps)
```

;;; The Card Objects

```
        (FNS NC.CardOrCardHolderFromUID NC.CardP NC.DeleteNoteCard NC.DeleteNoteCardInternal NC.SameCardP)
```

;;; Manipulate Note Card Representations

```
        (FNS NC.FetchStatus NC.SetNewCardFlg NC.FetchNewCardFlg NC.FetchNewCardPos NC.ActivateCard
             NC.ActiveCardP NC.DeactivateCard NC.FetchFromLinks NC.FetchGlobalLinks NC.FetchLinksDirtyFlg
             NC.FetchPropList NC.FetchRegion NC.FetchSavedRegion NC.FetchRegionViewed NC.FetchScale
             NC.FetchSlotNum NC.FetchSubstance NC.FetchTitle NC.FetchToLinks NC.FetchType NC.FetchWindow NC.IDP
             NC.MarkCardDirty NC.CardDirtyP NC.SetFromLinks NC.SetGlobalLinks NC.SetLinksDirtyFlg NC.SetPropList
             NC.SetRegion NC.SetSavedRegion NC.SetRegionViewed NC.SetScale NC.SetSubstance NC.SetTitle
             NC.SetToLinks NC.SetType NC.FetchTitleDirtyFlg NC.SetTitleDirtyFlg NC.FetchPropListDirtyFlg
             NC.SetPropListDirtyFlg NC.SetSubstanceDirtyFlg NC.FetchSubstanceDirtyFlg NC.TurnOffDirtyFlgs
             NC.FetchUserDataProp NC.SetUserDataProp NC.SetUserDataPropList NC.MakeTypeIconBitMapSet
             NC.MakeNewCardWindow NC.FlashHiddenWindow NC.OpenWindows)
```

;;; Read subscriptions and write locks on card parts

```
        (FNS NC.ObtainEditPermission NC.CardPartBusy)
```

;;; Stuff to handle read-only notefiles.

```
        (FNS NC.ReadOnlyCardP NC.CardSomehowDirtyP NC.CardReadOnlyOpenP)
```

;;; Retrieve Note Card Info

```
        (FNS NC.RetrieveFromLinks NC.RetrieveLinkLabels NC.StoreLinkLabels NC.RetrievePropList NC.RetrieveTitle
             NC.NameOfCardorNodeorImageObj NC.RetrieveToLinks NC.RetrieveGlobalLinks NC.RetrieveType
             NC.RetrieveTypeAndTitle NC.LinksCachedP NC.ShowInfo)
```

;;; General note card manipulations

```
        (FNS NC.AddParents NC.SeverExternalLinks NC.AssignTitle NC.DeleteNoteCards NC.CardsToDelete
             NC.EditNoteCard NC.MakeNoteCard NC.QuitCard NC.CloseW NC.CheckFiling NC.CheckTitle
             NC.ForceFilingForCardTypeP NC.InsureProperFiling NC.QuitWithoutSaving NC.UnfileNoteCard
             NC.UpdateUpdateList NC.CollectReferences NC.CardSaveFn NC.DetermineDisplayRegion NC.AbortCard
             NC.CardNeedsFilingP NC.TopLevelCardP NC.MakeDummyRegion NC.ValidCardP NC.TitleBarButtonEventFn
             NC.InstallTitleBarMiddleMenu)
```

;;; Prop List Editor

```
        (FNS NC.AddPropToPropList NC.CloseAllPropListEditors NC.ClosePropListEditor NC.DelPropFromList
             NC.EditPropButtonFN NC.EditProperties NC.EditPropList NC.ExtractPropList NC.OpenPropListEditor
             NC.ProcessEditedPropList NC.PropListEditorOpenP NC.SelectProperty NC.ShowLinks NC.StringIsListP
             NC.PutProp NC.GetProp NC.RemProp)
```

;;; Unknown ?????????

```
        (FNS NC.FetchBeingDeletedFlg NC.SetBeingDeletedFlg NC.RemoveDELETEMEImageObjsFromCard
             NC.InsureIntegerDate)
```

;;; Place marker ImageObjects

```
        (FNS NC.PlaceMarkerCopyFn NC.PlaceMarkerDisplayFn NC.PlaceMarkerGetFn NC.PlaceMarkerImageBoxFn
             NC.PlaceMarkerPutFn NC.MakePlaceMarker)
```

;;; Functions for handling dates.

```
        (FNS NC.FetchTitleDate NC.FetchItemDate NC.FetchPropListDate NC.FetchLinksDate
             NC.FetchLinkIconAttachedBitMap NC.FetchLinkDisplayMode NC.FetchDefaultWidth NC.FetchDefaultHeight
             NC.FetchLinkAnchorModesSupported)
        (FNS NC.SetTitleDate NC.SetItemDate NC.SetPropListDate NC.SetLinksDate)
        (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              NCCARDS)))
```

;;; Internal variables

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.SubBoxMarkerLabel NC.FiledCardMarkerLabel NC.PlaceMarkerDisplayFont USERNAME NC.SavePropEditMenu
       TEDIT.SELPENDING NC.SystemLinkLabels NC.SourceLinkLabel NC.FiledCardLinkLabel NC.CloseCardsOffScreenFlg
       NC.BringUpCardAtOldPosFlg NC.DefaultLinkIconAttachedBitMapHeights NC.SystemCardPropNames
       NC.UnknownLinkIconAttachedBitMap NC.UnknownLinkIconAttachedBitMaps)
)
```

(RPAQQ **NC.UnknownLinkIconAttachedBitMap**  )

```
(RPAQ? NC.SubBoxMarkerLabel "File Boxes")

(RPAQ? NC.FiledCardMarkerLabel "Note Cards")

(RPAQ? NC.PlaceMarkerDisplayFont (FONTCREATE 'HELVETICA 12 'ITALIC))

(RPAQ? NC.SystemCardPropNames '(Updates))

(RPAQ? NC.DefaultLinkIconAttachedBitMapHeights '(11 13 15 17 18 19 20 21 23 25 27 29 31 33 35 37 39))

(RPAQ? NC.UnknownLinkIconAttachedBitMaps NC.UnknownLinkIconAttachedBitMap)
```

;;; Setup the Card ICON info

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.CardIconMaster NC.CardIconMasterMask)
)
```

(RPAQQ **NC.CardIconMaster**  )

(RPAQQ **NC.CardIconMasterMask**  )

```
(RPAQ? NC.CardIconTileReg '(26 3 141 18))

(RPAQ? NC.IconFont (FONTCREATE 'HELVETICA 10 '(BOLD REGULAR REGULAR)
                        NIL
                        'DISPLAY))

(RPAQ? NC.IconProps NIL)

(DEFINEQ
```

## (NC.AddCardIconToIconProps

```
  [LAMBDA (CARD-TYPE SUB-ICON)
    (DECLARE (GLOBALVARS NC.CardIconMaster NC.CardIconMasterMask NC.CardIconTileReg))
                                                                  ; Edited 2-Dec-88 13:44 by krivacic
    (LET ((NEW-BITMAP (BITMAPCOPY NC.CardIconMaster)))
```

```
        (BITBLT SUB-ICON 0 0 NEW-BITMAP 2 4 21 19 'INPUT 'PAINT)
        (SETQ NC.IconProps (CONS CARD-TYPE (CONS (create TITLEDICON
                                                         ICON _ NEW-BITMAP
                                                         MASK _ NC.CardIconMasterMask
                                                         TITLEREG _ NC.CardIconTileReg)
                                                 NC.IconProps])

)

;;; The Card Objects

(DEFINEQ
```

## (NC.CardOrCardHolderFromUID

```
  [LAMBDA (CardUID NoteFileUID)                                    (* kirk%: "16-May-86 16:21")

          (* * Get the CardObject specified by CardUID and NoteFIleUID.
          This will have to be expanded a great deal when there are crosslinked NFs.)

          (* * fgh |5/7/86| Added card holder notion. If NF does not exists send back a new card object holding just the given ID's)

          (* * kirk 16May86 fixed above to work if NoteFile exists but is closed)

    (LET (NoteFile)
         (if (AND (SETQ NoteFile (NC.NoteFileFromNoteFileUID NoteFileUID))
                  (NC.CardFromUID CardUID NoteFile))
           else (create Card
                        UID _ CardUID
                        NoteFile _ NoteFileUID])
```

## (NC.CardP

```
  [LAMBDA (Card)                                                   (* rht%: "15-Nov-85 16:54")

          (* * Return non-NIL if the arg is a Card object.)

    (type? Card Card])
```

## (NC.DeleteNoteCard

```
  [LAMBDA (CardIdentifier DontClearFlg NoConfirmFlg QuietFlg InterestedWindow)
                                                                   (* pmi%: "15-Jan-88 11:49")

          (* * User interface level fn to delete a single note card from a NoteFile)

          (* * rht 3/25/87%: Added a bunch of new args. Fixed InterestedWindow stuff.)

          (* * 3/26/87%: Now clears InterestedWindow when done.)

          (* * pmi 3/27/87%: Changed call of NC.SeverAllLinks to NC.SeverExternalLinks.)

          (* * dsj 9/28/87%: Added call to NC.CardsToDelete which accesses the WhenDeletedFn prop of card type.
          This does not yet handle the general case of allowing me to bypass asking the user for confirmation, which is an operation
          that should be a super type's WhenDeletedFn.)

          (* * pmi 1/15/88%: Added dsj's change (see above comment))

    (LET ((Card (NC.CoerceToCard CardIdentifier)))
         (if (NC.ValidCardP Card)
             then (OR InterestedWindow (SETQ InterestedWindow (NC.CoerceToInterestedWindow Card)))
                  (NC.ProtectedCardOperation Card "Delete Note Card" InterestedWindow
                         (AND (if (NC.TopLevelCardP Card)
                                  then (NC.PrintMsg InterestedWindow T "You cannot delete this FileBox."
                                               (CHARACTER 13))
                                       (DISMISS 1000)
                                       (NC.ClearMsg InterestedWindow T)
                                       NIL
                                else T)
                              (NC.CheckForNotReadOnly Card InterestedWindow "Can't delete cards from a ")
                              (OR NoConfirmFlg (PROG1 (NC.AskYesOrNo "Are you sure you want to delete this?"
                                                             " -- " "Yes" (NULL DontClearFlg)
                                                             InterestedWindow NIL NIL)
                                                      (NC.ClearMsg InterestedWindow T)))
                              (if (NC.CardsToDelete Card)
                                  then T
                                else (NC.PrintMsg InterestedWindow T "This card cannot be deleted." (CHARACTER
                                                                                                        13))
                                     (DISMISS 1000)
                                     (NC.ClearMsg InterestedWindow T)
                                     NIL)
                              (PROGN

          (* * Mark UID of card about to be deleted.)

                                     (NC.UIDPutProp (fetch (Card UID) of Card)
                                            'AboutToBeDeletedFlg T)
```

(* * Sever all links into and out of Card)

```
(NC.SeverExternalLinks (LIST Card)
       QuietFlg InterestedWindow)
```

(* * Now delete the card)

```
(PROG1 (NC.DeleteNoteCardInternal Card QuietFlg InterestedWindow)
    (OR QuietFlg (NC.ClearMsg InterestedWindow T)))])
```

## (NC.DeleteNoteCardInternal

```
[LAMBDA (Card QuietFlg InterestedWindow IgnoreLinksFlg)        ; Edited 3-Dec-87 19:00 by rht:
```

(* * Delete a single note card from a NoteFile)

(* * rht |8/11/86| Now calls NC.DeleteReferencesToCardFromShowLinks to smash any link icons in show links menus pointing to this card.)

(* * kef 7/28/86%: Added code to obtain all of the write locks deemed necessary.)

(* * kef 7/30/86%: Modified to check for Client's concept of whether he owns the write lock or not, thus deciding whether or not to setup the release of the write lock afterwards.)

(* * kef 7/31/86%: Added the nesting of the obtaining writelocks with deactivating the card.)

(* * fgh |8/30/86| Translated APPLY* to NC.ApplyFn. Made cosmetic changes in FOR loop concewrning WriteLocks.)

(* * rht 1/16/87%: Moved call to MarkCardDeletedFn after call to NC.QuitCard.)

(* * rg |3/11/87| renamed)

(* * rht 3/13/87%: Added QuietFlg, InterestedWindow, and IgnoreLinksFlg args.
The latter non-nil causes card deletion but no link cutting.)

(* * rht 3/25/87%: Now calls NC.CoerceToInterestedWindow.)

(* * rht 5/27/87%: Changed to match reduced functionality of NC.ValidLinkP, now checks that DestinationCard and SourceCard are valid cards.)

(* * pmi 6/8/87%: Moved call to NC.GreyCard to beginning of this function.)

```
(RESETLST
    (RESETSAVE (CURSOR WAITINGCURSOR))
    (RESETSAVE NIL `(NC.SetBeingDeletedFlg ,Card NIL))
    (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
        (OR (OPENWP InterestedWindow)
            (SETQ InterestedWindow (NC.CoerceToInterestedWindow Card)))
        [LET ((WriteLocks (for CardPart in '(SUBSTANCE TITLE TOLINKS FROMLINKS GLOBALTOLINKS PROPLIST)
                             collect (CONS Card CardPart)))
              ToLinks FromLinks Window BusyPart)
             (NC.SetBeingDeletedFlg Card T)
```

(* * First grey the card being deleted.)

```
             (NC.GreyCard Card)
             [if (NOT IgnoreLinksFlg)
                 then (SETQ ToLinks (NC.RetrieveToLinks Card))
                      (SETQ FromLinks (NC.RetrieveFromLinks Card))
                      [for ToLink in ToLinks do (NCONC WriteLocks (for CardPart
                                                                     in '(TOLINKS FROMLINKS GLOBALTOLINKS)
                                                                     collect (CONS (fetch (Link DestinationCard)
                                                                                      of ToLink)
                                                                                   CardPart]
                      (for FromLink in FromLinks do (NCONC WriteLocks
                                                           (for CardPart
                                                              in '(SUBSTANCE TOLINKS FROMLINKS GLOBALTOLINKS)
                                                              collect (CONS (fetch (Link SourceCard)
                                                                               of FromLink)
                                                                            CardPart]
```

(* * The for... loop that follows is the condition that we can obtain all of the write locks on all of the card parts collected in WriteLocks. If we obtain a writelock, setup a RESETSAVE to release it upon exit.
Then finally, if we do obtain a write lock, return T so that the for loop's "always" condition is satisfied.
If we don't obtain a write lock for a given card part, save that one as a variable so we can report it to the user.
Then return NIL, so that the for loop's "always" condition is not satisfied, and we bump out of the for loop.)

```
             (OR QuietFlg (NC.PrintMsg InterestedWindow T "Collecting write locks..."))
             (COND
                ((for WriteLock in WriteLocks bind WLCard WLCardPart when (NC.ValidCardP (SETQ WLCard
                                                                                            (CAR WriteLock)))
                    always (SETQ WLCardPart (CDR WriteLock))
                           (COND
                              ((NC.ApplyFn ObtainWritePermissionFn WLCard WLCardPart)
                               [RESETSAVE NIL `(APPLY* ,(fetch (Card ReleaseWritePermissionFn) of WLCard)
                                                       ,WLCard
```

```
                                                    ,WLCardPart]
                                T)
                               (T (SETQ BusyPart WriteLock)
                                  NIL)))
```

(* * Call off to the MarkCardDeletedFn specific to the NoteFile.)

```
                        (RESETSAVE (for CardPart in '(SUBSTANCE TOLINKS GLOBALTOLINKS PROPLIST)
                                      do (NC.ApplyFn ObtainWritePermissionFn Card CardPart))
                               `(NC.DeactivateCard ,Card T))
                        (NC.ApplyFn MarkCardDeletedFn Card)
                        (NC.SetToLinks Card NIL)
                        (NC.SetFromLinks Card NIL)
                        (if (NOT IgnoreLinksFlg)
                            then (OR QuietFlg (NC.PrintMsg InterestedWindow NIL "Removing links to and from other
                                                    cards..."))
                                 (for ToLink in ToLinks bind DestCard
                                    when [AND (NC.ValidLinkP ToLink)
                                              (NC.ValidCardP (SETQ DestCard (fetch (Link DestinationCard)
                                                                                of ToLink]
                                    do (NC.DelFromLink ToLink)
                                       (NC.DelReferencesToCardFromShowLinks DestCard ToLink))
                                 (for FromLink in FromLinks bind SourceCard
                                    when [AND (NC.ValidLinkP FromLink)
                                              (NC.ValidCardP (SETQ SourceCard (fetch (Link SourceCard) of FromLink]
                                    do (NC.DelToLink FromLink)
                                       (NC.DelReferencesToCard SourceCard Card)
                                       (NC.DelReferencesToCardFromShowLinks SourceCard FromLink))
                                 (OR QuietFlg (NC.PrintMsg InterestedWindow NIL "..done.")))
                        (if (NC.ActiveCardP Card)
                            then (NC.TurnOffDirtyFlgs Card)
                                 (NC.CloseAllPropListEditors Card)
                                 (NC.QuitCard Card T T))
                        (NC.ApplyFn MarkCardDeletedFn Card)
                        (NC.SetNewCardFlg Card NIL))
                       (T (NC.CardPartBusy (CAR BusyPart)
                                 (CDR BusyPart)
                                 InterestedWindow]))])
```

## (NC.SameCardP

```
  [LAMBDA (Card1 Card2)                                    (* fgh%: "14-Nov-85 00:26")
```

(* * are Card1 and card2 the same Card?)

```
    (EQ Card1 Card2])

)
```

;;; Manipulate Note Card Representations

```
(DEFINEQ
```

## (NC.FetchStatus

```
  [LAMBDA (Card)                                           (* fgh%: "13-Nov-85 19:52")
```

(* * Return the status of the card specified by Card)

```
    (fetch (Card Status) of Card])
```

## (NC.SetNewCardFlg

```
  [LAMBDA (Card Value)                                     (* fgh%: "13-Nov-85 19:00")
                                                           (* Set the new card flg of ID to Value)
```

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh 11/11/85%: Updated to handle Card object.)

```
    (replace (Card NewCardFlg) of Card with Value])
```

## (NC.FetchNewCardFlg

```
  [LAMBDA (Card)                                           (* fgh%: "13-Nov-85 19:01")
                                                           (* Return the value of the new card flg of ID)
```

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh 11/11/85%: Updated to handle Card object)

```
    (fetch (Card NewCardFlg) of Card])
```

## (NC.FetchNewCardPos

```
  [LAMBDA (Card)                                           ; Edited  9-Jun-88 20:49 by Trigg
    ;; Get the default NewCardPos for this card.
```

```
;; This should do a (fetch (Card NewCardPos) of Card) with accessfns set up exactly like DefaultCardWidth slot works.  For now, it's hacked to grab
;; NewCardPos property of card type atom.
(GETPROP (NC.FetchType Card)
       'NewCardPos])
```

## (NC.ActivateCard

```
[LAMBDA (Card)                                                    (* fgh%: "13-Nov-85 19:02")

        (* * Set the active card flg for ID)

        (* * fgh |11/13/85| Updated to handle Card object)

   (replace (Card ActiveFlg) of Card with T])
```

## (NC.ActiveCardP

```
[LAMBDA (Card)                                                    (* fgh%: "13-Nov-85 18:41")

        (* * If the active card flg is set in the cache for ID, the return ID's type.
        Otherwise return NIL.)

        (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

        (* * fgh |11/13/85| Updated to handle Card object.)

   (AND Card (fetch (Card ActiveFlg) of Card)
        (NC.FetchType Card])
```

## (NC.DeactivateCard

```
[LAMBDA (Card DeleteTypeAndTitleFlg)                              (* fgh%: "30-Aug-86 00:11")

        (* * Remove all the information from the prop list of the NoteCard ID, except for the title which usually statys cached.)

        (* * rht 7/9/85%: Now also removes the new date properties.)

        (* * fgh |10/15/85| altered to use new caching mechanism)

        (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

        (* * fgh |11/13/85| Updated to handle Card object.)

        (* * kef 7/16/86%: Added the release of the write permission and cancel cache subscription.)

        (* * fgh |8/30/86| Converted APPLY* to NC.ApplyFn.)

   (replace (Card ActiveFlg) of Card with NIL)
   (replace (Card CardCache) of Card with NIL)
   (for CardPart in '(SUBSTANCE TOLINKS GLOBALTOLINKS PROPLIST) do (NC.ApplyFn ReleaseWritePermissionFn Card
                                                                          CardPart)
                                                                  (NC.ApplyFn CancelCacheSubscriptionFn Card
                                                                          CardPart))
   (if DeleteTypeAndTitleFlg
       then (replace (Card Type) of Card with NIL)
            (replace (Card Title) of Card with NIL)
            (NC.ApplyFn CancelCacheSubscriptionFn Card 'TITLE])
```

## (NC.FetchFromLinks

```
[LAMBDA (Card)                                                    (* fgh%: "13-Nov-85 19:17")

        (* * Fetch ID's links from the cache)

        (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

        (* * fgh |11/13/85| Updated to handle Card objects.)

   (fetch (Card FromLinks) of Card])
```

## (NC.FetchGlobalLinks

```
[LAMBDA (Card)                                                    (* fgh%: "13-Nov-85 19:18")

        (* * Fetch IDs global links from the cache)

        (* * fgh |11/13/85| Updated to handle Card objects.)

   (fetch (Card GlobalLinks) of Card])
```

## (NC.FetchLinksDirtyFlg

```
[LAMBDA (Card)                                                    (* fgh%: "13-Nov-85 19:18")

        (* * fetch IDs links dirty flag fom the cache)
```

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

(**fetch** (Card LinksDirtyFlg) **of** Card])

## (NC.FetchPropList

[LAMBDA (Card) (* fgh%: "13-Nov-85 19:19")

(* * fetych IDs prop list from the cache)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle card object.)

(**fetch** (Card PropList) **of** Card])

## (NC.FetchRegion

[LAMBDA (Card) (* fgh%: "13-Nov-85 19:19")

(* * fetch IDs region from the cache)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle card object.)

(**fetch** (Card Region) **of** Card])

## (NC.FetchSavedRegion

[LAMBDA (Card) (* fgh%: "13-Nov-85 19:20")

(* * fetch IDs saved region from the cache)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

(**fetch** (Card SavedRegion) **of** Card])

## (NC.FetchRegionViewed

[LAMBDA (Card) (* fgh%: "13-Nov-85 19:21")

(* * For sketch cards -- fetch from the UserData prop list the region viewed property)

(* * fgh |11/12/85| Updated to handle Card objects.)

```
(LISTGET (fetch (Card UserData) of Card)
       'RegionViewed])
```

## (NC.FetchScale

[LAMBDA (Card) (* fgh%: "17-Nov-85 18:10")

(* * For sketch cards -- fetch from the UserData prop list the region viewed property)

(* * fgh |11/12/85| Updated to handle Card objects)

(**NC.FetchUserDataProp** Card 'Scale])

## (NC.FetchSlotNum

[LAMBDA (Card) (* kirk%: " 9-Sep-86 08:55")

(* * compute card's slot number in the notefile)

```
(ADD1 (QUOTIENT [DIFFERENCE (fetch (Card IndexLoc) of Card)
                        (CONSTANT (fetch (NoteFileVersion NoteFileHeaderSize) of (NCLocalDevice.CurrentVersion
                                                                                 ]
                (CONSTANT (fetch (NoteFileVersion NoteFileIndexWidth) of (NCLocalDevice.CurrentVersion])
```

## (NC.FetchSubstance

[LAMBDA (Card) (* fgh%: "13-Nov-85 19:23")

(* * Return the substance of card ID)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handleCard object.)

(**fetch** (Card Substance) **of** Card])

```
(NC.FetchTitle
  [LAMBDA (Card)                                          (* fgh%: "13-Nov-85 19:23")

          (* * Fetch the title from IDs cache)

          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

          (* * fgh |11/13/85| Updated to handle Card object.)

    (fetch (Card Title) of Card])

(NC.FetchToLinks
  [LAMBDA (Card)                                          (* fgh%: "13-Nov-85 19:24")

          (* * fetch the to links from IDs cache)

          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

          (* * fgh |11/13/85| Updated to handl;e Card object.)

    (fetch (Card ToLinks) of Card])

(NC.FetchType
  [LAMBDA (Card)                                          (* fgh%: "13-Nov-85 19:24")

          (* * Fetch the note card type from IDs cache)

          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

          (* * fgh |11/13/85| Updated to handle card object.)

    (fetch (Card Type) of Card])

(NC.FetchWindow
  [LAMBDA (Card KEEP-UN-HIDDEN)                           ; Edited  9-Jan-89 12:26 by rtk

;;; Given an ID of a card, return the open window containing that card.

;;; rht 9/22/85: Rewrote slightly to remove the dangerous nested WINDOWPROP calls.

;;; fgh 11/15/85 Updated to handle card object.

;;; bk 1/9/89 Use NC.FlashHiddenWindow to flash hidden windows, also don't flash a shruken window if icon is in main window

    (if (NC.ActiveCardP Card)
        then (for Window in (NC.OpenWindows) bind UnshrunkenWin
                do (LET [(RETURN-WINDOW (COND
                                           ((NC.SameCardP Card (NC.CoerceToCard Window))
                                            Window)
                                           ((AND [WINDOWP (SETQ UnshrunkenWin (WINDOWPROP Window 'ICONFOR]
                                                 (NC.SameCardP Card (NC.CoerceToCard UnshrunkenWin)))
                                            UnshrunkenWin)
                                           (T NIL]
                         (AND RETURN-WINDOW (BOUNDP 'ROOMS:*ROOMS-SYSTEM-DATE*)
                              (NOT (FMEMB RETURN-WINDOW (OPENWINDOWS)))
                              (NOT (FMEMB (WINDOWPROP Window 'ICON)
                                          (OPENWINDOWS)))
                              (NC.FlashHiddenWindow RETURN-WINDOW KEEP-UN-HIDDEN))
                         (AND RETURN-WINDOW (RETURN RETURN-WINDOW])

(NC.IDP
  [LAMBDA (CardID)                                        (* kirk%: "23-Dec-85 00:43")

          (* * rht 10/29/85%: Totally rewritten. May get changed again soon! Stay tuned.)

          (* * fkr |11/8/85| Updated to new CardID scheme.)

    (type? UID CardID])

(NC.MarkCardDirty
  [LAMBDA (Card ResetFlg)                                 (* fgh%: " 6-Feb-86 22:03")

          (* Mark card specified by ID as being DIRTY and needing to be writtent to the database)

          (* * rht 2/1/85%: Now also sets/resets property on ID indicating substance dirty.)

          (* * fgh |11/13/85| Updated to handle Card object.)

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

    (NC.ApplyFn MarkDirtyFn Card ResetFlg)
    (NC.SetSubstanceDirtyFlg Card (NOT ResetFlg])
```

## (NC.CardDirtyP

```
[LAMBDA (Card)                                          (* fgh%: " 5-Feb-86 20:01")
```

(* * Return T if card ID has been changed.)

(* * rht 2/1/85%: Now also checks flag on prop list.)

(* * fgh |11/13/85| Updated to handle Card object.)

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

```
    (OR (NC.ApplyFn DirtyPFn Card)
        (NC.FetchSubstanceDirtyFlg Card])
```

## (NC.SetFromLinks

```
[LAMBDA (Card Links)                                    (* fgh%: "13-Nov-85 19:25")
```

(* * Cache the from links for ID)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (replace (Card FromLinks) of Card with Links])
```

## (NC.SetGlobalLinks

```
[LAMBDA (Card GlobalLinks)                              (* fgh%: "13-Nov-85 19:25")
```

(* * Cache the global links of ID)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (replace (Card GlobalLinks) of Card with GlobalLinks])
```

## (NC.SetLinksDirtyFlg

```
[LAMBDA (Card Value)                                    (* fgh%: "13-Nov-85 19:30")
```

(* * Set the flag indicating the links cache is dirty)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (replace (Card LinksDirtyFlg) of Card with Value])
```

## (NC.SetPropList

```
[LAMBDA (Card PropList)                                 (* fgh%: "13-Nov-85 19:32")
```

(* * Cache the prop list of ID)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (replace (Card PropList) of Card with PropList])
```

## (NC.SetRegion

```
[LAMBDA (Card Region)                                   (* fgh%: "13-Nov-85 19:32")
```

(* * Cache the region of ID)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (replace (Card Region) of Card with Region])
```

## (NC.SetSavedRegion

```
[LAMBDA (Card Region)                                   (* rht%: " 5-Jul-86 17:21")
```

(* * Cache the saved region of card ID)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

(* * rht 7/5/86%: Now makes sure region is on screen.)

```
    (replace (Card SavedRegion) of Card with (MAKEWITHINREGION (create REGION copying Region])
```

## (NC.SetRegionViewed

```
  [LAMBDA (Card RegionValue)                                    (* fgh%: "17-Nov-85 18:10")
          (* * For sketch cards -- save the cache the region viewd on the cache UserData prop list)
          (* * fgh |11/12/85| Updated to handle CardID and CardInfo objects.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (NC.SetUserDataProp Card 'RegionViewed RegionValue])
```

## (NC.SetScale

```
  [LAMBDA (Card ScaleValue)                                     (* fgh%: "18-Nov-85 00:01")
          (* * For sketch cards -- cache the scale on the cache UserData prop list)
          (* * fgh |11/12/85| Updated to handle CardID and CardInfo objects.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (NC.SetUserDataProp Card 'Scale ScaleValue])
```

## (NC.SetSubstance

```
  [LAMBDA (Card Substance)                                      (* fgh%: "13-Nov-85 19:41")
          (* * Set the cached substance of card ID to be substance)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (replace (Card Substance) of Card with Substance])
```

## (NC.SetTitle

```
  [LAMBDA (Card Title)                                          (* fgh%: "13-Nov-85 19:43")
          (* * Cache the title for card ID)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (replace (Card Title) of Card with Title])
```

## (NC.SetToLinks

```
  [LAMBDA (Card Links)                                          (* fgh%: "13-Nov-85 19:43")
          (* * Cache the to links for card ID)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (replace (Card ToLinks) of Card with Links])
```

## (NC.SetType

```
  [LAMBDA (Card NoteCardType)                                   (* fgh%: "13-Nov-85 19:43")
          (* * Cache the type of ID)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (replace (Card Type) of Card with NoteCardType])
```

## (NC.FetchTitleDirtyFlg

```
  [LAMBDA (Card)                                                (* fgh%: "13-Nov-85 19:44")
          (* * fetch the cached title dirty flag)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (fetch (Card TitleDirtyFlg) of Card])
```

## (NC.SetTitleDirtyFlg

```
  [LAMBDA (Card Value)                                    (* fgh%: "13-Nov-85 19:44")
          (* * Set the title cahce is dirty flag)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (replace (Card TitleDirtyFlg) of Card with Value])
```

## (NC.FetchPropListDirtyFlg

```
  [LAMBDA (Card)                                          (* fgh%: "13-Nov-85 19:44")
          (* * Fetch the prop liost list cache dirty flag)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (fetch (Card PropListDirtyFlg) of Card])
```

## (NC.SetPropListDirtyFlg

```
  [LAMBDA (Card Value)                                    (* fgh%: "13-Nov-85 19:45")
          (* * Set the prop list cache dirty flg)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (replace (Card PropListDirtyFlg) of Card with Value])
```

## (NC.SetSubstanceDirtyFlg

```
  [LAMBDA (Card Value)                                    (* fgh%: "13-Nov-85 19:45")
          (* * Set the substance cahce dirty flg)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (replace (Card SubstanceDirtyFlg) of Card with Value])
```

## (NC.FetchSubstanceDirtyFlg

```
  [LAMBDA (Card)                                          (* fgh%: "13-Nov-85 19:45")
          (* * fetch the substance cache dirty flag)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (fetch (Card SubstanceDirtyFlg) of Card])
```

## (NC.TurnOffDirtyFlgs

```
  [LAMBDA (Card)                                          (* fgh%: "13-Nov-85 20:13")
          (* * Make this card look not dirty by turning off all its dirty flags.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (NC.MarkCardDirty Card 'RESET)
    (NC.SetLinksDirtyFlg Card NIL)
    (NC.SetTitleDirtyFlg Card NIL)
    (NC.SetPropListDirtyFlg Card NIL])
```

## (NC.FetchUserDataProp

```
  [LAMBDA (Card Prop)                                     (* fgh%: "17-Nov-85 18:11")
          (* * Fetch a prop value from Card's UserData entry)
    (AND (fetch (Card UserData) of Card)
         (LISTGET (fetch (Card UserData) of Card)
                Prop])
```

## (NC.SetUserDataProp

```
  [LAMBDA (Card Prop Value)                               (* fgh%: "17-Nov-85 18:00")
          (* * Set a property on the USerData list)
```

```
    (LET ((UserDataPropList (fetch (Card UserData) of Card)))
         (COND
            ((LISTP UserDataPropList)
             (LISTPUT UserDataPropList Prop Value))
            (T (replace (Card UserData) of Card with (LIST Prop Value])
```

## (NC.SetUserDataPropList

```
  [LAMBDA (Card PropList)                                    (* rht%: "19-Dec-85 16:51")

          (* * Replace the user data prop list of Card.)

    (replace (Card UserData) of Card with PropList])
```

## (NC.MakeTypeIconBitMapSet

```
  [LAMBDA (Bitmap Heights)                                   (* rht%: "10-May-86 18:44")

          (* * Create a prop list of pairs of Height and scaled copies of Bitmap having that height.)

    (LET ((OriginalHeight (BITMAPHEIGHT Bitmap)))
         (OR [for Height in Heights join (LIST Height (SCALEBITMAP Bitmap (FQUOTIENT Height OriginalHeight]
             Bitmap])
```

## (NC.MakeNewCardWindow

```
  [LAMBDA (Card Title BorderWidth NoOpenFlg Region/Position)     (* rht%: "20-Sep-86 00:06")

          (* * Creates a window for Card and Title or "Untitled." If Region/Position NIL, then use default display region for Card.)

    (LET ((Window (CREATEW (NC.DetermineDisplayRegion Card Region/Position)
                           (OR Title "Untitled")
                          BorderWidth NoOpenFlg)))
         (WINDOWPROP Window 'NoteCardObject Card)
         Window])
```

## (NC.FlashHiddenWindow

```
  [LAMBDA (HIDDEN-WINDOW-TO-FLASH UN-HIDE-WINDOW)                ; Edited 23-Nov-88 16:55 by krivacic
    ;; Flash the HIDDEN-WINDOW-TO-FLASH.  UN-HIDE & RE-HIDE the window if necessary.
    ;;  Interacts with ROOMS if loaded, else just  flashes the window.
    (LET* [(ROOMS-LOADED (BOUNDP 'ROOMS:*ROOMS-SYSTEM-DATE*))
           (WINDOW-WAS-HIDDEN (AND ROOMS-LOADED (ROOMS:WINDOW-HIDDEN? HIDDEN-WINDOW-TO-FLASH]
          (AND WINDOW-WAS-HIDDEN (ROOMS:UN-HIDE-WINDOW HIDDEN-WINDOW-TO-FLASH))
          (FLASHWINDOW HIDDEN-WINDOW-TO-FLASH 3 100)
          (AND WINDOW-WAS-HIDDEN (NOT UN-HIDE-WINDOW)
               (ROOMS:HIDE-WINDOW HIDDEN-WINDOW-TO-FLASH])
```

## (NC.OpenWindows

```
  [LAMBDA NIL
    (OR (AND (BOUNDP 'ROOMS:*ROOMS-SYSTEM-DATE*)
             (ROOMS:ALL-WINDOWS T))
        (OPENWINDOWS])

)


;;; Read subscriptions and write locks on card parts

(DEFINEQ
```

## (NC.ObtainEditPermission

```
  [LAMBDA (Card)                                             (* Feuerman " 4-Aug-86 15:08")

          (* * Tries to get permission to edit the Card by obtaining write permission on the SUBSTANCE, TOLINKS and
          GLOBALTOLINKS of the Card. If any one of these fail, those that had been obtained so far are released and NIL is returns,
          otherwise T.)

          (* * kef 7/22/86%: Added PROPLIST to the set of card parts that need to have their write locks owned.)

          (* * kef 8/4/86%: Modified so that if a window already exists for the card, the write locks aren't re-obtained, thus not messing
          up the write lock count.)

    (PROG [(WRITEPERMISSIONFN (fetch (NoteFile ObtainWritePermissionFn) of (fetch (Card NoteFile) of Card)))
           (WRITERELEASEFN (fetch (NoteFile ReleaseWritePermissionFn) of (fetch (Card NoteFile) of Card]
          (RETURN (OR (NC.FetchWindow Card)
                      (COND
                         ((APPLY* WRITEPERMISSIONFN Card 'SUBSTANCE)
                          (COND
                             ((APPLY* WRITEPERMISSIONFN Card 'TOLINKS)
                              (COND
                                 ((APPLY* WRITEPERMISSIONFN Card 'GLOBALTOLINKS)
                                  (COND
                                     ((APPLY* WRITEPERMISSIONFN Card 'PROPLIST)
                                      T)
```

```
                                    (T (APPLY* WRITERELEASEFN Card 'GLOBALTOLINKS)
                                       (APPLY* WRITERELEASEFN Card 'SUBSTANCE)
                                       (APPLY* WRITERELEASEFN Card 'TOLINKS)
                                       NIL)))
                                 (T (APPLY* WRITERELEASEFN Card 'SUBSTANCE)
                                    (APPLY* WRITERELEASEFN Card 'TOLINKS)
                                    NIL)))
                              (T (APPLY* WRITERELEASEFN Card 'SUBSTANCE)
                                 NIL)))
                           (T NIL])
```

## (NC.CardPartBusy

```
  [LAMBDA (Card Part InterestedWindow)                    (* Feuerman " 6-Aug-86 12:51")
    (PRIN1 (CONCAT "Card part(s) busy for card %"" (NC.FetchTitle Card)
                   "%":  " Part)
           (OR InterestedWindow PROMPTWINDOW])

)

;;; Stuff to handle read-only notefiles.

(DEFINEQ
```

## (NC.ReadOnlyCardP

```
  [LAMBDA (Card)                                          (* rht%: " 4-Jul-86 18:53")

          (* * Return non-nil if card's notefile is open for read only. Someday it will get smarter.)

    (NC.ReadOnlyNoteFileP (fetch (Card NoteFile) of Card])
```

## (NC.CardSomehowDirtyP

```
  [LAMBDA (Card)                                          (* rht%: " 2-Jul-86 23:31")

          (* * Return non-nil if substance, title, props, or links of card is dirty.)

    (OR (NC.FetchNewCardFlg Card)
        (NC.CardDirtyP Card)
        (NC.FetchTitleDirtyFlg Card)
        (NC.FetchPropListDirtyFlg Card)
        (NC.FetchLinksDirtyFlg Card])
```

## (NC.CardReadOnlyOpenP

```
  [LAMBDA (Card)                                          ; Edited 15-Jan-88 16:24 by MacDonald
    (OR (NC.ReadOnlyNoteFileP (fetch (Card NoteFile) of Card])

)

;;; Retrieve Note Card Info

(DEFINEQ
```

## (NC.RetrieveFromLinks

```
  [LAMBDA (Card LeaveCachedFlg)                           (* rht%: "29-May-87 16:27")
                                                          (* Get or Fetch the from links for the card ID)

          (* * rht 11/10/85%: Updated to handle new CardID and hash array scheme.)

          (* * fgh 11/11/85%: Updated to handle CardInfo objects.)

          (* * fgh |11/13/85| Updated to handle Card object.)

          (* * rht&pmi 5/29/87%: Change to functionality. Default is now to not leave links cached if they weren't when we came in.
          If LeaveCachedFlg is non-nil, then get old functionality.)

    (LET ((LinksWereCachedFlg (NC.LinksCachedP Card)))
         (if (NOT LinksWereCachedFlg)
             then (NC.GetLinks Card))
         (PROG1 (NC.FetchFromLinks Card)
                (if (AND (NOT LinksWereCachedFlg)
                         (NOT LeaveCachedFlg))
                    then (NC.UncacheLinks Card)))])
```

## (NC.RetrieveLinkLabels

```
  [LAMBDA (NoteFile SystemLinksFlg)                       (* pmi%: " 6-Jan-88 14:55")

          (* Retrieve the list of link labels used in database specified by DatabaseStream.
          Include system maintained links only when specified by SystemLinksFlg)

          (* * fgh |11/12/85| Updated to handle NoteFile and Card objects.)

          (* * pmi 1/6/88%: Added check for LinkLabelsCard being cached before getting it from the notefile.)
```

```
    (DECLARE (GLOBALVARS NC.SystemLinkLabels))
    (LET ((LinkLabelsCard (fetch (NoteFile LinkLabelsCard) of NoteFile)))
         (UNION (AND SystemLinksFlg NC.SystemLinkLabels)
                (for Label in (OR (NC.FetchSubstance LinkLabelsCard)
                                  (NCP.CardCachedP LinkLabelsCard)
                                  (PROGN (NC.GetNoteCard LinkLabelsCard)
                                         (NC.FetchSubstance LinkLabelsCard)))
                   when (OR SystemLinksFlg (NULL (NC.SystemLinkLabelP Label))) collect Label]))
```

## (NC.StoreLinkLabels

```
[LAMBDA (NoteFile LinkLabels)                                   (* rht%: " 2-Mar-86 15:22")

          (* * Store the new set of links labels for NoteFile)

          (* * rht 3/2/86%: Added call to NC.GetType before call to NC.MarkCardDirty in order to get Type cached for the link labels
          card.)

    (LET ((LinkLabelsCard (fetch (NoteFile LinkLabelsCard) of NoteFile)))
         (NC.SetSubstance LinkLabelsCard LinkLabels)            (* Type has to be cached before NC.MarkCardDirty can be
                                                                called.)
         (if (NULL (NC.FetchType LinkLabelsCard))
             then (NC.GetType LinkLabelsCard))
         (NC.MarkCardDirty LinkLabelsCard])
```

## (NC.RetrievePropList

```
[LAMBDA (Card)                                                  (* fgh%: "13-Nov-85 19:55")
                                                                (* Fetch or get the prop list for the note card specified by ID)

          (* * rht 11/10/85%: Updated to handle new CardID and hasharray scheme.)

          (* * fgh 11/11/85%: UPdated to handle Card object.)

    (OR (NC.FetchPropList Card)
        (NC.GetPropList Card])
```

## (NC.RetrieveTitle

```
[LAMBDA (Card)                                                  (* fgh%: "13-Nov-85 19:55")

          (* If note card specified by NoteCardID is active then fetch its title.
          Otherwise activate the card, fetch title and then deactivate the card.)

          (* * fkr 10/29/85%: Rewrote to use new cache array stuff.)

          (* * rht 11/10/85%: Updated to handle new CardID and hasharray scheme.)

          (* * fgh 11/11/85%: Updated to handle Card object.)

    (OR (NC.FetchTitle Card)
        (NC.GetTitle Card])
```

## (NC.NameOfCardorNodeorImageObj

```
[LAMBDA (ITEM-TO-NAME)                                          ; Edited  9-Jan-89 10:56 by sye
    ;; Attempt to return a User-Friendly name for the ITEM-TO-NAME
    (LET ((NODELABEL (fetch (GRAPHNODE NODELABEL) of ITEM-TO-NAME)))
         (COND
            ((NOT (NC.LinkIconImageObjP NODELABEL))
             ;; It's a Label Object, return NODELABEL
             NODELABEL)
            (T ;; Get the Name from the Card Title
               (NC.RetrieveTitle (fetch (Link DestinationCard) of (IMAGEOBJPROP NODELABEL 'OBJECTDATUM])
```

## (NC.RetrieveToLinks

```
[LAMBDA (Card LeaveCachedFlg)                                   (* rht%: "29-May-87 16:26")
                                                                (* Get or Fetch the from links for the card ID)

          (* * rht 11/10/85%: Updated to handle new CardID and hasharray scheme.)

          (* * fgh 11/11/85%: Updated to handle Card object.)

          (* * rht&pmi 5/29/87%: Change to functionality. Default is now to not leave links cached if they weren't when we came in.
          If LeaveCachedFlg is non-nil, then get old functionality.)

    (LET ((LinksWereCachedFlg (NC.LinksCachedP Card)))
         (if (NOT LinksWereCachedFlg)
             then (NC.GetLinks Card))
         (PROG1 (NC.FetchToLinks Card)
             (if (AND (NOT LinksWereCachedFlg)
                      (NOT LeaveCachedFlg))
```

```
                then (NC.UncacheLinks Card)))])
```

## (NC.RetrieveGlobalLinks

```
[LAMBDA (Card LeaveCachedFlg)                                   (* rht%: "29-May-87 16:28")
```

(* * Fetch or get the global links of ID)

(* * rht 11/10/85%: Updated to handle new CardID and hasharray scheme.)

(* * fgh 11/11/85%: Updated to handle Card object.)

(* * rht&pmi 5/29/87%: Change to functionality. Default is now to not leave links cached if they weren't when we came in. If LeaveCachedFlg is non-nil, then get old functionality.)

```
    (LET ((LinksWereCachedFlg (NC.LinksCachedP Card)))
         (if (NOT LinksWereCachedFlg)
             then (NC.GetLinks Card))
         (PROG1 (NC.FetchGlobalLinks Card)
             (if (AND (NOT LinksWereCachedFlg)
                      (NOT LeaveCachedFlg))
                 then (NC.UncacheLinks Card)))])
```

## (NC.RetrieveType

```
[LAMBDA (Card)                                                  (* fgh%: "13-Nov-85 19:57")
                                                                (* Get or Fetch NoteCardType of ID)
```

(* * fkr 10/29/85%: Rewrote to use new cache array stuff.)

(* * rht 11/10/85%: Updated to handle new CardID and hasharray scheme.)

(* * fgh 11/11/85%: Updated to handle Card object.)

```
    (OR (NC.FetchType Card)
        (NC.GetType Card])
```

## (NC.RetrieveTypeAndTitle

```
[LAMBDA (Card)                                                  (* fgh%: " 3-Sep-86 21:03")
```

(* * Retrieve the type and title of a card using NC.GetCardInfo where possible to decrease Courier calls if using server.)

(* * fgh |9/3/86| First created.)

```
    (LET (Type Title Info)
         (if (EQ (NC.FetchStatus Card)
                 'ACTIVE)
             then [if (SETQ Type (NC.FetchType Card))
                      then [if (SETQ Title (NC.FetchTitle Card))
                               then NIL
                               else (NC.SetTitle Card (SETQ Title (NC.GetTitle Card]
                  else (if (SETQ Title (NC.FetchTitle Card))
                           then (NC.SetType Card (SETQ Type (NC.GetType Card)))
                           else [SETQ Info (NC.ApplyFn GetCardInfoFn Card '(TYPE TITLE]
                                [NC.SetTitle Card (SETQ Title (CDR (FASSOC 'TITLE Info]
                                (NC.SetType Card (SETQ Type (CDR (FASSOC 'TYPE Info]
                       (CONS Type Title])
```

## (NC.LinksCachedP

```
[LAMBDA (Card)                                                  (* rht%: "16-May-87 00:06")
```

(* * Does this card have links cahced?)

(* * rht 5/16/87%: Changed to also return non-nil if card is active and new.)

```
    (OR (fetch (Card Links) of Card)
        (AND (NC.ActiveCardP Card)
             (NC.FetchNewCardFlg Card])
```

## (NC.ShowInfo

```
[LAMBDA (Window)                                                ; Edited  3-Dec-87 19:00 by rht:
```

(* * Bring up an inspector on certain attribute/value pairs of this card.
The default ones are card part dates and Updates.)

(* * Need a hook so that users can provide other attribute/value pairs for given card types.)

(* * kirk%: 1May86 added NC.AttachNoteFileName)

(* * fgh |6/13/86| Now places and sizes window with less visual nnoise.)

(* * fgh&rht 7/4/86%: Now also print card type.)

(* * rht 7/14/86%: Now includes card object if global param is set.
No longer includes UID.)

```
          (* * rg |4/1/87| added NC.ProtectedCardOperation wrapper)

          (* * rg |4/6/87| removed NC.ProtectedCardOperation wrapper)

    (LET ((Card (NC.CoerceToCard Window))
          [Attributes `(Type ItemDate TitleDate LinksDate PropsDate Updates ,@(AND
                                                                                NC.IncludeCardObjectInShowInfo
                                                                                '(CardObject]
          Region AttributesAndValues InfoWindow)
         (NC.AttachNoteFileName Window)
         (SETQ Region (WINDOWREGION Window))
         (for Win in (ATTACHEDWINDOWS Window) when (WINDOWPROP Win 'ShowInfo) do (CLOSEW Win))
                                                                    (* close any previous info window)
         [SETQ AttributesAndValues `(Type ,(NC.FetchType Card)
                                          ItemDate
                                          ,(NC.FetchItemDate Card)
                                          TitleDate
                                          ,(NC.FetchTitleDate Card)
                                          LinksDate
                                          ,(NC.FetchLinksDate Card)
                                          PropsDate
                                          ,(NC.FetchPropListDate Card)
                                          Updates
                                          ,(LISTGET (NC.FetchPropList Card)
                                                  'Updates)
                                          ,@(AND NC.IncludeCardObjectInShowInfo `(CardObject ,Card]
         (SETQ InfoWindow (INSPECTW.CREATE AttributesAndValues Attributes (FUNCTION LISTGET)
                                  NIL "Can't set values of these attributes." NIL NIL "Card attributes" NIL
                                  (CREATEW (CREATEREGION (fetch (POSITION XCOORD) of NC.OffScreenPosition)
                                                         (fetch (POSITION YCOORD) of NC.OffScreenPosition)
                                                         (fetch (REGION WIDTH) of Region)
                                                         50)
                                           NIL NIL)))
         (SHAPEW InfoWindow (create REGION using (WINDOWPROP InfoWindow 'REGION)
                                                 HEIGHT _ (HEIGHTIFWINDOW (fetch (REGION HEIGHT)
                                                                                  of (WINDOWPROP InfoWindow
                                                                                                 'EXTENT))
                                                                          T)))
         (ATTACHWINDOW InfoWindow Window 'TOP 'JUSTIFY 'LOCALCLOSE)
         (NC.MoveWindowOntoScreen Window)
         (REDISPLAYW InfoWindow)
         (WINDOWPROP InfoWindow 'ShowInfo 'Showing)
         (WINDOWADDPROP InfoWindow 'CLOSEFN (FUNCTION FREEATTACHEDWINDOW)
                T])

)


;;; General note card manipulations

(DEFINEQ

(NC.AddParents
  [LAMBDA (WindowOrTextStream)                                    (* Randy.Gobbel " 2-Apr-87 15:38")

          (* Add a subtopic link from a contents card specified by the user to the contents card specified by WindowOrTextStream.
          But first check to make sure that this would not introduce any circularities in the contents lattice.)

          (* * rht 12/8/84%: Massive shaving. Now calls NC.MakeChildLink to do the tough work.)

          (* * rht 10/3/85%: No longer prints final, annoying, slow-to-disappear message in prompt window if nothing selected.)

          (* * fgh |11/13/85| Updated to handle Card object.)

          (* * fgh |6/9/86| Added code to check to make sure that another operation is not in progress on this card when this fn is
          called.)

          (* * rht 7/5/86%: Now checks for readonly cards.)

          (* * rht&pmi 11/24/86%: Surrounded call to NC.MakeChildLink with NC.ActivateCardAndDo so that ParentCard will be active
          for duration of the call.)

          (* * pmi 12/5/86%: Modified message to NC.SelectNoteCards to mention SHIFT-selection.)

          (* * pmi 12/12/86%: Removed obsolete ReturnLinksFlg argument in call to NC.SelectNoteCards.)

          (* * rht 1/28/87%: Took out call to NC.ActivateCardAndDo. Need to make sure ParentCard gets saved after getting link if
          wasn't active originally.)

          (* * rg |3/3/87| Enlarged scope of NC.ProtectedCardOperation)

          (* * rg |3/18/87| added NC.CardSelectionOperation wrapper)

          (* * rg |4/2/87| changed NC.CardSelectionOperation to NCP.WithLockedCards)

    (NCP.WithLockedCards (LET [Card NewParents (Window (OR (WINDOWP WindowOrTextStream)
```

```
                                                    (NC.TEditWindow WindowOrTextStream]
                                  (SETQ Card (NC.CoerceToCard Window))
                                  (NC.ProtectedCardOperation
                                   Card "Designate FileBoxes" NIL
                                   (if (NC.CheckForNotReadOnly Card Window "Can't do filing in ")
                                       then (SETQ NewParents (NC.SelectNoteCards NIL (FUNCTION NC.FileBoxP)
                                                                  NC.SelectingParentsMenu Card " Please
                                                                  shift-select the new parent FileBox(es)."))
                                            (AND NewParents Card
                                                 (for ParentCard in NewParents bind OneHook
                                                    when [LET ((WasActiveFlg (NC.ActiveCardP ParentCard)))
                                                              (OR WasActiveFlg (NC.GetNoteCard ParentCard))
                                                              (PROG1 (NC.MakeChildLink Card ParentCard Window)
                                                                  (OR WasActiveFlg
                                                                      (NC.QuitCard ParentCard NIL NIL NIL NIL NIL T))
)]
                                                    do (SETQ OneHook T) finally (RETURN OneHook])
```

## (NC.SeverExternalLinks

```
  [LAMBDA (ListOfCards QuietFlg InterestedWindow)                    ; Edited 9-Jun-88 18:09 by Trigg
```

;;; Delete all links in ListOfCards to or from cards not in ListOfCards. Furthermore, do it efficiently by caching an external card only long enough to delete ;;; all the links between it and ListOfCards. Note that we depend on the fact that every card in ListOfCards has its AboutToBeDeletedFlg set.

;; rht&pmi 5/29/87: Now passes non-nil LeaveCachedFlg to NC.RetrieveToLinks and NC.RetrieveFromLinks. Not sure if this is really necessary.

;; pmi 6/8/87: Added call to NC.GreyCard to grey cards being deleted before deleting all of their links.

;; dsj. 10/12/87. Changed to do lazy updating of links and only half-sever links for which this card is the destination: let the other half be severed ;; and updated when and if the source card is invoked.

;; rht 10/31/87: Now uncaches links after call to NC.PutLinks on 'external' source cards.

;; pmi 12/10/87: Merged dsj's and rht's changes; see last two comments above.

;; rht 6/9/88: Fixed the bug where A points to B and B points to A and we're deleting B results in all of A's fromlinks getting trashed.

```
(LET (LinksToSever NumLinksToSever)
     (OR QuietFlg (NC.PrintMsg InterestedWindow T "Gathering external links of " (LENGTH ListOfCards)
                         " cards."))
     [SETQ LinksToSever (NC.UnionListsOfLinks (for Card in ListOfCards
                                                 join          ; Grey the cards being deleted.
                                                      (NC.GreyCard Card)
                                                      (for Link in (NC.RetrieveToLinks Card T)
                                                         unless (NC.UIDGetProp (fetch (Card UID)
                                                                                  of (fetch (Link
                                                                                             DestinationCard
                                                                                             )
                                                                                        of Link))
                                                                                'AboutToBeDeletedFlg)
                                                         collect Link))
                                              (for Card in ListOfCards
                                                 join (for Link in (NC.RetrieveFromLinks Card T)
                                                         unless (NC.UIDGetProp (fetch (Card UID) of (fetch (Link SourceCard)
                                                                                                       of Link))
                                                                               'AboutToBeDeletedFlg)
                                                         collect Link]
```

;; Now sort links so that links with same external anchor card are grouped together. Furthermore, the links having that anchor card has ;; source card are grouped before the ones having that card as destination card.

```
     (OR QuietFlg (NC.PrintMsg InterestedWindow T "Sorting " (SETQ NumLinksToSever (LENGTH LinksToSever))
                         " links prior to severing."))
     [SORT LinksToSever (FUNCTION (LAMBDA (Link1 Link2)
                                    (LET (DestCard1 DestCard2 ExtCard1 ExtCard2 Link1SourceIsExtFlg)
                                         (SETQ ExtCard1 (if (NC.UIDGetProp (fetch (Card UID)
                                                                              of (SETQ DestCard1
                                                                                   (fetch (Link DestinationCard
                                                                                           )
                                                                                      of Link1)))
                                                                           'AboutToBeDeletedFlg)
                                                            then (SETQ Link1SourceIsExtFlg T)
                                                                 (fetch (Link SourceCard) of Link1)
                                                            else DestCard1))
                                         (SETQ ExtCard2 (if (NC.UIDGetProp (fetch (Card UID)
                                                                              of (SETQ DestCard2
                                                                                   (fetch (Link DestinationCard
                                                                                           )
                                                                                      of Link2)))
                                                                           'AboutToBeDeletedFlg)
                                                            then (fetch (Link SourceCard) of Link2)
                                                            else DestCard2))
                                         (if (NC.SameCardP ExtCard1 ExtCard2)
                                             then Link1SourceIsExtFlg
                                           else (LESSP (fetch (Card IndexLoc) of ExtCard1)
                                                       (fetch (Card IndexLoc) of ExtCard2]
```

;; Now walk down the list of links one by one activating the external anchor cards as needed.

```
(OR QuietFlg (NC.PrintMsg InterestedWindow T "Severing links: 1 out of " NumLinksToSever " ..."))
```

;; dsj. Changed to only half-sever links for which this card is the destination: let the other half be severed and updated when and if the ;; source card is invoked.

```
(for Link in LinksToSever as i from 1 bind PreviousExtCard WasNotActiveFlg PreviousExtCardIsSourceFlg
   eachtime (BLOCK) when (NC.ValidLinkP Link)
   do (OR QuietFlg (if (ZEROP (REMAINDER i 10))
                       then (NC.PrintMsg InterestedWindow T "Severing links: " i " out of "
                                  NumLinksToSever " ..."")))
      (LET (ExtCard ExtCardIsSourceFlg)
           (SETQ ExtCard (if (NC.UIDGetProp (fetch (Card UID) of (fetch (Link DestinationCard)
                                                                    of Link))
                                    'AboutToBeDeletedFlg)
                             then (SETQ ExtCardIsSourceFlg T)
                                  (fetch (Link SourceCard) of Link)
                           else (fetch (Link DestinationCard) of Link)))
           [if (OR (NOT (NC.SameCardP ExtCard PreviousExtCard))
                   (NOT (EQ ExtCardIsSourceFlg PreviousExtCardIsSourceFlg)))
               then                                          ; Write down changes to previous external card's substance.
                    (if WasNotActiveFlg
                        then (if PreviousExtCardIsSourceFlg
                                 then                        ; Have to call NC.CardSaveFn first and then NC.QuitCard with
                                                             ; Don'tSaveFlg to avoid insureProperFiling check.
                                                             ; dsj.  Disabled this.
                                                             ; (NC.CardSaveFn PreviousExtCard T) (NC.QuitCard
                                                             ; PreviousExtCard NIL T NIL NIL NIL T)

                                 else (NC.PutLinks PreviousExtCard)
                                      (NC.UncacheLinks PreviousExtCard)))
                                                             ; If ExtCard not active, then cache.
                    (if (SETQ WasNotActiveFlg (NOT (NC.ActiveCardP ExtCard)))
                        then (if ExtCardIsSourceFlg
                                 then                        ; Cache whole card if it's the link's source.
                                                             ; dsj.  Disabled this.
                                                             ; (NC.GetNoteCard ExtCard)

                                 else                        ; Else only need the links since we're deleting the from link.
                                      (NC.GetLinks ExtCard]  ; Delete the appropriate half of the link.
           (if ExtCardIsSourceFlg
               then                                          ; dsj.  Now delete TO link only if the card is active on the screen.
                    (AND (NOT WasNotActiveFlg)
                         (NC.DeleteToLink Link))
             else (NC.DeleteFromLink Link))
           (replace (Link UID) of Link with -1)
           (SETQ PreviousExtCard ExtCard)
           (SETQ PreviousExtCardIsSourceFlg ExtCardIsSourceFlg))
   finally (if (AND WasNotActiveFlg (NC.ValidCardP PreviousExtCard))
               then (if PreviousExtCardIsSourceFlg
                        then                                 ; Have to call NC.CardSaveFn first and then NC.QuitCard with
                                                             ; Don'tSaveFlg to avoid insureProperFiling check.
                                                             ; dsj.  Disabled this.
                                                             ; (NC.CardSaveFn PreviousExtCard T) (NC.QuitCard
                                                             ; PreviousExtCard NIL T NIL NIL NIL T)

                      else (NC.PutLinks PreviousExtCard)
                           (NC.UncacheLinks PreviousExtCard])
```

## (NC.AssignTitle

```
  [LAMBDA (CardIdentifier NoClearMsgFlg NewTitle InterestedWindow)
                                                             ; Edited 13-Dec-88 14:49 by krivacic
```

;;; Change the title of the card specified by the WindowOrTextStreamOrID

;;; rht 2/1/85: Changed from NC.PutTitle to NC.SetTitleDirtyFlg, unless card is not active. We shouldn't be writing to the notefile until save time.

;;; fgh 11/11/85: Added support for CardID, CardInfo and noteFile objects. Also entered call to Nc.StoreTitle.

;;; fgh 6/9/86 Added code to check to make sure that another operation is not in progress on this card when this fn is called.

;;; fgh 6/13/86 Now spawns mouse in case called under MOUSE process.

;;; fgh 6/27/86 returns T if completed okay.

;;; rht 7/4/86: Added check for readonly card.

;;; kef 7/16/86: Added obtain write permission.

;;; kef 7/24/86: Doesn't release the write lock if this is a new card.

;;; kef 7/30/86: Modified to check for Client's concept of whether he owns the write lock or not, thus deciding whether or not to setup the release of the
;;; write lock afterwards.

;;; fgh 8/30/86 Converted to use NC.IfCardPartNotBusy.

;;; rg 3/3/87 Enlarged scope of NC.ProtectedCardOperation

;;; rht 3/23/87: Now takes InterestedWindow arg.

;;; rht 3/24/87: Now calls NC.CoerceToInterestedWindow

;;; rht 5/27/87: Now passes title through cross-file link card if dest notefile is open.

;;; rht 5/29/87: Now uncaches links if they weren't cached when we came in.

;;; rht 11/20/87: Now updates ShowLinks menus if any for destinations of ToLinks from this card.

```
    (ALLOW.BUTTON.EVENTS)
    (LET ((Card (NC.CoerceToCard CardIdentifier))
          OldTitle Window)
         (NC.ProtectedCardOperation
          Card "Assign Title" NIL
          (NC.IfCardPartNotBusy
           Card
           'TITLE
           (OR InterestedWindow (SETQ InterestedWindow (NC.CoerceToInterestedWindow Card)))
           (if (NC.CheckForNotReadOnly Card InterestedWindow "Can't change titles for cards in ")
               then (COND
                       ((SETQ NewTitle (OR (STRINGP NewTitle)
                                           (AND NewTitle (OR (LITATOM NewTitle)
                                                             (NUMBERP NewTitle))
                                                (MKSTRING NewTitle))
                                           (NC.AskUser (CONCAT "Enter the title for this card" (CHARACTER 13))
                                                  "-->  "
                                                  (COND
                                                     ((AND (STREQUAL (SETQ OldTitle (NC.RetrieveTitle Card))
                                                                     "Untitled")
                                                           (NC.FetchNewCardFlg Card))
                                                      NIL)
                                                     (T OldTitle))
                                                  (NULL NoClearMsgFlg)
                                                  InterestedWindow)))
                        (NC.SetTitle Card NewTitle)
```

;;; Now do a PutTitle so that anyone else coming along will pick up on the new title. The only exception is if this card hasn't been written to the NoteFile
;;; yet, which is true when the NewCardFlg is T. In that case, we can't put the title down yet, so just mark it dirty.

```
                        (COND
                           ((fetch (Card NewCardFlg) of Card)
                            (NC.SetTitleDirtyFlg Card T))
                           (T (NC.PutTitle Card)
                              (NC.SetTitleDirtyFlg Card NIL)))
                        (AND (WINDOWP (SETQ Window (NC.FetchWindow Card)))
                             (WINDOWPROP Window 'TITLE NewTitle))
                        (LET ((LinksWereCachedFlg (NC.LinksCachedP Card)))
                             (if (NOT LinksWereCachedFlg)
                                 then (NC.GetLinks Card))
                             [for FromLink in (NC.FetchFromLinks Card)
                                do (LET ((ContainingCard (fetch (Link SourceCard) of FromLink))
                                         RemoteCrossFileLinkCard)
                                        (if (AND (NC.CrossFileLinkCardP ContainingCard)
                                                 (SETQ RemoteCrossFileLinkCard (NC.FetchRemoteCrossFileLinkCard
                                                                                   ContainingCard)))
                                            then (NC.CheckCrossFileLinkCardTitle RemoteCrossFileLinkCard Card)
                                          else (AND (NC.ActiveCardP ContainingCard)
                                                    (WINDOWP (NC.FetchWindow ContainingCard))
                                                    (NC.UpdateLinkImages ContainingCard Card]
                             [for ToLink in (NC.FetchToLinks Card)
                                do (LET ((ContainingCard (fetch (Link DestinationCard) of ToLink))
                                         RemoteCrossFileLinkCard)
                                        (if (AND (NC.CrossFileLinkCardP ContainingCard)
                                                 (SETQ RemoteCrossFileLinkCard (NC.FetchRemoteCrossFileLinkCard
                                                                                   ContainingCard)))
                                            then (NC.CheckCrossFileLinkCardTitle RemoteCrossFileLinkCard Card)
                                          else (NC.UpdateLinkImagesInShowLinks ContainingCard Card]
                             (if (NOT LinksWereCachedFlg)
                                 then (NC.UncacheLinks Card)))
                        T))
               else NIL])
```

## (NC.DeleteNoteCards

```
  [LAMBDA (CardIdentifiers NoIndividualConfirmFlg DontClearFlg InterestedWindow QuietFlg NoGroupConfirmFlg)
                                                                    ; Edited 6-Sep-88 15:00 by pmi
```

;; Delete note cards. If no card specified then get a list of note cards to be deleted. Then delete these cards.

;;; fgh 11/11/85: Updated to handle new Card objects. Also split off main work of deleteing a single note card into NC.DeleteNoteCard function.

;;; kirk 21Feb86 Added InterestedWindow

;;; kirk 29Apr86 Now returns CardIdentifiers

;;; fgh 6/9/86 Added checks to see if other operations are in progress

;;; rht 7/4/86: Now checks that card is not read-only.

;;; kirk 18Aug86 Added main window for windowless cards.

;;; rht 8/29/86: Reorganized and added call to NC.SeverAllLinks to make deleting more efficient. Added QuietFlg, NoGroupConfirmFlg and ;;; Don't PutToBeDeletedCardsFlg args.

;;; rht 9/5/86: Now forces NoGroupConfirmFlg to be non-nil if NoIndividualConfirmFlg is NIL and only one card to delete.

;;; pmi 12/5/86: Modified message to NC.SelectNoteCards to mention SHIFT-selection.

;;; pmi 12/12/86: Removed obsolete ReturnLinksFlg argument in call to NC.SelectNoteCards.

;;; rht 12/16/86: Removed obsolete Don't PutToBeDeletedCardsFlg arg.

;;; rht 3/9/87: Changed NC.DeleteSelectingMenu to NC.SelectingCardsMenu.

;;; rg 3/9/87 added NC.ProtectedSessionOperation wrapper

;;; rg 3/11/87 changed call of NC.DeleteNoteCard to NC.DeleteNoteCardInternal

;;; rg 3/18/87 changed NC.ProtectedSessionOperation to NC.CardSelectionOperation

;;; rht 3/30/87: Now calls NC.SeverExternalLinks rather than NC.SeverAllLinks.

;;; dsj 9/28/87: Added call to NC.CardsToDelete which accesses the WhenDeletedFn prop of card type. This does not yet handle the general case of ;;; allowing me to bypass asking the user for confirmation, which is an operation that should be a super type's WhenDeletedFn.

;;; pmi 1/15/88: Added dsj's change (see above comment)

;;; pmi 9/6/88: Removed redundant calls to NC.ClearMsg which were causing TTY window for mouse to appear because it was being called with no ;;; arguments.

```
(DECLARE (GLOBALVARS NC.SelectingCardsMenu))
(NCP.WithLockedCards
 (NC.IfAllCardsFree (NC.LockListOfCards (MKLIST CardIdentifiers)
                          "Delete Note Cards")
        (OR CardIdentifiers (SETQ CardIdentifiers (NC.SelectNoteCards NIL NIL NC.SelectingCardsMenu
                                                    InterestedWindow "Please shift-select the Note
                                                    Cards to be deleted.")))
```

;;; Kludge in case args are nil, say, when we're called from a card's menu.

```
            (if (AND (NULL NoIndividualConfirmFlg)
                     (NULL NoGroupConfirmFlg)
                     (EQ (LENGTH (MKLIST CardIdentifiers))
                         1))
                then (SETQ NoGroupConfirmFlg T)
                     (SETQ QuietFlg T))
```

;;; First collect cards that are deletable.

```
            (LET ((CardsToDelete (for CardIdentifier in (MKLIST CardIdentifiers) bind Card eachtime (BLOCK)
                                    when [AND (SETQ Card (NC.CoerceToCard CardIdentifier))
                                              (if (NOT (NC.TopLevelCardP Card))
                                                  else (NC.PrintMsg (NC.FetchWindow Card)
                                                             T "You cannot delete this FileBox." (CHARACTER 13))
                                                       (DISMISS 1000)
                                                       (NC.ClearMsg (NC.FetchWindow Card)
                                                             T)
                                                       NIL)
                                              (NC.CheckForNotReadOnly Card (NC.FetchWindow Card)
                                                    "Can't delete cards from a ")
                                              (OR NoIndividualConfirmFlg (PROG1 (NC.AskYesOrNo
                                                                                  "Are you sure you want to
                                                                                  delete this?" " -- " "Yes"
                                                                                  (NULL DontClearFlg)
                                                                                  (OR (NC.FetchWindow Card)
                                                                                      InterestedWindow)
                                                                                  NIL NIL]
                                    collect Card))
                  (NumSpecified (LENGTH (MKLIST CardIdentifiers)))
                  NumToDelete FinalNumToDelete)
                 (SETQ NumToDelete (LENGTH CardsToDelete))
                 (if [AND (GREATERP NumToDelete 0)
                          (if (EQUAL NumToDelete NumSpecified)
                              then (OR NoGroupConfirmFlg (PROG1 (NC.AskYesOrNo (CONCAT "You've specified "
                                                                                        NumToDelete " cards to
                                                                                        delete." (CHARACTER 13)
                                                                                        "Are you sure you want to
                                                                                        delete them? ")
                                                                                NIL "Yes" (NULL DontClearFlg)
                                                                                InterestedWindow)
                                                                ;; (NC.ClearMsg)
                                                                ))
                            else (PROG1 (NC.AskYesOrNo (CONCAT "Out of " NumSpecified " cards specified, "
                                                               (DIFFERENCE NumSpecified NumToDelete)
                                                               " are not deletable."
                                                               (CHARACTER 13)
```

```
                                                    "Want to delete the remaining " NumToDelete " cards?
")
                                            NIL "Yes" (NULL DontClearFlg)
                                            InterestedWindow]
                    then
```

;;; dsj 9/28/87: Let programmer have control over aborting deletion here.

```
                (SETQ CardsToDelete (NC.CardsToDelete CardsToDelete))
                (SETQ FinalNumToDelete (LENGTH CardsToDelete))
                (if (GREATERP FinalNumToDelete 0)
                    then (if (GREATERP NumToDelete FinalNumToDelete)
                             then (NC.PrintMsg InterestedWindow T "Can only delete " FinalNumToDelete
                                       " cards out of " NumToDelete " specified for deletion.")
                                  (DISMISS 3000)
                                  (NC.ClearMsg InterestedWindow))
```

;;; Mark UIDs of cards about to be deleted.

```
                (for Card in CardsToDelete do (NC.UIDPutProp (fetch (Card UID) of Card)
                                                             'AboutToBeDeletedFlg T))
```

;;; Sever all links into and out of CardsToDelete

```
                (NC.SeverExternalLinks CardsToDelete QuietFlg InterestedWindow)
```

;;; Now delete the cards one at a time.

```
                (OR QuietFlg (NC.PrintMsg InterestedWindow T "Deleting cards: 1 out of "
                                    FinalNumToDelete " ..."))
                (for Card in CardsToDelete as i from 1 eachtime (BLOCK)
                   do (OR QuietFlg (if (ZEROP (REMAINDER i 10))
                                       then (NC.PrintMsg InterestedWindow T "Deleting cards: "
                                                 i " out of " FinalNumToDelete " ...")))
                      (NC.DeleteNoteCardInternal Card QuietFlg InterestedWindow T))
                (OR QuietFlg (NC.ClearMsg InterestedWindow T)))
        CardIdentifiers])
```

## (NC.CardsToDelete

```
[LAMBDA (Cards)                                                    (* pmi%: "15-Jan-88 10:04")
```

(* * dsj 10/4/87%: This fn checks each card's card type WhenDeletedFn to see if we can go ahead with the delete. Return those cards for which it's okay to delete.)

(* * pmi 1/15/88%: Changed to not return a card if its WhenDeletedFn returns (QUOTE ABORT))

```
    (for Card in (MKLIST Cards) bind WhenDeletedFn collect Card
       when (NOT (AND (SETQ WhenDeletedFn (GETPROP (NCP.CardType Card)
                                                   'WhenDeletedFn))
                      (EQ (U-CASE (APPLY* WhenDeletedFn Card))
                          'ABORT])
```

## (NC.EditNoteCard

```
[LAMBDA (Card ReadOnly RegionOrPosition TypeSpecificArgs)      ; Edited  8-Sep-88 09:09 by pmi
```

;; Bring the already created NoteCard specified by ID onto the screen at Region or Position specified by RegionOrPosition

;; fgh 11/11/85: Updated to handle new Card object.

;; fgh 2/5/86 Added call to NC.ApplyFn

;; kirk 15May86 Added call to NC.AttachNoteFileName

;; rht 7/13/86: Added TypeSpecificArgs arg.

;; kef 7/16/86: Added NC.ObtainCardEditPermission.

;; kef 8/7/86: Added check to make sure that applying the EditFn worked. If not, then release those write locks, thus keeping the writelock count ;; consistent.

;; fgh 8/30/86 Converted APPLY* to NC.ApplyFn.

;; rht 10/6/86: Added checks before doing WINDOWPROP calls in case there was a recursive call to NC.EditNoteCard.

;; rg 3/30/87 added NC.ProtectedCardOperation wrapper

;; rht 5/13/87: Added call to new NC.InstallCopyButtonEventFn.

;; dsj 9/22/87: Included check for valid card type of destination card.

;; rg 11/4/87 added ReadOnly arg

;; pmi 12/30/87: Added dsj's change; see comment above.

;; pmi 2/10/88: Replaced bogus InterestedWindow argument with NIL in call to NCP.PrintMsg if card type definition is missing.

;; rht 5/30/88: Now deletes any links marked as Bad during running of EditFn.

;; pmi 9/8/88: Removed fix by dsj (9/22/87, see above). This is a case where a fix for IDE (checking for valid card type, which prevented ;; autoloading) hurt the rest of the NoteCards community.

```
    (DECLARE (GLOBALVARS NC.ShowNoteFileOnCards))
    (NC.ProtectedCardOperation Card "Edit NoteCard" NIL (RESETSAVE (CURSOR WAITINGCURSOR))
```

```
          (LET ((CardWindow (NCP.CardWindow Card)))
               (PROG (Window Substance EditResult)
                     [COND
                        ((AND (NC.ActiveCardP Card)
                              (NC.ObtainEditPermission Card))
                         (SETQ Substance (NC.FetchSubstance Card)))
                        ((NC.ObtainEditPermission Card)
                         (NC.GetNoteCard Card)
                         (SETQ Substance (NC.FetchSubstance Card)))
                        (T (RETURN (NC.CardPartBusy Card '(SUBSTANCE TOLINKS GLOBALTOLINKS PROPLIST]
                     (COND
                        ([AND (SETQ EditResult (ERSETQ (NC.ApplyFn EditFn Card Substance RegionOrPosition
                                                                   TypeSpecificArgs)))
                              (WINDOWP (SETQ Window (CAR EditResult]
                         (WINDOWADDPROP Window 'CLOSEFN (FUNCTION NC.QuitCard)
                                'FIRST)
                         (OR (NC.CardP (WINDOWPROP Window 'NoteCardObject))
                             (WINDOWPROP Window 'NoteCardObject Card))
                         (NC.InstallCopyButtonEventFn Window)
                         (if NC.ShowNoteFileOnCards
                             then (NC.AttachNoteFileName Window))
                         ;; Remove any bum links found while running the EditFn.
                         (for BadLink in (NC.FetchUserDataProp Card 'BadLinks) do (NC.DelReferencesToCard Card
                                                                                        BadLink))
                         (NC.SetUserDataProp Card 'BadLinks NIL))
                        (T                                    ; At this point, we've obtain the write locks but the edit failed, so
                                                              ; we'd better release them
                           (for CardPart in '(SUBSTANCE TOLINKS GLOBALTOLINKS PROPLIST)
                              do (NC.ApplyFn ReleaseWritePermissionFn Card CardPart))
                           (RETURN)))
                     (if ReadOnly
                         then (NC.ApplyFn MakeReadOnlyFn Card))
                     (RETURN Window])
```

## (NC.MakeNoteCard

```
  [LAMBDA (NoteCardType NoteFile Title NoDisplayFlg TypeSpecificArgs Card InterestedWindow RegionOrPosition)
                                                              ; Edited  5-Aug-88 15:56 by Trigg
                                                              ; Make a new note card of type NoteCardType.  If type note
                                                              ; specified, ask the user.
```

;; rht 2/1/85: Added call to NC.MarkCardDirty.

;; fgh 10/15/85 Added extra DatabaseStream argument for use by caching mechanism

;; fgh 11/11/85: Updated to handle new Card object.

;; fgh 2/5/86 Added call to NC.ApplyFn

;; kirk 15May86 Added call to NC.AttachNoteFileName

;; rht 7/4/86: Now checks for read-only notefile before proceeding.

;; kef 8/4/86: Updated to pass NoteCardType argument on to NC.GetNewCard.

;; rht 7/24/87: Replaced WINDOWPROP thing with call to NC.InstallCopyButtonEventFn.

;; pmi 2/25/88: Added InterestedWindow arg to be passed on to the card type's makefn.

;; rht 8/5/88: Added RegionOrPosition arg and passed to card type's makefn.

```
  (DECLARE (GLOBALVARS NC.ShowNoteFileOnCards))
  (if (NC.ReadOnlyNoteFileP NoteFile)
      then NIL
    else (LET (ReturnValue Window)
              (COND
                 ([SETQ NoteCardType (OR NoteCardType (NC.AskNoteCardType (fetch (NoteFile Menu) of NoteFile]
                  (SETQ Card (OR (PROGN (type? Card Card)
                                        Card)
                                 (NC.GetNewCard NoteFile NoteCardType)))
                  (NC.SetNewCardFlg Card T)
                  (NC.ActivateCard Card)
                  (NC.SetType Card NoteCardType)
                  [COND
                     ((OR [NULL (ERSETQ (SETQ ReturnValue (NC.ApplyFn MakeFn Card Title NoDisplayFlg
                                                                      TypeSpecificArgs InterestedWindow
                                                                      RegionOrPosition]
                          (NULL ReturnValue))
                      (NC.SetStatus Card 'DELETED)
                      (NC.DeactivateCard Card T))
                     (T (SETQ Window (WINDOWP ReturnValue))
                        [COND
                           ((NULL (NC.RetrieveTitle Card))
                            [SETQ Title (NC.SetTitle Card (SETQ Title (COND
                                                                        ((STRINGP Title)
                                                                         Title)
                                                                        ((AND Title (OR (LITATOM Title)
                                                                                        (NUMBERP Title)))
                                                                         (MKSTRING Title))
                                                                        (T "Untitled"]
                            (AND Window (WINDOWPROP Window 'TITLE Title)))
                           (T (NC.SetTitle Card (MKSTRING (NC.RetrieveTitle Card]
```

```
                    (COND
                       (Window (WINDOWADDPROP Window 'CLOSEFN (FUNCTION NC.QuitCard)
                                      'FIRST)
                              (WINDOWPROP Window 'NoteCardObject Card)
                              (NC.InstallCopyButtonEventFn Window)))
                    (NC.SetTitleDirtyFlg Card T)
                    ;; Reset the type in case of recursive calls change the type.  Always want the highest level type in a recursive descent
                    (NC.SetType Card NoteCardType)          ; Insure that a link ptr is set up during the first save
                    (NC.SetLinksDirtyFlg Card T)
                    (NC.SetPropListDirtyFlg Card T)         ; Mark that substance is dirty.
                    (NC.MarkCardDirty Card T)
                    (if NC.ShowNoteFileOnCards
                        then (NC.AttachNoteFileName Window]
                ReturnValue])
```

## (NC.QuitCard

```
  [LAMBDA (CardIdentifier CallCloseWFlg DontSaveFlg DontRecacheFlg InterestedWindow OperationMsg QuietFlg
                Don'tDeactivateFlg)                         ; Edited 23-Nov-88 16:55 by krivacic
```

;;; Force note card specified by ID to quit or stop

;; rht 2/9/85: New arg DontSaveFlg prevents NC.CardSaveFn from being called. Used when aborting a card. This is NOT equivalent to ;; NC.QuitWithoutSaving.

;; rht 6/25/85: Now moves card off screen before saving if NC.CloseCardsOffScreenFlg is non-nil.

;; rht 6/25/85: Brought the insure proper filing check back here from NC.CardSaveFn. Bails out if user cancelled operation inside of ;; NC.InsureProperFiling

;; fgh 11/11/85: Updated to handle CardID and CardInfo objects.

;; fgh 1/16/86 Put in code to insure that if one of the TopLevelCards is quit then it is reactivated immedialtely to make sure it stays cached for fast ;; access.

;; fgh 2/5/86 Added call to NC.ApplyFn

;; fgh 5/2/86 Added DontRecacheFlg arg

;; fgh 6/9/86 Added code to check to make sure other operations are not in progress. And DontCheckOpInProgressFlg arg to match

;; fgh 6/26/86 Added InterestedWindow & OperationMsg arg.

;; rht 7/2/86: Now bails out if notefile is readonly, user confirms, but we're supposed to write down changes.

;; rht 7/13/86: Now takes QuietFlg arg.

;; rht 7/14/86: Call NC.DeactivateCard from here instead of in card type QuitFn. Take a Don'tDeactivateFlg as well.

;; rht 10/7/86: Now removes DELETEME imageobj's from card substance.

;; rht 11/2/86: Now returns DON'T if operation in progress.

;; rht 11/13/86: Now closes open proplist editor if any before saving.

;; rg 3/4/87 rewritten to use new NC.ProtectedCardOperation, removed DontCheckOpInProgressFlg arg

;; rht 3/24/87: Now calls NC.CoerceToInterestedWindow and passes InterestedWindow to NC.InsureProperFiling.

;; rht 4/24/87: Fixed a vmem leak: when CallCloseWFlg is nil it doesn't clear CardObject windowprop.

;; pmi 9/16/87: Undoes previous fix to this function. Needed to get NCLOGGER working. It depends on getting the CardObject off of the Window ;; passed in as CardIdentifier.

;; rht 3/30/88: Restores fix of 4/24/87. NCLOGGER will have to finesse this somehow.

```
(DECLARE (GLOBALVARS NC.RemoveDELETEMEImageObjsFromCardFlg))
(LET ((Card (NC.CoerceToCard CardIdentifier))
      Window ReadOnlyCardFlg)
     (AND (NC.CardP Card)
          (NC.ProtectedCardOperation
           Card "Close Card" InterestedWindow
           (PROG NIL
                 (SETQ ReadOnlyCardFlg (NC.ReadOnlyCardP Card))
                                                        ; The window not being open should mean that it's shrunken.  If
                                                        ; so, expand it.
                 (SETQ Window (NC.FetchWindow Card T))
                 (OR InterestedWindow (SETQ InterestedWindow (NC.CoerceToInterestedWindow Card)))
                 (COND
                    ((AND Window (NOT (OPENWP Window)))
                     (EXPANDW Window)))
           ;; if proper filing says don't quit then get out
                 [OR DontSaveFlg ReadOnlyCardFlg (COND
                                                   ((EQ (NC.InsureProperFiling Card InterestedWindow)
                                                        'DON'T)
                                                    (RETURN 'DON'T]
           ;; If card is readonly but we've made changes that we're supposed to save, then get user confirmation and bail out.
                 (if [AND (NOT DontSaveFlg)
                          (NOT Don'tDeactivateFlg)
                          ReadOnlyCardFlg
                          (NC.CardSomehowDirtyP Card)
                          (NULL QuietFlg)
                          (NOT (PROGN (NC.PrintMsg InterestedWindow T "Card has been changed, but notefile is
                                                   readonly." (CHARACTER 13))
                                      (NC.AskYesOrNo "Want to quit anyway, flushing changes? " NIL
```

```
                                        'Yes NIL InterestedWindow]
                    then (RETURN 'DON'T))
        ;; Otherwise go ahead and quit
            (RETURN (PROGN                                  ; Close open proplist editor if any.
                        [AND Window (LET ((PropListEditorWindow (NC.PropListEditorOpenP Window)))
                                         (AND PropListEditorWindow (CLOSEW PropListEditorWindow]
                        (COND
                           ((AND Window NC.CloseCardsOffScreenFlg)
                            [COND
                               ((NOT (NC.FetchSavedRegion Card))
                                (NC.SetSavedRegion Card (WINDOWPROP Window 'REGION]
                            (MOVEW Window 1500 1500)))
                        (OR DontSaveFlg (if ReadOnlyCardFlg
                                            then (NC.TurnOffDirtyFlgs Card)
                                          else (AND NC.RemoveDELETEMEImageObjsFromCardFlg
                                                    (NC.RemoveDELETEMEImageObjsFromCard
                                                     Card
                                                     (FUNCTION NC.DELETEMEImageObjP)))
                                               (NC.CardSaveFn Card (OR NC.CloseCardsOffScreenFlg
                                                                       QuietFlg)
                                                                InterestedWindow OperationMsg)))
                        (AND Window (WINDOWDELPROP Window 'CLOSEFN (FUNCTION NC.QuitCard)))
                        (PROG1 (NC.ApplyFn QuitFn Card)
                            (AND Window (WINDOWPROP Window 'NoteCardObject NIL))
                            (AND CallCloseWFlg Window (NC.CloseW Window))
                            (OR Don'tDeactivateFlg (NC.DeactivateCard Card))
                            ;; if this is one of the top level cards, then make sure it stays cached
                            (if (AND (NC.TopLevelCardP Card)
                                     (NULL DontRecacheFlg)
                                     (NULL Don'tDeactivateFlg))
                                then (NCP.ActivateCards Card)))])
```

## (NC.CloseW

```
[LAMBDA (WINDOW)                                            ; Edited 30-Mar-89 14:59 by sye
  (AND (WINDOWP WINDOW)
       (OPENWP WINDOW)
       ;; (* * if (AND (BOUNDP 'ROOMS:*ROOMS-SYSTEM-DATE*) (ROOMS:ROOM-P ROOMS:*CURRENT-ROOM*)) then (*
       ;; ROOMS:INTERACTIVE-CLOSE-WINDOW WINDOW ROOMS:*CURRENT-ROOM*) (CLOSEW WINDOW) else (CLOSEW WINDOW))
       (CLOSEW WINDOW])
```

## (NC.CheckFiling

```
[LAMBDA (Card InterestedWindow)                             (* rht%: "26-Mar-87 19:43")

        (* Check to make sure this card has a contents hook of some sort.
        If not, hook it up to a contents card.)

        (* * rht 12/8/84%: Now checks whether both cards *and* fileboxes have been filed.)

        (* * rht 12/9/84%: Now files in orphan filebox if NC.ForceFiling flag is off, without bothering the user.)

        (* * rht 2/9/85%: Added call to NC.CardNeedsFilingP)

        (* * fgh |11/12/85| Updated to handle Card and NoteFile objects.)

        (* * fgh |6/9/86| Updated to set operation in progress indicator.)

        (* * rg |1/28/87| Make sure we always return a useful value)

        (* * rg |3/4/87| rewritten for new version of NC.ProtectedCardOperation)

        (* * rht 3/23/87%: Now takes InterestedWindow arg.)

        (* * Rht 3/24/87%: Now calls NC.CoerceToInterestedWindow)

        (* * rht 3/26/87%: Changed msg slightly.)

    (NC.ProtectedCardOperation Card "Filing" NIL (COND
                                                   [(NC.CardNeedsFilingP Card)
                                                    (OR InterestedWindow (SETQ InterestedWindow (
                                                                           NC.CoerceToInterestedWindow
                                                                                 Card)))
                                                    (COND
                                                       (NC.ForceFilingFlg (NC.MakeFilingLinks
                                                                           Card
                                                                           (CONCAT "This card (" (NC.RetrieveTitle
                                                                                                  Card)
                                                                                   ") is not currently filed in
                                                                                   a FileBox in this notefile."
                                                                                   (CHARACTER 13))
                                                                           InterestedWindow))
                                                       (T (NC.PrintMsg InterestedWindow T "This card ("
                                                                       (NC.RetrieveTitle Card)
```

```
                                                    ") is not currently filed in a FileBox."
                                                    (CHARACTER 13)
                                                    "It is being filed in the ToBeFiled FileBox."
                                                    (CHARACTER 13))
                                                (NC.HookToOrphanCard Card (fetch (NoteFile
                                                                                     ToBeFiledCard)
                                                                                  of (fetch (Card NoteFile)
                                                                                        of Card]
                                            (T T])
```

## (NC.CheckTitle

```
[LAMBDA (Card InterestedWindow)                                    (* rht%: "28-Apr-87 11:43")
                                                                   (* If card specified by ID has no title, ask the user for a title.)
```

(* * rht 11/19/84%: Now checks NC.ForceTitlesFlg before griping.)

(* * rht 12/6/84%: Now sends ID rather than Window to NC.AssignTitle.)

(* * fgh 11/11/85%: Updated to handle new Card objects.)

(* * rg |2/3/87| Now returns T if card already titled)

(* * rht 3/23/87%: Now takes InterestedWindow arg.)

(* * rg |4/22/87| now returns T even if card remains untitled ; we need a CANCEL button!)

(* * rht 4/28/87%: Now only calls NC.AssignTitle if current title is NIL, i.e.
this is a new card.)

```
  (DECLARE (GLOBALVARS NC.ForceTitlesFlg))
  (OR InterestedWindow (SETQ InterestedWindow (NC.FetchWindow Card)))
  (LET (Title)
       (COND
          ([AND NC.ForceTitlesFlg (OR (NULL (SETQ Title (NC.RetrieveTitle Card)))
                                      (AND (NC.FetchNewCardFlg Card)
                                           (STREQUAL Title "Untitled"]
           (NC.PrintMsg InterestedWindow T "This note card has no title." (CHARACTER 13))
           (NC.AssignTitle Card T NIL InterestedWindow))
          ((AND (NULL NC.ForceTitlesFlg)
                (NULL (NC.RetrieveTitle Card)))
           (NC.AssignTitle Card NIL "Untitled" InterestedWindow)))
       T])
```

## (NC.ForceFilingForCardTypeP

```
[LAMBDA (CardType)                                                 (* rht%: " 6-Oct-86 10:58")
```

(* * Does this card type have to be filed somewhere? For now checks property, should eventually be a slot in the card type def'n.)

```
  (NULL (GETPROP CardType 'Don'tForceFilingFlg])
```

## (NC.InsureProperFiling

```
[LAMBDA (Card InterestedWindow)                                    (* Randy.Gobbel " 1-Apr-87 17:33")
                                                                   (* Called when any type of note card is being quitted from, i.e., closed)
```

(* * rht 12/9/84%: Moved check of the NC.ForceFiling flag into NC.CheckContentsHooks.)

(* * fgh |11/12/85| Updated to handle Card and NoteFile objects.)

(* * fgh |6/27/86| Changed format to allow being killed by ERROR!)

(* * rg |1/28/87| Now returns CANCELLED if any of the component tests fail)

(* * rht 3/23/87%: Now takes InterestedWindow arg.)

(* * rg |4/1/87| now returns DON'T if component tests fail)

```
  (OR (AND (NULL (NC.FetchBeingDeletedFlg Card))
           (NC.CheckTitle Card InterestedWindow)
           (NC.CheckFiling Card InterestedWindow))
      'DON'T])
```

## (NC.QuitWithoutSaving

```
[LAMBDA (CardIdentifier)                                           (* Randy.Gobbel "16-Mar-87 18:25")
```

(* Quit from a note card without saving its contents on the database.
But must make sure that any updates that would have been done to this card even if it had been on the database are carried out on the old image currently on the database)

(* * rht 2/1/85%: Now only writes out links if it has to. Also resets dirty flags and calls normal quit procedure.
I think we've still got possible problems with recently changed titles, both ours and those of cards we point to.)

(* * rht 2/3/85%: Now handles title changes properly. Also fixed so that changes to from links will force links to be written.)

(* * rht 2/9/85%: Now accepts IDs as arg.)

(* * fgh 11/12/85%: Updated to handle Card objects. Removed DatabaseStream argument.)

(* * fgh |6/4/86| No longer asks for confirmation if no changes have been made.)

(* * kef 8/7/86%: Now compares ToLinks by NC.SameUIDP rather than EQ.
Situation comes up in Server, when the same UIDs may not be EQ.
Also, now obtains write lock on FROMLINKS before deleting FROMLINKS on the DestinationCard of a changed FROMLINK.)

(* * fgh |8/30/86| Converted to use NC.WithWrotePermission macro.)

(* * rht 10/16/86%: Changed NC.ProtectedCardOperation wrapper placement so that card deletion could still happen for new cards.)

(* * rg |3/3/87| Rewrote NC.ProtectedCardOperation wrapping)

(* * rg |3/16/87| Now uses NC.DeleteNoteCard instead of NC.DeleteNoteCards)

```
(LET (Window Card NewFromLinks ToLinks DeletedLinks Process TextObj MadeNewLinksP ChangedFromLinksP
          OldFromLinks OldToLinks)
     (SETQ Card (NC.CoerceToCard CardIdentifier))
     (NC.ProtectedCardOperation
      Card "Close w/o Saving" NIL (SETQ Window (NC.FetchWindow Card))
      (SETQ ToLinks (NC.FetchToLinks Card))
      (SETQ NewFromLinks (NC.FetchFromLinks Card))
      (if (OR (NULL (OR (NC.CardDirtyP Card)
                        (NC.FetchLinksDirtyFlg Card)
                        (NC.FetchTitleDirtyFlg Card)
                        (NC.FetchPropListDirtyFlg Card)))
              (NC.AskYesOrNo "Are you sure you want to flush your changes?" " -- " "Yes" T Window NIL NIL))
          then
```

(* * Elther no changes have been made or the user has confirmed.)

```
              (if (EQ (fetch (Card Status) of Card)
                      'ACTIVE)
                  then                                          (* Read old card parts off notefile.)
                       (NC.GetNoteCard Card)                    (* If from links have changed then we'll have to write the new ones out.)
                       (SETQ OldFromLinks (NC.FetchFromLinks Card))
                       (if (NOT (NC.SameLinksP NewFromLinks OldFromLinks))
                           then (NC.SetFromLinks Card NewFromLinks)
                                (SETQ ChangedFromLinksP T))
```

(* Delete the back links from all links that are not in the old to link set.)

```
                       (SETQ OldToLinks (NC.FetchToLinks Card))
                       [for ToLink in ToLinks unless (for Link in OldToLinks eachtime (BLOCK)
                                                          thereis (NC.SameLinkP Link ToLink))
                          do (LET ((DestinationCard (fetch (Link DestinationCard) of ToLink)))
                                                          (* Grab the writelock on the FROMLINKS before trying to write out the change.)
                                  (NC.WithWritePermission DestinationCard 'FROMLINKS (NC.DelFromLink ToLink)
                                         (SETQ MadeNewLinksP T]
```

(* Check to make sure none of the old To links point to a deleted card.
If so, replace with a deleted marker.)

```
                       (SETQ DeletedLinks (bind DestinationCard for Link in OldToLinks eachtime (BLOCK)
                                             when [PROGN (SETQ DestinationCard (fetch (Link DestinationCard)
                                                                                      of Link))
                                                         (AND (NOT (NC.ActiveCardP DestinationCard))
                                                              (EQ 'DELETED (NC.FetchStatus DestinationCard]
                                             collect (NC.DelReferencesToCard Card DestinationCard)
                                                     Link))
                       (NC.SetToLinks Card (LDIFFERENCE OldToLinks DeletedLinks))
```

(* Mark links dirty if we had to delete any to links or if the from links changed.)

```
                       (NC.SetLinksDirtyFlg Card (OR DeletedLinks ChangedFromLinksP))
                                                          (* Mark the substance dirty if had to delete any to links.)
                       (if DeletedLinks
                           then (NC.MarkCardDirty Card)
                         else (NC.MarkCardDirty Card 'RESET))
                       (NC.SetPropListDirtyFlg Card NIL)
                       (if (NC.FetchTitleDirtyFlg Card)
                           then
```

(* If we've changed title and any of the cards pointing to this one are up on the screen, then we need to update their images.)

```
                                (for FromLink in OldFromLinks bind SourceCard eachtime (BLOCK)
                                   when (AND (NC.ActiveCardP (SETQ SourceCard (fetch (Link SourceCard)
                                                                                     of FromLink)))
```

```
                                                (WINDOWP (NC.FetchWindow SourceCard)))
                                           do (NC.UpdateLinkImages SourceCard Card))
                                       (NC.SetTitleDirtyFlg Card NIL))
                                  (NC.QuitCard Card T)
                             else
```

(* If the card has never been written to the database quit w/o saving is equivalent to deleting the card.)

```
                        (NC.PrintMsg Window T "This card has never been saved." (CHARACTER 13)
                               "It will be deleted from the database."
                               (CHARACTER 13))
                        (NC.DeleteNoteCard Card T])
```

## (NC.UnfileNoteCard

```
[LAMBDA (WindowOrTextStream)                                    (* Randy.Gobbel " 2-Apr-87 15:38")
                                                                (* Take a notecard out of a file box.
                                                                Called fom title bar menu.)
```

(* * fgh |11/12/85| Updated to handle Card objects.)

(* * fgh |6/9/86| Added code to check to make sure that another operation is not in progress on this card when this fn is called.)

(* * pmi 12/5/86%: Modified message to NC.SelectNoteCards to mention SHIFT-selection.)

(* * pmi 12/12/86%: Removed obsolete ReturnLinksFlg argument in call to NC.SelectNoteCards.)

(* * rht 2/2/87%: Fixed bug |#418:| Trashed all the stuff about opening PropListEditor on the fromlinks. Now just uses SelectNoteCards to choose the fileboxes to delete from.)

(* * rg |3/4/87| rewritten for new NC.ProtectedCardOperation)

(* * rg |3/18/87| added NC.CardSelectionOperation wrapper)

(* * RG |4/2/87| changed NC.CardSelectionOperation to NCP.WithLockedCards)

```
(DECLARE (GLOBALVARS NC.SelectingParentsMenu))
(NCP.WithLockedCards (LET ((Card (NC.CoerceToCard WindowOrTextStream)))
                          (NC.ProtectedCardOperation
                           Card "Unfile" NIL (LET* [(FromLinks (NC.FetchFromLinks Card))
                                                    (Parents (for FromLink in FromLinks
                                                                when (FMEMB (fetch (Link Label) of FromLink)
                                                                            '(FiledCard SubBox))
                                                                collect (fetch (Link SourceCard) of FromLink]
                                                   (for Box in (NC.SelectNoteCards NIL
                                                                      [FUNCTION (LAMBDA (Box)
                                                                                  (AND (NC.FileBoxP Box)
                                                                                       (FMEMB Box Parents]
                                                                      NC.SelectingParentsMenu Card "Please
                                                                      shift-select the file box(es) from
                                                                      which to remove this card. ")
                                                      do (for FromLink in FromLinks
                                                            when (NC.SameCardP Box (fetch (Link SourceCard)
                                                                                          of FromLink))
                                                            do (NC.DeleteLink FromLink T])
```

## (NC.UpdateUpdateList

```
[LAMBDA (Card)                                                  (* fgh%: "13-Nov-85 20:36")
```

(* Update the list of update authors and times for the note card specified by ID)

(* * fgh |11/13/85| Updated to handle Card object.)

```
(COND
   ((NC.ActiveCardP Card)
    [COND
       [(NULL (NC.FetchPropList Card))
        (NC.SetPropList Card (LIST 'Updates (LIST (LIST USERNAME (DATE]
       (T (LISTPUT (NC.FetchPropList Card)
                   'Updates
                   (CONS (LIST USERNAME (DATE))
                         (LISTGET (NC.FetchPropList Card)
                                  'Updates]
    (NC.SetPropListDirtyFlg Card T))
   (T (NC.ReportError "NC.UpdateUpdateList" (CONCAT Card ": Card not active on screen"])
```

## (NC.CollectReferences

```
[LAMBDA (Card CheckAndDeleteFlg ReturnLinkIconsFlg ReturnLocationsFlg)
                                                                (* fgh%: " 5-Feb-86 19:54")
```

(* * Collect all the links in a card specified by ID. RETURNS the CONS of a list of link identifiers {described below} and a dirtyflg that is non-NIL if the Substance of ID has been modified {i.e., when a non-valid link is found and CheckAndDeleteFlg is NIL}. IF CheckAndDeleteFlg is non-NIL checks for valid links and deletes those that are not valid.
If ReturnLinkIconsFlg is NIL, returns link icons. Otherwise, returns links.
If ReturnLinkIconsFlg is NIL, returns just the links/link icons. Otherwise, returns the CONS of link/link icon and the

type-specific location of the link icon in the card.)

(* * fgh |11/12/85| Updated to handle Card objects.)

(* * rht 12/19/85%: Fixed to handle types with no CollectReferencesFn defined.)

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

```
(COND
   ((NC.ActiveCardP Card)
    (NC.ApplyFn CollectLinksFn Card CheckAndDeleteFlg ReturnLinkIconsFlg ReturnLocationsFlg))
   (T (NC.ReportError "NC.CollectReferences" (CONCAT "Attempt to call with inactive card:  " Card])
```

## (NC.CardSaveFn

```
[LAMBDA (WindowOrID QuietFlg InterestedWindow OperationMsg)    ; Edited 19-Jan-88 15:51 by Randy.Gobbel
```

(* * rht 2/1/85%: New function for saving ANY kind of card. All strangenesses are handled in NC.CardDirtyP and NC.MarkCardDirty. Added print statements to show what is being saved.
Lets NC.CardDirtyP take care of proper dirty checks.)

(* * rht 2/8/85%: Added InsureFilingFlg)

(* * rht 6/25/85%: Pulled out InsureFilingFlg. That check now done upstairs in NC.QuitCard.)

(* * rht 9/20/85%: Added QuietFlg.)

(* * fgh |11/12/85| Updated to handle Card objects. Removed DatabaseStream object.)

(* * kirk 29Jan86 replaced call on undefined NC.UpdateRegionData with NC.PutRegion)

(* * fgh |6/13/86| Added operations in progress code and DontCheckForOpsInProgressFlg arg.)

(* * fgh |6/26/86| Added InterestedWindow & OperationMsg arg.)

(* * rht 7/4/86%: Added check for readonly notefile.)

(* * kef 7/22/86%: Added something to obtain the write permission on the FROMLINKS if the links have been changed. FROMLINKS aren't ordinarily obtained at edit time like the rest of the links are.)

(* * kef 7/30/86%: Modified to check for Client's concept of whether he owns the write lock or not, thus deciding whether or not to setup the release of the write lock afterwards.)

(* * kef 7/30/86%: Added a check to see if the NewCardFlg was on, then release TITLE and FROMLINKS writelocks. This is needed since ordinary deactivation of cards won't do this;
i.e., only new cards have their TITLE and FROMLINKS also writelocked.)

(* * fgh |8/30/86| Changed APPLY* to NC.ApplyFn where possible.)

(* * rht&pmi 11/21/86%: Now calls WhenSavedFn for card type if any.)

(* * rg |3/4/87| rewritten for new NC.ProtectedCardOperation, removed DontCheckForOpsInProgressFlg)

(* * rht 3/23/87%: Fixed weirdness with InterestedWindow/Window.)

(* * Rht 3/24/87%: Now calls NC.CoerceToInterestedWindow)

(* * rht 3/30/87%: No longer prints messages if nothing got written down.)

(* * rg |3/31/87| fiddled w/ InterestedWindow stuff)

(* * rg |5/20/87| added FORCEOUTPUT on stream after all writes done.
WARNING! THIS IS A KLUDGE, AND WILL ONLY WORK FOR OLD-FASHIONED LOCAL NOTEFILES.
For remote notefiles, we really need to add another operation to the protocol.)

(* * pmi 12/10/87%: Now saves card's region if it has changed through either a reshape or a move.)

```
(LET ((Card (NC.CoerceToCard WindowOrID))
      Window OldRegion NewRegion DoneAPutP)
     (NC.ProtectedCardOperation Card "Save Card" InterestedWindow (LET [(WhenSavedFn (GETPROP (NC.FetchType
                                                                                               Card)
                                                                                       'WhenSavedFn]
                                                                            (AND WhenSavedFn (APPLY* WhenSavedFn
                                                                                               Card)))
           (SETQ Window (NC.FetchWindow Card))
           (OR InterestedWindow (SETQ InterestedWindow (NC.CoerceToInterestedWindow Card)))
           (if (NC.CheckForNotReadOnly Card InterestedWindow "Can't save cards in ")
               then (COND
                       ((OR (NC.CardDirtyP Card)
                            (NC.FetchNewCardFlg Card))
                        (OR QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                                  (NC.FetchTitle Card)
                                                  ": Saving ... "))
                        (OR QuietFlg (NC.PrintMsg InterestedWindow NIL "substance, "))
                        (NC.PutMainCardData Card T)
                        (SETQ DoneAPutP T)
                        (NC.MarkCardDirty Card 'RESET))
```

```
                        ([AND (NOT (NC.FetchBeingDeletedFlg Card))
                              Window
                              (OR [NOT (EQUAL (fetch (REGION WIDTH) of (SETQ OldRegion (NC.FetchRegion Card)))
                                              (fetch (REGION WIDTH) of (SETQ NewRegion (WINDOWPROP
                                                                                        Window
                                                                                        'REGION]
                                  (NOT (EQUAL (fetch (REGION HEIGHT) of OldRegion)
                                              (fetch (REGION HEIGHT) of NewRegion)))
                                  (NOT (EQUAL (fetch (REGION LEFT) of OldRegion)
                                              (fetch (REGION LEFT) of NewRegion)))
                                  (NOT (EQUAL (fetch (REGION BOTTOM) of OldRegion)
                                              (fetch (REGION BOTTOM) of NewRegion]
                         (OR DoneAPutP QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                                             (NC.FetchTitle Card)
                                                             ": Saving ... "))
                         (OR QuietFlg (NC.PrintMsg InterestedWindow NIL "region, "))
                         (NC.PutRegion Card)
                         (SETQ DoneAPutP T)))
                    (COND
                       ((NC.FetchTitleDirtyFlg Card)
                        (OR DoneAPutP QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                                            (NC.FetchTitle Card)
                                                            ": Saving ... "))
                        (OR QuietFlg (NC.PrintMsg InterestedWindow NIL "title, "))
                        (NC.PutTitle Card)
                        (SETQ DoneAPutP T)))
                    (COND
                       ((NC.FetchPropListDirtyFlg Card)
                        (OR DoneAPutP QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                                            (NC.FetchTitle Card)
                                                            ": Saving ... "))
                        (OR QuietFlg (NC.PrintMsg InterestedWindow NIL "proplist, "))
                        (NC.PutPropList Card)
                        (SETQ DoneAPutP T)))
                    [COND
                       ((NC.FetchLinksDirtyFlg Card)
                        (OR DoneAPutP QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                                            (NC.FetchTitle Card)
                                                            ": Saving ... "))
                        (OR QuietFlg (NC.PrintMsg InterestedWindow NIL "links, "))
```

(* Make sure that we have the FROMLINKS of this card. Only necessary because all of the LINKS are written together.)

```
                        (RESETLST
                            (until (NC.ApplyFn ObtainWritePermissionFn Card 'FROMLINKS)
                               do (BLOCK)
                                  (NC.PrintMsg InterestedWindow NIL "waiting for FROMLINKS write
                                                permission...."))
                            (RESETSAVE NIL `(APPLY* ,(fetch (Card ReleaseWritePermissionFn) of Card)
                                                    ,Card FROMLINKS))
                            (NC.PutLinks Card)
                            (SETQ DoneAPutP T))]            (* It's not a new card anymore.)
                    (FORCEOUTPUT (fetch (NoteFile Stream) of (fetch (Card NoteFile) of Card))
                          T)
                    (COND
                       ((NC.FetchNewCardFlg Card)
```

(* If a new card, then make sure we release the FROMLINKS and TITLE.
Necessary because DeactivateCard normally doesn't do this, because the FROMLINKS and TITLE aren't ordinarily owned on an active card.)

```
                        (NC.ApplyFn ReleaseWritePermissionFn Card 'FROMLINKS)
                        (NC.ApplyFn ReleaseWritePermissionFn Card 'TITLE)
                        (NC.SetNewCardFlg Card NIL)))
                    (OR QuietFlg (if DoneAPutP
                                     then (NC.PrintMsg InterestedWindow NIL "Done." (CHARACTER 13))
                                          (NC.ClearMsg InterestedWindow T])
```

## (NC.DetermineDisplayRegion

```
[LAMBDA (Card RegionOrPosition)                                    ; Edited 27-Jul-90 09:28 by tafel
```

;; fgh 11/13/85 Updated to handle Card object.

;; fgh 2/5/86 Changed names DefaultcardWidth to FeatchDefaultWidth ...

;; rht 3/25/87: Changed so as to account for the NoteFile indicator window height if present.

;; pmi 3/31/87: Changed prompt message for placing card to not say '... Card NIL' for a new card.

;; rht 5/16/87: Now takes attached windows into account for cases when RegionOrPosition is a region. But only when the card's window is already ;; open.

;; rg 9/16/87 now brings up card at old pos if BringUpCardAtOldPos prop is T

;; rht 6/9/88: Now checks to see if card's card type has a non-nil NewCardPos slot if RegionOrPosition is nil.

```
(DECLARE (GLOBALVARS NC.BringUpCardAtOldPosFlg NC.ShowNoteFileOnCards))
(LET ((Window (NC.FetchWindow Card))
      CardWinRegion TotalWinRegion Region)
     (if (WINDOWP Window)
```

```
            then (SETQ CardWinRegion (WINDOWPROP Window 'REGION))
                 (SETQ TotalWinRegion (WINDOWREGION Window)))
         (COND
            [(POSITIONP RegionOrPosition)
             (if (SETQ Region (OR TotalWinRegion (NC.FetchRegion Card)))
                 then (CREATEREGION (fetch (POSITION XCOORD) of RegionOrPosition)
                                    (fetch (POSITION YCOORD) of RegionOrPosition)
                                    (fetch (REGION WIDTH) of Region)
                                    (fetch (REGION HEIGHT) of Region))
               else (CREATEREGION (fetch (POSITION XCOORD) of RegionOrPosition)
                                  (fetch (POSITION YCOORD) of RegionOrPosition)
                                  (NC.FetchDefaultWidth Card)
                                  (NC.FetchDefaultHeight Card]
            ([SETQ Region (OR (REGIONP RegionOrPosition)
                              (AND (OR NC.BringUpCardAtOldPosFlg (NC.GetProp Card 'BringUpCardAtOldPos))
                                   (NC.FetchRegion Card]
             (if Window
                 then [create REGION using Region HEIGHT _ (PLUS (fetch (REGION HEIGHT) of TotalWinRegion)
                                                                  (DIFFERENCE (fetch (REGION HEIGHT) of Region)
                                                                              (fetch (REGION HEIGHT) of CardWinRegion)))
                                           WIDTH _ (PLUS (fetch (REGION WIDTH) of TotalWinRegion)
                                                         (DIFFERENCE (fetch (REGION WIDTH) of Region)
                                                                     (fetch (REGION WIDTH) of CardWinRegion]
               else Region))
            (Window TotalWinRegion)
            (T (LET (Width Height TotalHeight CardTitle NewRegion NewCardPos)
                    (if (SETQ Region (NC.FetchRegion Card))
                        then (SETQ Width (fetch (REGION WIDTH) of Region))
                             (SETQ Height (fetch (REGION HEIGHT) of Region))
                      else (SETQ Width (NC.FetchDefaultWidth Card))
                           (SETQ Height (NC.FetchDefaultHeight Card)))
                    (SETQ TotalHeight (if NC.ShowNoteFileOnCards
                                          then (PLUS Height (NC.NoteFileIndicatorWinHeight))
                                        else Height))
                    (if (POSITIONP (SETQ NewCardPos (NC.FetchNewCardPos Card)))
                        then (SETQ NewRegion (CREATEREGION (fetch (POSITION XCOORD) of NewCardPos)
                                                           (fetch (POSITION YCOORD) of NewCardPos)
                                                           Width Height))
                      else (SETQ CardTitle (NC.FetchTitle Card))
                           (replace (REGION HEIGHT) of [SETQ NewRegion (GETBOXREGION
                                                                        Width TotalHeight (GETMOUSEX)
                                                                        (IDIFFERENCE (GETMOUSEY)
                                                                               TotalHeight)
                                                                        NIL
                                                                        (if CardTitle
                                                                            then (CONCAT "Please specify location
                                                                                         for Note Card " CardTitle)
                                                                          else (CONCAT "Please specify location for
                                                                                       new " (NC.RetrieveType Card)
                                                                                       " Card"]
                                    with Height))
                          NewRegion])
```

## (NC.AbortCard

```
[LAMBDA (Card QuietFlg)                                    (* rht%: "16-May-87 20:50")
```

(* * Kill this card's process and its window.)

(* * fgh |11/13/85| Updated to handle Card object.)

(* * rht 7/6/86%: Now passes non-nil Don'tRecacheFlg to NC.QuitCard.)

(* * rht 7/13/86%: Now takes QuietFlg and passes to NC.QuitCard.)

(* * rht 5/16/87%: Now passes non-nil CloseWFlg to NC.QuitCard.)

```
   (NC.QuitCard Card T T T NIL NIL QuietFlg])
```

## (NC.CardNeedsFilingP

```
[LAMBDA (Card)                                             (* rht%: "26-Mar-87 19:30")
```

(* * Returns non-nil if this card is filed in some not currently filed anywhere unless it's a top level card.
Assumes card is active.)

(* * fgh |11/12/85| Updated to handle Card objects.)

(* * rht 10/6/86%: Now checks for card types that don't have to be filed.
Also makes sure that there's at least one filing link that's within this notefile.)

(* * rht 12/8/86%: Now checks for individual cards that don't require filing.)

(* * rg 12/18/86%: Added check of FilingLinkTypeFlg prop.)

(* * rht 3/26/87%: Fixed check for crossfilelink card.)

```
    (LET ((NoteFile (fetch (Card NoteFile) of Card)))
         (AND (NOT (NC.TopLevelCardP Card))
              (NC.ForceFilingForCardTypeP (NC.FetchType Card))
              (NOT (NCP.CardProp Card 'Don'tRequireFilingFlg))
              (for Link in (NC.FetchFromLinks Card) never (AND (OR (FMEMB (fetch (Link Label) of Link)
                                                                          '(FiledCard SubBox))
                                                                    (GETPROP (fetch (Link Label) of Link)
                                                                          'FilingLinkTypeFlg))
                                                               (NOT (NC.CrossFileLinkCardP (fetch (Link SourceCard)
                                                                                              of Link])
```

## (NC.TopLevelCardP

```
  [LAMBDA (Card)                                                    (* rht%: "13-Jul-86 17:35")

          (* * Is Card a top level card in its NoteFile?)

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

## (NC.MakeDummyRegion

```
  [LAMBDA (Card)                                                    (* fgh%: " 5-Jun-86 19:54")

          (* * Returns a region based at (0 0) with default width and height according to Type.)

          (* * fgh |2/5/86| Changed names DefaultcardWidth to FeatchDefaultWidth |...|)

          (* * fgh |6/5/86| Now leaves room for scroll bars.)

    (CREATEREGION 20 20 (NC.FetchDefaultWidth Card)
           (NC.FetchDefaultHeight Card])
```

## (NC.ValidCardP

```
  [LAMBDA (Card)                                                    (* rht%: "14-Nov-85 19:57")

          (* * Is ID a currently extant card or box?)

          (* * rht 1/31/85%: Now reads status from index array rather than file.)

          (* * fkr |11/8/85| Updated to handle NoteFile objects and new Card scheme.)

    (AND (NC.CardP Card)
         (OR (EQ 'ACTIVE (NC.FetchStatus Card))
             (NC.FetchNewCardFlg Card])
```

## (NC.TitleBarButtonEventFn

```
  [LAMBDA (Window)                                                  (* rht%: " 6-May-86 12:10")

          (* * if inside title region, bringup left or middle menu, otherwise just call the oldbuttoneventfn)

    (LET (MiddleButtonMenu LeftButtonMenu)
         (COND
            ((INSIDEP (DSPCLIPPINGREGION NIL Window)
                      (LASTMOUSEX Window)
                      (LASTMOUSEY Window))
             (APPLY* (WINDOWPROP Window 'OLDBUTTONEVENTFN)
                     Window))
            ([AND (LASTMOUSESTATE MIDDLE)
                  (type? MENU (SETQ MiddleButtonMenu (WINDOWPROP Window 'TitleBarMiddleButtonMenu]
             (APPLY* (OR (MENU MiddleButtonMenu)
                         (FUNCTION NILL))
                     Window))
            ([AND (LASTMOUSESTATE LEFT)
                  (type? MENU (SETQ LeftButtonMenu (WINDOWPROP Window 'TitleBarLeftButtonMenu]
             (APPLY* (OR (MENU LeftButtonMenu)
                         (FUNCTION NILL))
                     Window])
```

## (NC.InstallTitleBarMiddleMenu

```
  [LAMBDA (Window CardType)                                         (* pmi%: " 1-Apr-87 16:47")

          (* * Make a middle button title bar menu and install.)

          (* * pmi 4/1/87%: Added NC.MenuFont to all menus)

    (DECLARE (GLOBALVARS NC.MenuFont))
    (WINDOWPROP Window 'TitleBarMiddleButtonMenu (create MENU
                                                         ITEMS _ (NC.GetCardTypeField MiddleButtonMenuItems
                                                                       CardType)
                                                         CENTERFLG _ T
                                                         MENUFONT _ NC.MenuFont
                                                         ITEMHEIGHT _ (IPLUS (FONTPROP NC.MenuFont 'HEIGHT)
                                                                             1])
```

)

;;; Prop List Editor

(DEFINEQ

## (NC.AddPropToPropList

```
  [LAMBDA (editW)                                          (* rht%: "16-Jan-87 16:06")
                                                           (* Add a new prop to the propList being edited in editW)

          (* * fgh |2/10/86| Fix problem with ClearMsg deleting editing window.)

          (* * rht 4/11/86%: Fixed bug whereby it couldn't add property when none exist.)

          (* * rht 1/16/87%: Now uses prop names on windowprop rather than number of props.
          Makes sure that propname doesn't already exist and that it's not system-reserved.)

    (PROG [(promptWindow (GETPROMPTWINDOW editW 5 NIL NIL))
           selectedObject propName propValue insertPtr insertChars beginPtr (textStream (TEXTSTREAM editW))
           (PropNames (WINDOWPROP editW 'PROPERTYLIST.PROPNAMES]
          (SETQ propName (MKATOM (NC.AskUser "New Property Name is = " NIL NIL T promptWindow)))
          (COND
             ((FMEMB propName PropNames)
              (NC.PrintMsg promptWindow T (CONCAT "'" propName "' property already exists." (CHARACTER 13)))
              (RETURN))
             ((FMEMB propName NC.SystemCardPropNames)
              (NC.PrintMsg promptWindow T (CONCAT "'" propName "' is a system-reserved property name."
                                                  (CHARACTER 13)))
              (RETURN)))
          (SETQ propValue (NC.AskUser (CONCAT "Value for " propName " property = ")
                                      NIL NIL T promptWindow))       (* Insert these values into the editW)
          (if PropNames
              then                                                   (* Position just before selected button)
                   (NC.PrintMsg promptWindow T "Please select a property before which to insert this new
                         property." (CHARACTER 13))
                   (SETQ selectedObject (NC.SelectProperty editW))
                   (NC.ClearMsg promptWindow T)
                   (AND (NULL selectedObject)
                        (RETURN NIL))
                   (SETQ insertPtr (TEDIT.FIND.OBJECT textStream selectedObject))
            else                                                     (* No properties to insert before.)
                 (SETQ insertPtr 1))
          (SETQ beginPtr insertPtr)                                  (* Insert a button with this property name)
          (TEDIT.INSERT.OBJECT (MBUTTON.CREATE propName (FUNCTION NC.EditPropButtonFN)
                                       (FONTCREATE 'HELVETICA 10 'BOLD))
                 textStream insertPtr)                               (* Spacer between Prop and Value)
          (add insertPtr 1)
          (TEDIT.INSERT textStream (CHARACTER 9)
                 insertPtr)
          (TEDIT.LOOKS textStream '(PROTECTED ON)
                 insertPtr 1)
          (add insertPtr 1)                                          (* Create a new field (Copied from JBS's
                                                                     \TEXTMENU.DOC.CREATE))
          (TEDIT.INSERT textStream (CONCAT "  {}" (CHARACTER 13))
                 insertPtr)
          (TEDIT.LOOKS textStream '(PROTECTED ON)
                 insertPtr 5)
          (TEDIT.LOOKS textStream '(PROTECTED ON SELECTPOINT ON)
                 (IPLUS insertPtr 2)
                 1)
          (TEDIT.INSERT textStream (MKSTRING propValue)
                 (IPLUS insertPtr 3))
          (TEDIT.LOOKS textStream '(PROTECTED OFF SELECTPOINT OFF)
                 (IPLUS insertPtr 3)
                 (NCHARS (MKSTRING propValue)))
          (add insertPtr (NCHARS (MKSTRING propValue)))
          (add insertPtr 5)
          (WINDOWADDPROP editW 'PROPERTYLIST.PROPNAMES propName])
```

## (NC.CloseAllPropListEditors

```
  [LAMBDA (Card)                                           (* rht%: "11-Aug-86 19:11")
                                                           (* Force all prop list editors open on card ID to close without
                                                           saving changes.)

          (* * fgh 11/11/85%: Updated to handle new Card object.)

          (* * rht 8/11/86%: Now makes sure that proplist window gets closed.)

    (for AttachedWindow in (ALLATTACHEDWINDOWS (NC.FetchWindow Card)) when (WINDOWPROP AttachedWindow
                                                                                  'PropListEditor)
       eachtime (BLOCK) do (NC.ClosePropListEditor AttachedWindow 'NoSave)
                           (AND (OPENWP AttachedWindow)
                                (CLOSEW AttachedWindow])
```

## (NC.ClosePropListEditor

```
[LAMBDA (Window SaveFlg)                                    ; Edited  3-Dec-87 19:00 by rht:
```

(* Close the prop list editor, saving or not saving the edited prop list as specified by the SaveFlg or by the user if SaveFlg is NIL)

(* * fgh |11/13/85| Updated to handle Card object.)

(* * fgh |6/8/86| Added call to RAPOSITIONATTACHEDWINDOWS)

(* * rht |8/12/86| Now uses TEXTOBJ to get TextObj from window.)

(* * kef 7/16/86%: Added the call to release the write permission should the window property declare that it be necessary.)

(* * kef 7/22/86%: Now only releases the write permission if the card is inactive.
The reason is that it was determined that a card being edited would update the property list upon being closed.)

(* * fgh |8/30/86| Converted APPLY* to NC.ApplyFn. Added Card local var.)

(* * rht 11/12/86%: Now calls MAINWINDOW instead of WINDOWPROP to get Window's mainwindow and recovers card from a prop of Window.)

(* * rht 1/16/87%: Now checks that tedit stream has been modified before using user as to whether to save changes.
No longer puts up stupid save/cancel menu.)

(* * rht&rg 4/24/87%: Now smashes more windowprops to prevent storage leaks.)

(* * rg |11/20/87| now makes window read-write so TEDIT.QUIT will work -
UGH!!)

```
(DECLARE (GLOBALVARS NC.SavePropEditMenu))
(PROG (TextObj Answered OldPropList NewPropList MainWindow Card)
      (SETQ Card (WINDOWPROP Window 'SavedCardObject))
      (SETQ MainWindow (MAINWINDOW Window))
      (DETACHWINDOW Window)
      [SETQ TextObj (CAR (NLSETQ (TEXTOBJ Window]
      (SETQ OldPropList (WINDOWPROP Window 'PROPERTYLIST.BEING.EDITED))
      [AND TextObj (COND
                       ((EQ SaveFlg 'NoSave))
                       ((EQ (WINDOWPROP Window 'PropListEditor)
                            'ShowOnly))
                       ((OR (EQ SaveFlg 'Save)
                            (TEDIT.STREAMCHANGEDP TextObj))
                        (SETQ NewPropList (NC.ExtractPropList Window))
                        (NC.ProcessEditedPropList NewPropList OldPropList Card]
      (WINDOWPROP Window 'TEDIT.MENU NIL)
      (WINDOWPROP Window 'TEDIT.PROPS NIL)
      (WINDOWPROP Window 'PropListEditor NIL)
      (WINDOWPROP Window 'SavedCardObject NIL)
      (WINDOWPROP Window 'PROPERTYLIST.BEING.EDITED NIL)
      (WINDOWPROP Window 'PROPERTYLIST.PROPNAMES NIL)
      (WINDOWDELPROP Window 'CLOSEFN (FUNCTION NC.ClosePropListEditor))
      (COND
         ((AND (WINDOWPROP Window 'ReleaseWritePermissionP)
               (NOT (NC.ActiveCardP Card)))
          (NC.ApplyFn ReleaseWritePermissionFn Card 'PROPLIST)
          (WINDOWPROP Window 'ReleaseWritePermissionP NIL)))
      [COND
         (TextObj [if (NULL (WINDOWPROP Window 'PROCESS))
                      then (NC.MakeTEditReadWrite Window)
                           (LET ((Process (MAKE-NEW-TEDIT-PROCESS Window)))
                                (SETQ TEDIT-PROCESSES (CONS Process TEDIT-PROCESSES]
                  (replace (TEXTOBJ \DIRTY) of TextObj with NIL)
                  [\TEDIT.QUIT (CAR (MKLIST (fetch (TEXTOBJ \WINDOW) of TextObj]
                  (until (fetch (TEXTOBJ EDITFINISHEDFLG) of TextObj) do (BLOCK]
      (ADD.PROCESS `(PROGN (until (NULL (OPENWP ,Window)) do (BLOCK))
                           (REPOSITIONATTACHEDWINDOWS ,MainWindow])
```

## (NC.DelPropFromList

```
[LAMBDA (editW)                                             (* rht%: "16-Jan-87 15:52")
```

(* * rht 4/11/86%: Now decrements PROPERTYLIST.LENGTH windowprop.)

(* * rht 1/16/87%: Now uses prop names on windowprop rather than number of props.)

```
  (PROG ((promptWindow (GETPROMPTWINDOW editW 5 NIL NIL))
         (tobj (WINDOWPROP editW 'TEXTOBJ))
         (stream (WINDOWPROP editW 'TEXTSTREAM))
         selectedObject CH#)
        (if (NULL (WINDOWPROP editW 'PROPERTYLIST.PROPNAMES))
            then (NC.PrintMsg promptWindow T "No properties to delete.")
                 (RETURN))
        (NC.PrintMsg promptWindow T "Please select property to be deleted.")
        (SETQ selectedObject (NC.SelectProperty editW))
        (NC.ClearMsg promptWindow T)
```

```
          (if (IMAGEOBJPROP selectedObject 'EditPropListNoDelete)
              then (NC.PrintMsg promptWindow T "Selected property:value pair cannot be deleted." (CHARACTER
                                                                                                    13))
            else (SETQ CH# (TEDIT.FIND.OBJECT tobj selectedObject))
                 (MBUTTON.FIND.NEXT.FIELD tobj CH#)              (* Delete everything between the imageobj and the end-of-line.)
                 (TEDIT.LOOKS tobj '(PROTECTED OFF)
                        CH#
                        (IPLUS 2 (IDIFFERENCE (fetch CHLIM of (fetch SCRATCHSEL of tobj))
                                              CH#)))
                 (TEDIT.DELETE stream CH# (IPLUS 2 (IDIFFERENCE (fetch CHLIM of (fetch SCRATCHSEL of tobj))
                                                                CH#)))
                 (TEDIT.SETSEL stream (GETEOFPTR stream)
                        0
                        'RIGHT)
                 (WINDOWDELPROP editW 'PROPERTYLIST.PROPNAMES (IMAGEOBJPROP selectedObject 'MBTEXT])
```

## (NC.EditPropButtonFN

```
  [LAMBDA (imageObject sel window)                                  (* fgh%: "20-Apr-84 21:17")
```

(* This is the default function called whenever the user selects a button in the NC.EditPropList window.)

```
    (WINDOWPROP window 'SelectedObject imageObject])
```

## (NC.EditProperties

```
  [LAMBDA (TextStreamOrWindow)                                      (* Randy.Gobbel " 2-Dec-87 18:15")
```

(* Open a property list editor for the card corresponding to TextStream.
Called from Title bar menus.)

(* * fgh |11/13/85| Updated to handle Card object.)

(* * rht 4/11/86%: No longer sticks dates and Updates in property list.
Only user-defined stuff.)

(* * kef 7/22/86%: Added call to card's obtain write permission function for the PROPLIST.)

(* * fgh |8/30/86| Converted APPLY* to NC.ApplyFn.)

(* * rht 1/16/87%: Now uses NC.SystemCardPropNames globalvar.)

(* * rg |4/1/87| added NC.ProtectedCardOperation wrapper)

(* * rg |21/2/87| Now editor will be read-only if card is read-only)

```
    (DECLARE (GLOBALVARS NC.SystemCardPropNames))
    (LET ([Card (NC.CoerceToCard (OR (WINDOWP TextStreamOrWindow)
                                     (NC.TEditWindow TextStreamOrWindow]
          PropList PropEditorWindow)
         (NC.ProtectedCardOperation Card "Edit Properties" NIL (SETQ PropList (NC.FetchPropList Card))
                (COND
                   ((NC.ApplyFn ObtainWritePermissionFn Card 'PROPLIST)
                    (WINDOWPROP [SETQ PropEditorWindow (NC.OpenPropListEditor
                                                        TextStreamOrWindow
                                                        (for SubList on PropList by (CDDR SubList)
                                                           when (NOT (FMEMB (CAR SubList)
                                                                            NC.SystemCardPropNames))
                                                           join (LIST (CAR SubList)
                                                                      (CADR SubList)))
                                                        "Edit Property List"
                                                        (NC.FetchUserDataProp Card 'ReadOnly]
                                'ReleaseWritePermissionP T)
                     PropEditorWindow)
                    (T (NC.CardPartBusy Card 'PROPLIST])
```

## (NC.EditPropList

```
  [LAMBDA (propList window showOnlyFlg showLinksFlg)                ; Edited  7-Feb-89 17:04 by krivacic
```

;;; propList is a list of RECORDS of type PropListItem

;;; Edit a property list using the TEDIT menu-based editor.  The var window is the window to use.  If none supplied, get one from user.

;; rht 4/11/86: Now stashes length of propLis on WINDOWPROP.

;; rht 8/12/86: Moved code to add NC.ClosePropListEditor on CLOSEFN from NC.OpenPropListEditor to here so that it can before
;; TEDIT.DEACTIVATE.WINDOW on the CLOSEFN list.

;; rht 1/16/87: Now stashes prop names on windowprop rather than number of props.

;; pmi 3/25/87: Added NC.MenuFont to all menus

;; rg 11/19/87 menu items put on global vars (for make read-only)

;; rht/rg 11/20/87 Now kills any existing edit process in editW before starting new one.

;; rg 11/23/87 added showLinksFlg arg

;; rht 2/10/88: Now passes non-nil LEAVETTY flag as PROP in call to TEDIT in showlinks case.

;; bk 1/23/89: Fix timing bug of TEDIT-PROCESS-P.

```
;; bk 2/7/89: Add back the NC.ShowPropListMenu when |showOnlyFlg| |showLinksFlg| (lost in the TEDIT-PROCESS-P changes.
(DECLARE (GLOBALVARS NC.MenuFont NC.ShowPropListMenu NC.EditPropListMenu))
(PROG (menuStream textObj editW button editProcess (font (FONTCREATE 'HELVETICA 8))
            (CH# 1)
            (ENDCH# 1))                                          ; Init the editList and the propFnsList
                                                                 ; Create a TEDITMenu that reflects the structure of the proplist
      [SETQ menuStream (\TEXTMENU.DOC.CREATE
                        (for X in propList
                           join (NCONC [LIST (LIST 'MB.BUTTON (MKSTRING (fetch (PropListItem PropertyName)
                                                                        of X))
                                             (OR (fetch (PropListItem ButtonFn) of X)
                                                 (FUNCTION NC.EditPropButtonFN]
                                       (COND
                                          [(NOT (IMAGEOBJP (fetch (PropListItem Value) of X)))
                                           (COND
                                              ((fetch (PropListItem AllowEditFlg) of X)
                                               (LIST (LIST 'MB.TEXT (CONCAT (CHARACTER 9)
                                                                            "  ["
                                                                            (MKSTRING (fetch (PropListItem
                                                                                                 Value)
                                                                                             of X))
                                                                            "]"
                                                                            (CHARACTER 13))
                                                           font)))
                                              (T (LIST (LIST 'MB.TEXT (CHARACTER 9)
                                                             font)
                                                       (LIST 'MB.INSERT (MKSTRING (fetch (PropListItem Value)
                                                                                         of X)))
                                                       (LIST 'MB.TEXT (CHARACTER 13)
                                                             font]
                                          (T (LIST (LIST 'MB.TEXT (CHARACTER 9)
                                                         font)
                                                   (LIST 'MB.INSERT)
                                                   (LIST 'MB.TEXT (CHARACTER 13)
                                                         font]
      (SETQ textObj (TEXTOBJ menuStream))                        ; Go back and insert the ImageObjects into their value fields.
      (SETQ CH# 0)
      [for prop in propList when (OR (IMAGEOBJP (fetch (PropListItem Value) of prop))
                                     (NULL (fetch (PropListItem AllowEditFlg) of prop)))
         do (MBUTTON.FIND.NEXT.FIELD textObj (SETQ CH# (ADD1 CH#)))
            (SETQ CH# (fetch CH# of (fetch SCRATCHSEL of textObj)))
            (COND
               ((IMAGEOBJP (CADR prop))
                (TEDIT.INSERT.OBJECT (fetch (PropListItem Value) of prop)
                      menuStream CH#]
      (SETQ CH# 0)
      (for prop in propList do (SETQ button (MBUTTON.FIND.NEXT.BUTTON textObj (ADD1 CH#)))
                               (SETQ CH# (CDR button))           ; If the buttonProtect flag is on, protect the button
                               (AND (fetch (PropListItem AllowSelectFlg) of prop)
                                    (IMAGEOBJPROP (CAR button)
                                          'EditPropListNoDelete T)))
                                                                 ; Set up window and window title
      [SETQ editW (COND
                     (window window)
                     (T (CREATEW (GETREGION)
                                 "Edit Property List"]
;; Point to the proplist being edited so we can update it when this menu is closed.  (See NC.CloseEditPropListWindow)
      (WINDOWPROP editW 'PROPERTYLIST.BEING.EDITED propList)
      (WINDOWPROP editW 'PROPERTYLIST.PROPNAMES (for Item in propList collect (CAR Item)))
                                                                 ; Set the right margin to very-far-away.  Prevents stuff from
                                                                 ; wrapping around
      (TEDIT.PARALOOKS menuStream '(RIGHTMARGIN 1000 TABS (50 (80 . LEFT)))
             1
             (GETFILEINFO menuStream 'LENGTH))                   ; Set the first tab so the fields will line up correctly
                                                                 ; Set selection to the top -- make it look pretty
      (replace (SELECTION SET) of (fetch (TEXTOBJ SEL) of textObj) with NIL)
;; Kill any existing tedit process.  Notice we have to do WINDOWPROP because calling TEXTSTREAM breaks if window doesn't have one.
      (LET [(OldTextStream (WINDOWPROP editW 'TEXTSTREAM]
           (if (TEXTSTREAMP OldTextStream)
               then (TEDIT.KILL OldTextStream)))
      (LET ((OLD-TTY-PROCESS (TTY.PROCESS)))
           ;; Start the TEdit with the TTY so TEDIT-PROCESS-P will work
           [SETQ editProcess (TEDIT menuStream editW NIL (if (OR showOnlyFlg showLinksFlg)
                                                             then '(MENU ,NC.ShowPropListMenu NOTSPLITTABLE
                                                                         T)
                                                           else '(MENU ,NC.EditPropListMenu NOTSPLITTABLE T]
           (until (TEDIT-PROCESS-P editProcess) do (BLOCK))
           ;; Give Back the TTY if necessary
           (if (OR showOnlyFlg showLinksFlg)
               then (TTY.PROCESS OLD-TTY-PROCESS)))
      (if showOnlyFlg
          then (NC.MakeTEditReadOnly editW))
```

```
          (WINDOWADDPROP editW 'CLOSEFN (FUNCTION NC.ClosePropListEditor)
                 T])
```

## (NC.ExtractPropList

```
  [LAMBDA (editW)                                                    (* rht%: "16-Jan-87 12:29")
                                                                     (* Exctract the prop list from the TextStream in prop list editor
                                                                     window editW)

          (* * rht 1/16/87%: Now makes sure menuStream isn't empty before searching for items.)

    (LET [(menuStream (TEXTSTREAM editW))
          button propName (propValue T)
          EOL
          (CH# 1)
          (propList (WINDOWPROP editW 'PROPERTYLIST.BEING.EDITED]
                                                                     (* Move through each field/value pair in menu.
                                                                     Update each PROP in the list.)

          (* For each property in the list, move to the next field, and drop its value into the field slot.)
                                                                     (* for each button in the menu do)
         (AND (GREATERP (TEDIT.NCHARS menuStream)
                        1)
              (while (SETQ button (MBUTTON.FIND.NEXT.BUTTON (TEXTOBJ menuStream)
                                                            CH#))
                 collect                                             (* Convert the name of the button into the property name)

          (* Set up the looks of the fields so that MBUTTON.NEXT.FIELD will work correctly --
          Real kludge to get around limitations in TEditMenu stuff)

                        (TEDIT.LOOKS menuStream '(PROTECTED ON SELECTPOINT ON)
                               (IPLUS 4 (CDR button))
                               1)
                        [COND
                           ([SETQ EOL (MBUTTON.FIND.NEXT.BUTTON (TEXTOBJ menuStream)
                                                                 (IPLUS 4 (CDR button]
                            (SETQ EOL (IDIFFERENCE (CDR EOL)
                                                   2)))
                           (T (SETQ EOL (SUB1 (TEDIT.FIND menuStream (CHARACTER 13)
                                                          (CDR button]
                        [TEDIT.LOOKS menuStream '(PROTECTED OFF)
                               (IPLUS 4 (CDR button))
                               (IDIFFERENCE EOL (IPLUS 4 (CDR button]
                                                                     (* Get the name of the property from the butrton)
                        (SETQ propName (IMAGEOBJPROP (CAR button)
                                                     'MBTEXT))       (* Now get its value by reading the next field)
                        (SETQ propValue (MBUTTON.NEXT.FIELD.AS.TEXT.OR.IMAGEOBJ menuStream (CDR button)))
                        [COND
                           [(NULL propValue)
                            (SETQ propValue (fetch (PropListItem Value) of (FASSOC propName propList]
                           ((IMAGEOBJP propValue))
                           ((NC.StringIsListP propValue 0)           (* This string should really be interpreted as a LIST)
                            (SETQ propValue (READ propValue)))
                           (T                                        (* This really is a string)
                              (SETQ propValue (MKATOM propValue]      (* Keep track of where that property was in the menu stream)
                        (SETQ CH# (ADD1 (CDR button)))
                        (LIST propName propValue])
```

## (NC.OpenPropListEditor

```
  [LAMBDA (WindowOrTextStream PropList Title ShowOnlyFlg MakeImageObjFlg ShowLinksFlg)
                                                                     ; Edited  3-Dec-87 19:00 by rht:

          (* Open a property list editor above the card specified by TextStream
          (\, which) is either a TextStream or a Window)

          (* * rht 8/11/86%: Now takes ShowLinksFlg arg and passes to NC.PropListEditorOpenP so that we don't bail out if there's an
          open proplist not of our type.)

          (* * rht 11/13/86%: Now hangs Card object off prop list editor's window so that we can get to the card from the proplist editor
          at close time.)

          (* * rht 12/11/86%: Now only breaks out list if we're under ShowLinks.)

          (* * rht 9/30/87%: Now allows reuse of ShowLinks editor window.)

          (* * rg |11/23/87| passes ShowLinksFlg to NC.EditPropList)

    (PROG ((Window (OR (WINDOWP WindowOrTextStream)
                       (NC.TEditWindow WindowOrTextStream)))
           (SystemProperties '(ID Updates ItemDate LinksDate PropsDate TitleDate))
           EditWindow CardUID Card)

          (* Make sure there is no prop list editor already there. Okay, however, to reuse ShowLinks editor.)

          (if (AND (SETQ EditWindow (NC.PropListEditorOpenP Window ShowLinksFlg))
                   (NOT ShowLinksFlg))
```

```
        then (RETURN))
    [SETQ CardUID (fetch (Card UID) of (SETQ Card (NC.CoerceToCard Window]
```

(* FOR each prop/value pair with LISTP value. Make a series of individual prop/value pairs corresponding to the elements of the LISTP)

```
    [SETQ PropList
     (for Item on PropList by (CDDR Item)
        join (COND
                ((AND (LISTP (CADR Item))
                      (type? Link (CAADR Item)))
                 (for Element in (CADR Item)
                    collect (create PropListItem
                                    PropertyName _ (CAR Item)
                                    Value _ Element
                                    OriginalListFlg _ T
                                    AllowEditFlg _ NIL
                                    AllowSelectFlg _ NIL
                                    ButtonFn _ NIL)))
                (T (LIST (create PropListItem
                                 PropertyName _ (CAR Item)
                                 Value _ (CADR Item)
                                 OriginalListFlg _ NIL
                                 AllowEditFlg _ NIL
                                 AllowSelectFlg _ NIL
                                 ButtonFn _ NIL]
```

(* If specified, translate all NOTECARDLINK values into Image Objects for display.)

```
    [AND MakeImageObjFlg (for Item in PropList bind LinkIcon when (type? Link (fetch (PropListItem Value)
                                                                                 of Item))
                            do (replace (PropListItem Value) of Item with (SETQ LinkIcon
                                                                            (NC.MakeLinkIcon
                                                                             (fetch (PropListItem Value)
                                                                                of Item)))
                                                                          (IMAGEOBJPROP LinkIcon
                                                                                 'InsidePropListEditor T]
```

(* Indicate which properties can be edited by user.)

```
    [for Item in PropList do (COND
                                ((FMEMB (fetch (PropListItem PropertyName) of Item)
                                        SystemProperties)
                                 (replace (PropListItem AllowEditFlg) of Item with T)
                                 (replace (PropListItem AllowSelectFlg) of Item with T))
                                (ShowOnlyFlg (replace (PropListItem AllowEditFlg) of Item with NIL)
                                             (replace (PropListItem AllowSelectFlg) of Item with T))
                                (T (replace (PropListItem AllowEditFlg) of Item with NIL)
                                   (replace (PropListItem AllowSelectFlg) of Item with NIL]
```

(* Call the prop list editor)

```
    (OR (WINDOWP EditWindow)
        (ATTACHWINDOW (SETQ EditWindow (CREATEW (CREATEREGION 1000 2000 100 100)
                                                (OR Title "Edit Property List")
                                                NIL T))
                      Window
                      'TOP
                      'JUSTIFY
                      'LOCALCLOSE))
    (WINDOWADDPROP EditWindow 'CLOSEFN (FUNCTION FREEATTACHEDWINDOW)
           T)
    (WINDOWPROP EditWindow 'PropListEditor (COND
                                                (ShowOnlyFlg 'ShowOnly)
                                                (T T)))
    (WINDOWPROP EditWindow 'SavedCardObject Card)
    (NC.EditPropList PropList EditWindow ShowOnlyFlg ShowLinksFlg)
    (RETURN EditWindow])
```

## (NC.ProcessEditedPropList

```
  [LAMBDA (EditedPropList OldPropList Card)
```

(* rht%: "16-Jan-87 13:28")

(* Take an edited prop list and set the prop list of the card as required.)

(* * rht 2/1/85%: Removed call to NC.MarkCardDirty.)

(* * fgh |11/13/85| Updated to handle Card object.)

(* * rht 11/13/86%: No longer "spreads" out prop values.)

(* * rht 1/16/87%: Revised wholesale to fix bugs with prop ordering and prop deletion.)

```
    (DECLARE (GLOBALVARS NC.SystemCardPropNames))
    (LET ((CurPropList (NC.FetchPropList Card)))
         [NC.SetPropList Card (NCONC (for PropName in NC.SystemCardPropNames join (LIST PropName (LISTGET
                                                                                                  CurPropList
                                                                                                  PropName)))
                                     (for Item in EditedPropList
                                        join (LET* [(Prop (fetch (PropListItem PropertyName) of Item))
                                                    (NewValue (if (ZEROP (NCHARS (CADR Item)))
                                                                  then NIL
                                                                else (CADR Item]
```

```
                                                (LIST Prop (if (fetch (PropListItem OriginalListFlg)
                                                                  of (FASSOC Prop OldPropList))
                                                               then (LIST NewValue)
                                                             else NewValue]
        (NC.SetPropListDirtyFlg Card T])
```

## (NC.PropListEditorOpenP

```
  [LAMBDA (Window ShowLinksFlg)                                    (* rht%: "30-Sep-87 12:10")
                                                                   (* Is there a prop list editor opened on this window?)

          (* * rht 8/11/86%: Now doesn't return NIL unless there's an open proplist editor of our type.)

          (* * rht 9/30/87%: Now returns the attached window if successful.)

    (for AttachedWindow in (ALLATTACHEDWINDOWS Window) thereis (AND (WINDOWPROP AttachedWindow 'PropListEditor)
                                                                    [LET [(ShowLinks (WINDOWPROP AttachedWindow
                                                                                                 'ShowLinks]
                                                                   (* Check that ShowLinksFlg iff ShowLinks.)
                                                                         (OR (AND ShowLinksFlg ShowLinks)
                                                                             (AND (NULL ShowLinksFlg)
                                                                                  (NULL ShowLinks]
                                                                    AttachedWindow])
```

## (NC.SelectProperty

```
  [LAMBDA (Window)                                                 (* fgh%: "20-Apr-84 21:44")
                                                                   (* Select one of the properties in prop list editor window Window)
    (PROG ((ClippingRegion (DSPCLIPPINGREGION NIL Window)))
          (WINDOWPROP Window 'SelectedObject NIL)
          [until (WINDOWPROP Window 'SelectedObject) do (until (AND (MOUSESTATE LEFT)
                                                                    (EQ (WHICHW LASTMOUSEX LASTMOUSEY)
                                                                        Window)))
                                                        (COND
                                                           ((INSIDEP ClippingRegion (LASTMOUSEX Window)
                                                                     (LASTMOUSEY Window))
                                                            (APPLY* (WINDOWPROP Window 'BUTTONEVENTFN)
                                                                    Window]
          (AND (EQ TEDIT.SELPENDING (WINDOWPROP Window 'TEXTOBJ))
               (SETQ TEDIT.SELPENDING))
          (RETURN (WINDOWPROP Window 'SelectedObject NIL])
```

## (NC.ShowLinks

```
  [LAMBDA (CardIdentifier)                                         (* Randy.Gobbel "23-Nov-87 14:11")

          (* Open an inspector for the links for note card specified by TextStream above the window for the note card.)

          (* * fgh |11/13/85| Updated to handle Card object.)

          (* * fgh |5/2/86| Included calls to NC.InsureLinkDisplayMode to handle litatom link display modes.
          Added InsdiePropListEditor and Reverse indicators to UserData field of Links in show links editor.
          Also added ShowLinks property onto the editor window so other functions can detect that a window is a show links window.)

          (* * rht 8/11/86%: Now passes non-nil ShowLinksFlg to NC.OpenPropListEditor.)

          (* * rg |4/6/87| removed NC.ProtectedCardOperation wrapper.)

          (* * pmi 6/19/87%: Now places each link's UID on the corresponding link created for the inspector.
          Checks for the validity of each link's source and destination cards and deletes links that are bad
          (half-links.))

          (* * rht 9/30/87%: Now takes arbitrary CardIdentifier as arg.)

          (* * rg |11/23/87| now only opens prop list editor read-only if card is read-only)

    (LET ((Card (NC.CoerceToCard CardIdentifier))
          Links EditWindow)
         [SETQ Links (NCONC [for Link in (NC.FetchToLinks Card)
                               when (if (NC.ValidCardP (fetch (Link DestinationCard) of Link))
                                        then                       (* The link is good, include it in the show links window.)
                                             T
                                      else

          (* The link is bad, delete it and don't include it in the show links window.)

                                           (NC.DeleteLink Link)
                                           NIL)
                               join (LIST (COND
                                            ((EQ (fetch (Link AnchorMode) of Link)
                                                 'GlobalGlobal)
                                             "Global TO")
                                            (T "TO"))
                                          (LIST (create Link using Link DisplayMode _
                                                                   (create LINKDISPLAYMODE
                                                                      copying (NC.InsureLinkDisplayMode
                                                                                 (fetch (Link DisplayMode)
```

```
                                                                    of Link))
                                                              SHOWTITLEFLG _ T SHOWLINKTYPEFLG _ T)
                                                        UserData _ '(InsidePropListEditor T)
                                                        UID _ (fetch (Link UID) of Link]
                                    (for Link in (NC.FetchFromLinks Card)
                                       when (if (NC.ValidCardP (fetch (Link SourceCard) of Link))
                                                then                          (* The link is good, include it in the show links window.)
                                                     T
                                            else

        (* The link is bad, delete it and don't include it in the show links window.)

                                                  (NC.DeleteLink Link)
                                                 NIL)
                                       join (LIST "FROM" (LIST (create Link using Link DisplayMode _
                                                                              (create LINKDISPLAYMODE
                                                                                 copying (NC.InsureLinkDisplayMode
                                                                                           (fetch (Link DisplayMode)
                                                                                              of Link))
                                                                                       SHOWTITLEFLG _ T SHOWLINKTYPEFLG
                                                                                       _ T)
                                                                              SourceCard _ (fetch (Link DestinationCard)
                                                                                              of Link)
                                                                              DestinationCard _ (fetch (Link SourceCard)
                                                                                                   of Link)
                                                                              UserData _ '(InsidePropListEditor T
                                                                                            Reversed T)
                                                                              UID _ (fetch (Link UID) of Link]
        (WINDOWPROP (SETQ EditWindow (NC.OpenPropListEditor (NC.FetchWindow Card)
                                             Links "List of Links" (NC.FetchUserDataProp Card 'ReadOnly)
                                             T T))
              'ShowLinks T)
        EditWindow])
```

## (NC.StringIsListP

```
  [LAMBDA (string parenCount)                                   ; Edited  3-Dec-87 19:00 by rht:

        (* T if string has a balanced number of (and)%. Var "parenCount" counts open parens.)

    (PROG (nextParen)
          [COND
             ((NULL string)
              (COND
                 ((ZEROP parenCount)
                  (RETURN T))
                 (T (RETURN NIL]
          (SETQ nextParen (STRPOSL '(%( %))
                                   string))
          (COND
             ((NULL nextParen)
              (RETURN NIL)))
          [COND
             ([EQ (CHARCODE "(")
                  (EVAL `(CHARCODE ,(SUBSTRING string nextParen nextParen]
              (RETURN (NC.StringIsListP (SUBSTRING string (ADD1 nextParen))
                                        (ADD1 parenCount]
          [COND
             ([EQ (CHARCODE ")")
                  (EVAL `(CHARCODE ,(SUBSTRING string nextParen nextParen]
              (RETURN (NC.StringIsListP (SUBSTRING string (ADD1 nextParen))
                                        (SUB1 parenCount]
          (BREAK])
```

## (NC.PutProp

```
  [LAMBDA (Card Prop Value)                                     (* rht%: "25-Mar-87 16:11")

        (* * Put a property value pair on the NoteCardsPropList property of ID.
        ID must be active.)

        (* * fgh |11/13/85| Updated to handle Card object.)

        (* * rht 3/25/87%: Now calls NC.SetPropListDirtyFlg to mark prop list as dirty.)

    (PROG ((PropList (NC.FetchPropList Card)))
          [COND
             (PropList (LISTPUT PropList Prop Value)
                       (NC.SetPropList Card PropList))
             (T (NC.SetPropList Card (LIST Prop Value]
          (NC.SetPropListDirtyFlg Card T])
```

## (NC.GetProp

```
  [LAMBDA (Card Prop)                                           (* fgh%: "13-Nov-85 20:42")

        (* * Get the value of a property on the NoteCardsPropList property of ID.
        ID must be active.)
```

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (LISTGET (NC.FetchPropList Card)
           Prop])
```

## (NC.RemProp

```
  [LAMBDA (Card Prop)                                    (* rht%: "26-Nov-85 21:14")
```

(* * Remove given property from Card's property list. Someday maybe this could be smarter and save space.)

```
    (NC.PutProp Card Prop NIL])
)
```

;;; Unknown ?????????

```
(DEFINEQ
```

## (NC.FetchBeingDeletedFlg

```
  [LAMBDA (Card)                                         (* fgh%: "13-Nov-85 19:46")
```

(* * fetch IDs being deleted flag from the cache)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (fetch (Card BeingDeletedFlg) of Card])
```

## (NC.SetBeingDeletedFlg

```
  [LAMBDA (Card Value)                                   (* fgh%: "13-Nov-85 19:46")
```

(* * Set the being deleted flag in the cache)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (replace (Card BeingDeletedFlg) of Card with Value])
```

## (NC.RemoveDELETEMEImageObjsFromCard

```
  [LAMBDA (Card PredicateFn)                             (* rht%: " 7-Oct-86 01:20")
```

(* * If card needs cleaning, find and remove any image objects satisfying PredicateFn from Card.
Currently only works for text cards.)

```
    (if (AND (NC.FetchUserDataProp Card 'NeedsCleaningFlg)
             (NCP.TextBasedP Card))
        then (LET ((WasActiveFlg (NCP.CardCachedP Card))
                   TextStream)
                  (if (NOT WasActiveFlg)
                      then (NCP.CacheCards Card))
                  (SETQ TextStream (NCP.CardSubstance Card))      (* Need to reverse list so that we delete the last icon first.)
                  (for IconPair in (REVERSE (NC.TEditCollectObjects TextStream PredicateFn))
                     do (TEDIT.DELETE TextStream (CADR IconPair)
                                      1))
                  (if (NOT WasActiveFlg)
                      then (NCP.UncacheCards Card)))
             (NC.SetUserDataProp Card 'NeedsCleaningFlg NIL])
```

## (NC.InsureIntegerDate

```
  [LAMBDA (Date)                                         (* rht%: " 5-Sep-86 12:15")
```

(* * Convert a date to IDATE format if necessary)

(* * rht 9/5/86%: Now returns 0 if IDATE returns nil.)

```
    (COND
       ((NULL Date)
        0)
       ((FIXP Date))
       ((STRINGP Date)
        (OR (IDATE Date)
            0))
       (T (NC.ReportError NIL "Unknown date format"])
)
```

;;; Place marker ImageObjects

```
(DEFINEQ
```

### (NC.PlaceMarkerCopyFn

```
  [LAMBDA (ImageObj)                                          (* fgh%: " 5-Mar-84 23:22")
    (NC.MakePlaceMarker (IMAGEOBJPROP ImageObj 'OBJECTDATUM])
```

### (NC.PlaceMarkerDisplayFn

```
  [LAMBDA (ImageObj Stream)                                   (* rht%: " 7-Dec-84 19:33")

          (* * rht 9/24/84%: Now works for press and interpress as well as screen.)

    (PROG ((Label (IMAGEOBJPROP ImageObj 'OBJECTDATUM))
           (Scale (DSPSCALE NIL Stream))
           (Font (FONTCREATE 'HELVETICA 12 'ITALIC NIL Stream)))
          (RELMOVETO (ITIMES Scale 3)
                 0 Stream)
          (DSPFONT (PROG1 (DSPFONT Font Stream)
                          (PRIN1 (U-CASE Label)
                                 Stream))
                 Stream])
```

### (NC.PlaceMarkerGetFn

```
  [LAMBDA (FileStream TextStream)                             (* fgh%: " 5-Mar-84 23:25")
    (NC.MakePlaceMarker (READ FileStream])
```

### (NC.PlaceMarkerImageBoxFn

```
  [LAMBDA (ImageObj Stream)                                   (* rht%: " 7-Dec-84 19:33")

          (* * rht 9/24/84%: Now scales the box dimensions so can go to press and interpress.)

    (PROG ((Font (FONTCREATE 'HELVETICA 12 'ITALIC NIL Stream))
           (Label (IMAGEOBJPROP ImageObj 'OBJECTDATUM))
           (Scale (DSPSCALE NIL Stream)))
          (RETURN (create IMAGEBOX
                          XSIZE _ (IPLUS (TIMES 6 Scale)
                                         (STRINGWIDTH (U-CASE Label)
                                                      Font))
                          YSIZE _ (IPLUS (TIMES 18 Scale)
                                         (FONTPROP Font 'HEIGHT))
                          YDESC _ (IPLUS (TIMES 3 Scale)
                                         (FONTPROP Font 'DESCENT))
                          XKERN _ 0])
```

### (NC.PlaceMarkerPutFn

```
  [LAMBDA (ImageObj FileStream)                               (* fgh%: "29-Feb-84 19:15")
    (PROG [(Label (IMAGEOBJPROP ImageObj 'OBJECTDATUM]
          (PRIN2 (MKSTRING Label)
                 FileStream])
```

### (NC.MakePlaceMarker

```
  [LAMBDA (Label)                                             (* fgh%: " 5-Mar-84 01:43")
    (IMAGEOBJCREATE Label (IMAGEFNSCREATE (FUNCTION NC.PlaceMarkerDisplayFn)
                                  (FUNCTION NC.PlaceMarkerImageBoxFn)
                                  (FUNCTION NC.PlaceMarkerPutFn)
                                  (FUNCTION NC.PlaceMarkerGetFn)
                                  (FUNCTION NC.PlaceMarkerCopyFn)
                                  (FUNCTION NILL)
                                  (FUNCTION NILL)
                                  (FUNCTION NILL)
                                  (FUNCTION NILL)
                                  (FUNCTION NILL])

)


;;; Functions for handling dates.

(DEFINEQ
```

### (NC.FetchTitleDate

```
  [LAMBDA (Card DontConvertFlg)                               (* fgh%: "13-Nov-85 19:47")

          (* * Fetch IDs title date from the cache and and convert to string format if necessary)

          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

          (* * fgh |11/13/85| Updated to handle Card object.)

    (LET ((Date (fetch (Card TitleDate) of Card)))
         (COND
            ((ZEROP Date)
             NIL)
```

```
            (DontConvertFlg Date)
            (T (GDATE Date])
```

## (NC.FetchItemDate

```
[LAMBDA (Card DontConvertFlg)                                   (* fgh%: "13-Nov-85 19:47")
```

(* * Fetch IDs item date from the cache and and convert to string format if necessary)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (LET ((Date (fetch (Card ItemDate) of Card)))
         (COND
            ((ZEROP Date)
             NIL)
            (DontConvertFlg Date)
            (T (GDATE Date])
```

## (NC.FetchPropListDate

```
[LAMBDA (Card DontConvertFlg)                                   (* fgh%: "13-Nov-85 19:48")
```

(* * Fetch IDs prop list date from the cache and and convert to string format if necessary)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (LET ((Date (fetch (Card PropListDate) of Card)))
         (COND
            ((ZEROP Date)
             NIL)
            (DontConvertFlg Date)
            (T (GDATE Date])
```

## (NC.FetchLinksDate

```
[LAMBDA (Card DontConvertFlg)                                   (* fgh%: "13-Nov-85 19:48")
```

(* * Fetch IDs links date from the cache and and convert to string format if necessary)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (LET ((Date (fetch (Card LinksDate) of Card)))
         (COND
            ((ZEROP Date)
             NIL)
            (DontConvertFlg Date)
            (T (GDATE Date])
```

## (NC.FetchLinkIconAttachedBitMap

```
[LAMBDA (Card ScaledHeightToMatch Scale CachedCardType)         ; Edited 10-Jan-89 17:20 by rtk
```

;; Return the default link icon attached bit map corresponding to Card

;; fgh 2/5/86 First created.

;; rht 5/10/86: Now takes special action if BitMapVal is a list. In that case, it should be an ordered prop list of heights and bitmaps. We take the ;; one closest in height to HeightToMatch.

;; rht 8/7/86: Now converts single bitmap to list of bitmaps of different heights if necessary. Also now takes Scale argument.

;; pmi 11/3/87: Now uses a default bitmap (?) if Card is NIL or its card type is undefined.

;; pmi 2/9/88: Somehow lost the check for defined card type mentioned in previous comment. It's there now.

;; pmi 9/28/88: Now checks for an AttachedBitMapFn on the card type, and applies it to the card to get the bitmap or list of heights and bitmaps.

;; bk 1/10/89: Added CachedCardType for cross file links

```
(DECLARE (GLOBALVARS NC.DefaultLinkIconAttachedBitMapHeights NC.UnknownLinkIconAttachedBitMaps))
(LET [BitMapVal AttachedBitMapFn (CardType (OR CachedCardType (NCP.CardType Card]
     (if (AND Card (NCP.ValidCardType CardType))
         then (if (SETQ AttachedBitMapFn (GETPROP CardType 'AttachedBitMapFn))
                  then (SETQ BitMapVal (APPLY* AttachedBitMapFn Card ScaledHeightToMatch Scale))
                else (SETQ BitMapVal (fetch (Card LinkIconAttachedBitMap) of Card)))
              [if (BITMAPP BitMapVal)
                  then (replace (NoteCardType LinkIconAttachedBitMap) of (NC.CardTypeRecord CardType)
                          with (SETQ BitMapVal (NC.MakeTypeIconBitMapSet BitMapVal
                                                    NC.DefaultLinkIconAttachedBitMapHeights]

              ;; Check if the Cached card type can be used.

       elseif [AND (NCP.ValidCardType CardType)
                   (NOT (GETPROP CardType 'AttachedBitMapFn))
                   (SETQ BitMapVal (fetch (NoteCardType LinkIconAttachedBitMap) of (NC.CardTypeRecord CardType]
         then [if (BITMAPP BitMapVal)
```

```
                    then (replace (NoteCardType LinkIconAttachedBitMap) of (NC.CardTypeRecord CardType)
                            with (SETQ BitMapVal (NC.MakeTypeIconBitMapSet BitMapVal
                                                    NC.DefaultLinkIconAttachedBitMapHeights]
          elseif (BITMAPP NC.UnknownLinkIconAttachedBitMaps)
            then (SETQ NC.UnknownLinkIconAttachedBitMaps (SETQ BitMapVal (NC.MakeTypeIconBitMapSet
                                                                    NC.UnknownLinkIconAttachedBitMaps

                                                                  NC.DefaultLinkIconAttachedBitMapHeights
                                                                    )))
          else ;; Use the unknown bitmaps
               (SETQ BitMapVal NC.UnknownLinkIconAttachedBitMaps))
        (if (LISTP BitMapVal)
            then (OR ScaledHeightToMatch (SETQ ScaledHeightToMatch 0))
                 (OR Scale (SETQ Scale 1))
                 (LET (BitMap)
                      [for X on BitMapVal by (CDDR X) do (LET [(ScaledHeight (TIMES Scale (CAR X]
                                                             (if (OR (NULL BitMap)
                                                                     (LEQ ScaledHeight ScaledHeightToMatch))
                                                               then (SETQ BitMap (CADR X))
                                                              elseif (GREATERP ScaledHeight ScaledHeightToMatch)
                                                               then (RETURN]
                      BitMap])
```

## (NC.FetchLinkDisplayMode

```
  [LAMBDA (Card)                                                    (* fgh%: " 5-Feb-86 13:40")
```

(* * Fetch the default link display mode corresponding to Card)

(* * fgh |2/5/86| First created.)

```
    (fetch (Card LinkDisplayMode) of Card])
```

## (NC.FetchDefaultWidth

```
  [LAMBDA (Card)                                                    (* fgh%: " 5-Feb-86 13:42")
```

(* * Fetch the default width corresponding to Card)

(* * fgh |2/5/86| First created.)

```
    (fetch (Card DefaultWidth) of Card])
```

## (NC.FetchDefaultHeight

```
  [LAMBDA (Card)                                                    (* fgh%: " 5-Feb-86 13:43")
```

(* * Fetch default height corresponding to Card)

(* * fgh |2/5/86| First created.)

```
    (fetch (Card DefaultHeight) of Card])
```

## (NC.FetchLinkAnchorModesSupported

```
  [LAMBDA (Card)                                                    (* fgh%: " 5-Feb-86 13:45")
```

(* * Fetch the link anchor modes supported of Card)

```
    (fetch (Card LinkAnchorModesSupported) of Card])

)

(DEFINEQ
```

## (NC.SetTitleDate

```
  [LAMBDA (Card Date)                                               (* fgh%: "13-Nov-85 19:49")
```

(* * Set the title date in the cahce for card ID)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (replace (Card TitleDate) of Card with (NC.InsureIntegerDate Date))
   Date])
```

## (NC.SetItemDate

```
  [LAMBDA (Card Date)                                               (* fgh%: "13-Nov-85 19:49")
```

(* * Set the item date in the cache for ID)

(* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)

(* * fgh |11/13/85| Updated to handle Card object.)

```
    (replace (Card ItemDate) of Card with (NC.InsureIntegerDate Date))
    Date])
```

## (NC.SetPropListDate

```
  [LAMBDA (Card Date)                                    (* fgh%: "13-Nov-85 19:49")
          (* * Set the prop list date in the cache for ID)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (replace (Card PropListDate) of Card with (NC.InsureIntegerDate Date))
    Date])
```

## (NC.SetLinksDate

```
  [LAMBDA (Card Date)                                    (* fgh%: "13-Nov-85 19:50")
          (* * Set the links date in the cache for card ID)
          (* * rht 11/10/85%: Updated to handle CardID scheme and new version of NC.FetchCardCache.)
          (* * fgh |11/13/85| Updated to handle Card object.)
    (replace (Card LinksDate) of Card with (NC.InsureIntegerDate Date))
    Date])

)

(PUTPROPS NCCARDS FILETYPE :FAKE-COMPILE-FILE)

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

## FUNCTION INDEX



## VARIABLE INDEX

**PROPERTY INDEX**