File created: 19-May-2021 17:53:36 **{DSK}<home>pi>il>notecards>system>NCCONVERTVERSION2TO3.;7**

previous date: 2-Nov-2020 16:42:46 {DSK}<home>pi>il>notecards>system>NCCONVERTVERSION2TO3.;6

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1985-1990, 1993-1994, 2020-2021 by Venue & Xerox Corporation.

(RPAQQ NCCONVERTVERSION2TO3COMS
       (

;;; The junk in here is for converting a 1.2 style file into 1.3 style.

        (FNS NC.ConvertNoteFileVersion2To3 NC.BuildVersion2HashArray NC.GetAndPutVersion2Card
             NC.ReadVersion2Title NC.ReadVersion2Links NC.ReadVersion2PropList NC.ReadVersion2MainCardData
             NC.ReadVersion2LinkLabels NC.ConvertVersion2LinkIcon NC.ReadVersion2CardPartHeader
             NC.Version3CardFromVersion2ID)
        (FNS NC.ConvertLinkFormat)

;;; Functions to convert version 2 browsers to version 3.0 This is just the MergeNoteFiles browser code modified.

        (FNS NC.ConvertVersion2BrowserCard NC.ConvertVersion2GraphNodeID)

;;; Functions and vars basically stolen and renamed from the old 1.2 source.

        (GLOBALVARS NC.Version2LinkLabelsIdentifier NC.Version2ItemIdentifier NC.Version2LinksIdentifier
              NC.Version2PropsIdentifier NC.Version2TitlesIdentifier NC.DateStringLength)
        (INITVARS (NC.Version2LinkLabelsIdentifier '%###LABELS###)
               (NC.Version2ItemIdentifier '%###ITEM###)
               (NC.Version2LinksIdentifier '%###LINKS###)
               (NC.Version2PropsIdentifier '%###PROPS###)
               (NC.Version2TitlesIdentifier '%###TITLES###)
               (NC.DateStringLength 18))
        (FNS NC.IndexInFileFromID NC.ReadVersion2Ptrs NC.ReadVersion2Region NC.ReadVersion2Identifier
             NC.ReadVersion2Date)
        [DECLARE%: DONTEVAL@LOAD (P (NC.StoreAutoloadFnFile (FUNCTION NC.ScavengeDatabaseFile)
                                          'NCREPAIR
                                          'NOTECARDSDIRECTORIES]
        (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              NCCONVERTVERSION2TO3)))

;;; The junk in here is for converting a 1.2 style file into 1.3 style.

(DEFINEQ

(NC.ConvertNoteFileVersion2To3
  [LAMBDA (StreamOrFileName InterestedWindow)                    (* rht%: " 6-Nov-86 12:47")

          (* * Convert this notefile to version 3 outputting the version 3 notefile to new version with same filename.)

          (* * fgh |12/17/85| fixed call to CreateDatabaseFile to lop the version off of the file name.)

          (* * rht 12/18/85%: Now clears Version2HashArray in case uncollectable garbage is left around.)

          (* * rht 5/7/86%: No longer just passes IndexSizeInEntries on to NC.CreateDatabaseFile.
          Uses NextIndexNum and NC.DefaultIndexSizeInEntries to compute a reasonable hasharray size
          (hopefully smaller than 10240, the largest allowable.))

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

          (* * rht 11/6/86%: Now autoload apply's NC.ScavengeDatabaseFile in case NCREPAIR not loaded.)

    (DECLARE (GLOBALVARS NC.DefaultIndexSizeInEntries))
    (LET ((OperationMsg "Converting file to version 3.")
          CardTotal FileName FromStream FromNoteFile ToNoteFile Version2HashArray Version NextIndexNum
          IndexSizeInEntries NextLinkNum CheckptPtr ToFileName)
         [IF (STREAMP StreamOrFileName)
             THEN (SETQ FromStream StreamOrFileName)
                  (SETQ FileName (FULLNAME StreamOrFileName))
           ELSE (SETQ FileName StreamOrFileName)
                (SETQ FromStream (OPENSTREAM FileName 'BOTH]
         (SETFILEPTR FromStream 7)
         (IF (EQ (SETQ Version (NC.ReadPtr FromStream 1))
                 2)
             THEN                                                (* FromStream is indeed version 2.0)
                                                                 (* Get all the header info off the 1.2 stream.)
                  (SETFILEPTR FromStream 0)
                  (SETQ NextIndexNum (NC.ReadPtr FromStream 2))
```

```
                (SETQ CardTotal (SUB1 NextIndexNum))
                (SETQ IndexSizeInEntries (NC.ReadPtr FromStream 2))
                (SETQ NextLinkNum (NC.ReadPtr FromStream 3))    (* Skip version number.)
                (NC.ReadPtr FromStream 1)
                (SETQ CheckptPtr (NC.ReadPtr FromStream 3))
```

(* * Create a dummy notefile object for the 1.2 source notefile.)

```
                (SETQ FromNoteFile (create NoteFile
                                           UID _ (NC.MakeUID)
                                           Stream _ FromStream
                                           FullFileName _ (FULLNAME FileName)
                                           NextIndexNum _ NextIndexNum
                                           Version _ Version
                                           NextLinkNum _ NextLinkNum
                                           CheckptPtr _ CheckptPtr))
                (SETQ ToNoteFile (NC.CreateDatabaseFile (PACKFILENAME 'VERSION NIL 'BODY FileName)
                                         (MAX NC.DefaultIndexSizeInEntries (TIMES 2 NextIndexNum))
                                         OperationMsg T NIL NIL InterestedWindow))
                (SETQ ToFileName (fetch (NoteFile FullFileName) of ToNoteFile))
                (if (AND (type? NoteFile ToNoteFile)
                         (NC.OpenDatabaseFile ToNoteFile NIL T T NIL NIL NIL NIL T NIL InterestedWindow))
                    then
```

(* * Fill in NewNoteFile's hash array with new UIDs. Return a hash array mapping version 2 style ID atoms to version 3 style Cards.)

```
                        (NC.PrintMsg InterestedWindow T "Reading index of version 2 notefile ...")
                        (SETQ Version2HashArray (NC.BuildVersion2HashArray FromNoteFile ToNoteFile CardTotal))
```

(* * For each card in the old notefile, fill in a version 3 card object for it and put the filled in card down to NewNoteFile.)

```
                        (NC.PrintMsg InterestedWindow T OperationMsg (CHARACTER 13)
                               "Processing card ID 1 out of " CardTotal)
                        (for IDNum from 1 to CardTotal eachtime (BLOCK)
                           do (if (ZEROP (REMAINDER IDNum 10))
                                  then (NC.PrintMsg InterestedWindow T OperationMsg (CHARACTER 13)
                                                    "Processing card ID " IDNum " out of " CardTotal))
                              (NC.GetAndPutVersion2Card FromStream (NC.IDFromNumber IDNum)
                                    Version2HashArray FromNoteFile ToNoteFile))
                                                                (* Clear hash array to be sure uncollectable garbage isn't left around.)
                        (CLRHASH Version2HashArray)
                        (NC.CheckpointDatabase ToNoteFile T)
                        (NC.ForceDatabaseClose ToNoteFile)
                        (CLOSEF FromStream)                     (* Rebuild To and From links.)
                        (NC.AutoloadApply* (FUNCTION NC.ScavengeDatabaseFile)
                              ToNoteFile NIL NIL NIL InterestedWindow)
                        (NC.PrintMsg InterestedWindow T "Done.")
                        ToFileName)
     else (NC.ReportError "NC.ConvertNoteFileVersion2To3" (CONCAT "Filename " FileName " is version "
                                                                  Version ".  Can only convert version 2
                                                                  notefiles."))
          NIL])
```

(NC.BuildVersion2HashArray

```
  [LAMBDA (FromNoteFile ToNoteFile IndexSize)                   (* rht%: "26-Nov-85 21:50")
```

(* * Fill in ToNoteFile's hash array with new UIDs. Return a hash array mapping 1.2 style ID atoms to 1.3 style Cards.)

```
    (LET ((FromStream (fetch (NoteFile Stream) of FromNoteFile))
          (Version2HashArray (HASHARRAY IndexSize)))
```

(* * First fill in the special cards.)

```
         (PUTHASH 'NC00001 (fetch (NoteFile TableOfContentsCard) of ToNoteFile)
                Version2HashArray)
         (PUTHASH 'NC00002 (fetch (NoteFile OrphansCard) of ToNoteFile)
                Version2HashArray)
         (PUTHASH 'NC00003 (fetch (NoteFile ToBeFiledCard) of ToNoteFile)
                Version2HashArray)
         (PUTHASH 'NC00004 (fetch (NoteFile LinkLabelsCard) of ToNoteFile)
                Version2HashArray)
```

(* * Now fill in the rest of the entries with new card objects.)

```
         (for IDNum from 5 to IndexSize bind ID Card do (SETQ ID (NC.IDFromNumber IDNum))
                                                        (SETFILEPTR FromStream (NC.IndexInFileFromID ID))
```

(* * Only create card objects for those IDs that were active on the version 2 notefile.)

```
                                                        (if (EQ (NC.ReadStatus FromStream)
                                                                'ACTIVE)
                                                            then (SETQ Card (NC.GetNewCard ToNoteFile))
                                                                 (PUTHASH ID Card Version2HashArray)))
       Version2HashArray])
```

(NC.GetAndPutVersion2Card

```
[LAMBDA (Stream ID Version2HashArray FromNoteFile ToNoteFile)  (* rht%: " 6-Nov-86 11:54")

        (* * If ID has active status, then get its card parts off Stream and fill in the info into the corresponding card object.
        Find this object using Version2HashArray.)

        (* * rht 12/18/85%: Now deactivates card after putting to file.)

        (* * fgh |5/20/86| Added extra Version2HashArray arg to call to NC.ReadVersion2Links to support GlobalLinkConversion.
        Changed order in which Links and PropList are processed because global link conversion might need to put a NoSource
        property on the prop list.)

        (* * rht 8/27/86%: Now sets status to ACTIVE before reading card parts so that user code calling progintface functions will
        work okay, i.e. that card will be NC.ValidCardP.)

        (* * rht 11/6/86%: Now writes down link labels card to notefile.)

   (LET ((Pointers (NC.ReadVersion2Ptrs ID Stream))
         Card)
        (SELECTQ (fetch (POINTERLIST STATUS) of Pointers)
            (ACTIVE                                          (* Recover card object from the hash array.)
                    (SETQ Card (GETHASH ID Version2HashArray))

        (* * Set status to be active.)

                    (NC.SetStatus Card 'ACTIVE)

        (* * Read main data and substance and fill in Card object.)

                    (SETFILEPTR Stream (fetch (POINTERLIST MAINPTR) of Pointers))
                    (NC.ReadVersion2MainCardData Stream ID Card Version2HashArray FromNoteFile ToNoteFile)

        (* * Read title and fill in Card object.)

                    (SETFILEPTR Stream (fetch (POINTERLIST TITLEPTR) of Pointers))
                    (NC.ReadVersion2Title Stream ID Card)

        (* * Read prop list and fill in Card object.)

                    (SETFILEPTR Stream (fetch (POINTERLIST PROPSPTR) of Pointers))
                    (NC.ReadVersion2PropList Stream ID Card)

        (* * Read global links and fill in Card object.)

                    (SETFILEPTR Stream (fetch (POINTERLIST LINKSPTR) of Pointers))
                    (NC.ReadVersion2Links Stream ID Card Version2HashArray)

        (* * If card is a browser, then first convert to version 3 format.)

                    (if (NC.IsSubTypeOfP (NC.FetchType Card)
                                'Browser)
                        then (NC.ConvertVersion2BrowserCard Card Version2HashArray))

        (* * Write down the card parts to the ToNoteFile.)

                    (NC.PutMainCardData Card NIL T)
                    (NC.PutLinks Card T)
                    (NC.PutTitle Card T)
                    (NC.PutPropList Card T))
            (SPECIAL                                         (* Deal specially with link labels.)
                     (SETQ Card (NC.Version3CardFromVersion2ID ID Version2HashArray))

        (* Go get the link labels from version 2 stream and write down to version 3 notefile.)

                     (SETFILEPTR Stream (fetch (POINTERLIST MAINPTR) of Pointers))
                     (NC.StoreLinkLabels ToNoteFile (NC.ReadVersion2LinkLabels Stream ID Card))

        (* * Write down the linklabels card to the ToNoteFile.)

                     (NC.PutMainCardData Card NIL T))
            NIL)
        (AND (NC.CardP Card)
             (NC.DeactivateCard Card T))
        Card])
```

(NC.ReadVersion2Title

```
[LAMBDA (Stream ID Card)                                     (* rht%: "24-Nov-85 23:24")

        (* * Stream should be positioned at the Title card part of ID. Get the title and fill in for Card.)

   (NC.SetTitleDate Card (NC.ReadVersion2CardPartHeader Stream ID NC.Version2TitlesIdentifier))
   (NC.SetTitle Card (READ Stream])
```

(NC.ReadVersion2Links

```
[LAMBDA (Stream ID Card Version2HashArray)                    (* fgh%: "21-May-86 00:08")

        (* * Stream should be positioned at the Links card part of ID. Get the global links and fill in for Card.
        Ignore the To and From links -
        scavenger will rebuild them.)

        (* * fgh |5/20/86| Added call to NC.ConvertLinkFormat to convert the format of the global links from NOTECARDLINK
        records to Link datatypes. Also added Version2HashArray arg to support this.)

    (NC.SetLinksDate Card (NC.ReadVersion2CardPartHeader Stream ID NC.Version2LinksIdentifier))
                                                                 (* Skip the from and to links. The scavenger will rebuild them
                                                                 later.)
    (READ Stream)
    (READ Stream)
    (NC.SetGlobalLinks Card (NC.ConvertLinkFormat (READ Stream)
                                  Version2HashArray])
```

(NC.ReadVersion2PropList

```
[LAMBDA (Stream ID Card)                                     (* rht%: "24-Nov-85 23:23")

        (* * Stream should be positioned at the PropList card part of ID.
        Get the prop list and fill in for Card.)

    (NC.SetPropListDate Card (NC.ReadVersion2CardPartHeader Stream ID NC.Version2PropsIdentifier))
    (NC.SetPropList Card (READ Stream])
```

(NC.ReadVersion2MainCardData

```
[LAMBDA (Stream ID Card Version2HashArray FromNoteFile ToNoteFile)
                                                             (* fgh%: "27-May-86 20:43")

        (* * Stream should be positioned at the main data card part of ID.
        Get the main data and fill in for Card.)

        (* * fgh |12/17/85| changed Apply of CollectReferencesFn to be done only if there is a CollectReferencesFn for the card type)

        (* * rht 5/7/86%: Now only does the horrible kludge of smashing absolute pointers in the 1.2 file if the TEdit is judged to be
        formatted. We check that by looking for the TEdit password at the end of the substance.)

        (* * fgh |5/27/86| Added INTERSECTION call during conversion of links due to problems with browsers which have same
        link icon represented twice in the browser.)

    (LET (CardType Region StartPtr EndPtr Length TEditBasedFlg StartFormatPtr CollectReferencesFn
                FormattedTEditP LinkIcons)
         (NC.SetItemDate Card (NC.ReadVersion2CardPartHeader Stream ID NC.Version2ItemIdentifier))

          (* * Read card type and region)

         (NC.SetType Card (SETQ CardType (READ Stream)))
         (SETQ TEditBasedFlg (NC.TEditBasedP CardType))
         (READC Stream)
         (NC.SetRegion Card (NC.ReadVersion2Region Stream))

          (* * Read the substance pointers, compute the length, then call the substance get fn)

         (SETQ StartPtr (NC.ReadPtr Stream 3))
         (SETQ EndPtr (NC.ReadPtr Stream 3))
         (SETQ Length (DIFFERENCE EndPtr StartPtr))

          (* * Figure out whether the substance is TEdit formatted. In that case we have to smush absolute pointers.)

         (SETQ FormattedTEditP (AND TEditBasedFlg (GREATERP Length 2)
                                    (SETFILEPTR Stream (DIFFERENCE EndPtr 2))
                                    (EQ (QUOTIENT (NC.ReadPtr Stream 2)
                                                  100)
                                        NC.TEditPasswordDividedBy100)))

          (* * A horrible kludge%: Change the infamous file absolute pointer in the text stream to be file relative for duration of the
          GetSubstance call.)

         (if FormattedTEditP
             then (SETFILEPTR Stream (DIFFERENCE EndPtr 8))
                  (SETQ StartFormatPtr (NC.ReadPtr Stream 4))
                  (SETFILEPTR Stream (DIFFERENCE EndPtr 8))
                  (NC.WritePtr Stream (DIFFERENCE StartFormatPtr StartPtr)
                         4))
         (SETFILEPTR Stream StartPtr)
         (NC.SetSubstance Card (NC.ApplyFn GetFn Card Length Stream -1))

          (* * Now put back the infamous file absolute pointer.)

         (if FormattedTEditP
             then (SETFILEPTR Stream (DIFFERENCE EndPtr 8))
                  (NC.WritePtr Stream StartFormatPtr 4)
                  (SETFILEPTR Stream EndPtr))
```

```
          (* * Now convert each link in the embedded link icons in the substance.)

          (if (fetch (Card CollectLinksFn) of Card)
              then (for LinkIcon in (INTERSECTION (SETQ LinkIcons (CAR (NC.ApplyFn CollectLinksFn Card NIL T)))
                                                  LinkIcons)
                      eachtime (BLOCK) do (NC.ConvertVersion2LinkIcon LinkIcon Card Version2HashArray])
```

(NC.ReadVersion2LinkLabels

```
  [LAMBDA (Stream ID Card)                                          (* rht%: "24-Nov-85 23:21")

          (* * Stream should be positioned at the links labels.)

    (LET (CardType Region StartPtr EndPtr)
         (NC.SetItemDate Card (NC.ReadVersion2CardPartHeader Stream ID NC.Version2LinkLabelsIdentifier))
         (READ Stream])
```

(NC.ConvertVersion2LinkIcon

```
  [LAMBDA (LinkIcon SourceCard Version2HashArray)                   (* rht%: "26-Nov-85 21:03")

          (* * Convert the link inside LinkIcon into a version 3 link. Use Version2HashArray to convert the Destination ID into a card
          object.)

    (LET ((Version2Link (fetch (IMAGEOBJ OBJECTDATUM) of LinkIcon)))
                                                                    (* Make sure that version 2 link is in TYPERECORD format.)
         (if (NEQ (CAR Version2Link)
                  'NOTECARDLINK)
             then (SETQ Version2Link (CONS 'NOTECARDLINK Version2Link)))
                                                                    (* Create and plug in a new link.)
         (replace (IMAGEOBJ OBJECTDATUM) of LinkIcon with (create Link
                                                                  UID _ (NC.MakeUID)
                                                                  SourceCard _ SourceCard
                                                                  DestinationCard _ (NC.Version3CardFromVersion2ID
                                                                                     (fetch (NOTECARDLINK
                                                                                                 DESTINATIONID)
                                                                                        of Version2Link)
                                                                                     Version2HashArray)
                                                                  AnchorMode _ (fetch (NOTECARDLINK ANCHORMODE)
                                                                                  of Version2Link)
                                                                  Label _ (fetch (NOTECARDLINK LINKLABEL)
                                                                             of Version2Link)
                                                                  DisplayMode _ (fetch (NOTECARDLINK DISPLAYMODE)
                                                                                   of Version2Link)))
                                                                    (* Change display mode to record format if it's currently atomic.)
         (NC.CheckDisplayModeFormat Version2Link])
```

(NC.ReadVersion2CardPartHeader

```
  [LAMBDA (Stream ID Identifier)                                    (* rht%: "24-Nov-85 23:52")

          (* * Read a card part header from a version 2 style notefile stream.)

    (LET (VersionNumber ActualID Date)
         [COND
            ((NOT (SETQ VersionNumber (NC.ReadVersion2Identifier Stream Identifier)))
             (NC.ReportError "NC.ReadVersion2CardPartHeader" (CONCAT ID " Error while reading NoteFile" " --
                                                                    incorrect identifier."]
         [COND
            ((GEQ VersionNumber 1)
             (SETQ Date (NC.ReadVersion2Date Stream]
         (SETQ ActualID (READ Stream))
         [COND
            ((NOT (EQ ActualID ID))
             (NC.ReportError "NC.ReadVersion2CardPartHeader" (CONCAT "ID mismatch while reading item.  Expected
                                                                    ID: " ID "   Found ID: " ActualID]
        Date])
```

(NC.Version3CardFromVersion2ID

```
  [LAMBDA (ID Version2HashArray)                                    (* rht%: "26-Nov-85 21:02")

          (* * Return the card object corresponding to given ID using given hash array.)

    (GETHASH ID Version2HashArray])

)

(DEFINEQ
```

(NC.ConvertLinkFormat

```
  [LAMBDA (ListOfLinks Version2HashArray)                           (* fgh%: "25-May-86 18:08")

          (* * Convert links from NOTECARDLINK record to Link datatype.
          Note the conversion of NoSource from a dangling link to a property list item.)
```

```
          (* * fgh |5/21/86| First created.)

    (bind NewLink ConvertedLinks PropList Card for Link in ListOfLinks
       do (OR (EQ (CAR Link)
                  'NOTECARDLINK)
              (SETQ Link (CONS 'NOTECARDLINK Link)))
          [if (NEQ (fetch (NOTECARDLINK DESTINATIONID) of Link)
                   'NC00000)
              then (SETQ NewLink (create Link
                                         UID _ (NC.MakeUID)
                                         SourceCard _ (GETHASH (fetch (NOTECARDLINK SOURCEID) of Link)
                                                               Version2HashArray)
                                         DestinationCard _ (GETHASH (fetch (NOTECARDLINK DESTINATIONID)
                                                                           of Link)
                                                                    Version2HashArray)
                                         AnchorMode _ (fetch (NOTECARDLINK ANCHORMODE) of Link)
                                         Label _ (fetch (NOTECARDLINK LINKLABEL) of Link)
                                         DisplayMode _ (fetch (NOTECARDLINK DISPLAYMODE) of Link)))
                   (NC.CheckDisplayModeFormat NewLink)
                   (SETQ ConvertedLinks (CONS NewLink ConvertedLinks))
            elseif (SETQ Card (GETHASH (fetch (NOTECARDLINK SOURCEID) of Link)
                                       Version2HashArray))
              then

          (* * This is the old form of NoSource. Convert to new form by putting on the prop list.)

                   [SETQ PropList (NC.FetchPropList (SETQ Card (GETHASH (fetch (NOTECARDLINK SOURCEID)
                                                                               of Link)
                                                                        Version2HashArray]
                   (NC.SetPropList Card (if PropList
                                            then (LISTPUT PropList 'NoSource T)
                                                 PropList
                                          else (LIST 'NoSource T]
       finally (RETURN ConvertedLinks])

)

;;; Functions to convert version 2 browsers to version 3.0 This is just the MergeNoteFiles browser code modified.

(DEFINEQ

(NC.ConvertVersion2BrowserCard
  [LAMBDA (Card Version2HashArray)                                    (* rht%: "13-May-87 23:40")

          (* * Fix the browser roots and graphnode IDs to be version 3 -
          this is just modified version of the code in NCMERGEFILES that processes browser cards.)

          (* * rht 4/5/86%: Now removes NODEID and DESTNODEID fields from LinkParams of TOLINKS of graphnodes.
          This effectively converts multiple links between pairs of nodes into single links.
          Thus, users will have to ReconnectNodesInBrowser in order to recover the multiple links they had in 1.2.)

          (* * fgh |5/28/86| Changed the way the TONODES and FROMNODES lists are converted.
          Now uses an ASSOC list to map from old to new NodeIds when convreting the NODEID field.
          This is then used to convert each NodeID in the TO and FTOM lists.
          This way virtual nodes we can be sure for virtual nodes that all references are EQ not just EQUAL as required by grapher.)

          (* * rht 5/13/87%: Now puts the information regarding link dashing onto the individual graph node id props.
          This gets saved when the browser is "put" to the 1.3 notefile.)

    (DECLARE (GLOBALVARS NC.DashingStyles))                           (* Get various stuff off browser's prop list.)
    (LET (MappingList LinksLegend)
         [NC.SetBrowserLinkLabels Card (CAR (NC.GetProp Card 'BrowserLinkLabels]
         (NC.RemProp Card 'BrowserLinkLabels)
         [NC.SetBrowserLinksLegend Card (SETQ LinksLegend (CAR (NC.GetProp Card 'BrowserLinksLegend]
         (NC.RemProp Card 'BrowserLinksLegend)
         (NC.SetBrowserDepth Card (NC.GetProp Card 'BrowserDepth))
         (NC.RemProp Card 'BrowserDepth)
         [NC.SetBrowserFormat Card (CAR (NC.GetProp Card 'BrowserFormat]
         (NC.RemProp Card 'BrowserFormat)
         [NC.SetSpecialBrowserSpecs Card (CAR (NC.GetProp Card 'SpecialBrowserSpecs]
         (NC.RemProp Card 'SpecialBrowserSpecs)                       (* Fix up browser roots)
         (NC.SetBrowserRoots Card (for BrowserRootID in (CAR (NC.GetProp Card 'BrowserRoots))
                                     collect (NC.Version3CardFromVersion2ID BrowserRootID Version2HashArray)))
                                                                      (* Throw away old browser roots.)
         (NC.RemProp Card 'BrowserRoots)

          (* * Fix up graph nodeids and store a mapping between old and new IDs on MappingList)

         (bind OldNodeID NewNodeID for GraphNode in (fetch (GRAPH GRAPHNODES) of (NC.FetchSubstance Card))
            do (replace (GRAPHNODE NODEID) of GraphNode with (SETQ NewNodeID (NC.ConvertVersion2GraphNodeID
                                                                                (SETQ OldNodeID (fetch (GRAPHNODE
                                                                                                        NODEID)
                                                                                                   of GraphNode))
                                                                                Card Version2HashArray)))
               (push MappingList (CONS OldNodeID NewNodeID)))
```

(* * Fix up the TONODES and FROMNODES for each GRAPHNODE using the new IDs created above and stored in MappingList.)

```
(for GraphNode in (fetch (GRAPH GRAPHNODES) of (NC.FetchSubstance Card))
   do [LET ((GraphNodeID (NC.CoerceToGraphNodeID GraphNode)))
           (replace (GRAPHNODE TONODES) of GraphNode
              with (for ToNode in (fetch (GRAPHNODE TONODES) of GraphNode)
                      collect (LET ((ToNodeDashingStyle (if (EQ (CAR ToNode)
                                                                LINKPARAMS)
                                                            then (LISTGET ToNode 'DASHING)
                                                          else NIL))
                                    ToNodeID)
                                   (SETQ ToNodeID (CDR (ASSOC (if (EQ (CAR ToNode)
                                                                      LINKPARAMS)
                                                                  then (CADR ToNode)
                                                                else ToNode)
                                                              MappingList)))
                                   [NC.GraphNodeIDPutProp GraphNodeID ToNodeID
                                          (for DashingStyle in NC.DashingStyles as DashingPair in
                                                                                                LinksLegend
                                             when (EQUAL DashingStyle ToNodeDashingStyle)
                                             do (RETURN DashingPair) finally (RETURN (CAR LinksLegend]
                                   (if (EQ (CAR ToNode)
                                           LINKPARAMS)
                                       then (RPLACA (CDR ToNode)
                                                    ToNodeID)
                                            (AND (LISTGET ToNode 'NODEID)
                                                 (LISTPUT ToNode 'NODEID NIL))
                                            (AND (LISTGET ToNode 'DESTNODEID)
                                                 (LISTPUT ToNode 'DESTNODEID NIL))
                                            ToNode
                                     else ToNodeID]
           (replace (GRAPHNODE FROMNODES) of GraphNode
              with (for NodeID in (fetch (GRAPHNODE FROMNODES) of GraphNode)
                      collect (COND
                                 [(EQ (CAR NodeID)
                                      LINKPARAMS)
                                  (RPLACA (CDR NodeID)
                                          (CDR (ASSOC (CADR NodeID)
                                                      MappingList]
                                 (T (CDR (ASSOC NodeID MappingList])
```

(NC.ConvertVersion2GraphNodeID

```
[LAMBDA (NodeID BrowserCard Version2HashArray)                    (* rht%: "26-Nov-85 21:32")
```

(* * Convert a graph node ID from version 2 to version 3.0 This is modification of NCMERGEFILES code to convert graph node ids.)

```
   (if (LISTP NodeID)
       then (LIST (NC.GetBrowserNodeID BrowserCard (NC.Version3CardFromVersion2ID (SUBATOM (CAR NodeID)
                                                                                           8)
                                                                                  Version2HashArray)))
     else (NC.GetBrowserNodeID BrowserCard (NC.Version3CardFromVersion2ID (SUBATOM NodeID 8)
                                                                          Version2HashArray])

)
```

;;; Functions and vars basically stolen and renamed from the old 1.2 source.

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.Version2LinkLabelsIdentifier NC.Version2ItemIdentifier NC.Version2LinksIdentifier
       NC.Version2PropsIdentifier NC.Version2TitlesIdentifier NC.DateStringLength)
)

(RPAQ? NC.Version2LinkLabelsIdentifier '%###LABELS###)

(RPAQ? NC.Version2ItemIdentifier '%###ITEM###)

(RPAQ? NC.Version2LinksIdentifier '%###LINKS###)

(RPAQ? NC.Version2PropsIdentifier '%###PROPS###)

(RPAQ? NC.Version2TitlesIdentifier '%###TITLES###)

(RPAQ? NC.DateStringLength 18)

(DEFINEQ
```

(NC.IndexInFileFromID

```
[LAMBDA (ID)                                                      (* rht%: "24-Nov-85 22:55")
```

(* * Returns the ptr into the current notefile corresponding to the index entry of ID.)

(* * rht 11/24/85%: Took out NC.IDP test.)

```
    (LSH (SUBATOM ID 3)
         4])
```

(NC.ReadVersion2Ptrs

```
  [LAMBDA (ID Stream)                                    (* rht%: "24-Nov-85 23:46")

          (* * Return a list of pointers from the index of card ID)

          (* * rht 11/24/85%: Stolen from old 1.2i NC.GetPtrs function. Took out WITH.MONITOR call.)

    (LET (Index Ptr LinksPtr TitlePtr PropsPtr Status PtrList EofPtr)
         (SETQ Index (NC.IndexInFileFromID ID))
         (SETFILEPTR Stream Index)
         (SETQ Status (NC.ReadStatus Stream))
         (SETQ Ptr (NC.ReadPtr Stream 3))
         (SETQ LinksPtr (NC.ReadPtr Stream 3))
         (SETQ TitlePtr (NC.ReadPtr Stream 3))
         (SETQ PropsPtr (NC.ReadPtr Stream 3))
         (SETQ PtrList
          (create POINTERLIST
                 STATUS _ Status
                 MAINPTR _ Ptr
                 LINKSPTR _ LinksPtr
                 TITLEPTR _ TitlePtr
                 PROPSPTR _ PropsPtr
                 INDEXPTR _ Index))
         (SETQ EofPtr (GETEOFPTR Stream))
         [AND (EQ Status 'ACTIVE)
              (for Ptr in (CDR PtrList) when (OR (IGREATERP Ptr EofPtr)
                                                 (MINUSP Ptr))
                 do (replace (POINTERLIST STATUS) of PtrList with 'BADPOINTER]
        PtrList])
```

(NC.ReadVersion2Region

```
  [LAMBDA (Stream)                                       (* rht%: "24-Nov-85 20:52")

          (* * Get a region from Stream.)

    (CREATEREGION (NC.ReadPtr Stream 2)
           (NC.ReadPtr Stream 2)
           (NC.ReadPtr Stream 2)
           (NC.ReadPtr Stream 2])
```

(NC.ReadVersion2Identifier

```
  [LAMBDA (Stream Identifier)                            (* rht%: "10-Dec-85 14:47")
                                                         (* Return T if next item on databaseStream is the identifier
                                                         specified by Identifier)

          (* * rht 2/4/85%: A horrible hack for the case of titles identifier. This is because a previous typo was causing NOBIND to get
          written for titles identifiers.)

          (* * rht 7/9/85%: Now checks for new data format. This is indicated by identifiers with the last two %#'s clipped off.
          Then comes the one-byte version number of the data format. If identifier is not clipped then it's old style and there is no
          version number. Return version number if there is one, 0 if old style, and NIL if can't match identifier.)

    (LET ((ThingRead (READ Stream NC.OrigReadTable))
          (ClippedIdentifier (SUBATOM Identifier 1 -3)))
         (COND
            ([OR (EQ ThingRead Identifier)
                 (AND (EQ Identifier NC.Version2TitlesIdentifier)
                      (EQ ThingRead 'NOBIND]
             0)
            [(AND (EQ ThingRead ClippedIdentifier)
                  (NUMBERP (PROGN                        (* First char is separator. Next is one-byte version number.)
                                  (READC Stream)
                                  (NC.GetPtr Stream 1]
            (T NIL])
```

(NC.ReadVersion2Date

```
  [LAMBDA (Stream)                                       (* rht%: "24-Nov-85 22:08")

          (* * Read a date string from Stream. All dates have the same length, so can use that as a check.
          I'm allowing null date since we may be compacting an old style
          (non-dated) notefile. Thus we won't give it a misleadingly new date.)

    (LET ((Date (READ Stream)))
         (if (OR (NULL Date)
                 (EQ (NCHARS Date)
                     NC.DateStringLength))
            then Date
          else (NC.ReportError "NC.ReadVersion2Date" (CONCAT Date " is not a proper date."))
               NIL])
```

```
)

(DECLARE%: DONTEVAL@LOAD

(NC.StoreAutoloadFnFile (FUNCTION NC.ScavengeDatabaseFile)
       'NCREPAIR
       'NOTECARDSDIRECTORIES)
)

(PUTPROPS NCCONVERTVERSION2TO3 FILETYPE :FAKE-COMPILE-FILE)

(PUTPROPS NCCONVERTVERSION2TO3 MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP" :BASE 10))

(PUTPROPS NCCONVERTVERSION2TO3 COPYRIGHT ("Venue & Xerox Corporation" 1985 1986 1987 1988 1989 1990 1993 1994
                                          2020 2021))
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX