<AFFIRM>REQUEST..38 30-Sep-81 15:36:09

affirmWhereIs	16
askIfNew	17
askToContinue	18
askToOwn	19
askWhereToGo	20
checkIfOwn	21
CheckPoint	53
checkUser	22
CleanupFile	1
CleanupFunction	2
CleanupRequest	55
CreatePatchFile	54
decrUserCount	23
deleteEntryFile	24
deleteEntryFunction	25
findDefinition	26
getComment	27
getEntry	28
getEntryFile	29
getEntryFunction	30
grantAccessFile	31
grantAccessFunction	32
InitializeDataFiles	3
lockFile	33
mapEntryFile	34
mapEntryFunction	35
nextSystem	36
NoteFiles	4
printChanged	37
printChanged1	38
putEntry	39
putEntryFile	40
putEntryFunction	41
ratomsRetto	42
readPreviousVersion	43
readReleaseFile	44
readRetfrom	45
redoFilePointers	46
ReleaseFile	5
ReleaseFunction	6
removeAccessFunction	47
requestCommentToFile	48
RequestFile	7
RequestFunction	8
systemBuild	49
systemCheck	50
unLockFile	51
unUpdateFunction	52
Update	9
UpdateFile	10
UpdateFunction	11
VerifyRequest	56
WhatChanged	12
WhoOwns	13
WhoOwnsFile	14
WhoOwnsFunction	15

```
(FILECREATED "26-Sep-81 17:04:50" <AFFIRM>REQUEST..38 47019

     changes to:  mapEntryFunction readReleaseFile CreatePatchFile

     previous date: "17-Jun-81 17:53:16" <AFFIRM>REQUEST..37)


(PRETTYCOMPRINT REQUESTCOMS)

(RPAQQ REQUESTCOMS [(FNS * REQUESTFNS)
        (* Normal Users should NOT call these functions.)
        (FNS * RequestSystemFNS)
        (RECORDS * REQUESTRECORDS)
        (VARS (AskFromFile NIL)
              FilesToUpdate SystemKey (PREVIOUSVERSION (QUOTE <AFFIRM>PREVIOUSVERSION))
              (COMMENT\READ\TABLE (COPYREADTABLE (QUOTE ORIG)))
              (AUTOUPDATEFLAG T)
              (HOMEMACHINE HOSTNAME)
              REQUESTDATAFILE REQUESTDATAFUNCTION RELEASEDATAFILE REQUESTCOMMENTS)
        (P (SETBRK (QUOTE (26))
                   1 COMMENT\READ\TABLE))
        (ADVISE * REQUESTADVISE)
        (MACROS Sprintout)
        (BLOCKS (REQUESTBLOCK CleanupFile CleanupFunction ReleaseFile ReleaseFunction RequestFile
                              RequestFunction Update UpdateFile UpdateFunction WhatChanged WhoOwns
                              WhoOwnsFile WhoOwnsFunction affirmWhereIs askIfNew askToContinue
                              askToOwn checkIfOwn CheckPoint checkUser CleanupRequest CreatePatchFile
                              decrUserCount deleteEntryFunction deleteEntryFile getEntry getEntryFile
                              getEntryFunction grantAccessFile grantAccessFunction lockFile
                              mapEntryFile mapEntryFunction nextSystem printChanged printChanged1
                              putEntry putEntryFile putEntryFunction readPreviousVersion
                              redoFilePointers removeAccessFunction systemBuild systemCheck
                              unLockFile unUpdateFunction VerifyRequest
                              (ENTRIES CleanupFile CleanupFunction ReleaseFile ReleaseFunction
                                       RequestFile RequestFunction Update UpdateFile UpdateFunction
                                       WhatChanged WhoOwns WhoOwnsFile WhoOwnsFunction CheckPoint
                                       CleanupRequest CreatePatchFile getEntryFile getEntryFunction
                                       mapEntryFile mapEntryFunction putEntryFile putEntryFunction
                                       redoFilePointers systemBuild VerifyRequest)
                              (GLOBALVARS AUTOUPDATEFLAG BADFILENAMES CHANGEDVARSLST FULLSYSTEMFILES
                                          HELPFLAG HELPTIME HOMEMACHINE HOSTNAME LASTWORD
                                          PREVIOUSVERSION PV\DIRECTORY PreviousVersion
                                          RELEASEDATAFILE REQUESTDATAFILE REQUESTDATAFUNCTION
                                          SYSTEMFILES SystemKey USERNAME value)
                              (NOLINKFNS . T])

(RPAQQ REQUESTFNS (CleanupFile CleanupFunction InitializeDataFiles NoteFiles ReleaseFile
                               ReleaseFunction RequestFile RequestFunction Update UpdateFile
                               UpdateFunction WhatChanged WhoOwns WhoOwnsFile WhoOwnsFunction
                               affirmWhereIs askIfNew askToContinue askToOwn askWhereToGo checkIfOwn
                               checkUser decrUserCount deleteEntryFile deleteEntryFunction
                               findDefinition getComment getEntry getEntryFile getEntryFunction
                               grantAccessFile grantAccessFunction lockFile mapEntryFile
                               mapEntryFunction nextSystem printChanged printChanged1 putEntry
                               putEntryFile putEntryFunction ratomsRetto readPreviousVersion
                               readReleaseFile readRetfrom redoFilePointers removeAccessFunction
                               requestCommentToFile systemBuild systemCheck unLockFile
                               unUpdateFunction))
(DEFINEQ
```

1

(CleanupFile

```
  [LAMBDA (File User)                                  (* R.Bates "28-FEB-79 15:47")
    (PROG (hashFile entry patchFile fromWhere)
          (File←(U-CASE File))
          (patchFile←(if (INFILEP File)
                         else (PRINTLINES "Can not find the file" File T)
                              (ERROR!)))
          (User←(checkUser User))
          (hashFile←(lockFile REQUESTDATAFILE))
          (entry←(checkIfOwn File User hashFile))
          (fromWhere←(if entry:PatchFile
                         else entry:File))
          (entry:State←'UPDATE)
          (entry:User←NIL)
          (entry:PatchFile←patchFile)
          (PUTHASHFILE File entry hashFile)
          (unLockFile hashFile)
          (nextSystem File 'FILE User patchFile fromWhere])
```

2

(CleanupFunction
```
  [LAMBDA (Functions User NoRelease)                    (* R.Bates " 8-Nov-79 15:25")
    (PROG (fileList patchFileList hashFile entryList fromWhereList)
          (if (NLISTP Functions)
              then Functions←(LIST Functions))
          (User←(checkUser User))
          (hashFile←(lockFile REQUESTDATAFUNCTION))
          (entryList←(for fun in Functions collect (checkIfOwn fun User hashFile)))
          (fileList←(for entry in entryList collect entry:File))
          (fromWhereList←(for entry in entryList collect (if entry:PatchFile
                                                            else entry:File)))
          [patchFileList←(for (entry error patchfile) in entryList as fun in Functions
                             collect (patchfile←(for file in (WHEREIS fun 'FNS T) collect file
                                                     when ~(MEMBER file SYSTEMFILES)
                                                         and file←(INFILEP (GETPROP file 'FILEDATES)
                                                                               :1::1)))
                                     (if patchfile=NIL
                                         then (printout T "Can not find where the new version of " fun
                                                        " is"
                                                        T)
                                              error←T
                                              NIL
                                       elseif patchfile::1~=NIL
                                         then (askWhereToGo fun patchfile NoRelease)
                                       else patchfile:1)
                             finally (if error
                                         then (askToContinue hashFile]
          (for entry in entryList as fun in Functions as patch in patchFileList
             collect (entry:State←'UPDATE)
                     (entry:User←NIL)
                     (entry:PatchFile←patch)
                     (PUTHASHFILE fun entry hashFile)
             when patch)
          (unLockFile hashFile)
          (for (file entry) in fileList as patch in patchFileList first hashFile←(lockFile
                                                                                  REQUESTDATAFILE)
             when patch and file~='NEW do (decrUserCount hashFile file User)
             finally (unLockFile hashFile))
          (if NoRelease=NIL
              then (for fun in Functions as patch in patchFileList as fromWhere in fromWhereList
                          do (nextSystem fun 'FUNCTION User patch fromWhere) when patch])
```

3

(InitializeDataFiles
```
  [LAMBDA NIL                                           (* R.Bates " 3-Apr-79 09:45")
    <(CLOSEF REQUESTDATAFILE←(HASHFILENAME (CREATEHASHFILE 'REQUESTDATAFILE 'EXPR)))
    (CLOSEF REQUESTDATAFUNCTION←(HASHFILENAME (CREATEHASHFILE 'REQUESTDATAFUNCTION 'EXPR)))
    (CLOSEF RELEASEDATAFILE←(OUTPUT (OUTFILE 'RELEASEDATAFILE)))
    >])
```

4

(NoteFiles
```
  [LAMBDA (FileList SystemLoad)                         (* R.Bates "18-Nov-80 20:38")
    (PROG (notMakeSys)
          (notMakeSys←(STRINGP MAKESYSDATE))
          (if (NLISTP FileList)
              then FileList← <FileList>)
          [if SystemLoad
              then (if notMakeSys
                       then (for file in FileList do (WHEREISNOTICE file)
                               first (if (FILENAMEFIELD WHEREIS.HASH 'DIRECTORY)='LISPUSERS
                                         then WHEREIS.HASH←(PACKFILENAME 'DIRECTORY PV\DIRECTORY
                                                                         'NAME 'TEMP 'EXTENSION 'HASH)
                                              (WHEREISNOTICE FileList:1 T)
                                              $$LST1←$$LST1::1)))
                   (for (file root hashFile) in FileList first (if notMakeSys
                                                                   then hashFile←(OPENHASHFILE
                                                                                  WHEREIS.HASH))
                      do (SYSFILES←(DREMOVE root←(FILENAMEFIELD file 'NAME)
                                            SYSFILES))
                         (FILELST←(ATTACH root FILELST))
                         (if notMakeSys
                             then (PUTPROP 'FILEDATES root <(GETHASHFILE file hashFile) ! file>]
        (if FULLSYSTEMFILES='NOBIND
            then FULLSYSTEMFILES←NIL)
        (FULLSYSTEMFILES← < !(for file in FileList collect <(FILENAMEFIELD file 'NAME) ! file>)
                            ! FULLSYSTEMFILES>)
        (SYSTEMFILES←(for file in FULLSYSTEMFILES collect file:1))
        (for file in SYSTEMFILES do (PUTPROP file 'DATABASE 'NO])
```

5

(ReleaseFile

```
  [LAMBDA (File User)                                   (* R.Bates "13-Dec-79 16:08")
    (PROG (entry hashFile)
          (File←(U-CASE File))
          (if User=NIL
              then User←USERNAME
            else User←(U-CASE User))
          (hashFile←(lockFile REQUESTDATAFILE))
          (entry←(checkIfOwn File User hashFile))
          (entry:User←NIL)
          (entry:State←NIL)
          (PUTHASHFILE File entry hashFile)
          (unLockFile hashFile])
```

6

(ReleaseFunction

```
  [LAMBDA (Functions User)                              (* R.Bates "14-FEB-79 16:36")
    (if (NLISTP Functions)
        then Functions← <Functions>)
    (for fun in Functions do (removeAccessFunction fun User])
```

7

(RequestFile

```
  [LAMBDA (File User)                                   (* R.Bates "23-FEB-79 15:54")
    (PROG (owner)
          (File←(U-CASE File))
          (if File ~MEMB SYSTEMFILES
              then (ERROR "File is not a member of the system files" File))
          (if owner←(WhoOwnsFile File)
              then (if (ATOM owner)
                       then (ERROR "File Already owned by" owner)
                     else (ERROR "File is partially owned by" owner::1)))
          (UpdateFile File)
          (grantAccessFile File User])
```

8

(RequestFunction

```
  [LAMBDA (Functions User EDITF SystemBuild)            (* R.Bates "12-Jul-79 15:13")
    (if HOSTNAME~=HOMEMACHINE
        then 'WrongMachine
      else (PROG (fileList updatedfns)
                 (if Functions=NIL
                     then Functions←LASTWORD)
                 (if (NLISTP Functions)
                     then Functions←(LIST Functions))
                 (if fileList←[for fun in Functions
                                  collect (if (affirmWhereIs fun)
                                              else (if EDITF
                                                       then (if (FNTYP fun)=NIL
                                                                then (printout T fun ,
                                                                               "is not a function."
                                                                               T
                                                     " RequestFunction call ignored, Please recall"
                                                                               T))
                                                                     (RETURN NIL)
                                                     elseif (askIfNew fun]=NIL
                     then (RETURN))
                 [Functions←(for (file owner error) in fileList as fun in Functions
                                collect (if owner←(WhoOwnsFile file T)
                                            then (PRINTLINES "The file" file "is owned by" owner
                                                             "which has the function"
                                                             fun "on it" T)
                                                 error←T
                                                 NIL
                                          elseif owner←(WhoOwnsFunction fun)
```

```
                                        then (if owner=USERNAME and EDITF
                                                then (RETURN))
                                              (PRINTLINES "The function" fun "is owned by" owner T)
                                              error←T
                                              NIL
                                       else fun)
                          finally (if error
                                        then (if EDITF
                                                then (askToContinue)
                                              else (ERROR!]
               (updatedfns←(for fun in Functions as file in fileList
                              collect (UpdateFunction fun file EDITF) when fun and ~SystemBuild))
               (for fun in Functions as file in fileList
                  do (if ~EDITF or file←(askToOwn fun file)
                          then (grantAccessFunction fun User file SystemBuild))
                  when fun)
               (RETURN updatedfns])
```

9

(Update

```
  [LAMBDA NIL                                              (* R.Bates " 1-MAR-79 15:48")
    (for entry in (mapEntryFile) do (UpdateFile entry:1) when entry:2:State='UPDATE)
    (for entry in (mapEntryFunction) do (UpdateFunction entry:1) when entry:2:State='UPDATE])
```

10

(UpdateFile

```
  [LAMBDA (File)                                           (* R.Bates "23-FEB-79 13:41")
    (PROG (entry hashFile)
          (File←(U-CASE File))
          (hashFile←(lockFile REQUESTDATAFILE))
          (entry←(GETHASHFILE File hashFile))
          (if entry:State='UPDATE
              then (PRINTLINES "loading from" entry:PatchFile T)
                   (LOAD entry:PatchFile))
          (unLockFile hashFile])
```

11

(UpdateFunction

```
  [LAMBDA (Function AffirmFile EDITF)                      (* R.Bates "26-Feb-80 11:20")
    (VIRGINFN Function T)
    (PROG ((AUTOUPDATEFLAG (if EDITF
                                then 'ASK
                              elseif (GETPROP Function 'EXPR) or (EXPRP Function)
                                then 'ASK
                              else AUTOUPDATEFLAG))
           entry hashFile keyList patchfile patchfileversion question temp topfile topfileversion)
          (hashFile←(lockFile REQUESTDATAFUNCTION))
          (entry←(GETHASHFILE Function hashFile))
          (unLockFile hashFile)
          (if AffirmFile MEMB BADFILENAMES
              then BADFILENAMES← <Function ! BADFILENAMES>)
          (if entry:State='UPDATE
              then (if AUTOUPDATEFLAG=T or AUTOUPDATEFLAG
                          and (AFFIRMUSER NIL NIL (CONCAT "Do you want to update " Function
                                                          " from file "
                                                          entry:PatchFile " ? "))='Y
                      then patchfile←(INFILEP entry:PatchFile)
                           topfile←(UNPACKFILENAME entry:PatchFile)
                           (if temp←('VERSION MEMB topfile)
                               then (temp:2←NIL))
                           topfile←(INFILEP (PACKFILENAME topfile))
                           (if patchfile=NIL and topfile
                               then (printout NIL .TAB0 0 "*** Can find only the top version of" ,
                                              entry:PatchFile T)
                                    patchfile←topfile
                             elseif patchfile=topfile
                               then (if patchfile=NIL
                                        then (printout NIL .TAB0 0 "*** Can't find any version of" ,
                                                       entry:PatchFile T)
                                             (RETURN NIL))
                             else topfileversion←(FILENAMEFIELD topfile 'VERSION)
                                  patchfileversion←(FILENAMEFIELD patchfile 'VERSION)
                                  (printout NIL .TAB0 0 "There are two versions of the file" ,
                                            topfile , ":" , topfileversion , "and" ,
                                            patchfileversion T)
                                  question←"Which version do you want? " keyList←
                                  <<patchfileversion "" > <topfileversion "" > '(Top "") >
                                  (if ~(EQUAL patchfileversion (AFFIRMUSER NIL NIL question keyList))
                                      then patchfile←topfile))
                          (printout NIL .TAB0 0 "loading from" , patchfile T)
                          temp←(LOADFNS Function patchfile):1
                          (if (LISTP temp)
                              then (printout NIL .TAB0 0 "*** Can't find" , Function , "on file" ,
                                             patchfile T)
                                   (RETURN NIL)
                            else (RETURN temp)))
          elseif entry:State~='OWN and AffirmFile MEMB SYSTEMFILES
            then (findDefinition Function])
```

12

(WhatChanged

```
  [LAMBDA (NumberOrFunction File Scribe)                    (* R.Bates " 4-Feb-80 11:08")
    (PROG (fileList openFiles)
          (if File=NIL
              then File←T
            else File←(OPENFILE File 'OUTPUT))
          (openFiles←(OPENP))
          (Sprintout File "@make(Manual)" T "@device(XGP)" T "@Style(Indent=0)" T "@font(Nonie12)" T
                     "@newpage"
                     T
                     (DATE)
                     T)
          [if NumberOrFunction=NIL
              then fileList←(readReleaseFile)
                   (printChanged fileList File Scribe)
            elseif (NUMBERP NumberOrFunction)
              then fileList←(readPreviousVersion)
                   (for x in fileList
                      do (if Scribe
                             then (printout File
                     "@PageHeading(left %"@Value(AffirmVersion)%",Right %"Page @Value(Page)%") "
                                            T "@string(AffirmVersion=%"Changes to version " x:1 "%")"
                                            T "@newpage" T "@SubHeading(@Value(AffirmVersion))" T)
                           else (printout File "Changes to version" , x:1 T))
                         (printChanged x:2 File Scribe x:1)
                      when (IGEQ x:1 NumberOrFunction))
            elseif (LITATOM NumberOrFunction)
              then fileList← <<'NextVersion (readReleaseFile) > !(readPreviousVersion) >
                   (for (x funList) in fileList do (funList←x:2)
                                                   (while funList←(SOME funList
                                                                        (FUNCTION [LAMBDA (Z)
                                                                            Z:1=NumberOrFunction]))
                                                      do (printout File "From version" , x:1 T)
                                                         (printChanged <funList:1> File Scribe)
                                                         (funList←funList::1]
          (printout File T)
          (for file in (LDIFFERENCE (OPENP)
                                    openFiles)
             do (CLOSEF? file))
          (if File~=T
              then (Sprintout File "@PageHeading(Right %"Page @Value(Page)%") " T)
                   (RETURN (CLOSEF File])
```

13

(WhoOwns

```
  [LAMBDA (FileOrFunction User)                             (* R.Bates "13-Dec-79 16:14")
    User←(U-CASE User)
    (if FileOrFunction=T
        then (if User=NIL
                 then User←USERNAME)
             < !(for entry in (mapEntryFunction) collect entry:1 when entry:2:State='OWN
                                                                   and entry:2:User=User)
               !(for entry in (mapEntryFile) collect entry:1 when entry:2:State='OWN
                                                               and entry:2:User=User)
             >
      else (PROG (userFile userFunction value)
                 (userFile←(WhoOwnsFile FileOrFunction))
                 (userFunction←(WhoOwnsFunction FileOrFunction))
                 (if userFile~=NIL and (NLISTP userFile)
                     then userFile← <userFile>)
                 (if userFunction~=NIL and (NLISTP userFunction)
                     then userFunction← <userFunction>)
```

```
                (value← < ! userFile ! userFunction>)
                (RETURN (INTERSECTION value value])
```

14

```
(WhoOwnsFile
  [LAMBDA (File Full)                                       (* R.Erickson "27-Aug-79 18:50")
    (RESETLST (PROG (entry hashFile value)
                    (File←(U-CASE File))
                    (hashFile←(lockFile REQUESTDATAFILE T))
                                                            (* undone by RESETSAVE)
                    (entry←(GETHASHFILE File hashFile))
                    (if entry:State='OWN
                        then value←entry:User
                      elseif ~Full and entry:State='PARTIAL
                        then value← <'PartiallyOwnedBy !(for u in entry:User collect u:1)
                                     >)
                    (RETURN value])
```

15

```
(WhoOwnsFunction
  [LAMBDA (Function)                                        (* R.Erickson "27-Aug-79 18:05")
    (RESETLST (PROG (entry hashFile value)
                    (hashFile←(lockFile REQUESTDATAFUNCTION T))
                                                            (* undone by RESETSAVE)
                    (entry←(GETHASHFILE Function hashFile))
                    (if entry:State='OWN
                        then value←entry:User)
                    (RETURN value])
```

16

```
(affirmWhereIs
  [LAMBDA (Function)                                        (* R.Bates "12-Jun-79 09:30")
    (PROG (entry hashFile file value)
          (hashFile←(lockFile REQUESTDATAFUNCTION))
          (entry←(GETHASHFILE Function hashFile))
          (value←(if entry:File
                   elseif file←(for i in (WHEREIS Function 'FNS T) thereis i MEMB SYSTEMFILES)
                     then entry←(create RequestFileRecord)
                          entry:File←file
                          (PUTHASHFILE Function entry hashFile)
                          file))
          (unLockFile hashFile)
          (RETURN value])
```

17

```
(askIfNew
  [LAMBDA (fun)                                             (* D.Thompson "28-Jan-80 10:42")
    (if (AFFIRMUSER NIL NIL (CONCAT "Is the function " fun " new to the system? "))='N
        then (ERROR!)
      else 'NEW])
```

18

```
(askToContinue
  [LAMBDA (HashFile)                                        (* D.Thompson "28-Jan-80 10:42")
    (if (AFFIRMUSER NIL NIL "Do you want to continue? ")='N
        then (if HashFile
                 then (unLockFile HashFile))
             (ERROR!])
```

19

```
(askToOwn
  [LAMBDA (Fun File)                                        (* D.Thompson " 5-Feb-80 15:31")
    (if (AFFIRMUSER NIL NIL (CONCAT "Do you want to own " Fun "? "))='N
        then NIL
      else (if File
               else (askIfNew Fun])
```

20

```
(askWhereToGo
  [LAMBDA (Function Files MasterPatchFile)                    (* D.Thompson " 6-Feb-80 09:06")
    (if MasterPatchFile~=NIL and MasterPatchFile~=T and (INFILEP MasterPatchFile)
      else (PROG (keyList optionList question)
                 (optionList←'(AUTOCOMPLETEFLG T))
                 (keyList←(for file in Files collect <file "" >))
                 (question←"Which file is the proper patch file? ")
                 (AFFIRMMAPRINT (CONCAT "The function " Function " appears on files")
                                Files T T)
                 (RETURN (AFFIRMUSER NIL NIL question keyList NIL NIL optionList])
```

21

```
(checkIfOwn
  [LAMBDA (Key User HashFile)                                 (* R.Bates "13-Dec-79 16:08")
    (PROG (entry)
          (User←(U-CASE User))
          (entry←(GETHASHFILE Key HashFile))
          (if entry:State~='OWN
              then (PRINTLINES Key "is not owned by anyone" T)
                   (unLockFile HashFile)
                   (ERROR!))
          (if entry:User~=User
              then (PRINTLINES "User should be" entry:User "and not" User T)
                   (unLockFile HashFile)
                   (ERROR!))
          (RETURN entry])
```

22

```
(checkUser
  [LAMBDA (User)                                              (* D.Thompson "28-Jan-80 10:44")
    (if User~=NIL and User~=USERNAME
        then (if (AFFIRMUSER NIL NIL (CONCAT "Do you really want to transfer ownership to " User "? ")
                             )='Y
                 then (U-CASE User)
               else (ERROR!))
      else USERNAME])
```

23

```
(decrUserCount
  [LAMBDA (HashFile AffirmFile User)                          (* R.Bates "14-FEB-79 12:29")
    (PROG (entry key)
          (entry←(GETHASHFILE AffirmFile HashFile))
          (if key←(SASSOC User entry:User)
              then (key::1←key::1-1)
            else (ERROR "Can find not user" User))
          (if key::1=0
              then (entry:User←(REMOVE key entry:User)))
          (if entry:User=NIL
              then (entry:State←NIL))
          (PUTHASHFILE AffirmFile entry HashFile])
```

24

```
(deleteEntryFile
  [LAMBDA (File)                                              (* R.Bates "18-Jun-79 14:30")
    File←(U-CASE File)
    (putEntry REQUESTDATAFILE File NIL])
```

25

```
(deleteEntryFunction
  [LAMBDA (Function)                                          (* R.Bates "18-Jun-79 14:34")
    (putEntry REQUESTDATAFUNCTION Function NIL])
```

26

```
(findDefinition
  [LAMBDA (Function)                                          (* R.Bates "21-Jul-80 11:02")
    (PROG (file inCoreDefn)
          (VIRGINFN Function T)
          (inCoreDefn←((GETPROP Function 'EXPR) or (EXPRP Function)))
```

```
        (if ~((GETPROP Function 'REQUEST) and inCoreDefn)
            then (if ~inCoreDefn or (AFFIRMUSER NIL NIL (CONCAT "Is the in-core definition of "
                                                                 Function " correct? "))='N
                     then file←(EDITLOADFNS? Function)
                          (if file
                              then (if (EXPRP Function)
                                       then (LOADFNS Function file)
                                     else (LOADFNS Function file 'PROP)
                                          (PUTPROP Function 'REQUEST T))
                          (RETURN file])
```

27

```
(getComment
  [LAMBDA (Key)                                           (* R.Bates " 8-Feb-80 20:29")
    (PROG (file DFile start end temp)
          (printout T .TAB0 0 "Please comment about " Key T)
          (file←(OPENFILE 'TEMP.EDITOR 'OUTPUT))
          (DFile←(OPENFILE TEMPRELEASECOMMENTS 'APPEND NIL NIL 'WAIT))
          (do temp←(CallEditor file 'File) repeatuntil temp~='ABORTED)
          (start←(GETFILEPTR DFile))
          (COPYBYTES file DFile)
          (end←(GETFILEPTR DFile))
          (CLOSEF? DFile)
          (CLOSEF? file)
          (DELFILE file)
          (RETURN (create RequestComment
                          DFile ← DFile
                          CStart ← start
                          CEnd ← end])
```

28

```
(getEntry
  [LAMBDA (File Key)                                      (* R.Bates "17-Dec-79 14:44")
    (PROG (hashFile entry)
          (hashFile←(lockFile File))
          (entry←(for item in (if (NLISTP Key)
                                  then <Key>
                                else Key)
                    collect (GETHASHFILE item hashFile)))
          (unLockFile hashFile)
          (RETURN (if (NLISTP Key)
                      then entry:1
                    else entry])
```

29

```
(getEntryFile
  [LAMBDA (File)                                          (* R.Bates "23-FEB-79 13:53")
    File←(U-CASE File)
    (getEntry REQUESTDATAFILE File])
```

30

```
(getEntryFunction
  [LAMBDA (Function)                                      (* R.Bates " 9-FEB-79 15:53")
    (getEntry REQUESTDATAFUNCTION Function])
```

31

```
(grantAccessFile
  [LAMBDA (File User)                                     (* R.Bates "23-FEB-79 11:33")
    (PROG (hashFile entry)
          (User←(checkUser User))
          (hashFile←(lockFile REQUESTDATAFILE))
          (entry←(GETHASHFILE File hashFile))
          (if entry=NIL
              then entry←(create RequestFileRecord))
          (entry:State←'OWN)
          (entry:User←User)
          (if entry:File=NIL
              then (entry:File←(INFILEP File)))
          (PUTHASHFILE File entry hashFile)
          (unLockFile hashFile])
```

32

```
(grantAccessFunction
  [LAMBDA (Function User AffirmFile SystemBuild)              (* R.Bates " 5-Jun-79 10:54")
    (PROG (hashFile entry key)
          (if SystemBuild
              then (if User=NIL
                       then (ERROR "User = NIL"))
            else User←(checkUser User))
          (hashFile←(lockFile REQUESTDATAFUNCTION))
          (systemCheck hashFile SystemBuild)
          (entry←(GETHASHFILE Function hashFile))
          (if entry=NIL
              then entry←(create RequestFileRecord))
          (entry:State←'OWN)
          (entry:User←User)
          (if AffirmFile='NEW
              then (entry:File←'NEW))
          (PUTHASHFILE Function entry hashFile)
          (unLockFile hashFile)
          (if AffirmFile~='NEW
              then hashFile←(lockFile REQUESTDATAFILE)
                   (if entry←(GETHASHFILE AffirmFile hashFile)=NIL
                       then entry←(create RequestFunctionRecord))
                   (if entry:File=NIL
                       then (entry:File←(SASSOC AffirmFile FULLSYSTEMFILES)::1))
                   (if key←(SASSOC User entry:User)
                       then (key::1←key::1+1)
                     else (entry:User← <<User ! 1> ! entry:User>))
                   entry:State←'PARTIAL
                   (PUTHASHFILE AffirmFile entry hashFile)
                   (unLockFile hashFile])
```

33

```
(lockFile
  [LAMBDA (File resetsave)                                    (* R.Erickson "27-Aug-79 18:05")
    (while $$VAL←[ERSETQ (PROG (HELPTIME HELPFLAG)
                               (RETURN (OPENHASHFILE File 'BOTH]=NIL
       do (PRINTLINES T "Having problems opening" File "..." T)
          (DISMISS 5000)
       finally (if resetsave
                   then (RESETSAVE NIL <'CLOSEHASHFILE $$VAL:1>))
               (RETURN $$VAL:1])
```

34

```
(mapEntryFile
  [LAMBDA NIL                                                 (* R.Erickson "27-Aug-79 18:09")
    (RESETLST (PROG (value hashFile)
                    (hashFile←(lockFile REQUESTDATAFILE T))
                                                              (* undone by RESETSAVE)
                    (MAPHASHFILE hashFile (FUNCTION [LAMBDA (key keyvalue)
                                 value← <<(MKATOM key)
                                           keyvalue> ! value>]))
                    (RETURN value])
```

35

```
(mapEntryFunction
  [LAMBDA NIL                                                 (* R.Erickson "22-Jul-81 17:54")

          (* * Returns list of <key entry> pairs for all fns)


    (RESETLST (PROG (value hashFile)
                    (hashFile←(lockFile REQUESTDATAFUNCTION T))
                                                              (* file closed by RESETSAVE)
                    [MAPHASHFILE hashFile (FUNCTION (LAMBDA (key keyvalue)
                                 (push value (<(MKATOM key)
                                           keyvalue>]
                    (RETURN value])
```

36

```
(nextSystem
  [LAMBDA (Key FileorFunction User PatchFile FromWhere)       (* R.Bates "2-MAR-79 14:32")
    (PROG (jfn entry)
          (entry←(create RequestFileRecord))
          (entry:What←Key)
          (entry:FileorFunction←FileorFunction)
          (entry:When←(IDATE))
          (entry:Who←User)
          (entry:WhereNow←PatchFile)
          (entry:FromWhere←FromWhere)
          (entry:Why←(getComment Key))
          (jfn←(OPENFILE RELEASEDATAFILE 'APPEND NIL NIL '(WAIT)))
          (PRIN2 entry jfn)
          (PRIN1 "
" jfn)
          (CLOSEF jfn])
```

37

```
(printChanged
  [LAMBDA (FileList OutFile Scribe IndexNumber)              (* R.Bates "26-Feb-80 09:35")
    (PROG (functionList)
          (FileList←(for x in FileList join (if x:FileorFunction~='FILE
                                                 then functionList←(ATTACH x functionList)
                                                      NIL
                                               else <x>)))
          (if (LISTP FileList)
              then (for fileEntry in FileList do (printChanged1 fileEntry OutFile Scribe IndexNumber)
                        first (printout OutFile T "These files have changed:" T T)
                              (Sprintout OutFile .TAB0 0 "@begin(Description,group)" T)
                        finally (Sprintout OutFile .TAB0 0 "@end(Description)" T)))
          (if (LISTP functionList)
              then (for functionEntry in functionList do (printChanged1 functionEntry OutFile Scribe
                                                                        IndexNumber)
                        first (printout OutFile T "These functions have changed:" T T)
                              (Sprintout OutFile .TAB0 0 "@begin(Description,group)" T)
                        finally (Sprintout OutFile .TAB0 0 "@end(Description)" T])
```

38

```
(printChanged1
  [LAMBDA (Entry OutFile Scribe IndexNumber)                 (* R.Bates "23-Jul-80 20:38")
    (if Scribe
        then (printout OutFile "@hinge" T "@begin(multiple)" T "@B[" Entry:What "]@\" "Changed by" ,
                       Entry:Who , "at" , (GDATE Entry:When)
                       , "from" T Entry:FromWhere "@*" , T)
             (if (NUMBERP IndexNumber)
                 then (printout OutFile "@IndexEntry(Key=%" " Entry:What "%",Entry=%"" Entry:What
                                "%")"
                                T "@IndexEntry(Key=%" " Entry:What "%",Entry=%"" "@ @ @ @ @ version "
                                IndexNumber "%",Number)" T)
                        " T] ")
      else (printout OutFile .PARA 2 78 (<Entry:What "changed by" Entry:Who "at" (GDATE Entry:When)
                       "from" Entry:FromWhere>)
                     T))
    (if (NUMBERP Entry:Why:CStart) and (NUMBERP Entry:Why:CEnd)
        then (if ~(OPENP Entry:Why:DFile 'INPUT)
                 then Entry:Why:DFile←(OPENFILE Entry:Why:DFile 'INPUT))
             (COPYBYTES Entry:Why:DFile OutFile Entry:Why:CStart Entry:Why:CEnd)
      else (printout OutFile .TAB 10 .PARA 10 60 Entry:Why T))
    (Sprintout OutFile "@end[multiple]" T)
    (printout OutFile T T])
```

39

```
(putEntry
  [LAMBDA (File Key Entry)                                   (* R.Bates "22-May-79 15:07")
    (PROG (hashFile)
          (if (NLISTP Key)
              then Key← <Key>)
          (hashFile←(lockFile File))
          (for item in Key do (PUTHASHFILE item Entry hashFile))
          (unLockFile hashFile])
```

40

```
(putEntryFile
  [LAMBDA (File Entry)                                       (* R.Bates "22-May-79 15:08")
```

```
    (putEntry REQUESTDATAFILE File Entry])
```

41

```
(putEntryFunction
  [LAMBDA (Function Entry)                                   (* R.Bates "22-May-79 15:09")
    (putEntry REQUESTDATAFUNCTION Function Entry])
```

42

```
(ratomsRetto
  [LAMBDA (File)                                             (* R.Erickson "28-Aug-79 16:13")
    (CLOSEF File)
    (RETFROM 'RATOM 'STOP])
```

43

```
(readPreviousVersion
  [LAMBDA NIL                                                (* R.Bates "30-Jan-80 15:00")
    (if (NLISTP (EVALV 'PreviousVersion))
        then (LOADVARS 'PreviousVersion PREVIOUSVERSION))
    PreviousVersion])
```

44

```
(readReleaseFile
  [LAMBDA NIL                                                (* R.Erickson "22-Jul-81 17:55")
    (PROG (file sexpr (ERRORTYPELST (<<16 NIL> ! ERRORTYPELST>)))
                                                             (* allow EOF errors to occur in READFILE)
          (file←(OPENFILE RELEASEDATAFILE 'INPUT NIL NIL '(WAIT)))
          (sexpr←(READFILE file))
          (CLOSEF? file)
          (RETURN sexpr])
```

45

```
(readRetfrom
  [LAMBDA (File)                                             (* R.Bates " 2-MAR-79 14:28")
    (CLOSEF File)
    (RETFROM 'READ 'STOP])
```

46

```
(redoFilePointers
  [LAMBDA NIL                                                (* D.Thompson "29-Aug-79 14:24")
    (PROG (listFunctions listFiles temp)
          (if ~ GOODGUY
              then (ERROR "You can't!"))
          (if (VerifyRequest)=NIL
              then (RETURN NIL))
          (temp←(CheckPoint))
          (listFunctions←(mapEntryFunction))
          (listFiles←(mapEntryFile))
          (for entry in listFiles do (if entry:2:State=NIL or entry:2:State='PARTIAL
                                                then (deleteEntryFile entry:1)))
          (for entry in listFunctions do (if entry:2:State=NIL
                                                  then (deleteEntryFunction entry:1)
                                                elseif entry:2:State='OWN
                                                  then (deleteEntryFunction entry:1)
                                                       (RequestFunction entry:1 entry:2:User NIL T)
                                                elseif entry:2:State='UPDATE
                                                  then (deleteEntryFunction entry:1)
                                                       entry:2:File←(if (affirmWhereIs entry:1)
                                                                        else 'NEW)
                                                       (putEntryFunction entry:1 entry:2)))
          (REHASHFILE REQUESTDATAFILE)
          (REHASHFILE REQUESTDATAFUNCTION)
          (RETURN temp])
```

47

```
(removeAccessFunction
  [LAMBDA (Function User)                                    (* R.Bates "22-FEB-79 16:29")
    (PROG (entry hashFile affirmFile)
        (if User=NIL
            then User←USERNAME)
        (hashFile←(lockFile REQUESTDATAFUNCTION))
        (entry←(checkIfOwn Function User hashFile))
        (affirmFile←entry:File)
        (entry:User←NIL)
        (if entry:PatchFile
            then (entry:State←'UPDATE)
          else (entry:State←NIL))
        (PUTHASHFILE Function entry hashFile)
        (unLockFile hashFile)
        (if affirmFile~='NEW
            then hashFile←(lockFile REQUESTDATAFILE)
                 (decrUserCount hashFile affirmFile User)
                 (unLockFile hashFile])
```

48

```
(requestCommentToFile
  [LAMBDA (File Entries)                              (* R.Bates " 6-Feb-80 15:11")
    (PROG (releaseCommentFile endPoint beginPoint)
          (File←(OPENFILE File 'APPEND NIL NIL '(WAIT)))
          (beginPoint←(GETFILEPTR File))
          (for entry in Entries
             do (if (NUMBERP entry:Why:CStart) and (NUMBERP entry:Why:CEnd)
                    then releaseCommentFile←(OPENFILE entry:Why:DFile 'INPUT NIL NIL '(WAIT))
                         (COPYBYTES releaseCommentFile File entry:Why:CStart entry:Why:CEnd)
                         (CLOSEF releaseCommentFile)
                  else (printout File .TAB 10 .PARA 10 60 entry:Why T T))
                (endPoint←(GETFILEPTR File))
                (entry:Why←(create RequestComment
                                   DFile ← File
                                   CStart ← beginPoint
                                   CEnd ← endPoint))
                (beginPoint←endPoint))
          (RETURN (CLOSEF File])
```

49

```
(systemBuild
  [LAMBDA (OnOff)                                     (* R.Bates " 5-Jun-79 10:29")
    (putEntryFunction SystemKey OnOff])
```

50

```
(systemCheck
  [LAMBDA (HashFile SystemBuild)                      (* R.Bates " 5-Jun-79 11:42")
    (if ~SystemBuild
        then (PROG (closeHashFile)
                   (if ~HashFile
                       then closeHashFile←T
                            HashFile←(lockFile REQUESTDATAFUNCTION))
                   (if (GETHASHFILE SystemKey HashFile)
                       then (unLockFile HashFile)
                            (ERROR "Can not do a request when system build in progress"))
                   (if closeHashFile
                       then (unLockFile HashFile])
```

51

```
(unLockFile
  [LAMBDA (File)                                      (* R.Bates " 7-FEB-79 11:43")
    (CLOSEHASHFILE File])
```

52

```
(unUpdateFunction
  [LAMBDA (Function File)                             (* D.Thompson "28-Jan-80 10:56")
    (PROG (entry hashFile)
          (hashFile←(lockFile REQUESTDATAFUNCTION))
          (entry←(GETHASHFILE Function hashFile))
          (if entry:State~='UPDATE
              then (unLockFile hashFile)
                   (printout NIL .TAB0 0 "The function" , Function , "is in state" , entry:State ,
                             "and not in the UPDATE state!"
                             T)
```

```
                (RETURN NIL))
          (if ~(for x in '(DIRECTORY NAME EXTENSION) always (FILENAMEFIELD File x)=(FILENAMEFIELD
                                                                   entry:PatchFile x))
              then (printout NIL .TAB0 0 "The system says the patch file came from" , entry:PatchFile
                             , "and not" , File T)
                   (if (AFFIRMUSER NIL NIL (CONCAT "Do you want to unUpdate " Function "? "))='N
                       then (unLockFile hashFile)
                            (RETURN NIL)))
          (entry:State←NIL)
          (entry:PatchFile←NIL)
          (PUTHASHFILE Function entry hashFile)
          (unLockFile hashFile)
          (RETURN Function])
)
[DECLARE: DONTEVAL@LOAD DONTCOPY
```

(* Normal Users should NOT call these functions.)]

```
(RPAQQ RequestSystemFNS (CheckPoint CreatePatchFile CleanupRequest VerifyRequest))
(DEFINEQ
```

53

(CheckPoint

```
  [LAMBDA NIL                                          (* R.Bates " 2-Aug-79 16:58")
    <(COPYHASHFILE REQUESTDATAFILE 'BACKUP.DATAFILE)
    (COPYHASHFILE REQUESTDATAFUNCTION 'BACKUP.DATAFUNCTION)
    (PROG (scr dst)
          (dst←(OPENFILE 'BACKUP.RELEASE 'OUTPUT))
          (scr←(OPENFILE RELEASEDATAFILE 'INPUT))
          (COPYBYTES scr dst)
          (CLOSEF dst)
          (CLOSEF scr)
          (RETURN dst))
    >])
```

54

(CreatePatchFile

```
  [LAMBDA NIL                                          (* R.Erickson "25-Sep-81 20:25")

          (* * Create <affirm>cpatch-nnn.)


    (PROG (fileRootPair funDirPair CPComs CPComsName userFNS newComs temp CPName file OldCPFns)
          (if (EVALV 'AFFIRMVERSION)='NOBIND
              then (ERROR "u.b.a. AFFIRMVERSION"))
          (if (MKATOM (USERNAME T))
              ~=PV\DIRECTORY
              then (ERROR "not connected to" PV\DIRECTORY))
          (CPName←(PACK* 'CPATCH- AFFIRMVERSION))
          (CPComsName←(FILECOMS CPName))
          (if ~(GETPROP CPName 'FILEDATES) and (file←(INFILEP (PACKFILENAME 'NAME CPName 'EXTENSION
                                                                                 'COM))
                                                     or file←(INFILEP CPName))
              then                                     (* not loaded, but patch exists -
                                                       load it)
                   (LOAD? file))
          (CPComs←(if (EVALV CPComsName)='NOBIND
                       then                            (* no COMS set)
                            NIL
                     else (EVALV CPComsName)))
          (OldCPFns←(FILEFNSLST CPName))

          (* * Request all fns which are in UPDATE state, so :PatchFile can later be transferred to the new CPatch.
          Those fns where we succeed are used to build records of user patch files, in -
          fileRootPair has entries "(<dir>name.exe name)" used for fetching COMS, -
          funDirPair has "(fn dir)" used to building userFNS.)


          [for fnAndEntry in (mapEntryFunction) bind AUTOUPDATEFLG unpak
             when fnAndEntry:2:State='UPDATE
             do (AUTOUPDATEFLG←(if (FILENAMEFIELD fnAndEntry:2:PatchFile 'NAME)=CPName
                                    and (FILENAMEFIELD fnAndEntry:2:PatchFile 'DIRECTORY)
                                        =PV\DIRECTORY
                                  then NIL
                                else T))
                (if (RequestFunction fnAndEntry:1):1
                    then unpak←(UNPACKFILENAME fnAndEntry:2:PatchFile)
                         (push fileRootPair (<(PACKFILENAME (for x in '(DIRECTORY NAME EXTENSION)
                                                               join <x (LISTGET unpak x)
                                                                       >))
                                              (LISTGET unpak 'NAME)
                                              >))
                         (push funDirPair (<fnAndEntry:1 (LISTGET unpak 'DIRECTORY)
                                              >]
          (OR OldCPFns fileRootPair (ERROR "No fns to update"))

          (* * Create the userFNS lists according to dirs of patch files. add to CPComs. userFNS may already have stuff.)


          (userFNS←(for (pair funList) in funDirPair when pair:1 ~MEMB OldCPFns
                      collect (funList←(PACK* pair:2 'FNS))
                              (if (EVALV funList)='NOBIND
                                   then (/SET funList NIL))
                                (/SET funList <pair:1 !(EVALV funList)
                                                       >)
                                funList))                       (* names of --FNS)
        (for funList in (REMOVEDUPLICATES userFNS) do [/SET funList (SORT (REMOVEDUPLICATES
                                                                           (EVALV funList]
                                                        (funList← <'FNS '* funList>)
                                                        (if ~(MEMBER funList CPComs)
                                                            then (push CPComs funList)))
                                                        (* load up non FNS patches, add to coms)
        (newComs←(COPY (for part in (for file in (REMOVEDUPLICATES fileRootPair)
                                         join (LOADVARS T file:1)
                                                  (EVALV (FILECOMS file:2)))
                            when part:1~='FNS collect part)))
        (CPComs← < ! CPComs ! newComs>)
        (/SET CPComsName CPComs)
        (APPLY* 'EDITV CPComsName)
        (MAKEFILE CPName)
        (CleanupFunction (INTERSECTION (WhoOwns T)
                                       (FILEFNSLST CPName))
                         NIL CPName)
        (RETURN CPName])
```

55

```
(CleanupRequest
  [LAMBDA (File FunsList)                                 (* R.Bates "11-Jul-80 14:25")
    (PROG (temp temp2)
          (if FunsList=NIL
              then FunsList←(FILEFNSLST File))
          (File←(if (INFILEP File)
                    else (ERROR "Can find file" File)))
          (if (VerifyRequest)=NIL
              then (ERROR "VerifyRequest() failed"))

          (* This function assumes that everything on the release file should go on previous versions file.
          unUpdateFunction does some checking to make sure it is ok to unUpdate a function.)


          (for fun in FunsList do (unUpdateFunction fun File))
                                                        (* redoFilePointers does a CheckPoint.)
          (if temp2←(redoFilePointers)=NIL
              then (ERROR "unUpdateFunction() didn't work"))
          (temp←(REVERSE (readReleaseFile)))
          (requestCommentToFile REQUESTCOMMENTS temp)
          (LOAD PREVIOUSVERSION)
          (/SET 'PreviousVersion <<AFFIRMVERSION-1 temp> ! PreviousVersion>)
          (/SET 'CHANGEDVARSLST <'PreviousVersion ! CHANGEDVARSLST>)
          (FILES?)

          (* Still have to clean up the Release file by hand !!!! The right way to fix this problem is to create a new
          ReleaseFile and rename it over the old one.)


          (RETURN temp2])
```

56

```
(VerifyRequest
  [LAMBDA NIL                                           (* R.Bates "17-Dec-79 09:17")
    (PROG (listFunctions listFiles funUserList fileUserList temp error)
          (listFunctions←(mapEntryFunction))
          (listFiles←(mapEntryFile))
          (funUserList←(for entry in listFunctions collect <entry:2:User entry:2:File>
                            when entry:2:State='OWN and entry:2:File~='NEW))
          (fileUserList←(for entry in listFiles
                             join (for user in entry:2:User
                                       join (for i from 1 to user::1 collect <user:1 entry:1>))
                             when entry:2:State='PARTIAL))
          (for user in funUserList do (if ~(temp←(MEMBER user fileUserList))
                                          then error←T
                                               (LISPXPRIN1 user:1)
                                               (LISPXPRIN1 " should have partial access in file ")
                                               (LISPXPRIN1 user:2)
                                               (LISPXTERPRI)
                                        else (temp:1←NIL)))
          (for user in fileUserList do (error←T)
                                       (LISPXPRIN1 user:1)
                                       (LISPXPRIN1 " extra partial access in file ")
```

```
                                                       (LISPXPRIN1 user:2)
                                                       (LISPXTERPRI)
              when user)
          (RETURN ~error])
)

(RPAQQ REQUESTRECORDS (RequestComment RequestFileRecord RequestFunctionRecord))
[DECLARE: EVAL@COMPILE

(RECORD RequestComment (DFile CStart CEnd))

(RECORD RequestFileRecord (What FileorFunction When Who Why FromWhere WhereNow))

(RECORD RequestFunctionRecord (State User File PatchFile))
]

(RPAQ AskFromFile NIL)

(RPAQQ FilesToUpdate (PGU DATABASE INIT.LISP))

(RPAQQ SystemKey SystemKey% % % )

(RPAQQ PREVIOUSVERSION <AFFIRM>PREVIOUSVERSION)

(RPAQ COMMENT\READ\TABLE (COPYREADTABLE (QUOTE ORIG)))

(RPAQ AUTOUPDATEFLAG T)

(RPAQ HOMEMACHINE HOSTNAME)

(RPAQQ REQUESTDATAFILE <AFFIRM>REQUESTDATAFILE..11)

(RPAQQ REQUESTDATAFUNCTION <AFFIRM>REQUESTDATAFUNCTION..11)

(RPAQQ RELEASEDATAFILE <AFFIRM>RELEASEDATAFILE..10)

(RPAQQ REQUESTCOMMENTS <AFFIRM>COMMENTS.DB.1)
(SETBRK (QUOTE (26))
        1 COMMENT\READ\TABLE)

(RPAQQ REQUESTADVISE (EDITF))

(PUTPROPS EDITF ARGNAMES (NIL (FN COM1 ... COMN) . EDITFX))

(PUTPROPS EDITF READVICE [NIL (BEFORE NIL (AND [OR (LITATOM EDITFX)
                                                    (AND (LISTP EDITFX)
                                                         (NULL (CDR EDITFX]
                                                (RequestFunction EDITFX NIL T])
(READVISE EDITF)
(DECLARE: EVAL@COMPILE

(PUTPROPS Sprintout MACRO [X (LIST (QUOTE COND)
                                   (LIST (QUOTE Scribe)
                                         (CONS (QUOTE printout)
                                               X])
)
[DECLARE: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY
(BLOCK: REQUESTBLOCK CleanupFile CleanupFunction ReleaseFile ReleaseFunction RequestFile
        RequestFunction Update UpdateFile UpdateFunction WhatChanged WhoOwns WhoOwnsFile
        WhoOwnsFunction affirmWhereIs askIfNew askToContinue askToOwn checkIfOwn CheckPoint checkUser
        CleanupRequest CreatePatchFile decrUserCount deleteEntryFunction deleteEntryFile getEntry
        getEntryFile getEntryFunction grantAccessFile grantAccessFunction lockFile mapEntryFile
        mapEntryFunction nextSystem printChanged printChanged1 putEntry putEntryFile putEntryFunction
        readPreviousVersion redoFilePointers removeAccessFunction systemBuild systemCheck unLockFile
        unUpdateFunction VerifyRequest
        (ENTRIES CleanupFile CleanupFunction ReleaseFile ReleaseFunction RequestFile RequestFunction
                Update UpdateFile UpdateFunction WhatChanged WhoOwns WhoOwnsFile WhoOwnsFunction
                CheckPoint CleanupRequest CreatePatchFile getEntryFile getEntryFunction mapEntryFile
                mapEntryFunction putEntryFile putEntryFunction redoFilePointers systemBuild
                VerifyRequest)
        (GLOBALVARS AUTOUPDATEFLAG BADFILENAMES CHANGEDVARSLST FULLSYSTEMFILES HELPFLAG HELPTIME
                   HOMEMACHINE HOSTNAME LASTWORD PREVIOUSVERSION PV\DIRECTORY PreviousVersion
                   RELEASEDATAFILE REQUESTDATAFILE REQUESTDATAFUNCTION SYSTEMFILES SystemKey
                   USERNAME value)
        (NOLINKFNS . T))
]
(DECLARE: DONTCOPY
  (FILEMAP (NIL (3183 37265 (CleanupFile 3195 . 3985) (CleanupFunction 3989 . 6182) (InitializeDataFiles
 6186 . 6559) (NoteFiles 6563 . 7995) (ReleaseFile 7999 . 8512) (ReleaseFunction 8516 . 8777) (
RequestFile 8781 . 9344) (RequestFunction 9348 . 11282) (Update 11286 . 11621) (UpdateFile 11625 .
12065) (UpdateFunction 12069 . 14569) (WhatChanged 14573 . 16444) (WhoOwns 16448 . 17425) (WhoOwnsFile
```