File created: 28-Apr-88 17:36:47 **{ERIS}<CUTTING>RPC>RPCCOMMON.;5**

changes to: (IL:FUNCTIONS REINITIALIZE-RPCSTREAM CLOSE-RPCSTREAM RPC-CREATE-UDP-STREAM RECLAIM-XDR-DATA-BLOCK
ALLOCATE-XDR-DATA-BLOCK XDR-PUT-CELL XDR-PUT-BYTES XDR-PUT-BYTE XDR-INITIALIZE-CACHE
XDR-GET-CELL XDR-GET-BYTES XDR-GET-BYTE OS-UDP-PUTOFFSET OS-UDP-GETOFFSET OS-UDP-PUTCELL
OS-UDP-GETCELL OS-UDP-PUTBYTES OS-UDP-PUTBYTE OS-UDP-GETBYTES OS-UDP-GETBYTE UNFOLD FOLDLO)
(IL:VARS IL:RPCCOMMONCOMS)
(IL:RECORDS XDR-DATA-BLOCK)
(IL:VARIABLES *WORDS-PER-CELL* *CELLS-PER-XDR-DATA-BLOCK* *MAX-XDR-DATA-BLOCKS*
FREE-XDR-DATA-BLOCKS)
(FILE-ENVIRONMENTS IL:RPCCOMMON)

previous date: 28-Apr-88 17:22:33 {ERIS}<CUTTING>RPC>RPCCOMMON.;4

Read Table: XCL

Package: RPC2

Format: XCCS


```
(IL:RPAQQ IL:RPCCOMMONCOMS
          ((IL:PROPS (IL:RPCCOMMON IL:MAKEFILE-ENVIRONMENT IL:FILETYPE))
           (IL:P (EXPORT '(DEFINE-REMOTE-PROGRAM UNDEFINE-REMOTE-PROGRAM REMOTE-PROCEDURE-CALL
                                CREATE-UNIX-AUTHENTICATION SETUP-RPC PERFORM-RPC OPEN-RPCSTREAM CLOSE-RPCSTREAM
                                LIST-REMOTE-PROGRAMS INSPECT-STRING INSPECT-PACKET INSPECT-STRING1))
                 (EXPORT '(*DEBUG* *COMPILE-XDR-CODE* *RPC-PROGRAMS* *MSEC-UNTIL-TIMEOUT* *MSEC-BETWEEN-TRIES*
                                *RPC-OK-TO-CACHE* *RPC-SOCKET-CACHE* *RPC-WELL-KNOWN-SOCKETS*
                                *BYTES-PER-RM-OUTREC*)))
           (IL:VARIABLES *BYTES-PER-RM-OUTREC* *RPC-GETCELL-TEMP*)
           (IL:FUNCTIONS FORMAT-T)
           ;; Other Utilities
           (IL:FUNCTIONS LIST-REMOTE-PROGRAMS FIND-RPC-PROGRAM FIND-RPC-TYPEDEF FIND-RPC-TYPENAME
                  FIND-RPC-PROCEDURE FIND-XDR-CONSTANT INSPECT-STRING INSPECT-STRING1 INSPECT-PACKET)

;;; RPC Streams

           (IL:FUNCTIONS OPEN-RPCSTREAM CLOSE-RPCSTREAM REINITIALIZE-RPCSTREAM)
           (IL:FUNCTIONS GETBYTE GETBYTES GETCELL GETOFFSET PUTBYTE PUTBYTES PUTCELL GETUNSIGNED PUTUNSIGNED
                  PUTOFFSET)

;;; UDP Protocol RPC Streams

           (IL:FUNCTIONS RPC-CREATE-UDP-STREAM UDP-GETBYTE UDP-PUTBYTE UDP-GETCELL UDP-PUTCELL UDP-GETOFFSET
                  UDP-PUTOFFSET UDP-PUTBYTES UDP-GETBYTES)

;;; TCP Protocol RPC Streams

           (IL:FUNCTIONS RPC-CREATE-TCP-STREAM TCP-GETBYTE TCP-GETBYTES TCP-PUTBYTES TCP-PUTBYTE TCP-GETCELL
                  TCP-PUTCELL RM-FORCEOUTPUT RM-INITIALIZE-OUTSTREAM RM-INITIALIZE-INSTREAM RM-NEW-INPUT-RECORD)

;;; String RPC Stream

           (IL:FUNCTIONS CREATE-STRING-RPC-STREAM)

;;; TTY RPC Stream

           (IL:FUNCTIONS CREATE-TTY-RPC-STREAM)))

(IL:PUTPROPS IL:RPCCOMMON IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "RPC2"))

(IL:PUTPROPS IL:RPCCOMMON IL:FILETYPE :COMPILE-FILE)

(EXPORT '(DEFINE-REMOTE-PROGRAM UNDEFINE-REMOTE-PROGRAM REMOTE-PROCEDURE-CALL CREATE-UNIX-AUTHENTICATION
               SETUP-RPC PERFORM-RPC OPEN-RPCSTREAM CLOSE-RPCSTREAM LIST-REMOTE-PROGRAMS INSPECT-STRING
               INSPECT-PACKET INSPECT-STRING1))

(EXPORT '(*DEBUG* *COMPILE-XDR-CODE* *RPC-PROGRAMS* *MSEC-UNTIL-TIMEOUT* *MSEC-BETWEEN-TRIES* *RPC-OK-TO-CACHE*
               *RPC-SOCKET-CACHE* *RPC-WELL-KNOWN-SOCKETS* *BYTES-PER-RM-OUTREC*))

(DEFCONSTANT *BYTES-PER-RM-OUTREC* 8192
   "Size of string in which to store outgoing RPC/RM/TCP/IP messages fragments.")

(DEFGLOBALVAR *RPC-GETCELL-TEMP* " Á  "
   "A string into which TCP-GETCELL reads four bytes for puting together as an integer.")

(DEFMACRO FORMAT-T (&REST ARGS)
   "
```

```
   Use in low-level code in place of (FORMAT T ...) to avoid disaster.

   The problem is that Xerox Common Lisp, when given (FORMAT <stream> ...), rebinds
   *STANDARD-OUTPUT* to <stream> under the assumption that none of the
   implementation of FORMAT will ever use *STANDARD-OUTPUT*.  Thus, if you try to
   write to *STANDARD-OUTPUT* in any code called by COMMON LISP I/O routines writing
   to another stream, the output goes into the other stream rather than the original
   *STANDARD-OUTPUT*.  This routine is a quick fix for a lot of (FORMAT T ...) calls to
   send their output to *DEBUG-IO*, which is where the output should have gone in the
    first place.
   "
   `(FORMAT *DEBUG-IO* ,@ARGS))
```

;; Other Utilities

```
(DEFUN LIST-REMOTE-PROGRAMS ()
   ;; Return list of (name number version protocol)  for each defined remote program.
   (MAP 'LIST #'(LAMBDA (R)
                     (LIST (RPC-PROGRAM-NAME R)
                           (RPC-PROGRAM-NUMBER R)
                           (RPC-PROGRAM-VERSION R)
                           (RPC-PROGRAM-PROTOCOL R)))
        *RPC-PROGRAMS*))

(DEFUN FIND-RPC-PROGRAM (&KEY NUMBER VERSION NAME PROTOCOL)
   "
   Returns the RPC-PROGRAM struct for the given identifiers from among
   all the remote programs known to RPC2:*RPC-PROGRAMS*.

   Name is assumed to uniquely specify the program. If NAME is specified,
   then NUMBER, VERSION, and PROTOCOL are ignored.

   If NAME is not specified, then VERSION defaults to the highest existing version,
   and if PROTOCOL is specified, it must match the PROTOCOL of any remote program
   found.
   "
   (COND
      (NAME (FIND-IF #'(LAMBDA (RPC)
                           (EQL (RPC-PROGRAM-NAME RPC)
                                NAME))
                     *RPC-PROGRAMS*))
      (NUMBER (FIND-IF #'(LAMBDA (RPC)
                               (AND (EQL (RPC-PROGRAM-NUMBER RPC)
                                         NUMBER)
                                    (SETQ VERSION (OR VERSION (DO ((RPC *RPC-PROGRAMS* (CDR RPC))
                                                                   (LATEST)
                                                                   (LATEST-VERSION 0))
                                                                  ((NULL RPC)
                                                                   LATEST)
                                                                (IF (AND (EQL NUMBER (RPC-PROGRAM-NUMBER
                                                                                         (CAR RPC)))
                                                                         (EQL PROTOCOL (RPC-PROGRAM-PROTOCOL
                                                                                           (CAR RPC)))
                                                                         (> (RPC-PROGRAM-VERSION (CAR RPC))
                                                                            LATEST-VERSION))
                                                                    (SETQ LATEST (CAR RPC)
                                                                          LATEST-VERSION
                                                                          (RPC-PROGRAM-VERSION (CAR RPC)))))))
                                    (EQL (RPC-PROGRAM-VERSION RPC)
                                         VERSION)
                                    (OR (NULL PROTOCOL)
                                        (EQL (RPC-PROGRAM-PROTOCOL RPC)
                                             PROTOCOL)))))
              *RPC-PROGRAMS*)
      (T (ERROR "Invalid RPC Program Specifier Number: ~a Version: ~a Name: ~a Protocol: ~a" NUMBER VERSION NAME
                PROTOCOL))))

(DEFUN FIND-RPC-TYPEDEF (CONTEXT TYPE)
   "Returns the type definition for TYPE defined in RPC CONTEXT
   (CONTEXT may be a name or RPC-PROGRAM structure) if any, or else
   returns NIL.
   "
   (LET ((PRGSTR (ETYPECASE CONTEXT
                     (SYMBOL (FIND-RPC-PROGRAM :NAME CONTEXT))
                     (RPC-PROGRAM CONTEXT))))
        (SECOND (ASSOC TYPE (RPC-PROGRAM-TYPES PRGSTR)))))

(DEFUN FIND-RPC-TYPENAME (CONTEXT TYPE)
   "
   Returns TYPE, if TYPE defined in RPC CONTEXT
   (CONTEXT may be a name or RPC-PROGRAM structure) if any, or else
   returns NIL.
```

```
   "
   (LET ((PRGSTR (ETYPECASE CONTEXT
                   (SYMBOL (FIND-RPC-PROGRAM :NAME CONTEXT))
                   (RPC-PROGRAM CONTEXT))))
        (FIRST (ASSOC TYPE (RPC-PROGRAM-TYPES PRGSTR)))))

(DEFUN FIND-RPC-PROCEDURE (RPC-PROCS PROCID)
   "Finds (and returns) RPC-PROCEDURE structure specified by PROCID from among RPC-PROCS, a list of
   RPC-PROCEDURE structures. PROCID may be either an integer or a symbol.
    Áa  "
   (CTYPECASE PROCID ((INTEGER 0 *)
                      (FIND PROCID RPC-PROCS :KEY #'RPC-PROCEDURE-PROCNUM))
          ((AND SYMBOL (NOT NULL))
           (FIND PROCID RPC-PROCS :KEY #'RPC-PROCEDURE-NAME))))

(DEFUN FIND-XDR-CONSTANT (CONTEXT CONSTANT)
   "Find (and return) the constant definition for symbol CONSTANT among the constants for RPC-PROGRAM structure
   CONTEXT. "
   (CHECK-TYPE CONSTANT SYMBOL)
   (SECOND (ASSOC CONSTANT (RPC-PROGRAM-CONSTANTS CONTEXT))))

(DEFUN INSPECT-STRING (S)
   "Utility function for seeing the bytes in an unprintable string."
   (DO ((I 0 (+ 1 I))
        (WORD 0))
       ((>= I (LENGTH S)))
     (SETQ WORD (+ (CHAR-INT (ELT S I))
                   (ASH WORD 8)))
     (IF (AND (> I 0)
              (EQL 0 (MOD (+ 1 I)
                          4)))
         (PROGN (FORMAT-T "Word(~2d)=~a~%" (FLOOR (/ I 4))
                          WORD)
                (SETQ WORD 0)))))

(DEFUN INSPECT-STRING1 (S &OPTIONAL NBYTES)
   (DO ((BYTE 0 (+ BYTE 4))
        (I 0 (+ I 1))
        (NBYTES (OR NBYTES (LENGTH S)))
        (BYTE1)
        (BYTE2)
        (BYTE3)
        (BYTE4)
        (WORD)
        (NEXTWORD)
        (CELL)
        (STRINGREP))
       ((>= BYTE NBYTES)
        T)
     (SETQ BYTE1 (CHAR-INT (ELT S BYTE)))
     (SETQ BYTE2 (IF (< (+ 1 BYTE)
                        NBYTES)
                     (CHAR-INT (ELT S (+ 1 BYTE)))
                     0))
     (SETQ BYTE3 (IF (< (+ 2 BYTE)
                        NBYTES)
                     (CHAR-INT (ELT S (+ 2 BYTE)))
                     0))
     (SETQ BYTE4 (IF (< (+ 3 BYTE)
                        NBYTES)
                     (CHAR-INT (ELT S (+ 3 BYTE)))
                     0))
     (SETQ WORD (LOGIOR (ASH BYTE1 8)
                        BYTE2))
     (SETQ NEXTWORD (LOGIOR (ASH BYTE3 8)
                            BYTE4))
     (SETQ CELL (LOGIOR (ASH WORD 16)
                        NEXTWORD))
     (SETQ STRINGREP (MAP 'STRING #'(LAMBDA (C)
                                      (IF (GRAPHIC-CHAR-P (INT-CHAR C))
                                          (INT-CHAR C)
                                          #\-))
                          (LIST BYTE1 BYTE2 BYTE3 BYTE4)))
     (FORMAT-T "~3d(~3d): ~12d  ~6d  ~6d   ~4d  ~4d  ~4d  ~4d    ~a~%" I BYTE CELL WORD NEXTWORD BYTE1 BYTE2
            BYTE3 BYTE4 STRINGREP)))

(DEFUN INSPECT-PACKET (PACKET DIR &OPTIONAL (ARGNUM 0))
   "Utility function for seeing the bytes, words and cells of a UDP packet for a remote procedure call or reply.
   PACKET is an IL:ETHERPACKET and DIR is one of RPC2::CALL or RPC2::REPLY. This procedure does not know how
   long authenticaitons are. That's too bad.
   "
   (LET* ((NAME-ARRAY (VECTOR 20))
```

```
          WORD NEXTWORD CELL (INIT-OFFSET 30)
          (UDP-PACKET-LENGTH (/ (IL:\\GETBASE PACKET (+ INIT-OFFSET 12))
                                4))
          BYTE1 BYTE2 BYTE3 BYTE4 STRINGREP
          (CALL-NAMES '("Vers|HdrL|Serv||TotLn" "IPIP||Frag" "Time|Prot||Chsum" "IP Source Address" "IP
                              Destination Address" "Source Port| Dest Port" "Length | Checksum" "XID" "Msg
                              Type" "RPC Protocol Version" "RPC Program" "RPC Program Version" "RPC Procedure
                              Number" "Auth1-type" "Auth1-len" "Auth2-type" "Auth2-len"))
          (REPLY-NAMES '("Vers|HdrL|Serv||TotLn" "IPIP||Frag" "Time|Prot||Chsum" "IP Source Address" "IP
                               Destination Address" "Source Port| Dest Port" "Length | Checksum" "XID" "Msg
                               Type" "Reply Status" "Auth-Type/Reject-Stat" "Auth-Length/Low" "Accept
                               Status/High")))
         (DO ((I 0 (+ I 1))
              (NAMES (IF (EQUAL (SYMBOL-NAME DIR)
                                "CALL")
                         CALL-NAMES
                         REPLY-NAMES)
                     (CDR NAMES)))
             ((>= I (+ 5 UDP-PACKET-LENGTH))
              T)
           (SETQ WORD (IL:\\GETBASE PACKET (+ INIT-OFFSET (* 2 I))))
           (SETQ NEXTWORD (IL:\\GETBASE PACKET (+ INIT-OFFSET (+ 1 (* 2 I)))))
           (SETQ CELL (+ (ASH WORD 16)
                         NEXTWORD))
           (SETQ BYTE1 (ASH WORD -8)
                 BYTE2
                 (LOGAND WORD 255)
                 BYTE3
                 (ASH NEXTWORD -8)
                 BYTE4
                 (LOGAND NEXTWORD 255))
           (SETQ STRINGREP (MAP 'STRING #'(LAMBDA (C)
                                            (IF (GRAPHIC-CHAR-P (INT-CHAR C))
                                                (INT-CHAR C)
                                                #\-))
                                (LIST BYTE1 BYTE2 BYTE3 BYTE4)))
           (FORMAT-T "~3d:  ~23a   ~12d  ~6d  ~6d   ~4d  ~4d  ~4d  ~4d    ~a~%" I
                  (OR (FIRST NAMES)
                      (PROGN (SETQ ARGNUM (+ 1 ARGNUM))
                             (CONCATENATE 'STRING "Arg" (PRIN1-TO-STRING ARGNUM))))
                  CELL WORD NEXTWORD BYTE1 BYTE2 BYTE3 BYTE4 STRINGREP))))
```

;;; RPC Streams

```
(DEFUN OPEN-RPCSTREAM (PROTOCOL DESTADDR DESTSOCKET)
   "Create and return a new RPC-STREAM."
   (ECASE PROTOCOL
       (UDP (RPC-CREATE-UDP-STREAM))
       (TCP (RPC-CREATE-TCP-STREAM DESTADDR DESTSOCKET))))

(DEFUN CLOSE-RPCSTREAM (RPCSTREAM)
   "Deallocate an RPC Stream. Tries to cleanup after itself."
   (ECASE (RPC-STREAM-PROTOCOL RPCSTREAM)
       (UDP (IF *USE-OS-NETWORKING*
                (PROGN (RECLAIM-XDR-DATA-BLOCK (RPC-STREAM-OUTSTREAM RPCSTREAM))
                       (RECLAIM-XDR-DATA-BLOCK (RPC-STREAM-INSTREAM RPCSTREAM)))
                (IL:UDP.CLOSE.SOCKET (RPC-STREAM-IPSOCKET RPCSTREAM))))
       (TCP
          (CLOSE (RPC-STREAM-OUTSTREAM RPCSTREAM))
          (CLOSE (RPC-STREAM-INSTREAM RPCSTREAM))
          T)))

(DEFUN REINITIALIZE-RPCSTREAM (STREAM DESTADDR DESTSOCKET)
   "Reuse an existing RPC Stream to send a new packet. Resets length counters, reinitializes packets, etc."
   (CCASE (RPC-STREAM-PROTOCOL STREAM)
          (UDP (COND
                  (*USE-OS-NETWORKING* (SETF (RPC-STREAM-IPSOCKET STREAM)
                                             DESTSOCKET)
                         (SETF (RPC-STREAM-OS-DESTADDR STREAM)
                               DESTADDR)
                         (SETF (RPC-STREAM-OUTBYTEPTR STREAM)
                               0))
                  (T (WHEN (TYPEP (RPC-STREAM-INSTREAM STREAM)
                                  'IL:ETHERPACKET)
                           ;; Release  Etherpacket used for previous input from remote host. This could be done earlier, when
                           ;; PARSE-RPC-STREAM finishes with the packet, but since *RPCSTREAM* still points at the stream for debugging, it
                           ;; is better to wait until now..
                           (IL:\\RELEASE.ETHERPACKET (RPC-STREAM-INSTREAM STREAM))
                           (SETF (RPC-STREAM-INSTREAM STREAM)
                                 NIL))
                     (CHECK-TYPE (RPC-STREAM-OUTSTREAM STREAM)
                            IL:ETHERPACKET)
```

```
                    (IL:UDP.SETUP (RPC-STREAM-OUTSTREAM STREAM)
                                  DESTADDR DESTSOCKET 0 (RPC-STREAM-IPSOCKET STREAM))))
               (SETF (RPC-STREAM-INBYTEPTR STREAM)
                     0))
          (TCP (RM-INITIALIZE-OUTSTREAM STREAM)
               (RM-INITIALIZE-INSTREAM STREAM)
               T)))
```

```
(DEFMACRO GETBYTE (XDRSTREAM)
   "Macro that calls function from GETBYTE field of an RPC Stream on that RPC Stream
   to read in and return the next byte of the stream. "
   `(FUNCALL (RPC-STREAM-GETBYTE ,XDRSTREAM)
           ,XDRSTREAM))
```

```
(DEFMACRO GETBYTES (XDRSTREAM NBYTES)
   "Macro that calls function from GETBYTES field of an RPC Stream on that RPC Stream
   to read in and return the next NBYTES bytes of the stream. "
   `(FUNCALL (RPC-STREAM-GETBYTES ,XDRSTREAM)
           ,XDRSTREAM
           ,NBYTES))
```

```
(DEFMACRO GETCELL (XDRSTREAM)
   "Macro that calls function from GETCELL field of an RPC Stream on that RPC Stream
   to read in and return the next cell of the stream. A cell is a 32-bit two's complement integer.  "
   `(FUNCALL (RPC-STREAM-GETCELL ,XDRSTREAM)
           ,XDRSTREAM))
```

```
(DEFMACRO GETOFFSET (XDRSTREAM)
   "
   Returns dotted pair (base . byteoffset), pointing at current position in incoming packet
   "
   `(FUNCALL (RPC-STREAM-GETOFFSET ,XDRSTREAM)
           ,XDRSTREAM))
```

```
(DEFMACRO PUTBYTE (RPCSTREAM VALUE)
   "Macro that calls function from PUTBYTE field of an RPC Stream on that RPC Stream
   to write the byte VALUE on that stream. VALUE is an integer between 0 and 255 inclusive.
   "
   `(FUNCALL (RPC-STREAM-PUTBYTE ,RPCSTREAM)
           ,RPCSTREAM
           ,VALUE))
```

```
(DEFMACRO PUTBYTES (RPCSTREAM STRING)
   "Macro that calls function from PUTBYTES field of an RPC Stream on that RPC Stream
   to write the bytes from STRING on that stream. Each character of STRING is converted to the corresponding
   integer value between 0 and 255.

   "
   `(FUNCALL (RPC-STREAM-PUTBYTES ,RPCSTREAM)
           ,RPCSTREAM
           ,STRING))
```

```
(DEFMACRO PUTCELL (RPCSTREAM VALUE)
   "Macro that calls function from PUTCELL field of an RPC Stream on that RPC Stream
   to write the cell VALUE on that stream. A cell is a 32-bit two's complement integer.

   "
   `(FUNCALL (RPC-STREAM-PUTCELL ,RPCSTREAM)
           ,RPCSTREAM
           ,VALUE))
```

```
(DEFUN GETUNSIGNED (RPCSTREAM &AUX VALUE)
   "Macro that calls function from GETUNSIGNED field of an RPC Stream on that RPC Stream
   to read and return the next unsgned from that stream. An unsigned is a 32-bit unsigned integer.

   "
   (IF (< (SETQ VALUE (GETCELL RPCSTREAM))
          0)
       (+ TWOTO32ND VALUE)
       VALUE))
```

```
(DEFUN PUTUNSIGNED (RPCSTREAM VALUE)
   "Macro that calls function from GETUNSIGNED field of an RPC Stream on that RPC Stream
   to read and return an unsigned number from that stream. An unsigned number is a 32-bit unsigned number.

   "
   (IF (> VALUE TWOTO31MINUSONE)
       (SETQ VALUE (- VALUE TWOTO32ND)))
```

```
   (PUTCELL RPCSTREAM VALUE))

(DEFMACRO PUTOFFSET (XDRSTREAM BYTEOFFSET)
   "
   Sets byteoffset in incoming packet
   "
   `(FUNCALL (RPC-STREAM-PUTOFFSET ,XDRSTREAM)
           ,XDRSTREAM
           ,BYTEOFFSET))
```

;;; UDP Protocol RPC Streams

```
(DEFUN RPC-CREATE-UDP-STREAM ()
   "Create a new RPC Stream with the vector of functions set up to for UDP Protocol Datagrams.
   "
   (IF *USE-OS-NETWORKING*
       (MAKE-RPC-STREAM :PROTOCOL 'UDP :OUTSTREAM (ALLOCATE-XDR-DATA-BLOCK)
              :INSTREAM
              (ALLOCATE-XDR-DATA-BLOCK)
              :INBYTEPTR 0 :OUTBYTEPTR 0 :GETBYTE #'OS-UDP-GETBYTE :GETBYTES #'OS-UDP-GETBYTES :PUTBYTE
              #'OS-UDP-PUTBYTE :PUTBYTES #'OS-UDP-PUTBYTES :GETCELL #'OS-UDP-GETCELL :PUTCELL #'OS-UDP-PUTCELL
              :GETOFFSET #'OS-UDP-GETOFFSET :PUTOFFSET #'OS-UDP-PUTOFFSET)
       (MAKE-RPC-STREAM :PROTOCOL 'UDP :IPSOCKET (IL:UDP.OPEN.SOCKET)
              :OUTSTREAM
              (IL:\\ALLOCATE.ETHERPACKET)
              :GETBYTE
              #'UDP-GETBYTE :GETBYTES #'UDP-GETBYTES :PUTBYTE #'UDP-PUTBYTE :PUTBYTES #'UDP-PUTBYTES :GETCELL
              #'UDP-GETCELL :GETOFFSET #'UDP-GETOFFSET :PUTCELL #'UDP-PUTCELL :PUTOFFSET #'UDP-PUTOFFSET
              :INBYTEPTR 0)))

(DEFUN UDP-GETBYTE (RPCSTREAM)
   "NIL"
   (LET ((OFFSET (RPC-STREAM-INBYTEPTR RPCSTREAM)))
        (PROG1 (IL:UDP.GET.BYTE (RPC-STREAM-INSTREAM RPCSTREAM)
                       OFFSET)
               (SETF (RPC-STREAM-INBYTEPTR RPCSTREAM)
                     (+ 1 OFFSET)))))

(DEFUN UDP-PUTBYTE (RPCSTREAM BYTE)
   "NIL"
   (IL:UDP.APPEND.BYTE (RPC-STREAM-OUTSTREAM RPCSTREAM)
          BYTE))

(DEFUN UDP-GETCELL (RPCSTREAM)
   "NIL"
   (LET ((BYTEOFFSET (RPC-STREAM-INBYTEPTR RPCSTREAM)))
        (PROG1 (IL:UDP.GET.CELL (RPC-STREAM-INSTREAM RPCSTREAM)
                       (ASH BYTEOFFSET -2))
               (SETF (RPC-STREAM-INBYTEPTR RPCSTREAM)
                     (+ 4 BYTEOFFSET)))))

(DEFUN UDP-PUTCELL (XDRSTREAM VALUE)
   "NIL"
   (IL:UDP.APPEND.CELL (RPC-STREAM-OUTSTREAM XDRSTREAM)
          VALUE))

(DEFUN UDP-GETOFFSET (RPCSTREAM)
   (CONS (RPC-STREAM-INSTREAM RPCSTREAM)
         (RPC-STREAM-INBYTEPTR RPCSTREAM)))

(DEFUN UDP-PUTOFFSET (RPCSTREAM BYTEOFFSET)
   (SETF (RPC-STREAM-INBYTEPTR RPCSTREAM)
         BYTEOFFSET))

(DEFUN UDP-PUTBYTES (XDRSTREAM STRING)
   (IL:UDP.APPEND.STRING (RPC-STREAM-OUTSTREAM XDRSTREAM)
          STRING))

(DEFUN UDP-GETBYTES (XDRSTREAM NBYTES)
   (PROG1 (IL:UDP.MYGET.STRING (RPC-STREAM-INSTREAM XDRSTREAM)
                (RPC-STREAM-INBYTEPTR XDRSTREAM)
                NBYTES)
       (INCF (RPC-STREAM-INBYTEPTR XDRSTREAM)
             NBYTES)))
```

;;; TCP Protocol RPC Streams

```
(DEFUN RPC-CREATE-TCP-STREAM (DESTADDR DESTSOCKET)
   "Create a new RPC Stream with the vector of functions handling a bi-directional TCP stream between the
   devices."
   (LET* ((OSTR (IL:TCP.OPEN DESTADDR DESTSOCKET NIL 'IL:ACTIVE 'IL:OUTPUT))
          (RPCSTREAM (MAKE-RPC-STREAM :PROTOCOL 'TCP :OUTSTREAM OSTR :INSTREAM (IL:TCP.OTHER.STREAM OSTR)
                                      :GETBYTE
                                      #'TCP-GETBYTE :GETCELL #'TCP-GETCELL :GETBYTES #'TCP-GETBYTES :PUTBYTE
                                      #'TCP-PUTBYTE :PUTBYTES #'TCP-PUTBYTES :PUTCELL #'TCP-PUTCELL :OUTSTRING
                                      (MAKE-STRING *BYTES-PER-RM-OUTREC*))))
          (REINITIALIZE-RPCSTREAM RPCSTREAM DESTADDR DESTSOCKET)
          RPCSTREAM))

(DEFUN TCP-GETBYTE (RPCSTREAM)
   "Read in one byte from an RM Record"
   (WHEN (ZEROP (THE INTEGER (RPC-STREAM-INBYTEPTR RPCSTREAM)))
         (RM-NEW-INPUT-RECORD RPCSTREAM))
   (DECF (THE INTEGER (RPC-STREAM-INBYTEPTR RPCSTREAM))
         1)
   (IL:BIN (RPC-STREAM-INSTREAM RPCSTREAM)))

(DEFUN TCP-GETBYTES (RPCSTREAM NBYTES)
   "Read NBYTES bytes into a new string from as many RM records as needed."
   (IF (ZEROP NBYTES)
       (RETURN-FROM TCP-GETBYTES ""))
   (LET ((FIRST 0)
         (STR (MAKE-STRING NBYTES :INITIAL-ELEMENT #\Null))
         (INSTREAM (RPC-STREAM-INSTREAM RPCSTREAM)))
        ;;  FIRST is the index of the next char to be read.
        ;; NSTRBYTES is the number of bytes remaining to be read.
        ;; NRMBYTES is the number of  bytes remaining in the current RM Record.
        (DO ((NRMBYTES (RPC-STREAM-INBYTEPTR RPCSTREAM)
                       (RPC-STREAM-INBYTEPTR RPCSTREAM))
             (NSTRBYTES NBYTES))
            ((<= NSTRBYTES NRMBYTES)
             ;; Here is the real case --- the string all comes from the same RM record..
             (OR (ZEROP NSTRBYTES)
                 (IL:STRING.BINS INSTREAM STR FIRST NSTRBYTES))
             (DECF (RPC-STREAM-INBYTEPTR RPCSTREAM)
                   NSTRBYTES)
             (WHEN (AND (NUMBERP *DEBUG*)
                        (> *DEBUG* 1))
                 (FORMAT-T "Inspecting string after TCP-GETBYTES.~%")
                 (INSPECT-STRING1 STR (RPC-STREAM-INBYTEPTR RPCSTREAM)))
             STR)
          ;; Hypothetical Case: String is too big. Write out the beginning of it and start over.
          (IL:STRING.BINS INSTREAM STR FIRST NRMBYTES)
          (RM-NEW-INPUT-RECORD RPCSTREAM)
          (INCF FIRST NRMBYTES)
          (DECF (RPC-STREAM-INBYTEPTR RPCSTREAM)
                NRMBYTES))))

(DEFUN TCP-PUTBYTES (RPCSTREAM STRING)
   (LET ((FIRST 0))
        ;;  FIRST is the index of the next char to be written.
        ;; NSTRBYTES is the number of bytes remaining to be written out.
        ;; NRMBYTES is the number of unused bytes remaining in the current RM Record.
        (DO ((NRMBYTES (- *BYTES-PER-RM-OUTREC* (RPC-STREAM-OUTBYTEPTR RPCSTREAM))
                       (- *BYTES-PER-RM-OUTREC* (RPC-STREAM-OUTBYTEPTR RPCSTREAM)))
             (NSTRBYTES (LENGTH STRING)
                        (- NSTRBYTES FIRST)))
            ((<= NSTRBYTES NRMBYTES)
             ;; Here is the real case. Our string fits just fine into the outgoing RM record. Just write it and bump OUTBYTEPTR
             ;; .
             (OR (= NSTRBYTES 0)
                 (REPLACE (RPC-STREAM-OUTSTRING RPCSTREAM)
                          STRING :START1 (RPC-STREAM-OUTBYTEPTR RPCSTREAM)
                          :START2 FIRST :END2 (1- (+ FIRST NSTRBYTES))))
             (INCF (RPC-STREAM-OUTBYTEPTR RPCSTREAM)
                   NSTRBYTES)
             (WHEN (AND (NUMBERP *DEBUG*)
                        (> *DEBUG* 3))
                 (FORMAT-T "Inspecting string after TCP-PUTBYTES.~%")
                 (INSPECT-STRING1 (RPC-STREAM-OUTSTRING RPCSTREAM)
                        (RPC-STREAM-OUTBYTEPTR RPCSTREAM))))))
```

```
          ;; Hypothetical Case: String is too big. Write out the beginning of it and start over.
          (REPLACE (RPC-STREAM-OUTSTRING RPCSTREAM)
                 STRING :START1 (RPC-STREAM-OUTBYTEPTR RPCSTREAM)
                 :START2 FIRST :END2 (1- (+ FIRST NRMBYTES)))
          (SETQ FIRST (+ FIRST NRMBYTES))
          (INCF (RPC-STREAM-OUTBYTEPTR RPCSTREAM)
                NRMBYTES)
          (RM-FORCEOUTPUT RPCSTREAM NIL)                    ; Force it out!!!
          (RM-INITIALIZE-OUTSTREAM RPCSTREAM))))

(DEFUN TCP-PUTBYTE (RPCSTREAM BYTE)
   (WHEN (>= (RPC-STREAM-OUTBYTEPTR RPCSTREAM)
             *BYTES-PER-RM-OUTREC*)
       (RM-FORCEOUTPUT RPCSTREAM NIL)
       (RM-INITIALIZE-OUTSTREAM RPCSTREAM))
   (SETF (SCHAR (RPC-STREAM-OUTSTRING RPCSTREAM)
                (RPC-STREAM-OUTBYTEPTR RPCSTREAM))
         (INT-CHAR BYTE))
   (INCF (RPC-STREAM-OUTBYTEPTR RPCSTREAM)
         1)
   (WHEN (> *DEBUG* 3)
       (FORMAT-T "Inspecting string after TCP-PUTBYTE.~%")
       (INSPECT-STRING1 (RPC-STREAM-OUTSTRING RPCSTREAM)
              (RPC-STREAM-OUTBYTEPTR RPCSTREAM))))

(DEFUN TCP-GETCELL (RPCSTREAM)
   "Read in a 4 byte signed integer."
   (IF (< (THE INTEGER (RPC-STREAM-INBYTEPTR RPCSTREAM))
          4)
       ;; Since it calls TCP-GETBYTE, does not have to check for breaking across RM records.
       (+ (ASH (TCP-GETBYTE RPCSTREAM)
               24)
          (ASH (TCP-GETBYTE RPCSTREAM)
               16)
          (ASH (TCP-GETBYTE RPCSTREAM)
               8)
          (TCP-GETBYTE RPCSTREAM))
       (PROGN (IL:STRING.BINS (RPC-STREAM-INSTREAM RPCSTREAM)
                     *RPC-GETCELL-TEMP* 0 4)
              (DECF (THE INTEGER (RPC-STREAM-INBYTEPTR RPCSTREAM))
                    4)
              (+ (ASH (CHAR-INT (SCHAR *RPC-GETCELL-TEMP* 0))
                      24)
                 (ASH (CHAR-INT (SCHAR *RPC-GETCELL-TEMP* 1))
                      16)
                 (ASH (CHAR-INT (SCHAR *RPC-GETCELL-TEMP* 2))
                      8)
                 (CHAR-INT (SCHAR *RPC-GETCELL-TEMP* 3))))))

(DEFUN TCP-PUTCELL (RPCSTREAM VALUE)
   (LET ((OUTSTRING (RPC-STREAM-OUTSTRING RPCSTREAM))
         (INDX (RPC-STREAM-OUTBYTEPTR RPCSTREAM)))
        (WHEN (> INDX (- *BYTES-PER-RM-OUTREC* 4))
            (RM-FORCEOUTPUT RPCSTREAM NIL)
            (RM-INITIALIZE-OUTSTREAM RPCSTREAM))
        (SETF (SCHAR OUTSTRING INDX)
              (INT-CHAR (ASH VALUE -24)))
        (SETF (SCHAR OUTSTRING (+ INDX 1))
              (INT-CHAR (LOGAND 255 (ASH VALUE -16))))
        (SETF (SCHAR OUTSTRING (+ INDX 2))
              (INT-CHAR (LOGAND 255 (ASH VALUE -8))))
        (SETF (SCHAR OUTSTRING (+ INDX 3))
              (INT-CHAR (LOGAND 255 VALUE)))
        (INCF (RPC-STREAM-OUTBYTEPTR RPCSTREAM)
              4)
        (WHEN (AND (NUMBERP *DEBUG*)
                   (> *DEBUG* 3))
            (FORMAT-T "Inspecting string after TCP-PUTCELL of ~d.~%" VALUE)
            (INSPECT-STRING1 (RPC-STREAM-OUTSTRING RPCSTREAM)
                   (RPC-STREAM-OUTBYTEPTR RPCSTREAM)))))

(DEFUN RM-FORCEOUTPUT (RPCSTREAM FINAL-FRAGMENT-FLAG)
   (LET* ((OUTSTRING (RPC-STREAM-OUTSTRING RPCSTREAM))
          (OUTSTREAM (RPC-STREAM-OUTSTREAM RPCSTREAM))
          (TOTAL-LENGTH (RPC-STREAM-OUTBYTEPTR RPCSTREAM))
          (NET-LENGTH (- (THE INTEGER TOTAL-LENGTH)
                         4)))
         ;; Stuff RM header into outstring.
         ;; If this is the final fragment of the RM record, OR in a one to high order bit of RM header.
         (SETF (SCHAR OUTSTRING 0)
```

```
              (IF FINAL-FRAGMENT-FLAG
                  (INT-CHAR (LOGIOR 128 (ASH NET-LENGTH -24)))
                  (INT-CHAR (ASH NET-LENGTH -24))))
          (SETF (SCHAR OUTSTRING 1)
                (INT-CHAR (LOGAND 255 (ASH NET-LENGTH -16))))
          (SETF (SCHAR OUTSTRING 2)
                (INT-CHAR (LOGAND 255 (ASH NET-LENGTH -8))))
          (SETF (SCHAR OUTSTRING 3)
                (INT-CHAR (LOGAND 255 NET-LENGTH)))
          (IL:STRING.BOUTS OUTSTREAM OUTSTRING 0 TOTAL-LENGTH)))
```

```
(DEFUN RM-INITIALIZE-OUTSTREAM (RPCSTREAM)
```

;; Zero out the four bytes of RM header and leave OUTBYTEPTR pointing after them.

```
   (IL:STRING.ZEROBYTES (RPC-STREAM-OUTSTRING RPCSTREAM)
          0 4)
   (SETF (RPC-STREAM-OUTBYTEPTR RPCSTREAM)
         4))
```

```
(DEFUN RM-INITIALIZE-INSTREAM (RPCSTREAM)
   T)
```

```
(DEFUN RM-NEW-INPUT-RECORD (RPCSTREAM)
   "Read the four byte unsigned record length for a new rm record and store it in INBYTEPTR."
   (LET* ((INSTREAM (RPC-STREAM-INSTREAM RPCSTREAM))
          (CELL (+ (ASH (LOGAND (IL:BIN INSTREAM)
                                2147483648)
                        24)                                    ; Kill high order bit, if any.
                   (ASH (IL:BIN INSTREAM)
                        16)
                   (ASH (IL:BIN INSTREAM)
                        8)
                   (IL:BIN INSTREAM)))
          (NBYTES (IF (< CELL 0)
                      (+ TWOTO32ND CELL)
                      CELL)))
         (WHEN *DEBUG* (FORMAT-T "RM Record is to be ~d bytes.~%" NBYTES))
         (SETF (RPC-STREAM-INBYTEPTR RPCSTREAM)
               NBYTES)))
```

;;; String RPC Stream

```
(DEFUN CREATE-STRING-RPC-STREAM ()
   "Create RPC STREAM that writes data to a string-output-stream."
   (MAKE-RPC-STREAM :OUTSTREAM (MAKE-STRING-OUTPUT-STREAM)
          :PUTCELL
          #'(LAMBDA (STR VALUE)
                    (IF (< VALUE 0)
                        (SETQ VALUE (+ VALUE TWOTO32ND)))
                    (WRITE-CHAR (INT-CHAR (ASH VALUE -24))
                           (RPC-STREAM-OUTSTREAM STR))
                    (WRITE-CHAR (INT-CHAR (LOGAND 255 (ASH VALUE -16)))
                           (RPC-STREAM-OUTSTREAM STR))
                    (WRITE-CHAR (INT-CHAR (LOGAND 255 (ASH VALUE -8)))
                           (RPC-STREAM-OUTSTREAM STR))
                    (WRITE-CHAR (INT-CHAR (LOGAND 255 VALUE))
                           (RPC-STREAM-OUTSTREAM STR)))
          :GETCELL
          #'(LAMBDA (STR)
                    (LET ((V (+ (ASH (GETBYTE (RPC-STREAM-OUTSTREAM STR))
                                     24)
                                (ASH (GETBYTE (RPC-STREAM-OUTSTREAM STR))
                                     16)
                                (ASH (GETBYTE (RPC-STREAM-OUTSTREAM STR))
                                     8)
                                (GETBYTE (RPC-STREAM-OUTSTREAM STR)))))
                         (IF (> V TWOTO31MINUSONE)
                             (- V TWOTO32ND)
                             V)))
          :PUTBYTES
          #'(LAMBDA (RPCSTREAM VALUE)
                    (DO ((I 0 (+ 1 I)))
                        ((>= I (LENGTH VALUE)))
                      (PUTBYTE RPCSTREAM (CHAR-INT (ELT VALUE I)))))
          :PUTBYTE
          #'(LAMBDA (STR VAL)
                    (WRITE-CHAR (INT-CHAR VAL)
                           (RPC-STREAM-OUTSTREAM STR)))
          :GETBYTE
          #'(LAMBDA (STR)
                    (CHAR-INT (READ-FROM-STRING (RPC-STREAM-OUTSTREAM STR))))))
```

;;; TTY RPC Stream

```
(DEFUN CREATE-TTY-RPC-STREAM (&OPTIONAL (INSTRING T READP))
   "For debugging using the TTY as the output device or an optional string INSTRING from which to take data."
   (MAKE-RPC-STREAM :INSTREAM (IF READP (MAKE-STRING-INPUT-STREAM INSTRING))
          :OUTSTREAM *STANDARD-OUTPUT* :PUTCELL #'(LAMBDA (STR VALUE)
                                                          (IF (< VALUE 0)
                                                              (SETQ VALUE (+ VALUE TWOTO32ND)))
                                                          (FORMAT (RPC-STREAM-OUTSTREAM STR)
                                                                 "~a~%"
                                                                 (ASH VALUE -24))
                                                          (FORMAT (RPC-STREAM-OUTSTREAM STR)
                                                                 "~a~%"
                                                                 (LOGAND 255 (ASH VALUE -16)))
                                                          (FORMAT (RPC-STREAM-OUTSTREAM STR)
                                                                 "~a~%"
                                                                 (LOGAND 255 (ASH VALUE -8)))
                                                          (FORMAT (RPC-STREAM-OUTSTREAM STR)
                                                                 "~a~%"
                                                                 (LOGAND 255 VALUE)))
          :GETCELL
          #'(LAMBDA (STR)
                    (LET ((V (+ (ASH (GETBYTE STR)
                                     24)
                                (ASH (GETBYTE STR)
                                     16)
                                (ASH (GETBYTE STR)
                                     8)
                                (GETBYTE STR))))
                         (IF (> V TWOTO31MINUSONE)
                             (- V TWOTO32ND)
                             V)))
          :PUTBYTES
          #'(LAMBDA (RPCSTREAM VALUE)
                    (DO ((I 0 (+ 1 I)))
                        ((>= I (LENGTH VALUE)))
                      (PUTBYTE RPCSTREAM (CHAR-INT (ELT VALUE I)))))
          :PUTBYTE
          #'(LAMBDA (STR VAL)
                    (FORMAT (RPC-STREAM-OUTSTREAM STR)
                           "~a~%"
                           (INT-CHAR VAL)))
          :GETBYTES
          #'(LAMBDA (STR N)
                    (LET ((S (MAKE-STRING N)))
                         (DOTIMES (I N S)
                             (SETF (ELT S I)
                                   (INT-CHAR (GETBYTE STR))))))
          :GETBYTE
          #'(LAMBDA (STR)
                    (LET (B)
                         (FORMAT (RPC-STREAM-OUTSTREAM STR)
                                "~a~%"
                                (SETQ B (CHAR-INT (READ-CHAR (RPC-STREAM-INSTREAM STR)))))
                        B))))

(IL:PUTPROPS IL:RPCCOMMON IL:COPYRIGHT ("Stanford University and Xerox Corporation" 1987 1988))
```

FUNCTION INDEX

MACRO INDEX

VARIABLE INDEX

CONSTANT INDEX

PROPERTY INDEX