File created: 12-Jun-90 16:18:18 **{DSK}<usr>local>lde>lispcore>library>TCPDOMAIN.;3**

changes to: (VARS TCPDOMAINCOMS)

previous date: 28-Feb-89 18:35:51 {DSK}<usr>local>lde>lispcore>library>TCPDOMAIN.;2

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ TCPDOMAINCOMS
       ((COMS ;; TCP/IP Domain resolver implementation.  RFC882, RFC883, RFC973
              )
        (COMS ;; UDP protocol functions
              (DECLARE%: DONTCOPY (EXPORT (CONSTANTS (\UDPDOMAIN.WDS 6))
                                          (RECORDS DOMAIN.HEADER)))
              (INITVARS (\UDPDOMAIN.IPSOCKET))
              (GLOBALVARS \UDPDOMAIN.IPSOCKET)
              (FILES (SYSLOAD)
                     TCPUDP)
              (FNS \UDPDOM.PROCESS.RESPONSE \UDPDOM.QUERY \UDPDOM.IPSOCKET))
        (COMS ;; Protocol independent functions
              [DECLARE%: DONTCOPY (EXPORT (CONSTANTS * DOMAIN.OPCODES)
                                          (CONSTANTS * DOMAIN.RCODES)
                                          (CONSTANTS * DOMAIN.RRTYPES)
                                          (CONSTANTS * DOMAIN.CLASSTYPES)
                                          (CONSTANTS (\DOMAIN.PORT 53]
              (INITVARS (\DOMAIN.DEFAULT.SERVER))
              (GLOBALVARS \DOMAIN.DEFAULT.SERVER)
              (FNS \DOMAIN.NAME \DOMAIN.PACK.NAME.LIST \DOMAIN.PARSE.NAME \DOMAIN.RCODE.ERROR
                   \DOMAIN.PROCESS.REDIRECT \DOMAIN.PROCESS.RESPONSE \DOMAIN.PROCESS.RR \DOMAIN.READ.ADDRESS
                   \DOMAIN.READ.NAME.FROM.STREAM \DOMAIN.READ.STRING.FROM.STREAM
                   \DOMAIN.SEARCH.FOR.CANONICAL.NAME \DOMAIN.SKIP.NAME.IN.STREAM \DOMAIN.SKIP.QUESTION
                   \DOMAIN.SKIP.RR))
        (COMS ;; Functions to maintain the domain tree structure
              (RECORDS DOMAIN.TREE.NODE DOMAIN.SERVER)
              (INITRECORDS DOMAIN.TREE.NODE)
              (FNS USTRINGHASHBITS)
              (INITVARS (\DOMAIN.ROOT (create DOMAIN.TREE.NODE NAME _ ""))
                     (\DOMAIN.NAMESERVERS (HASHARRAY 50 1.2 (FUNCTION USTRINGHASHBITS)
                                                 (FUNCTION STRING-EQUAL)))
                     (\DOMAIN.UNKNOWN.DOMAINS)
                     (\DOMAIN.GC.INTERVAL 600000)
                     (\DOMAIN.GC.TIMER (SETUPTIMER \DOMAIN.GC.INTERVAL)))
              (GLOBALVARS \DOMAIN.ROOT \DOMAIN.NAMESERVERS \DOMAIN.UNKNOWN.DOMAINS \DOMAIN.GC.TIMER
                     \DOMAIN.GC.INTERVAL)
              (FNS \DOMAIN.ADD.NEW.DOMAIN \DOMAIN.ADD.NAMESERVER \DOMAIN.AUGMENT.TREE
                   \DOMAIN.CHOOSE.BEST.SERVERS \DOMAIN.FIND.DOMAIN.IN.TREE \DOMAIN.INIT \DOMAIN.INSERT.IN.TREE
                   \DOMAIN.PATH \DOMAIN.SEARCH.RESOURCE.LIST \DOMAIN.DELETE.NAMESERVER \DOMAIN.AROUND.EXIT
                   \DOMAIN.DELETE.TREE \DOMAIN.BACKGROUND \DOMAIN.GC.NAMESERVERS \DOMAIN.SORT.BY.SVC.TIME)
              (ADDVARS (BACKGROUNDFNS \DOMAIN.BACKGROUND)))
        (COMS ;; Programmer's interface
              (INITVARS (DOMAIN.TRACE.FLG)
                     (DOMAIN.TRACE.FILE)
                     (INTERNET.LOCAL.DOMAIN))
              (GLOBALVARS DOMAIN.TRACE.FLG DOMAIN.TRACE.FILE INTERNET.LOCAL.DOMAIN)
              (FNS DOMAIN.INIT DOMAIN.LOOKUP.ADDRESS DOMAIN.LOOKUP.NAMESERVER DOMAIN.LOOKUP.OSTYPE DOMAIN.LOOKUP
                   DOMAIN.GRAPH DOMAIN.NAME.EQUAL DOMAIN.TRACE DOMAIN.TRACEWINDOW.BUTTONFN))
        (P (DOMAIN.INIT))))
```

;; TCP/IP Domain resolver implementation. RFC882, RFC883, RFC973

;; UDP protocol functions

```
(DECLARE%: DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(RPAQQ \UDPDOMAIN.WDS 6)

(CONSTANTS (\UDPDOMAIN.WDS 6))
)

(DECLARE%: EVAL@COMPILE
```

```
(BLOCKRECORD DOMAIN.HEADER ((ID WORD)
                            (RESPONSEFLG FLAG)
                            (OPCODE BITS 4)
                            (AUTHORITYFLG FLAG)
                            (TRUNCATEDFLG FLAG)
                            (WANTRECURSEFLG FLAG)
                            (CANRECURSEFLG FLAG)
                            (NIL BITS 3)
                            (RESPONSECODE BITS 4)
                            (QDCOUNT WORD)
                            (ANCOUNT WORD)
                            (NSCOUNT WORD)
                            (ARCOUNT WORD)))
)
)
```

;; END EXPORTED DEFINITIONS

```
(RPAQ? \UDPDOMAIN.IPSOCKET )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \UDPDOMAIN.IPSOCKET)
)

(FILESLOAD (SYSLOAD)
       TCPUDP)

(DEFINEQ
```

(\UDPDOM.PROCESS.RESPONSE

```
  [LAMBDA (DOMAIN.PATH RESPONSE)                                    (* ejs%: " 5-Nov-86 13:38")

          (* * This function parses a query reponse packet)

    (LET ((RESPONSEBASE (fetch (UDP UDPCONTENTS) of RESPONSE)))
         (COND
            ((NEQ 0 (fetch (DOMAIN.HEADER ANCOUNT) of RESPONSEBASE))

          (* * The response packet has the information we requested)

               (PROG1 (\DOMAIN.PROCESS.RESPONSE (\MAKEBASEBYTESTREAM RESPONSEBASE 0 (IDIFFERENCE
                                                                                      (fetch (UDP UDPLENGTH)
                                                                                         of RESPONSE)
                                                                                      \UDPOVLEN)
                                                                                'INPUT))
                      (\RELEASE.ETHERPACKET RESPONSE)))
            ((OR (NEQ 0 (fetch (DOMAIN.HEADER NSCOUNT) of RESPONSEBASE))
                 (NEQ 0 (fetch (DOMAIN.HEADER ARCOUNT) of RESPONSEBASE)))

          (* * The server we asked didn't know, but did tell us the name of a server which might know)

               (PROG1 (\DOMAIN.PROCESS.REDIRECT (\MAKEBASEBYTESTREAM RESPONSEBASE 0 (IDIFFERENCE
                                                                                      (fetch (UDP UDPLENGTH)
                                                                                         of RESPONSE)
                                                                                      \UDPOVLEN)
                                                                               'INPUT))
                      (\RELEASE.ETHERPACKET RESPONSE)))
            (T (\RELEASE.ETHERPACKET RESPONSE)
               'FAILED])
```

(\UDPDOM.QUERY

```
  [LAMBDA (DOMAIN TYPE CLASS SERVER)                                (* ejs%: " 5-Nov-86 13:40")

          (* * Make a domain query. Argument semantics should be self-evident if you've read RFC882 and RFC883.
          Returns a list of answers, or atoms to indicate failure--USE.TCP, etc)

    (LET* ((QUERY (\ALLOCATE.ETHERPACKET))
           (ID (RAND 1 65534))
           ANSWER DOMAINBASE)

          (* * Do basic QUERY initialization)

          (UDP.SETUP QUERY (OR SERVER \DOMAIN.DEFAULT.SERVER)
                 \DOMAIN.PORT ID (\UDPDOM.IPSOCKET))
          (SETQ DOMAINBASE (fetch (UDP UDPCONTENTS) of QUERY))

          (* * Format header section)

          (replace (DOMAIN.HEADER ID) of DOMAINBASE with ID)
          (replace (DOMAIN.HEADER RESPONSEFLG) of DOMAINBASE with NIL)
          (replace (DOMAIN.HEADER AUTHORITYFLG) of DOMAINBASE with NIL)
          (replace (DOMAIN.HEADER TRUNCATEDFLG) of DOMAINBASE with NIL)
          (replace (DOMAIN.HEADER WANTRECURSEFLG) of DOMAINBASE with NIL)
          (replace (DOMAIN.HEADER CANRECURSEFLG) of DOMAINBASE with NIL)
```

```
          (replace (DOMAIN.HEADER OPCODE) of DOMAINBASE with DOMAIN.QUERY)
          (replace (DOMAIN.HEADER RESPONSECODE) of DOMAINBASE with 0)
          (replace (DOMAIN.HEADER QDCOUNT) of DOMAINBASE with 1)
          (replace (DOMAIN.HEADER ANCOUNT) of DOMAINBASE with 0)
          (replace (DOMAIN.HEADER NSCOUNT) of DOMAINBASE with 0)
          (replace (DOMAIN.HEADER ARCOUNT) of DOMAINBASE with 0)
          (UDP.INCREMENT.LENGTH QUERY (UNFOLD \UDPDOMAIN.WDS BYTESPERWORD))

          (* * Add Query)

          [COND
             ((AND (NOT (NULL DOMAIN))
                    (NLISTP DOMAIN))
              (SETQ DOMAIN (\DOMAIN.PARSE.NAME DOMAIN]
          (for NAME in DOMAIN do (UDP.APPEND.BYTE QUERY (NCHARS NAME))
                                 (UDP.APPEND.STRING QUERY (MKSTRING NAME))
             finally (UDP.APPEND.BYTE QUERY 0))
          (UDP.APPEND.WORD QUERY TYPE)
          (UDP.APPEND.WORD QUERY CLASS)

          (* * Do the query)

          (bind RESPONSE RESPONSEBASE for I from 1 to \MAXETHERTRIES
             do (COND
                   [(SETQ RESPONSE (UDP.EXCHANGE (\UDPDOM.IPSOCKET)
                                                 QUERY 10000))
                    (SETQ RESPONSEBASE (fetch (UDP UDPCONTENTS) of RESPONSE))
                    (COND
                       [(AND (EQ (fetch (DOMAIN.HEADER ID) of RESPONSEBASE)
                                 ID)
                             (fetch (DOMAIN.HEADER RESPONSEFLG) of RESPONSEBASE))
                        (COND
                           ((AND (fetch (DOMAIN.HEADER TRUNCATEDFLG) of RESPONSEBASE)
                                 (EQ (fetch (DOMAIN.HEADER ANCOUNT) of RESPONSEBASE)
                                     0)
                                 (EQ (fetch (DOMAIN.HEADER NSCOUNT) of RESPONSEBASE)
                                     0)
                                 (EQ (fetch (DOMAIN.HEADER RESPONSECODE) of RESPONSEBASE)
                                     RCODE.OK))
                            (SETQ ANSWER 'USE.TCP)
                            (\RELEASE.ETHERPACKET RESPONSE)
                            (GO $$OUT))
                           ((NEQ (fetch (DOMAIN.HEADER RESPONSECODE) of RESPONSEBASE)
                                 RCODE.OK)
                            (SETQ ANSWER (\DOMAIN.RCODE.ERROR (fetch (DOMAIN.HEADER RESPONSECODE) of
                                                                                                  RESPONSEBASE
                                                                                                  )))
                            (COND
                               (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE)
                                                 (printout DOMAIN.TRACE.FILE "Error on query: " ANSWER)))
                            (\RELEASE.ETHERPACKET RESPONSE)
                            (GO $$OUT))
                           (T (SETQ ANSWER (\UDPDOM.PROCESS.RESPONSE DOMAIN RESPONSE))
                              (GO $$OUT]
                       (T (\RELEASE.ETHERPACKET RESPONSE]
                   (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE)
                          (printout DOMAIN.TRACE.FILE "Query to " (\IP.ADDRESS.TO.STRING (fetch (IP
                                                                                                IPDESTINATIONADDRESS
                                                                                                )
                                                                                          of QUERY))
                                    " timed out.")))
             finally (\RELEASE.ETHERPACKET QUERY)
                     (RETURN ANSWER])
```

(**UDPDOM.IPSOCKET**

```
  [LAMBDA NIL                                                    (* ejs%: "12-Apr-86 20:39")
    [COND
       ((NULL \UDPDOMAIN.IPSOCKET)
        (SETQ \UDPDOMAIN.IPSOCKET (UDP.OPEN.SOCKET)))
       ((NOT (\IP.FIND.SOCKET (fetch (IPSOCKET IPSOCKET) of \UDPDOMAIN.IPSOCKET)
                    (\IP.FIND.PROTOCOL \UDP.PROTOCOL)))
        (SETQ \UDPDOMAIN.IPSOCKET (UDP.OPEN.SOCKET NIL 'ACCEPT]
    \UDPDOMAIN.IPSOCKET])

)

;; Protocol independent functions

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(RPAQQ DOMAIN.OPCODES ((DOMAIN.QUERY 0)
                       (DOMAIN.IQUERY 1)
                       (DOMAIN.CQUERYM 2)
```

```
                        (DOMAIN.CQUERYU 3)))

(DECLARE%: EVAL@COMPILE

(RPAQQ DOMAIN.QUERY 0)

(RPAQQ DOMAIN.IQUERY 1)

(RPAQQ DOMAIN.CQUERYM 2)

(RPAQQ DOMAIN.CQUERYU 3)

(CONSTANTS (DOMAIN.QUERY 0)
       (DOMAIN.IQUERY 1)
       (DOMAIN.CQUERYM 2)
       (DOMAIN.CQUERYU 3))
)

(RPAQQ DOMAIN.RCODES ((RCODE.OK 0)
                      (RCODE.FORMATERROR 1)
                      (RCODE.SERVERFAILED 2)
                      (RCODE.NAMEERROR 3)
                      (RCODE.NOTIMPLEMENTED 4)
                      (RCODE.REFUSED 5)))

(DECLARE%: EVAL@COMPILE

(RPAQQ RCODE.OK 0)

(RPAQQ RCODE.FORMATERROR 1)

(RPAQQ RCODE.SERVERFAILED 2)

(RPAQQ RCODE.NAMEERROR 3)

(RPAQQ RCODE.NOTIMPLEMENTED 4)

(RPAQQ RCODE.REFUSED 5)

(CONSTANTS (RCODE.OK 0)
       (RCODE.FORMATERROR 1)
       (RCODE.SERVERFAILED 2)
       (RCODE.NAMEERROR 3)
       (RCODE.NOTIMPLEMENTED 4)
       (RCODE.REFUSED 5))
)

(RPAQQ DOMAIN.RRTYPES
       ((RRTYPE.A 1)
        (RRTYPE.NS 2)
        (RRTYPE.MD 3)
        (RRTYPE.MF 4)
        (RRTYPE.CNAME 5)
        (RRTYPE.SOA 6)
        (RRTYPE.MB 7)
        (RRTYPE.MG 8)
        (RRTYPE.MR 9)
        (RRTYPE.NULL 10)
        (RRTYPE.WKS 11)
        (RRTYPE.PTR 12)
        (RRTYPE.HINFO 13)
        (RRTYPE.MINFO 14)
        (RRTYPE.MX 15)))

(DECLARE%: EVAL@COMPILE

(RPAQQ RRTYPE.A 1)

(RPAQQ RRTYPE.NS 2)

(RPAQQ RRTYPE.MD 3)

(RPAQQ RRTYPE.MF 4)

(RPAQQ RRTYPE.CNAME 5)

(RPAQQ RRTYPE.SOA 6)

(RPAQQ RRTYPE.MB 7)

(RPAQQ RRTYPE.MG 8)

(RPAQQ RRTYPE.MR 9)

(RPAQQ RRTYPE.NULL 10)

(RPAQQ RRTYPE.WKS 11)
```

```
(RPAQQ RRTYPE.PTR 12)

(RPAQQ RRTYPE.HINFO 13)

(RPAQQ RRTYPE.MINFO 14)

(RPAQQ RRTYPE.MX 15)

(CONSTANTS (RRTYPE.A 1)
       (RRTYPE.NS 2)
       (RRTYPE.MD 3)
       (RRTYPE.MF 4)
       (RRTYPE.CNAME 5)
       (RRTYPE.SOA 6)
       (RRTYPE.MB 7)
       (RRTYPE.MG 8)
       (RRTYPE.MR 9)
       (RRTYPE.NULL 10)
       (RRTYPE.WKS 11)
       (RRTYPE.PTR 12)
       (RRTYPE.HINFO 13)
       (RRTYPE.MINFO 14)
       (RRTYPE.MX 15))
)

(RPAQQ DOMAIN.CLASSTYPES ((CLASSTYPE.IN 1)
                          (CLASSTYPE.CSNET 2)
                          (CLASSTYPE.CHAOS 3)))

(DECLARE%: EVAL@COMPILE

(RPAQQ CLASSTYPE.IN 1)

(RPAQQ CLASSTYPE.CSNET 2)

(RPAQQ CLASSTYPE.CHAOS 3)

(CONSTANTS (CLASSTYPE.IN 1)
       (CLASSTYPE.CSNET 2)
       (CLASSTYPE.CHAOS 3))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \DOMAIN.PORT 53)

(CONSTANTS (\DOMAIN.PORT 53))
)
)
```

;; END EXPORTED DEFINITIONS

```
(RPAQ? \DOMAIN.DEFAULT.SERVER )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \DOMAIN.DEFAULT.SERVER)
)

(DEFINEQ
```

(\DOMAIN.NAME

```
  [LAMBDA (DOMAIN.TREE.NODE)                                    (* ejs%: "13-Apr-86 15:38")

          (* * Generate a list of domain names along the path to the root of the domain tree)

    (COND
       ((NULL (fetch (DOMAIN.TREE.NODE SUPERDOMAIN) of DOMAIN.TREE.NODE))
        NIL)
       (T (LET [(SUFFIX (\DOMAIN.NAME (fetch (DOMAIN.TREE.NODE SUPERDOMAIN) of DOMAIN.TREE.NODE]
               (COND
                  (SUFFIX (CONCAT (fetch (DOMAIN.TREE.NODE NAME) of DOMAIN.TREE.NODE)
                                  "." SUFFIX))
                  (T (fetch (DOMAIN.TREE.NODE NAME) of DOMAIN.TREE.NODE])
```

(\DOMAIN.PACK.NAME.LIST

```
  [LAMBDA (LIST)                                                (* ejs%: "12-Apr-86 20:29")
    (COND
       ((LISTP LIST)
        (LET [(DOMAIN.NAME (ALLOCSTRING (IPLUS (SUB1 (LENGTH LIST))
                                               (for NAME in LIST sum (NCHARS NAME]
             [bind (I _ 1) for NAME in LIST do (RPLSTRING DOMAIN.NAME I NAME)
                                               (add I (NCHARS NAME))
                                               (COND
                                                  ((ILESSP I (NCHARS DOMAIN.NAME))
```

```
                                                (RPLCHARCODE DOMAIN.NAME I (CHARCODE %.))
                                                (add I 1]
            DOMAIN.NAME))
       (T (ALLOCSTRING 0])
```

(\DOMAIN.PARSE.NAME

```
[LAMBDA (NAME)                                                  (* ejs%: "12-Apr-86 18:11")
```

(* * This function parses a domain name (e.g. SUMEX.STANFORD.EDU)%, and returns a list of domain labels (SUMEX STANFORD EDU))

```
    (bind (SCRATCHSTRING _ (CONSTANT (ALLOCSTRING 63)))
          NAMELIST
          (LENGTH _ 0) for CHAR instring (MKSTRING NAME)
       do (COND
             [(EQ CHAR (CHARCODE %.))
              (COND
                 ((NEQ 0 LENGTH)
                  [SETQ NAMELIST (NCONC1 NAMELIST (CONCAT (SUBSTRING SCRATCHSTRING 1 LENGTH]
                  (SETQ LENGTH 0]
             ((IGREATERP LENGTH 63)
              (ERROR "Domain name too long" SCRATCHSTRING))
             (T (RPLCHARCODE SCRATCHSTRING (add LENGTH 1)
                             CHAR)))
       finally (RETURN (COND
                         [(NEQ LENGTH 0)
                          (NCONC1 NAMELIST (CONCAT (SUBSTRING SCRATCHSTRING 1 LENGTH]
                         (T NAMELIST])
```

(\DOMAIN.RCODE.ERROR

```
[LAMBDA (CODE)                                                  (* ejs%: "12-Apr-86 19:15")
  (SELECTC CODE
       (RCODE.OK 'OK)
       (RCODE.FORMATERROR
           'FORMAT.ERROR)
       (RCODE.SERVERFAILED
           'SERVER.FAILED)
       (RCODE.NAMEERROR
           'NAME.ERROR)
       (RCODE.NOTIMPLEMENTED
           'NOT.IMPLEMENTED)
       (RCODE.REFUSED
           'REFUSED)
      NIL])
```

(\DOMAIN.PROCESS.REDIRECT

```
[LAMBDA (STREAM)                                                (* ejs%: "12-Apr-86 21:04")
```

(* * Skip past the header and query section to get to the answer section)

(* * Past ID and flags in header)

```
  (\WIN STREAM)
  (\WIN STREAM)
  (LET ((%#QUESTIONS (\WIN STREAM))
        (%#ANSWERS (\WIN STREAM))
        (%#NSERVERS (\WIN STREAM))
        (%#ADDITIONAL (\WIN STREAM)))
```

(* * Past questions)

```
       (for I from 1 to %#QUESTIONS do (\DOMAIN.SKIP.QUESTION STREAM))
```

(* * Collect answers)

```
       (for I from 1 to %#ANSWERS collect (\DOMAIN.SKIP.RR STREAM))
```

(* * Collect rest)

```
       (APPEND (for I from 1 to %#NSERVERS collect (\DOMAIN.PROCESS.RR STREAM))
               (for I from 1 to %#ADDITIONAL collect (\DOMAIN.PROCESS.RR STREAM])
```

(\DOMAIN.PROCESS.RESPONSE

```
[LAMBDA (STREAM)                                                (* ejs%: "12-Apr-86 19:58")
```

(* * Skip past the header and query section to get to the answer section)

(* * Past ID and flags in header)

```
  (\WIN STREAM)
  (\WIN STREAM)
  (LET ((%#QUESTIONS (\WIN STREAM))
        (%#ANSWERS (\WIN STREAM)))
```

```
          (* * Past rest of header)

         (\WIN STREAM)
         (\WIN STREAM)

          (* * Past questions)

         (for I from 1 to %#QUESTIONS do (\DOMAIN.SKIP.QUESTION STREAM))

          (* * Collect answers)

         (for I from 1 to %#ANSWERS collect (\DOMAIN.PROCESS.RR STREAM])
```

(\DOMAIN.PROCESS.RR

```
  [LAMBDA (STREAM)                                                    (* ejs%: "13-Apr-86 17:09")

          (* * Process a resource record beginning at the current point in the stream)

    (LET ((NAME (\DOMAIN.READ.NAME.FROM.STREAM STREAM))
          (TYPE (\WIN STREAM))
          (CLASS (\WIN STREAM))
          (TTL (\MAKENUMBER (\WIN STREAM)
                    (\WIN STREAM)))
          (RDLEN (\WIN STREAM))
          ANSWER)
         [SETQ ANSWER `(NAME %, NAME TYPE %, TYPE CLASS %, CLASS TTL %, TTL DATA %,
                         (SELECTC TYPE
                              (RRTYPE.A (\DOMAIN.READ.ADDRESS STREAM CLASS (FOLDLO RDLEN BYTESPERCELL)))
                              ((LIST RRTYPE.CNAME RRTYPE.NS)
                                   (\DOMAIN.READ.NAME.FROM.STREAM STREAM))
                              (RRTYPE.HINFO (CONS (\DOMAIN.READ.STRING.FROM.STREAM STREAM)
                                                  (\DOMAIN.READ.STRING.FROM.STREAM STREAM)))
                              (PROGN (for I from 1 to RDLEN do (BIN STREAM))
                                     NIL]
         [COND
            (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE (printout DOMAIN.TRACE.FILE "Answer received: "
                                                                     ANSWER]
         ANSWER])
```

(\DOMAIN.READ.ADDRESS

```
  [LAMBDA (STREAM CLASS %#ADDRESSES)                                  (* ejs%: "12-Apr-86 20:56")
    (SELECTC CLASS
         (CLASSTYPE.IN [COND
                           ((EQ %#ADDRESSES 0)
                            NIL)
                           [(NEQ %#ADDRESSES 1)
                            (for I from 1 to %#ADDRESSES collect (\MAKENUMBER (\WIN STREAM)
                                                                              (\WIN STREAM]
                           (T (\MAKENUMBER (\WIN STREAM)
                                           (\WIN STREAM])
         NIL])
```

(\DOMAIN.READ.NAME.FROM.STREAM

```
  [LAMBDA (STREAM)                                                    (* ejs%: "12-Apr-86 20:54")
    (bind NAMELEN NAMELST until (EQ 0 (SETQ NAMELEN (BIN STREAM)))
       do [COND
             [(EQ 3 (LRSH NAMELEN 6))

          (* * Process a pointer redirection)

              (LET ((CONTINUATIONADDR (create WORD
                                              HIBYTE _ (LOGAND NAMELEN (MASK.1'S 0 6))
                                              LOBYTE _ (BIN STREAM)))
                    (STREAMPTR (GETFILEPTR STREAM)))
                   (SETFILEPTR STREAM CONTINUATIONADDR)
                   (RETURN (PROG1 (COND
                                     (NAMELST (CONCAT (\DOMAIN.PACK.NAME.LIST (DREVERSE NAMELST))
                                                      "."
                                                      (\DOMAIN.READ.NAME.FROM.STREAM STREAM)))
                                     (T (\DOMAIN.READ.NAME.FROM.STREAM STREAM)))
                                  (SETFILEPTR STREAM STREAMPTR]
             (T

          (* * Normal name segment)

                (LET ((NAME (ALLOCSTRING NAMELEN)))
                     (\BINS STREAM (fetch (STRINGP BASE) of NAME)
                            (fetch (STRINGP OFFST) of NAME)
                            NAMELEN)
                     (push NAMELST NAME]
       finally (RETURN (\DOMAIN.PACK.NAME.LIST (DREVERSE NAMELST])
```

```
(\DOMAIN.READ.STRING.FROM.STREAM
  [LAMBDA (STREAM)                                                    (* ejs%: "13-Apr-86 02:33")
    (LET* ((NAMELEN (BIN STREAM))
           (STRING (ALLOCSTRING NAMELEN)))
          (for I from 1 to NAMELEN do (RPLCHARCODE STRING I (BIN STREAM)))
          STRING])

(\DOMAIN.SEARCH.FOR.CANONICAL.NAME
  [LAMBDA (NAME RRLST)                                                (* ejs%: "14-Nov-86 14:44")
    (bind FOUNDIT DATA for RR in RRLST thereis (AND (EQ RRTYPE.CNAME (LISTGET RR 'TYPE))
                                                    (DOMAIN.NAME.EQUAL (LISTGET RR 'NAME)
                                                           NAME)
                                                    (SETQ FOUNDIT T))
       finally (RETURN (AND FOUNDIT (LISTGET RR 'DATA])

(\DOMAIN.SKIP.NAME.IN.STREAM
  [LAMBDA (STREAM)                                                    (* ejs%: "12-Apr-86 21:06")
    (bind NAMELEN NAMELST until (EQ 0 (SETQ NAMELEN (BIN STREAM)))
       do (COND
             ((EQ 3 (LRSH NAMELEN 6))

          (* * Process a pointer redirection)

              (BIN STREAM))
             (T

          (* * Normal name segment)

                (for I from 1 to NAMELEN do (BIN STREAM])

(\DOMAIN.SKIP.QUESTION
  [LAMBDA (STREAM)                                                    (* ejs%: "12-Apr-86 21:06")

          (* * Skip over a question section--composed of compressed name, QTYPE, and QCLASS fields)

    (\DOMAIN.SKIP.NAME.IN.STREAM STREAM)
    (\WIN STREAM)
    (\WIN STREAM])

(\DOMAIN.SKIP.RR
  [LAMBDA (STREAM)                                                    (* ejs%: "12-Apr-86 21:10")

          (* * Skip a resource record beginning at the current point in the stream)

          (* * Name)

    (\DOMAIN.SKIP.NAME.IN.STREAM STREAM)

          (* * Type)

    (\WIN STREAM)

          (* * Class)

    (\WIN STREAM)

          (* * Time to Live)

    (\WIN STREAM)
    (\WIN STREAM)

          (* * RDATA Length)

    (for I from 0 to (\WIN STREAM) do (BIN STREAM])

)

;; Functions to maintain the domain tree structure

(DECLARE%: EVAL@COMPILE

(DATATYPE DOMAIN.TREE.NODE ((NAME POINTER)                            (* The name of this domain)
                            (SUBDOMAINS POINTER)                      (* List of domains inferior to this one)
                            (SUPERDOMAIN POINTER)                     (* The domain of which this domain is a part)
                            (NAMESERVERS POINTER)                     (* The list of designated name servers for this domain)
                            ))

(RECORD DOMAIN.SERVER (NAME ADDRESSES EXPIRATION.DATE FOR.DOMAINS AVG.SVC.TIME)
       AVG.SVC.TIME _ 0)
)

(/DECLAREDATATYPE 'DOMAIN.TREE.NODE '(POINTER POINTER POINTER POINTER)
```

```
        ;; ---field descriptor list elided by lister---
        '8)

(/DECLAREDATATYPE 'DOMAIN.TREE.NODE '(POINTER POINTER POINTER POINTER)
        ;; ---field descriptor list elided by lister---
        '8)

(DEFINEQ
```

(USTRINGHASHBITS

```
  [LAMBDA (STRING)                                          (* ejs%: " 5-Nov-86 13:20")
    (for C inthinstring (MKSTRING STRING) bind (HASHBITS _ 0)
       do [SETQ HASHBITS (IPLUS16 (ELT UPPERCASEARRAY C)
                                  (IPLUS16 (SETQ HASHBITS (IPLUS16 HASHBITS (LLSH (LOGAND HASHBITS 4095)
                                                                                  2)))
                                           (LLSH (LOGAND HASHBITS 255)
                                                 8]
       finally (RETURN HASHBITS])

)

(RPAQ? \DOMAIN.ROOT (create DOMAIN.TREE.NODE NAME _ ""))

(RPAQ? \DOMAIN.NAMESERVERS (HASHARRAY 50 1.2 (FUNCTION USTRINGHASHBITS)
                                  (FUNCTION STRING-EQUAL)))

(RPAQ? \DOMAIN.UNKNOWN.DOMAINS )

(RPAQ? \DOMAIN.GC.INTERVAL 600000)

(RPAQ? \DOMAIN.GC.TIMER (SETUPTIMER \DOMAIN.GC.INTERVAL))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \DOMAIN.ROOT \DOMAIN.NAMESERVERS \DOMAIN.UNKNOWN.DOMAINS \DOMAIN.GC.TIMER \DOMAIN.GC.INTERVAL)
)

(DEFINEQ
```

(\DOMAIN.ADD.NEW.DOMAIN

```
  [LAMBDA (NODE DOMAIN NAMESERVER ADDRESSES TTL)            (* ejs%: "25-Apr-86 12:25")

          (* * Add DOMAIN as a subdomain of NODE, with name service by NAMESERVER, at addresses ADDRESSES, with
          expiration TTL seconds from now)

    (LET ((SUBDOMAIN (create DOMAIN.TREE.NODE
                             SUPERDOMAIN _ NODE
                             NAME _ DOMAIN)))
         (push (fetch (DOMAIN.TREE.NODE SUBDOMAINS) of NODE)
               SUBDOMAIN)
         [COND
            (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE)
                   (printout DOMAIN.TRACE.FILE "Adding " DOMAIN " as subdomain of " (\DOMAIN.NAME NODE]
         (COND
            (NAMESERVER                                     (* Add name server information to new subdomain)
                   (COND
                      (DOMAIN.TRACE.FLG (printout DOMAIN.TRACE.FILE " with name server " NAMESERVER)))
                   (\DOMAIN.ADD.NAMESERVER SUBDOMAIN NAMESERVER ADDRESSES TTL])
```

(\DOMAIN.ADD.NAMESERVER

```
  [LAMBDA (NODE NAMESERVER ADDRESSES TTL)                   (* ejs%: "25-Apr-86 12:34")

          (* * Function called to add name server information to a node in the domain tree.
          If ADDRESSES is NIL, this function will query the internet to resolve the information)

    (COND
       (NAMESERVER (LET [(DOMAIN.SERVER (OR (GETHASH NAMESERVER \DOMAIN.NAMESERVERS)
                                            (PUTHASH NAMESERVER
                                                   (create DOMAIN.SERVER
                                                          NAME _ NAMESERVER
                                                          ADDRESSES _ ADDRESSES
                                                          EXPIRATION.DATE _ (IPLUS (IDATE)
                                                                                   (OR (NUMBERP TTL)
                                                                                       3600)))
                                                   \DOMAIN.NAMESERVERS]
                         [COND
                            ([AND (NULL ADDRESSES)
                                  (NULL (SETQ ADDRESSES (fetch (DOMAIN.SERVER ADDRESSES) of DOMAIN.SERVER]
                             (SETQ ADDRESSES (replace (DOMAIN.SERVER ADDRESSES) of DOMAIN.SERVER
                                                with (OR ADDRESSES (DOMAIN.LOOKUP.ADDRESS NAMESERVER NIL T]
                         (COND
                            [ADDRESSES (COND
                                          ((NOT (for SERVER in (fetch (DOMAIN.TREE.NODE NAMESERVERS) of NODE)
                                                   thereis (STRING-EQUAL SERVER NAMESERVER)))
                                           [COND
```

```
                                        (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE)
                                               (printout DOMAIN.TRACE.FILE "Adding " NAMESERVER " as new
                                                      name server for " (\DOMAIN.NAME NODE]
                                           (push (fetch (DOMAIN.TREE.NODE NAMESERVERS) of NODE)
                                                 NAMESERVER)
                                           (push (fetch (DOMAIN.SERVER FOR.DOMAINS) of DOMAIN.SERVER)
                                                 NODE]
                                 (T (PUTHASH NAMESERVER NIL \DOMAIN.NAMESERVERS])
```

(\DOMAIN.AUGMENT.TREE

```
[LAMBDA (RRLST)                                                    (* ejs%: "14-Nov-86 14:30")
```

(* * RRLST is a list of RRTYPE.NS and/or RRTYPE.A records.
Build up our model of the internet domain tree by processing the information in RRLST)

```
(bind NAMESERVER for RR in RRLST do (COND
                                       ((EQ (LISTGET RR 'TYPE)
                                            RRTYPE.NS)
                                        (SETQ NAMESERVER (LISTGET RR 'DATA))
                                        (\DOMAIN.INSERT.IN.TREE (LISTGET RR 'NAME)
                                               NAMESERVER
                                               (\DOMAIN.SEARCH.RESOURCE.LIST RRLST NAMESERVER RRTYPE.A NIL)
                                               (LISTGET RR 'TTL])
```

(\DOMAIN.CHOOSE.BEST.SERVERS

```
[LAMBDA (DOMAIN)                                                   (* ejs%: " 1-May-86 17:15")
```

(* * This function chooses the best servers for a query to resolve DOMAIN)

```
(LET* [(PATH (COND
                ((AND (NLISTP DOMAIN)
                      DOMAIN)
                 (DREVERSE (\DOMAIN.PARSE.NAME DOMAIN)))
                (T DOMAIN)))
       (BEST.CHOICE (bind NEXT (CURRENT _ \DOMAIN.ROOT) for NAME in PATH
                          while [SETQ NEXT (for SUBDOMAIN in (fetch (DOMAIN.TREE.NODE SUBDOMAINS) of CURRENT)
                                              thereis (STRING-EQUAL NAME (fetch (DOMAIN.TREE.NODE NAME)
                                                                                of SUBDOMAIN]
                          do (SETQ CURRENT NEXT) finally (RETURN CURRENT]
      [while BEST.CHOICE do (COND
                               ((fetch (DOMAIN.TREE.NODE NAMESERVERS) of BEST.CHOICE)
                                (RETURN))
                               (T (SETQ BEST.CHOICE (fetch (DOMAIN.TREE.NODE SUPERDOMAIN) of BEST.CHOICE]
      [COND
         ((EQ BEST.CHOICE \DOMAIN.ROOT)
```

(* Here we have a problem. Is the request for a subdomain of ROOT
(e.g. COM, GOV, EDU, etc)%, or for a local name in our own domain?)

```
          (COND
             [(AND (EQLENGTH PATH 1)
                   (for SUBDOMAIN in (fetch (DOMAIN.TREE.NODE SUBDOMAINS) of \DOMAIN.ROOT)
                      thereis (STRING-EQUAL (CAR PATH)
                                            (fetch (DOMAIN.TREE.NODE NAME) of SUBDOMAIN]
             (T
```

(* Heuristic%: If the domain doesn't appear to be a subdomain of the root, assume that the local domain server will know it.
If we're wrong, the local name server will tell us)

```
                (SETQ BEST.CHOICE NIL]
      (COND
         [(NULL BEST.CHOICE)
          (COND
             ((OR (EQLENGTH PATH 1)
                  (NULL (fetch (DOMAIN.TREE.NODE NAMESERVERS) of \DOMAIN.ROOT)))
              (COND
                 (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE)
                        (printout DOMAIN.TRACE.FILE "Best choice for " DOMAIN " is our local server: "
                                  \DOMAIN.DEFAULT.SERVER)))
              (SORT (MKLIST \DOMAIN.DEFAULT.SERVER)
                    (FUNCTION \DOMAIN.SORT.BY.SVC.TIME)))
             (T (COND
                   (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE)
                           (printout DOMAIN.TRACE.FILE "Best choice for " DOMAIN " is the root server")))
                (SORT (fetch (DOMAIN.TREE.NODE NAMESERVERS) of \DOMAIN.ROOT)
                      (FUNCTION \DOMAIN.SORT.BY.SVC.TIME]
         (T [COND
               (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE)
                      (printout DOMAIN.TRACE.FILE "Best choice(s) for " DOMAIN ": " (fetch (DOMAIN.TREE.NODE
                                                                                           NAMESERVERS)
                                                                                      of BEST.CHOICE]
            (SORT (fetch (DOMAIN.TREE.NODE NAMESERVERS) of BEST.CHOICE)
                  (FUNCTION \DOMAIN.SORT.BY.SVC.TIME])
```

(\DOMAIN.FIND.DOMAIN.IN.TREE

```
[LAMBDA (NAME)                                                    (* ejs%: "13-Apr-86 01:25")
  (COND
     ((STREQUAL NAME "")
      \DOMAIN.ROOT)
     (T (LET ([PATH (COND
                       ((LISTP NAME)
                        (REVERSE NAME))
                       (T (DREVERSE (\DOMAIN.PARSE.NAME NAME]
              (CURRENT \DOMAIN.ROOT))
             (bind NEXT for NODE on PATH do (COND
                                               ([NOT (SETQ NEXT (for SUBDOMAIN in (fetch (DOMAIN.TREE.NODE
                                                                                           SUBDOMAINS)
                                                                                    of CURRENT)
                                                                  thereis (STRING-EQUAL (CAR NODE)
                                                                                        (fetch (DOMAIN.TREE.NODE NAME)
                                                                                           of SUBDOMAIN]
                                                (RETURN (CONS CURRENT NODE)))
                                               (T (SETQ CURRENT NEXT)))
               finally (RETURN CURRENT])
```

(\DOMAIN.INIT

```
[LAMBDA (EVENT)                                                   (* ejs%: " 1-May-86 15:46")
  (SETQ \DOMAIN.DEFAULT.SERVER (bind NAME for SERVER in (fetch (IPINIT DOMAIN.SERVERS) of
                                                                                    \IP.DEFAULT.CONFIGURATION
                                                          )
                                  as SERVER# from 1
                                  collect (SETQ NAME (CONCAT "Local-Domain-Server-" SERVER#))
                                          (PUTHASH NAME (create DOMAIN.SERVER
                                                               NAME _ NAME
                                                               EXPIRATION.DATE _ MAX.FIXP
                                                               ADDRESSES _ (LIST (\IP.READ.STRING.ADDRESS SERVER
                                                                                 )))
                                          \DOMAIN.NAMESERVERS)
                                  NAME])
```

(\DOMAIN.INSERT.IN.TREE

```
[LAMBDA (DOMAIN NAMESERVER ADDRESSES TTL)                         (* ejs%: "25-Apr-86 12:21")

          (* * Given information from an RRTYPE.NS record, add an entry to the domain tree)

   (LET ((PARTIAL.PATH (\DOMAIN.FIND.DOMAIN.IN.TREE DOMAIN)))
        (COND
           ((type? DOMAIN.TREE.NODE PARTIAL.PATH)                 (* Found it)
            (\DOMAIN.ADD.NAMESERVER PARTIAL.PATH NAMESERVER ADDRESSES TTL))
           ((EQLENGTH PARTIAL.PATH 2)                             (* Only one away from previous knowledge?)
            (\DOMAIN.ADD.NEW.DOMAIN (CAR PARTIAL.PATH)
                   (CADR PARTIAL.PATH)
                   NAMESERVER ADDRESSES TTL))
           (T                                                     (* Some number of domains between our deepest knowledge
                                                                     and the desired domain)
              (\DOMAIN.ADD.NEW.DOMAIN (CAR PARTIAL.PATH)
                     (CADR PARTIAL.PATH))
              (\DOMAIN.INSERT.IN.TREE DOMAIN NAMESERVER ADDRESSES TTL])
```

(\DOMAIN.PATH

```
[LAMBDA (DOMAIN.TREE.NODE)                                        (* ejs%: "13-Apr-86 14:44")

          (* * Generate a list of domain names along the path to the root of the domain tree)

   (COND
      ((NULL (fetch (DOMAIN.TREE.NODE SUPERDOMAIN) of DOMAIN.TREE.NODE))
       NIL)
      (T (CONS (fetch (DOMAIN.TREE.NODE NAME) of DOMAIN.TREE.NODE)
               (\DOMAIN.PATH (fetch (DOMAIN.TREE.NODE SUPERDOMAIN) of DOMAIN.TREE.NODE])
```

(\DOMAIN.SEARCH.RESOURCE.LIST

```
[LAMBDA (RRLST NAME TYPE OK.TO.RETURN.NAME)                       (* ejs%: "14-Nov-86 14:40")
   (LET [(ANSWER (bind DATA for RR in RRLST collect (SETQ DATA (LISTGET RR 'DATA))
                                                    (COND
                                                       ((AND DATA (EQ TYPE RRTYPE.A)
                                                             OK.TO.RETURN.NAME)
                                                        (LISTGET RR 'NAME))
                                                       (T DATA))
                                 when (AND (EQ TYPE (LISTGET RR 'TYPE))
                                           (DOMAIN.NAME.EQUAL (LISTGET RR 'NAME)
                                                  NAME]
        (COND
           (ANSWER)
           (T (LET [(CANONICAL.NAME (bind FOUNDIT DATA for RR in RRLST
                                       thereis (AND (EQ RRTYPE.CNAME (LISTGET RR 'TYPE))
                                                    (DOMAIN.NAME.EQUAL (LISTGET RR 'NAME)
                                                           NAME)
```

```
                                        (SETQ FOUNDIT T))
                                    finally (RETURN (AND FOUNDIT (LISTGET RR 'DATA]
                    (COND
                       (CANONICAL.NAME (\DOMAIN.SEARCH.RESOURCE.LIST RRLST CANONICAL.NAME TYPE
                                             OK.TO.RETURN.NAME])
```

(\DOMAIN.DELETE.NAMESERVER

```
[LAMBDA (NAMESERVER)                                                    (* ejs%: "13-Apr-86 18:35")
  (LET ((DOMAIN.SERVER (GETHASH NAMESERVER \DOMAIN.NAMESERVERS)))
       (COND
          (DOMAIN.SERVER [bind NAMESERVERS for DOMAIN in (fetch (DOMAIN.SERVER FOR.DOMAINS) of DOMAIN.SERVER)
                            do (SETQ NAMESERVERS (fetch (DOMAIN.TREE.NODE NAMESERVERS) of DOMAIN))
                               (bind for NAME in NAMESERVERS when (STRING-EQUAL NAME NAMESERVER)
                                  do (replace (DOMAIN.TREE.NODE NAMESERVERS) of DOMAIN with (DREMOVE NAME
                                                                                                    NAMESERVERS]
                  (PUTHASH NAMESERVER NIL \DOMAIN.NAMESERVERS])
```

(\DOMAIN.AROUND.EXIT

```
[LAMBDA (EVENT)                                                         (* ejs%: "13-Apr-86 18:30")
  (SELECTQ EVENT
       ((NIL AFTERLOGOUT AFTERSYSOUT AFTERMAKESYS AFTERSAVEVM)
            (\DOMAIN.DELETE.TREE))
       NIL])
```

(\DOMAIN.DELETE.TREE

```
[LAMBDA NIL                                                             (* ejs%: "13-Apr-86 17:39")

          (* * Undoes circularity in pointers between levels of the tree)

  (bind (OPEN _ (LIST \DOMAIN.ROOT))
        CLOSED CURRENT while OPEN do (SETQ CURRENT (pop OPEN))
                                     (SETQ OPEN (APPEND (fetch (DOMAIN.TREE.NODE SUBDOMAINS) of CURRENT)
                                                        OPEN))
                                     (replace (DOMAIN.TREE.NODE SUBDOMAINS) of CURRENT with NIL)
                                     (replace (DOMAIN.TREE.NODE NAME) of CURRENT with NIL)
                                     (replace (DOMAIN.TREE.NODE NAMESERVERS) of CURRENT with NIL)
                                     (replace (DOMAIN.TREE.NODE SUPERDOMAIN) of CURRENT with NIL))
  [MAPHASH \DOMAIN.NAMESERVERS (FUNCTION (LAMBDA (DOMAIN.SERVER NAME)
                                           (replace (DOMAIN.SERVER FOR.DOMAINS) of DOMAIN.SERVER with NIL]
  (CLRHASH \DOMAIN.NAMESERVERS)
  NIL])
```

(\DOMAIN.BACKGROUND

```
[LAMBDA NIL                                                             (* ejs%: "13-Apr-86 18:24")
  (COND
     ((TIMEREXPIRED? \DOMAIN.GC.TIMER)
      (\DOMAIN.GC.NAMESERVERS)
      (SETQ \DOMAIN.GC.TIMER (SETUPTIMER \DOMAIN.GC.INTERVAL \DOMAIN.GC.TIMER])
```

(\DOMAIN.GC.NAMESERVERS

```
[LAMBDA NIL                                                             ; Edited 11-Feb-89 12:36 by akw:

          (* * This function maps over the name server hash array, and removes old servers which have timed out)

  (LET ((TIME (IDATE)))
       (DECLARE (SPECVARS TIME))
       [MAPHASH \DOMAIN.NAMESERVERS (FUNCTION (LAMBDA (DOMAIN.SERVER NAME)
                                                (DECLARE (USEDFREE TIME))
                                                (COND
                                                   ((MEMBER NAME (fetch (IPINIT DOMAIN.SERVERS) of
                                                                                              \IP.DEFAULT.CONFIGURATION
                                                                 ))
                                                    T)
                                                   ((ILESSP (fetch (DOMAIN.SERVER EXPIRATION.DATE) of
                                                                                                   DOMAIN.SERVER
                                                                )
                                                            TIME)
                                                    (COND
                                                       (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE)
                                                              (printout DOMAIN.TRACE.FILE "Name server " NAME
                                                                     " has expired; deleting...")))
                                                    (\DOMAIN.DELETE.NAMESERVER NAME]
       NIL])
```

(\DOMAIN.SORT.BY.SVC.TIME

```
[LAMBDA (NAME1 NAME2)                                                   (* ejs%: "13-Apr-86 18:14")
  (LET ((R1 (GETHASH NAME1 \DOMAIN.NAMESERVERS))
        (R2 (GETHASH NAME2 \DOMAIN.NAMESERVERS)))
       (ILESSP (OR (fetch (DOMAIN.SERVER AVG.SVC.TIME) of R1)
                   0)
               (OR (fetch (DOMAIN.SERVER AVG.SVC.TIME) of R2)
```

```
                    0])

)

(ADDTOVAR BACKGROUNDFNS \DOMAIN.BACKGROUND)

;; Programmer's interface

(RPAQ? DOMAIN.TRACE.FLG )

(RPAQ? DOMAIN.TRACE.FILE )

(RPAQ? INTERNET.LOCAL.DOMAIN )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS DOMAIN.TRACE.FLG DOMAIN.TRACE.FILE INTERNET.LOCAL.DOMAIN)
)

(DEFINEQ

(DOMAIN.INIT
  [LAMBDA NIL                                                    ; Edited 15-Feb-88 17:26 by Snow
    ;; Called to initialize the domain service for this host
    (DECLARE (GLOBALVARS \DOMAIN.DEFAULT.SERVER INTERNET.LOCAL.DOMAIN))
    (if (NOT \IP.DEFAULT.CONFIGURATION)
        then (PROMPTPRINT "Internet domain code is loaded, but disabled.")
      else (LET [(LOCAL.DOMAIN.SERVERS (fetch (IPINIT DOMAIN.SERVERS) of \IP.DEFAULT.CONFIGURATION))
                 (LOCAL.DOMAIN (MKSTRING (fetch (IPINIT LOCAL.DOMAIN) of \IP.DEFAULT.CONFIGURATION]
                (COND
                   ((AND LOCAL.DOMAIN.SERVERS LOCAL.DOMAIN)
                    (SETQ \DOMAIN.DEFAULT.SERVER (for ADDR inside LOCAL.DOMAIN.SERVERS collect (MKSTRING ADDR)))
                    (SETQ INTERNET.LOCAL.DOMAIN LOCAL.DOMAIN)
                    (for NAMESERVER in LOCAL.DOMAIN.SERVERS do (\DOMAIN.INSERT.IN.TREE LOCAL.DOMAIN (MKSTRING
                                                                                                      NAMESERVER
                                                                                                       )
                                                                                          (LIST (DODIP.HOSTP NAMESERVER))
                                                                                          MAX.FIXP)))
                   (T (PROMPTPRINT "Internet domain code is loaded, but disabled."]))

(DOMAIN.LOOKUP.ADDRESS
  [LAMBDA (NAME SERVER DONT.GET.OSTYPE)                          ; Edited 15-Feb-89 15:14 by welch

          (* * Programmer's interface to lookup IP Internet host name using the domain system)

    (bind (OPEN _ (OR (MKLIST SERVER)
                      (\DOMAIN.CHOOSE.BEST.SERVERS NAME)))
          CANONICAL.NAME CLOSED ADDRESSES THIS.SERVER ANSWER OSTYPE (ATOMIC-NAME _ (MKATOM (U-CASE NAME)))
      while OPEN do (SETQ THIS.SERVER (pop OPEN))
                    (push CLOSED THIS.SERVER)
                    (SETQ ANSWER (DOMAIN.LOOKUP NAME RRTYPE.A THIS.SERVER))
                    (COND
                       ((SETQ ADDRESSES (\DOMAIN.SEARCH.RESOURCE.LIST ANSWER NAME RRTYPE.A))
                        (\DOMAIN.AUGMENT.TREE ANSWER)
                        [SETQ OSTYPE (COND
                                        (DONT.GET.OSTYPE NIL)
                                        (T (DOMAIN.LOOKUP.OSTYPE NAME]
                        (PUTHASH ATOMIC-NAME (create HOSTS.TXT.ENTRY
                                                     HTE.TYPE _ 'HOST
                                                     HTE.ADDRESSES _ ADDRESSES
                                                     HTE.NAMES _ (LIST ATOMIC-NAME)
                                                     HTE.OS.TYPE _ OSTYPE)
                               \IP.HOSTNAMES)
                        (RETURN ADDRESSES))
                       (ANSWER (COND
                                  ([SETQ CANONICAL.NAME (MKATOM (U-CASE (\DOMAIN.SEARCH.FOR.CANONICAL.NAME
                                                                          NAME ANSWER]
                                   (SETQ ADDRESSES (DOMAIN.LOOKUP.ADDRESS CANONICAL.NAME SERVER))
                                   (PUTHASH ATOMIC-NAME (GETHASH CANONICAL.NAME \IP.HOSTNAMES)
                                          \IP.HOSTNAMES)
                                   (RETURN ADDRESSES))
                                  (T (\DOMAIN.AUGMENT.TREE ANSWER)
                                     (SETQ OPEN (APPEND (for NEXT.SERVER in (\DOMAIN.SEARCH.RESOURCE.LIST
                                                                              ANSWER
                                                                              '* RRTYPE.A T)
                                                           when (NOT (MEMBER NEXT.SERVER CLOSED)) collect
                                                                                                   NEXT.SERVER
                                                         )
                                                        OPEN]))

(DOMAIN.LOOKUP.NAMESERVER
  [LAMBDA (NAME SERVER)                                          (* ejs%: "25-Apr-86 12:55")

          (* * Programmer's interface to lookup IP Internet host name using the domain system)
```

```
    (bind (OPEN _ (OR (MKLIST SERVER)
                      (\DOMAIN.CHOOSE.BEST.SERVERS NAME)))
          CLOSED NAMESERVERS THIS.SERVER ANSWER while OPEN
       do (SETQ THIS.SERVER (pop OPEN))
          (push CLOSED THIS.SERVER)
          (SETQ ANSWER (DOMAIN.LOOKUP NAME RRTYPE.NS THIS.SERVER))
          (COND
             ((SETQ NAMESERVERS (\DOMAIN.SEARCH.RESOURCE.LIST ANSWER NAME RRTYPE.NS))
              (\DOMAIN.AUGMENT.TREE ANSWER)
              (RETURN NAMESERVERS))
             (ANSWER (\DOMAIN.AUGMENT.TREE ANSWER)
                    (SETQ OPEN (APPEND (for NEXT.SERVER in (\DOMAIN.SEARCH.RESOURCE.LIST ANSWER '* RRTYPE.A T)
                                          when (NOT (MEMBER NEXT.SERVER CLOSED)) collect NEXT.SERVER)
                                       OPEN])
```

(DOMAIN.LOOKUP.OSTYPE

```
  [LAMBDA (NAME SERVER)                                          (* ejs%: "14-Nov-86 14:46")
```

(* * Programmer's interface to lookup IP Internet host name using the domain system)

```
    (bind (OPEN _ (OR (MKLIST SERVER)
                      (\DOMAIN.CHOOSE.BEST.SERVERS NAME)))
          CANONICAL.NAME CLOSED CPU.OSTYPES THIS.SERVER ANSWER while OPEN
       do (SETQ THIS.SERVER (pop OPEN))
          (push CLOSED THIS.SERVER)
          (SETQ ANSWER (DOMAIN.LOOKUP NAME RRTYPE.HINFO THIS.SERVER))
          (COND
             [(SETQ CPU.OSTYPES (\DOMAIN.SEARCH.RESOURCE.LIST ANSWER NAME RRTYPE.HINFO))
              (\DOMAIN.AUGMENT.TREE ANSWER)
              (RETURN (MKATOM (U-CASE (CDAR CPU.OSTYPES]
             (ANSWER (COND
                        ((SETQ CANONICAL.NAME (\DOMAIN.SEARCH.FOR.CANONICAL.NAME NAME ANSWER))
                         (RETURN (DOMAIN.LOOKUP.OSTYPE CANONICAL.NAME SERVER)))
                        (T (\DOMAIN.AUGMENT.TREE ANSWER)
                           (SETQ OPEN (APPEND (for NEXT.SERVER in (\DOMAIN.SEARCH.RESOURCE.LIST ANSWER
                                                                                '* RRTYPE.A T)
                                                 when (NOT (MEMBER NEXT.SERVER CLOSED)) collect NEXT.SERVER)
                                              OPEN])
```

(DOMAIN.LOOKUP

```
  [LAMBDA (NAME TYPE SERVER)                                     ; Edited 15-Feb-88 17:24 by Snow
```

;;; Programmer's interface to lookup IP Internet host name using the domain system

```
    (PROG ((DOMAIN.PATH (\DOMAIN.PARSE.NAME NAME))
           (RETRYCOUNT 0)
           ANSWER ADDRESS TIMINGFLG START.TIME)
          (OR TYPE (SETQ TYPE RRTYPE.A))
          [COND
             [(LISTP SERVER)
              (SETQ ADDRESS (COND
                               [(LISTP (CAR SERVER))            ; Handles a list of DOMAIN.SERVER records
                                (CAR (fetch (DOMAIN.SERVER ADDRESSES) of (CAR SERVER]
                               (T                               ; Handles a list of addresses
                                  (CAR SERVER]
             (SERVER                                            ; Handles a single address
                    (SETQ ADDRESS SERVER))
             (T (SETQ SERVER \DOMAIN.DEFAULT.SERVER)
                (SETQ ADDRESS (CAR SERVER]
          [COND
             ((STRINGP ADDRESS)
              (SETQ ADDRESS (CAR (fetch (DOMAIN.SERVER ADDRESSES) of (SETQ TIMINGFLG (GETHASH ADDRESS
                                                                                      \DOMAIN.NAMESERVERS]
          (COND
             (DOMAIN.TRACE.FLG (FRESHLINE DOMAIN.TRACE.FILE)
                    (PRIN1 "Type " DOMAIN.TRACE.FILE)
                    (PRINTCONSTANT TYPE DOMAIN.RRTYPES DOMAIN.TRACE.FILE)
                    (printout DOMAIN.TRACE.FILE " query to " (COND
                                                               ((NUMBERP SERVER)
                                                                (\IP.ADDRESS.TO.STRING SERVER))
                                                               (T SERVER))
                              " for " NAME T)))
      LOOP
          (add RETRYCOUNT 1)
          [COND
             (TIMINGFLG (SETQ START.TIME (IDATE]
          [SETQ ANSWER (COND
                          ((NULL ANSWER)
                           (\UDPDOM.QUERY DOMAIN.PATH TYPE CLASSTYPE.IN ADDRESS))
                          ((EQ ANSWER 'USE.TCP)
                           (\TCPDOM.QUERY DOMAIN.PATH TYPE CLASSTYPE.IN ADDRESS]
          [COND
             (TIMINGFLG (replace (DOMAIN.SERVER AVG.SVC.TIME) of TIMINGFLG with (IDIFFERENCE (IDATE)
                                                                                   START.TIME]
```

```
          (COND
             ((LITATOM ANSWER)
              (SELECTQ ANSWER
                   (NIL (COND
                          ((LISTP SERVER)
                           (SETQ SERVER (CDR SERVER))
                           [SETQ ADDRESS (CAR (fetch (DOMAIN.SERVER ADDRESSES) of (CAR SERVER]
                           (SETQ RETRYCOUNT 0)
                           (GO LOOP))
                          (T (RETURN ANSWER))))
                   (NAME.ERROR (RETURN NIL))
                   (USE.TCP (COND
                              ((EQ RETRYCOUNT 1)
                               (GO LOOP))
                              (T (RETURN NIL))))
                   (RETURN ANSWER)))
             (T (RETURN ANSWER])
```

(DOMAIN.GRAPH

```
[LAMBDA (WINDOW)                                        ; Edited 19-Mar-87 16:58 by FS
  (LET ((OPENLIST (LIST \DOMAIN.ROOT))
        NODELST)
       (bind NODE while OPENLIST do (SETQ NODE (pop OPENLIST))
                                    (push NODELST (create GRAPHNODE
                                                          NODELABEL _ (COND
                                                                        ((NULL (fetch (DOMAIN.TREE.NODE
                                                                                       SUPERDOMAIN)
                                                                                  of NODE))
                                                                         "*ROOT*")
                                                                        (T (fetch (DOMAIN.TREE.NODE NAME)
                                                                                  of NODE)))
                                                          NODEID _ NODE
                                                          TONODES _ (fetch (DOMAIN.TREE.NODE SUBDOMAINS)
                                                                           of NODE)))
                                    (SETQ OPENLIST (APPEND (fetch (DOMAIN.TREE.NODE SUBDOMAINS) of NODE)
                                                           OPENLIST)))
       (SHOWGRAPH (LAYOUTGRAPH NODELST (LIST \DOMAIN.ROOT)
                               'HORIZONTAL)
              WINDOW
              (FUNCTION (LAMBDA (NODE W)
                          (COND
                             (NODE (INSPECT (fetch (GRAPHNODE NODEID) of NODE)))
                             (T (DOMAIN.GRAPH W])
```

(DOMAIN.NAME.EQUAL

```
[LAMBDA (NAME1 NAME2)                                   (* ejs%: "13-Apr-86 17:23")
  (COND
     ((OR (EQ NAME1 '*)
          (EQ NAME2 '*))
      T)
     (T (OR (LISTP NAME1)
            (SETQ NAME1 (\DOMAIN.PARSE.NAME NAME1)))
        (OR (LISTP NAME2)
            (SETQ NAME2 (\DOMAIN.PARSE.NAME NAME2)))
        (COND
           ((OR (AND (NULL NAME1)
                     NAME2)
                (AND (NULL NAME2)
                     NAME1))
            NIL)
           (T (for X in NAME1 as Y in NAME2 always (STRING-EQUAL X Y])
```

(DOMAIN.TRACE

```
[LAMBDA (MODE)                                          (* ejs%: "13-Apr-86 16:12")
  [COND
     ((WINDOWP DOMAIN.TRACE.FILE)
      (OPENW DOMAIN.TRACE.FILE))
     (T (SETQ DOMAIN.TRACE.FILE (CREATEW NIL "Domain Trace File"))
        (DSPSCROLL 'ON DOMAIN.TRACE.FILE)
        (DSPFONT '(GACHA 8)
               DOMAIN.TRACE.FILE)
        (WINDOWPROP DOMAIN.TRACE.FILE 'BUTTONEVENTFN (FUNCTION DOMAIN.TRACEWINDOW.BUTTONFN))
        (WINDOWPROP DOMAIN.TRACE.FILE 'CLOSEFN (FUNCTION (LAMBDA NIL
                                                           (SETQ DOMAIN.TRACE.FLG NIL)
                                                           (SETQ DOMAIN.TRACE.FILE]
  (SETQ DOMAIN.TRACE.FLG MODE])
```

(DOMAIN.TRACEWINDOW.BUTTONFN

```
[LAMBDA (WINDOW)                                        (* ejs%: "13-Apr-86 15:49")
  (COND
     ((MOUSESTATE (NOT UP))
      (SETQ DOMAIN.TRACE.FLG (SELECTQ DOMAIN.TRACE.FLG
                                  (NIL T)
```

```
                                    (T NIL)
                                    NIL))
        (printout WINDOW T "[Tracing " (SELECTQ DOMAIN.TRACE.FLG
                                             (T "on")
                                             "off")
               "]" T])

)
```

(**DOMAIN.INIT**)

```
(PUTPROPS TCPDOMAIN COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1989 1990))
```

FUNCTION INDEX

CONSTANT INDEX

VARIABLE INDEX

RECORD INDEX