File created: 1-Aug-88 11:51:33 **{ERINYES}<LISPUSERS>MEDLEY>RPCRPC.;2**

changes to: (IL:FUNCTIONS DEFINE-REMOTE-PROGRAM)

previous date: 28-Apr-88 17:26:39 {ERINYES}<LISPUSERS>MEDLEY>RPCRPC.;1

Read Table: XCL

Package: RPC2

Format: XCCS

; Copyright (c) 1987, 1988 by Stanford University and Xerox Corporation. All rights reserved.

```
(IL:RPAQQ IL:RPCRPCCOMS
          ((IL:PROPS (IL:RPCRPC IL:MAKEFILE-ENVIRONMENT IL:FILETYPE))
           (IL:VARIABLES *DEBUG* *RPC-CALL* *RPC-VERSION* *RPC-PROGRAMS* *MSEC-UNTIL-TIMEOUT*
                  *MSEC-BETWEEN-TRIES* *INTERNAL-TIME-UNITS-PER-MSEC* *RPC-REPLY-STATS* *RPC-ACCEPT-STATS*
                  *RPC-REJECT-STATS* *RPC-AUTHENTICATION-STATS* *RPC-OK-TO-CACHE* *RPC-SOCKET-CACHE* *XID-COUNT*
                  *RPC-DEF-IN-PROGRESS* *RPC-WELL-KNOWN-SOCKETS* *RPC-PROTOCOLS* *RPCSTREAM* *RPC-PGNAME*
                  *RPC-PCNAME*)

;;; Define RPC Program

           (IL:FUNCTIONS DEFINE-REMOTE-PROGRAM DEFINE-REMOTE-PROG CONS-UP-RPC-PROCS CLEAR-ANY-NAME-CONFLICTS
                  DEF-RPC-TYPES DEF-RPC-INHERITS DEF-RPC-PROCEDURES DEF-RPC-PROCEDURE DEF-RPC-CONSTANTS
                  UNDEFINE-REMOTE-PROGRAM XDR-GENCODE-MAKEFCN XDR-GENCODE-INLINE)

;;; Remote Procedure Call

           (IL:FUNCTIONS REMOTE-PROCEDURE-CALL SETUP-RPC PERFORM-RPC RPC-RESOLVE-HOST RPC-RESOLVE-PROG
                  RPC-RESOLVE-PROC RPC-FIND-SOCKET ENCODE-RPC-ARGS ACTUALLY-DO-THE-RPC EXCHANGE-UDP-PACKETS
                  EXCHANGE-TCP-PACKETS PARSE-RPC-REPLY CREATE-XID)

;;; RPC Utility Functions

           (IL:FUNCTIONS GET-REPLY-STAT GET-ACCEPT-STAT GET-REJECT-STAT GET-AUTHENTICATION-STAT
                  GET-PROTOCOL-NUMBER FIND-CACHED-SOCKET)

;;; RPC Error Messages

           (IL:FUNCTIONS RPC-ERROR-PRM-MISMATCH RPC-ERROR-PRM-UNAVAILABLE RPC-ERROR-PRC-UNAVAILABLE
                  RPC-ERROR-GARBAGE-ARGS RPC-ERROR-MISMATCH RPC-ERROR-AUTHENTICATION)

;;; Authentication

           (IL:VARIABLES *AUTHENTICATION-TYPEDEF* *NULL-AUTHENTICATION*)
           (IL:FUNCTIONS CREATE-UNIX-AUTHENTICATION ENCODE-AUTHENTICATION DECODE-AUTHENTICATION)))

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

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


(DEFGLOBALPARAMETER *DEBUG* NIL
   "T for printout, NUMBER for even more.")


(DEFCONSTANT *RPC-CALL* 0
   "Constant 0 in packet means RPC call, 1 means reply")


(DEFCONSTANT *RPC-VERSION* 2
   "This code will only work for SUN RPC Version 2")


(DEFGLOBALVAR *RPC-PROGRAMS* NIL
   "
   A list of RPC-PROGRAM structs.

   This list is consulted by various routines to find infomation about known
   remote programs.

   It is assumed that a given NAME field uniquely identifies a (NUMBER, VERSION, PROTOCOL).
   On the other hand, there may be several NAMEs (and hence, several RPC-STRUCTs) for
   a given (NUMBER, VERSION, PROTOCOL).

   ")


(DEFPARAMETER *MSEC-UNTIL-TIMEOUT* 10000
   "Total time in msec before giving up on UDP exchange with remote host")
```

```
(DEFPARAMETER *MSEC-BETWEEN-TRIES* 1000
   "Time in msec between UDP retries")

(DEFCONSTANT *INTERNAL-TIME-UNITS-PER-MSEC* (/ INTERNAL-TIME-UNITS-PER-SECOND 1000)
                                            "This gets used in EXCHANGE-UDP-PACKETS.")

(DEFCONSTANT *RPC-REPLY-STATS* '((0 . ACCEPTED)
                                 (1 . REJECTED))
                               "
                               Assoc list for internal use by PARSE-RPC-REPLY.
                               ")

(DEFCONSTANT *RPC-ACCEPT-STATS* '((0 . SUCCESS)
                                  (1 . PROGRAM-UNAVAILABLE)
                                  (2 . PROGRAM-MISMATCH)
                                  (3 . PROCEDURE-UNAVAILABLE)
                                  (4 . GARBAGE-ARGUMENTS))
                                "
                                Assoc list for internal use by PARSE-RPC-REPLY.
                                ")

(DEFCONSTANT *RPC-REJECT-STATS* '((0 . RPC-MISMATCH)
                                  (1 . AUTHENTICATION-ERROR))
                                "
                                Assoc list for internal use by PARSE-RPC-REPLY.
                                ")

(DEFCONSTANT *RPC-AUTHENTICATION-STATS* '((1 . BAD-CREDENTIAL)
                                          (2 . REJECTED-CREDENTIAL)
                                          (3 . BAD-VERIFIER)
                                          (4 . REJECTED-VERIFIER)
                                          (5 TOO-WEAK))
                                        "NIL")

(DEFPARAMETER *RPC-OK-TO-CACHE* T
   "
   If NIL, does not attempt to cache socket numbers for non-well-known sockets
   ")

(DEFVAR *RPC-SOCKET-CACHE* NIL
   "
   A list of (<iphost-address> <remote-program-name> <remote-program-version>
              <protocol> <ipsocket-number>) quintuples.")

(DEFVAR *XID-COUNT* 0
   "Contains the  XID stamp of the next remote procedure call")

(DEFVAR *RPC-DEF-IN-PROGRESS* NIL
   "Used for debugging only")

(DEFGLOBALVAR *RPC-WELL-KNOWN-SOCKETS*
   `((* 100000 2 UDP 111)
     (* 100000 2 TCP 111)
     (* 100003 2 UDP 2049))
   "
   List of well-known RPC programs and their sockets.
   Each element is a list:
     (host-address prog-number prog-version protocol socket-number)

   Host-address may be *, in which case it matches any host address.
   Protocol should be either rpc2::UDP or rpc2::TCP.
   ")

(DEFVAR *RPC-PROTOCOLS* '((TCP . 6)
                          (UDP . 17)))

(DEFVAR *RPCSTREAM* NIL
   "This global is not used exceptin debugging.
    It holds a copy of the RPC-STREAM even after the RPC-CALL returns.")

(DEFGLOBALVAR *RPC-PGNAME* NIL
   "Name of RPC Program. Used only for *debug* printout.")
```

```
(DEFGLOBALVAR *RPC-PCNAME* NIL
   "Name of RPC Procedure. Used only for *debug* printout.")
```

;;; Define RPC Program

```
(DEFMACRO DEFINE-REMOTE-PROGRAM (NAME NUMBER VERSION PROTOCOL &KEY CONSTANTS TYPES INHERITS PROCEDURES)
   "
   This macro expands into code to add a new RPC-PROGRAM struct to
   *RPC-PROGRAMS*. The generated code checks first to see that there
   are no name conflicts with existing remote programs and then adds the new
   structure to *RPC-PROGRAMS*.
   "
   (LET ((ENAME (EVAL NAME))
         (ENUMBER (EVAL NUMBER))
         (EVERSION (EVAL VERSION))
         (EPROTOCOL (OR (EVAL PROTOCOL)
                        'UDP))
         (ECONSTANTS (EVAL CONSTANTS))
         (ETYPES (EVAL TYPES))
         (EINHERITS (EVAL INHERITS))
         (EPROCEDURES (EVAL PROCEDURES)))
        (CHECK-TYPE ENAME SYMBOL)
        (CHECK-TYPE ENUMBER NUMBER)
        (CHECK-TYPE EVERSION NUMBER)
        (COND
           ((MEMBER EPROTOCOL '(UDP TCP))
            (IF (AND *USE-OS-NETWORKING* (EQ EPROTOCOL 'TCP))
                (ERROR "~a is an unsupported protocol." EPROTOCOL)
                T))
           ((EQUAL "UDP" (STRING EPROTOCOL))
            (SETQ EPROTOCOL 'UDP))
           ((EQUAL "TCP" (STRING EPROTOCOL))
            (IF *USE-OS-NETWORKING*
                (ERROR "~a is an unsupported protocol." EPROTOCOL)
                (SETQ EPROTOCOL 'TCP)))
           ((ERROR "~a is unknown prototype." EPROTOCOL)))
        (LET ((RPROG (DEFINE-REMOTE-PROG ENAME ENUMBER EVERSION EPROTOCOL ECONSTANTS ETYPES EINHERITS
                           EPROCEDURES)))
             `(LET ((DUMMY (FORMAT-T "Defining remote program ~a, version ~a~%" ',ENAME ',EVERSION))
                    (NEWPROG (MAKE-RPC-PROGRAM :NUMBER ,ENUMBER :VERSION ,EVERSION :NAME ',ENAME :PROTOCOL
                                               ',EPROTOCOL :TYPES ',(RPC-PROGRAM-TYPES RPROG)
                                               :CONSTANTS
                                               ',(RPC-PROGRAM-CONSTANTS RPROG)
                                               :INHERITS
                                               ',(RPC-PROGRAM-INHERITS RPROG)
                                               :PROCEDURES
                                               ,(CONS-UP-RPC-PROCS (RPC-PROGRAM-PROCEDURES RPROG)))))
                   (IF (CLEAR-ANY-NAME-CONFLICTS ',ENAME ',ENUMBER ',EVERSION ',EPROTOCOL)
                       (PROGN (UNDEFINE-REMOTE-PROGRAM ',ENAME ',ENUMBER ',EVERSION)
                              (PUSH NEWPROG *RPC-PROGRAMS*)
                              ',ENAME)
                       (PROGN (FORMAT-T "Old RPC program not overwritten.~%")
                              NIL))))))
```

```
(DEFUN DEFINE-REMOTE-PROG (NAME NUMBER VERSION PROTOCOL CONSTANTS TYPES INHERITS PROCEDURES)
   ;; This guy does the work, so that DEFINE-REMOTE-PROGRAM can cons up the macro easily.
   ;; An RPC-PROGRAM struct RPROG is passed back to DEFINE-REMOTE-PROGRAM. Its innards are then used by DEFINE-REMOTE-PROGRAM
   ;; to build up the big cons that will cons up the proper RPC-PROGRAM later.
   (LET (RPROG)
        (FORMAT-T "Building XDR routines for remote program ~a, version ~a~%" NAME VERSION)
        (SETQ RPROG (MAKE-RPC-PROGRAM :NUMBER NUMBER :VERSION VERSION :NAME NAME :PROTOCOL PROTOCOL)
              *RPC-DEF-IN-PROGRESS* RPROG)
        (SETF (RPC-PROGRAM-TYPES RPROG)
              (DEF-RPC-TYPES RPROG TYPES))
        (SETF (RPC-PROGRAM-INHERITS RPROG)
              (DEF-RPC-INHERITS RPROG INHERITS))
        (SETF (RPC-PROGRAM-CONSTANTS RPROG)
              (DEF-RPC-CONSTANTS RPROG CONSTANTS))
        (SETF (RPC-PROGRAM-PROCEDURES RPROG)
              (DEF-RPC-PROCEDURES RPROG PROCEDURES))
        RPROG))
```

```
(DEFUN CONS-UP-RPC-PROCS (PROCS)
   "
   Given a list of RPC-PROCEDURE structs, conses up code to produce that set of
   RPC-PROCEDURE structs.
   "
   `(LIST
     ,@(MAP
        'LIST
        #'(LAMBDA (PROC)
```

```
                `(MAKE-RPC-PROCEDURE
                  :NAME
                  ',(RPC-PROCEDURE-NAME PROC)
                  :PROCNUM
                  ',(RPC-PROCEDURE-PROCNUM PROC)
                  :ARGTYPES
                  ,(IF (RPC-PROCEDURE-ARGTYPES PROC)
                       `(LIST ,@(MAP 'LIST #'(LAMBDA (FCN)
                                               (LIST 'FUNCTION FCN))
                                     (RPC-PROCEDURE-ARGTYPES PROC))))
                  :RESULTTYPES
                  ,(IF (RPC-PROCEDURE-RESULTTYPES PROC)
                       `(LIST ,@(MAP 'LIST #'(LAMBDA (FCN)
                                               (LIST 'FUNCTION FCN))
                                     (RPC-PROCEDURE-RESULTTYPES PROC))))))
        PROCS)))

(DEFUN CLEAR-ANY-NAME-CONFLICTS (NAME NUMBER VERSION PROTOCOL)
   "
   Determines whether a proposed (NAME, NUMBER, VERSION, PROTOCOL) would violate
   the assumption that a NAME uniquely specifies the other three components.

   If there exists a violation, the user is given a chance to remove the old program.

   Returns T if no violation of assumption (or violation is resolved by removing old program),
   Returns NIL if there is an unresolved violation.

   "
   (LET (OLDRPC)
        (COND
           ((AND (SETQ OLDRPC (FIND-RPC-PROGRAM :NAME NAME))
                 (OR (/= NUMBER (RPC-PROGRAM-NUMBER OLDRPC))
                     (/= VERSION (RPC-PROGRAM-VERSION OLDRPC))
                     (NOT (EQL PROTOCOL (RPC-PROGRAM-PROTOCOL OLDRPC)))))
            (FORMAT *QUERY-IO* "Remote program name conflict with existing program:~%   Name ~a, Protocol ~A,
                   Number ~a, Version ~a~%" NAME (RPC-PROGRAM-PROTOCOL OLDRPC)
                   (RPC-PROGRAM-NUMBER OLDRPC)
                   (RPC-PROGRAM-VERSION OLDRPC))
            (AND (YES-OR-NO-P "Do you want to remove the old program? ")
                 (UNDEFINE-REMOTE-PROGRAM (RPC-PROGRAM-NAME OLDRPC)
                        (RPC-PROGRAM-NUMBER OLDRPC)
                        (RPC-PROGRAM-VERSION OLDRPC)
                        (RPC-PROGRAM-PROTOCOL OLDRPC))))
           (T T))))

(DEFUN DEF-RPC-TYPES (CONTEXT TYPEDEFS)
   "
   Essentially a no-op, as typedefs are copied directly from the DEFINE-REMOTE-PROGRAM
   into the RPC-PROGRAM struct. Just prints out the name of each type as it is encountered.
   "
   (IF TYPEDEFS (FORMAT-T "    Types~%"))
   (DOLIST (I TYPEDEFS)
       (FORMAT-T "        ~A~%" (FIRST I)))
   TYPEDEFS)

(DEFUN DEF-RPC-INHERITS (CONTEXT PROGLIST)
   "
   Checks remote program inherited by this one to make sure that it exists.
   Issues a warning if it cannot find the program to be inherited.
   "
   (IF PROGLIST (FORMAT-T "    Inherits~%"))
   (DOLIST (PRG PROGLIST PROGLIST)
       (FORMAT-T "        ~A~%" PRG)
       (IF (NOT (AND (SYMBOLP PRG)
                     (FIND-RPC-PROGRAM :NAME PRG)))
           (WARN "Trying to inherit from remote program ~a, but ~a not found.~%" PRG PRG))))

(DEFUN DEF-RPC-PROCEDURES (CONTEXT PROCS)
   "Returns a list of RPC-PROCEDURE structs returned by DEF-RPC-PROCEDURE."
   (CHECK-TYPE PROCS LIST "A list of RPC procedure declarations")
   (IF PROCS (FORMAT-T "    Procedures~%"))
   (MAP 'LIST #'(LAMBDA (PROC)
                  (DEF-RPC-PROCEDURE CONTEXT PROC))
        PROCS))

(DEFUN DEF-RPC-PROCEDURE (CONTEXT PROC)
   "
   For a procedure specified to DEFINE-REMOTE-PROGRAM's :PROCEDURES argument,
   creates and returns an RPC-PROCEDURE struct.

   XDR procedure code is generated via the call to XDR-GENCODE-MAKEFCN.
   "
```

```
   (CHECK-TYPE (FIRST PROC)
          (AND SYMBOL (NOT NULL))
          "a non-null symbol naming the RPC procedure.")
   (CHECK-TYPE (SECOND PROC)
          (INTEGER 0 *)
          "a non-negative integer RPC procedure number")
   (CHECK-TYPE (THIRD PROC)
          LIST)
   (CHECK-TYPE (FOURTH PROC)
          LIST)
   (LET ((RP (MAKE-RPC-PROCEDURE)))
        (SETF (RPC-PROCEDURE-NAME RP)
              (FIRST PROC))
        (SETF (RPC-PROCEDURE-PROCNUM RP)
              (SECOND PROC))
        (SETF (RPC-PROCEDURE-ARGTYPES RP)
              (MAP 'LIST #'(LAMBDA (TD)
                                   (XDR-GENCODE-MAKEFCN CONTEXT TD 'WRITE))
                   (THIRD PROC)))
        (SETF (RPC-PROCEDURE-RESULTTYPES RP)
              (MAP 'LIST #'(LAMBDA (TD)
                                   (XDR-GENCODE-MAKEFCN CONTEXT TD 'READ))
                   (FOURTH PROC)))
        (FORMAT-T "        ~A~%" (RPC-PROCEDURE-NAME RP))
        RP))
```

```
(DEFUN DEF-RPC-CONSTANTS (CONTEXT PAIRS)
   "
   Checks that constants specified to DEFINE-REMOTE-PROGRAM are syntactically
   reasonable.
   "
   (IF PAIRS (FORMAT-T "    Constants~%"))
   (DOLIST (PAIR PAIRS)
       (CHECK-TYPE (FIRST PAIR)
              (AND (NOT NULL)
                   SYMBOL))
       (CHECK-TYPE (SECOND PAIR)
              (AND (NOT NULL)
                   NUMBER))
       (FORMAT-T "        ~A~%" (FIRST PAIR)))
   PAIRS)
```

```
(DEFUN UNDEFINE-REMOTE-PROGRAM (NAME NUMBER VERSION &OPTIONAL (PROTOCOL 'UDP))
   "
   If finds NAME-NUMBER-VERSION-PROTOCOL match in *RPC-PROGRAMS*, deletes.
   If finds NUMBER-VERSION match with NAME mismatch, asks first.
   If deletes something, returns NAME of DELETED program, otherwise NIL."
                                                                    ;
   (LET ((RPC (FIND-RPC-PROGRAM :NUMBER NUMBER :VERSION VERSION :NAME NAME :PROTOCOL PROTOCOL)))
        (IF RPC
            (IF (OR (EQL NAME (RPC-PROGRAM-NAME RPC))
                    (YES-OR-NO-P "Do you really want to remove/overwrite RPC program ~a?" (RPC-PROGRAM-NAME
                                                                                            RPC)))
                (PROGN (SETQ *RPC-PROGRAMS* (DELETE RPC *RPC-PROGRAMS*))
                       (RPC-PROGRAM-NAME RPC))))))
```

```
(DEFUN XDR-GENCODE-MAKEFCN (CONTEXT TYPEDEF OPER &OPTIONAL COMPILESW)
   "
   Calls XDR-CODEGEN to generate an XDR function for TYPEDEF.
   If COMPILESW, then compiles the function. COMPILESW is not
   used anymore since DEFINE-REMOTE-PROGRAM became a macro.
   "
   (LET ((CODE (XDR-CODEGEN CONTEXT TYPEDEF OPER)))
        (IF COMPILESW
            (COMPILE NIL CODE)
            CODE)))
```

```
(DEFMACRO XDR-GENCODE-INLINE (CONTEXT TYPEDEF OPER &REST VARS)
   "NIL"
   ;; Note that using a NIL context is valid here. It just means that no typedefs from other Remote Program Definitions are available.
   "NIL"
   `(FUNCALL #',(XDR-CODEGEN CONTEXT (EVAL TYPEDEF)
                             (EVAL OPER))
             ,.VARS))
```

;;; Remote Procedure Call

```
(DEFUN REMOTE-PROCEDURE-CALL (DESTINATION PROGRAM PROCID ARGLIST &KEY (PROTOCOL 'UDP)
                                          REMOTESOCKET VERSION CREDENTIALS DYNAMIC-PROGNUM (DYNAMIC-VERSION
                                                                                            1)
```

```
                                        (ERRORFLG T)
                                        LEAVE-STREAM-OPEN
                                        (MSEC-UNTIL-TIMEOUT *MSEC-UNTIL-TIMEOUT*)
                                        (MSEC-BETWEEN-TRIES *MSEC-BETWEEN-TRIES*)
                                        RESULTS)
   "
   This is the high-level way of making a remote procedure call (PERFORM-RPC is the low-level
   way).

   REMOTE-PROCEDURE-CALL resolves all the arguments, creates a new RPC-STREAM, makes the call, optionally closes
   the RPC-STREAM, and returns the results of the call.

   The resolution of arguments is designed such that all arguments may be either
   unresolved (e.g., a remote host name), or already resolved (e.g., an IP address).
   "
   (WHEN (NUMBERP *DEBUG*)
       (FORMAT-T "Remote-Procedure-Call...~%")
       (FORMAT-T "  Destination=~A~%" DESTINATION)
       (FORMAT-T "  Program=~A~%" PROGRAM)
       (FORMAT-T "  ProcID=~A~%" PROCID)
       (FORMAT-T "  ArgList=~A~%" ARGLIST))
   (MULTIPLE-VALUE-BIND (DESTADDR DESTSOCKET RPROG RPROC RPCSTREAM)
       (SETUP-RPC DESTINATION PROGRAM PROCID REMOTESOCKET VERSION DYNAMIC-PROGNUM DYNAMIC-VERSION PROTOCOL)
     (SETQ RPCSTREAM (OPEN-RPCSTREAM (RPC-PROGRAM-PROTOCOL RPROG)
                                     DESTADDR DESTSOCKET))
     (SETQ RESULTS (PERFORM-RPC DESTADDR DESTSOCKET RPROG RPROC RPCSTREAM ARGLIST CREDENTIALS :ERRORFLG
                                ERRORFLG :MSEC-UNTIL-TIMEOUT MSEC-UNTIL-TIMEOUT :MSEC-BETWEEN-TRIES MSEC-BETWEEN-TRIES
                                ))
     (UNLESS LEAVE-STREAM-OPEN (CLOSE-RPCSTREAM RPCSTREAM))
     RESULTS))


(DEFUN SETUP-RPC (DESTINATION PROGRAM PROCID &OPTIONAL DESTSOCKET VERSION DYNAMIC-PROGNUM DYNAMIC-VERSION
                              (PROTOCOL 'UDP))
   "
   Resolves arguments to REMOTE-PROCEDURE-CALL. Takes arguments in more or less
   any reasonable form and returns multiple values (destination-address, socket-number,
   RPC-PROGRAM struct, RPC-PROCEDURE struct).

   See individual RPC-RESOLVE-* programs for details on what inputs are acceptable.
   "
   (LET* ((DESTADDR (RPC-RESOLVE-HOST DESTINATION))
          (RPROG (RPC-RESOLVE-PROG PROGRAM VERSION PROTOCOL))
          (DUMMY                                              ; This code may set RPROG
                 (WHEN DYNAMIC-PROGNUM
                     (SETF RPROG (COPY-RPC-PROGRAM RPROG))
                     (SETF (RPC-PROGRAM-NUMBER RPROG)
                           DYNAMIC-PROGNUM)
                     (SETF (RPC-PROGRAM-VERSION RPROG)
                           DYNAMIC-VERSION)))
          (RPROC (RPC-RESOLVE-PROC RPROG PROCID))
          (SOCKET (OR DESTSOCKET (RPC-FIND-SOCKET DESTADDR RPROG (RPC-PROGRAM-PROTOCOL RPROG)))))
         (VALUES DESTADDR SOCKET RPROG RPROC)))


(DEFUN PERFORM-RPC (DESTADDR DESTSOCKET RPROG RPROC STREAM ARGLIST CREDENTIALS &KEY (ERRORFLG T)
                             (MSEC-UNTIL-TIMEOUT *MSEC-UNTIL-TIMEOUT*)
                             (MSEC-BETWEEN-TRIES *MSEC-BETWEEN-TRIES*))
   "
   The low-level remote procedure call function.
   "
   (LET (RETVALS)
        (REINITIALIZE-RPCSTREAM STREAM DESTADDR DESTSOCKET)
        (PROGN ;; These are for debugging printouts only
               (SETQ *RPCSTREAM* STREAM)
               (SETQ *RPC-PGNAME* (RPC-PROGRAM-NAME RPROG))
               (SETQ *RPC-PCNAME* (RPC-PROCEDURE-NAME RPROC)))
        (XDR-UNSIGNED STREAM (CREATE-XID))
        (XDR-UNSIGNED STREAM *RPC-CALL*)
        (XDR-UNSIGNED STREAM *RPC-VERSION*)
        (XDR-UNSIGNED STREAM (RPC-PROGRAM-NUMBER RPROG))
        (XDR-UNSIGNED STREAM (RPC-PROGRAM-VERSION RPROG))
        (XDR-UNSIGNED STREAM (RPC-PROCEDURE-PROCNUM RPROC))
        (ENCODE-AUTHENTICATION STREAM CREDENTIALS)
        (ENCODE-AUTHENTICATION STREAM *NULL-AUTHENTICATION*)
        (ENCODE-RPC-ARGS STREAM ARGLIST RPROC)
        (SETQ RETVALS (CATCH 'GOFORIT
                             (ACTUALLY-DO-THE-RPC STREAM MSEC-UNTIL-TIMEOUT MSEC-BETWEEN-TRIES ERRORFLG)
                             (PARSE-RPC-REPLY STREAM (RPC-PROCEDURE-RESULTTYPES RPROC)
                                              ERRORFLG)))
        (WHEN (AND (NUMBERP *DEBUG*)
                   (> *DEBUG* 0))
              (FORMAT-T " Values Returned by RPC: ~A~%" RETVALS))
        RETVALS))
```

```
(DEFUN RPC-RESOLVE-HOST (DESTINATION)
   "
   Takes an IPADDRESS, symbol, or string and tries to find an IPADDRESS for a remote
   host. Signals an error if it cannot resolve the host.
   "
   (OR (TYPECASE DESTINATION
          (NUMBER DESTINATION)
          (SYMBOL (IF *USE-OS-NETWORKING*
                      (OS-RESOLVE-HOST (STRING DESTINATION))
                      (IL:IPHOSTADDRESS DESTINATION)))
          (STRING (IF *USE-OS-NETWORKING*
                      (OS-RESOLVE-HOST DESTINATION)
                      (IL:IPHOSTADDRESS (INTERN DESTINATION))))
          (T (IL:\\ILLEGAL.ARG DESTINATION)))
       (ERROR "Could not find an IP net address for DESTINATION ~A" DESTINATION)))

(DEFUN RPC-RESOLVE-PROG (PROGRAM &OPTIONAL VERSION PROTOCOL)
   "
   Takes an RPC-PROGRAM, a number, a symbol, or a string along with an optional VERSION and PROTOCOL and tries
   to find the matching RPC-PROGRAM.
   Signals an error if it cannot find the intended program.
   "
   (COND
      ((TYPEP PROGRAM 'RPC-PROGRAM)
       PROGRAM)
      ((AND (TYPEP PROGRAM 'SYMBOL)
            (FIND-RPC-PROGRAM :NAME PROGRAM :VERSION VERSION :PROTOCOL PROTOCOL)))
      ((AND (NUMBERP PROGRAM)
            (FIND-RPC-PROGRAM :NUMBER PROGRAM :VERSION VERSION :PROTOCOL PROTOCOL)))
      ((AND (STRINGP PROGRAM)
            (FIND-RPC-PROGRAM :NAME (INTERN PROGRAM)
                  :VERSION VERSION :PROTOCOL PROTOCOL)))
      (T (ERROR "Could not find definition for program ~a~a~a.~%" PROGRAM (IF VERSION
                                                                              (FORMAT NIL ", version ~a" VERSION
                                                                                     )
                                                                              "")
                (IF PROTOCOL
                    (FORMAT NIL ", protocol ~a" PROTOCOL)
                    "")))))

(DEFUN RPC-RESOLVE-PROC (RPROG PROCID)
   "
   Given an RPC-PROGRAM struct RPROG, tries to find and return an RPC-PROCEDURE in
   RPROG specified by a number, string,  symbol, or RPC-PROCEDURE.

   Signals an error if it cannot find the intended rpc-procedure
   "
   (COND
      ((TYPEP PROCID 'RPC-PROCEDURE)
       PROCID)
      ((AND (OR (NUMBERP PROCID)
                (SYMBOLP PROCID))
            (FIND-RPC-PROCEDURE (RPC-PROGRAM-PROCEDURES RPROG)
                  PROCID)))
      ((AND (STRINGP PROCID)
            (FIND-RPC-PROCEDURE (RPC-PROGRAM-PROCEDURES RPROG)
                  (INTERN PROCID))))
      (T (ERROR "Could not find definition for program ~a, procedure ~a~%" (RPC-PROGRAM-NAME RPROG)
                PROCID))))

(DEFUN RPC-FIND-SOCKET (DESTADDR PRG PROTOCOL)
   "
   Tries to find and return a remote socket number.

   (1) Looks in *RPC-WELL-KNOWN-SOCKETS*,
   (2) Looks in *RPC-SOCKET-CACHE*, but only if *RPC-OK-TO-CACHE*,
   (3) Requests socket number via remote procedure call to Portmapper
   on remote machine. If found and *RPC-OK-TO-CACHE*, caches the new
   socket number on *RPC-SOCKET-CACHE*.
   (4) If all the above have failed, signals an error.
   "
   (LET ((PROGNUM (RPC-PROGRAM-NUMBER PRG))
         (PROGVERS (RPC-PROGRAM-VERSION PRG))
         SKT)
        (COND
           ((SETQ SKT (FIND-CACHED-SOCKET '* PROGNUM PROGVERS PROTOCOL *RPC-WELL-KNOWN-SOCKETS*))
            (IF *DEBUG*
                (FORMAT-T "Cached well-known socket ~a found for program ~a~%" SKT (RPC-PROGRAM-NAME PRG)))
            SKT)
           ((AND *RPC-OK-TO-CACHE* (SETQ SKT (FIND-CACHED-SOCKET DESTADDR PROGNUM PROGVERS PROTOCOL
                                                 *RPC-SOCKET-CACHE*)))
            (IF *DEBUG*
                (FORMAT-T "Cached non-well-known socket ~a found for program ~a~%" SKT (RPC-PROGRAM-NAME PRG)))
            SKT)
```

```
          ((PROGN (IF *DEBUG*
                      (FORMAT-T "Looking up socket for program ~a on ~a.~%" (RPC-PROGRAM-NAME PRG)
                             DESTADDR))
                  (SETQ SKT (FIRST (REMOTE-PROCEDURE-CALL DESTADDR 'PORTMAPPER 'LOOKUP
                                                          `(,(RPC-PROGRAM-NUMBER PRG)
                                                            ,(RPC-PROGRAM-VERSION PRG)
                                                            ,(GET-PROTOCOL-NUMBER PROTOCOL)
                                                            0)
                                                          :REMOTESOCKET 111)))
                  (IF *DEBUG*
                      (FORMAT-T "Socket ~a found via portampper on ~a for program ~a~%" SKT DESTADDR
                             (RPC-PROGRAM-NAME PRG)))
                  (IF (AND *RPC-OK-TO-CACHE* (> SKT 0))
                      (PUSH `(,DESTADDR ,PROGNUM ,PROGVERS ,PROTOCOL ,SKT)
                            *RPC-SOCKET-CACHE*)
                      SKT)
                  (IF (> SKT 0)
                      SKT)))
           ((ERROR "Could not find remote socket number for~%~
                    Host ~a, Remote Program ~a, Number ~a, Version ~a, Protocol ~a" DESTADDR (RPC-PROGRAM-NAME
                                                                                              PRG)
                  PROGNUM PROGVERS PROTOCOL)))))

(DEFUN ENCODE-RPC-ARGS (STREAM ARGLIST RPC-PROC)
   "
   Takes a list of arguments and the corresponding list of XDR procedures and
   converts the arguments into XDR, writing them into the RPC-STREAM.
   "
   (WHEN (AND (NUMBERP *DEBUG*)
              (> *DEBUG* 0))
         (FORMAT-T "  RPC Arguments: ~A~%" ARGLIST))
   (DO ((XDR-FNS (RPC-PROCEDURE-ARGTYPES RPC-PROC)
                 (REST XDR-FNS))
        (ARGS ARGLIST (REST ARGS)))
       ((OR (NULL ARGS)
            (NULL XDR-FNS))
        (IF (OR XDR-FNS ARGS)
            (ERROR "Mismatch of arguments and parameters to RPC call.~
                                Number or arguments:~a, Number of parameters:~a" (LENGTH ARGLIST)
                   (LENGTH (RPC-PROCEDURE-ARGTYPES RPC-PROC)))
            (RPC-PROCEDURE-NAME RPC-PROC)))
     (FUNCALL (FIRST XDR-FNS)
              STREAM
              (FIRST ARGS))))

(DEFUN ACTUALLY-DO-THE-RPC (STREAM MSEC-UNTIL-TIMEOUT MSEC-BETWEEN-TRIES ERRORFLG)
   "
   Calls the appropriate function (for the protocol) to actually send the packets over
   the net and await an answer.
   "
   (ECASE (RPC-STREAM-PROTOCOL STREAM)
       (UDP (IF *USE-OS-NETWORKING*
                (OS-EXCHANGE-UDP-PACKETS STREAM MSEC-UNTIL-TIMEOUT MSEC-BETWEEN-TRIES ERRORFLG)
                (EXCHANGE-UDP-PACKETS STREAM MSEC-UNTIL-TIMEOUT MSEC-BETWEEN-TRIES ERRORFLG)))
       (TCP (EXCHANGE-TCP-PACKETS STREAM MSEC-UNTIL-TIMEOUT ERRORFLG))))

(DEFUN EXCHANGE-UDP-PACKETS (STREAM MSEC-UNTIL-TIMEOUT MSEC-BETWEEN-TRIES ERRORFLG)
   "
   Given the specified timeout and time between tries, this routine continues
   to send out UDP packets until it either gets a reply or times out.
   "
   (IF (AND (NUMBERP *DEBUG*)
            (> *DEBUG* 5))
       (BREAK "Packet ready to go from PACKET of *RPCSTREAM*"))
   (DO* ((INIT-TIME (GET-INTERNAL-REAL-TIME))
         (FINAL-TIME (+ INIT-TIME (* MSEC-UNTIL-TIMEOUT *INTERNAL-TIME-UNITS-PER-MSEC*))))
        ((>= (GET-INTERNAL-REAL-TIME)
             FINAL-TIME)
         (CASE ERRORFLG
             (:NOERRORS (THROW 'GOFORIT NIL))
             (:RETURNERRORS (THROW 'GOFORIT '(ERROR TIMEOUT)))
             (OTHERWISE (ERROR "Timeout of RPC Call"))))
      (WHEN *DEBUG* (FORMAT-T "Trying RPC Call: Program ~a, Procedure ~a...~%" *RPC-PGNAME* *RPC-PCNAME*))
      (IF (SETF (RPC-STREAM-INSTREAM STREAM)
                (IL:UDP.EXCHANGE (RPC-STREAM-IPSOCKET STREAM)
                       (RPC-STREAM-OUTSTREAM STREAM)
                       MSEC-BETWEEN-TRIES))
          (PROGN (WHEN *DEBUG*
                     (FORMAT-T "It returned!~%")
                     (AND (NUMBERP *DEBUG*)
                          (> *DEBUG* 5)
                          (BREAK "Reply Packet in INSTREAM of RPC-STREAM *RPCSTREAM*")))
                 (RETURN T)))))
```

```
(DEFUN EXCHANGE-TCP-PACKETS (RPCSTREAM TIMEOUT &OPTIONAL ERRORFLG)
   "
   Given the specified timeout, this routine writes onto the TCP stream and
   waits until it either gets a reply or times out.
   "
   ;; Yes, I know EXCHANGE-TCP-PACKETS is a misnomer, but I wanted it to parallel Exchange-UDP-Packets
   (LET* ((OUTSTRING (RPC-STREAM-OUTSTRING RPCSTREAM))
          (OUTSTREAM (RPC-STREAM-OUTSTREAM RPCSTREAM))
          (INSTREAM (RPC-STREAM-INSTREAM RPCSTREAM))
          (EVENT (IL:TCP.SOCKET.EVENT (IL:TCP.STREAM.SOCKET (RPC-STREAM-OUTSTREAM RPCSTREAM)))))
         (WHEN (NUMBERP *DEBUG*)
             (INSPECT-STRING1 OUTSTRING (RPC-STREAM-OUTBYTEPTR RPCSTREAM))
             (AND (> *DEBUG* 4)
                  (BREAK "Ready to write to tcp stream")))
         (RM-FORCEOUTPUT RPCSTREAM T)
         (IL:FORCEOUTPUT OUTSTREAM T)
         (IF *DEBUG* (FORMAT-T "Output forced out. Will wait ~a msec for reply~%" TIMEOUT))
         (IL:AWAIT.EVENT (IL:TCP.SOCKET.EVENT (IL:TCP.STREAM.SOCKET (RPC-STREAM-OUTSTREAM RPCSTREAM)))
             TIMEOUT NIL)
         (IF (IL:READP INSTREAM)
             (PROGN (IF *DEBUG* (FORMAT-T "It returned!!!!~%"))
                    (RM-NEW-INPUT-RECORD RPCSTREAM)
                    T)
             (CASE ERRORFLG
                 (:NOERRORS (THROW 'GOFORIT NIL))
                 (:RETURNERRORS (THROW 'GOFORIT '(ERROR TIMEOUT)))
                 (OTHERWISE (ERROR "Timeout of TCP Call after ~a msec.~%" TIMEOUT))))))


(DEFUN PARSE-RPC-REPLY (RPCSTREAM RETTYPES &OPTIONAL ERRORFLG)
   "
   Parses a reply message. If all goes well, returns a list of the values returned (or T if RETTYPES is NIL).

   If RPC was REJECTED, or ACCEPTED but with an ACCEPT-STAT other than SUCCESS,
   then  (Following Courier) the response depends on the value of ERRORFLG:
        If ERRORFLG = 'NOERROR, then returns NIL
        If ERRORFLG = 'RETURNERRORS, then returns a list of the form
                (ERROR reply-stat accept-or-reject-stat otherinfo)
        If ERRORFLG = anything else, signals Lisp error.

   "                                                          ;
   (LET (XID MSGTYPE REPLY-STAT VERF ACCEPT-STAT REJECT-STAT)
        (SETQ XID (XDR-UNSIGNED RPCSTREAM))
        (SETQ MSGTYPE (XDR-UNSIGNED RPCSTREAM))
        (IF (NOT (EQL MSGTYPE 1))
            (ERROR "RPC message is not a reply. MSGTYPE is ~A" MSGTYPE))
        (CASE (GET-REPLY-STAT (SETQ REPLY-STAT (XDR-UNSIGNED RPCSTREAM)))
            (ACCEPTED
               (SETQ VERF (DECODE-AUTHENTICATION RPCSTREAM))
               (CASE (GET-ACCEPT-STAT (SETQ ACCEPT-STAT (XDR-UNSIGNED RPCSTREAM)))
                   (SUCCESS (IF (NULL RETTYPES)
                                T
                                (DO ((RS RETTYPES (CDR RS))
                                     (VALS))
                                    ((NULL RS)
                                     (NREVERSE VALS))
                                  (PUSH (FUNCALL (CAR RS)
                                                 RPCSTREAM)
                                        VALS))))
                   (PROGRAM-MISMATCH (RPC-ERROR-PRM-MISMATCH ERRORFLG REPLY-STAT ACCEPT-STAT (XDR-UNSIGNED
                                                                                               RPCSTREAM)
                                                             (XDR-UNSIGNED RPCSTREAM)))
                   (PROGRAM-UNAVAILABLE (RPC-ERROR-PRM-UNAVAILABLE ERRORFLG REPLY-STAT ACCEPT-STAT))
                   (PROCEDURE-UNAVAILABLE (RPC-ERROR-PRC-UNAVAILABLE ERRORFLG REPLY-STAT ACCEPT-STAT))
                   (GARBAGE-ARGUMENTS (RPC-ERROR-GARBAGE-ARGS ERRORFLG REPLY-STAT ACCEPT-STAT))))
            (REJECTED (CASE (GET-REJECT-STAT (SETQ REJECT-STAT (XDR-UNSIGNED RPCSTREAM)))
                          (RPC-MISMATCH (RPC-ERROR-MISMATCH ERRORFLG REPLY-STAT ACCEPT-STAT (XDR-UNSIGNED
                                                                                             RPCSTREAM)
                                                            (XDR-UNSIGNED RPCSTREAM)))
                          (AUTHENTICATION-ERROR (RPC-ERROR-AUTHENTICATION ERRORFLG REPLY-STAT REJECT-STAT
                                                                          (XDR-UNSIGNED RPCSTREAM)))
                          (OTHERWISE (ERROR "Unknown  RPC reply status: ~A" REPLY-STAT)))))))


(DEFUN CREATE-XID ()
   "Returns a number to use as the ID of a given transmisssion."
   (SETQ *XID-COUNT* (LOGAND TWOTO32MINUSONE (+ 1 *XID-COUNT*))))


;;; RPC Utility Functions

(DEFUN GET-REPLY-STAT (NUMBER)
   "Map number to corresponding reply-stat symbol of remote procedure call"
   (CDR (ASSOC NUMBER *RPC-REPLY-STATS*)))
```

```
(DEFUN GET-ACCEPT-STAT (NUMBER)
   "Map number to corresponding accept-stat symbol of remote procedure call"
   (CDR (ASSOC NUMBER *RPC-ACCEPT-STATS*)))

(DEFUN GET-REJECT-STAT (NUMBER)
   "Map number to corresponding reject-stat symbol of remote procedure call"
   (CDR (ASSOC NUMBER *RPC-REJECT-STATS*)))

(DEFUN GET-AUTHENTICATION-STAT (NUMBER)
   "Map number to corresponding authentication-stat symbol of remote procedure call"
   (CDR (ASSOC NUMBER *RPC-AUTHENTICATION-STATS*)))

(DEFUN GET-PROTOCOL-NUMBER (PROTOCOL)
   "Map protocol name (e.g., RPC2::UDP) to corresponding protocol number (e.g., 17)"
   (OR (CDR (ASSOC PROTOCOL *RPC-PROTOCOLS*))
       (ERROR "Could not find number for protocol ~a in *RPC-PROTOCOLS*" PROTOCOL)))

(DEFUN FIND-CACHED-SOCKET (DESTADDR PROGNUM PROGVERS PROTOCOL CACHE)
   "Looks up a given (DESTADDR, PROGNUM, PROGVERS, PROTOCOL) in the specified CACHE."
   (FIFTH (FIND-IF #'(LAMBDA (QUINT)
                           (AND (EQL (FIRST QUINT)
                                     DESTADDR)
                                (EQL (SECOND QUINT)
                                     PROGNUM)
                                (EQL (THIRD QUINT)
                                     PROGVERS)
                                (EQL (FOURTH QUINT)
                                     PROTOCOL)))
                   CACHE)))
```

;;; RPC Error Messages

```
(DEFUN RPC-ERROR-PRM-MISMATCH (ERRORFLG REPLY-STAT ACCEPT-STAT LOW HIGH)
   "NIL"
   (CASE ERRORFLG
       (:NOERRORS NIL)
       (:RETURNERRORS `(ERROR ,(GET-REPLY-STAT REPLY-STAT)
                              ,(GET-ACCEPT-STAT ACCEPT-STAT)
                              `(,LOW ,HIGH)))
       (OTHERWISE (ERROR "RPC Program Mismatch: High: ~A Low: ~A" LOW HIGH))))

(DEFUN RPC-ERROR-PRM-UNAVAILABLE (ERRORFLG REPLY-STAT ACCEPT-STAT)
   "NIL"
   (CASE ERRORFLG
       (:NOERRORS NIL)
       (:RETURNERRORS `(ERROR ,(GET-REPLY-STAT REPLY-STAT)
                              ,(GET-ACCEPT-STAT ACCEPT-STAT)))
       (OTHERWISE (ERROR "RPC Program Unavailable"))))

(DEFUN RPC-ERROR-PRC-UNAVAILABLE (ERRORFLG REPLY-STAT ACCEPT-STAT)
   "NIL"
   (CASE ERRORFLG
       (:NOERRORS NIL)
       (:RETURNERRORS `(ERROR ,(GET-REPLY-STAT REPLY-STAT)
                              ,(GET-ACCEPT-STAT ACCEPT-STAT)))
       (OTHERWISE (ERROR "RPC Procedure Unavailable"))))

(DEFUN RPC-ERROR-GARBAGE-ARGS (ERRORFLG REPLY-STAT ACCEPT-STAT)
   "NIL"
   (CASE ERRORFLG
       (:NOERRORS NIL)
       (:RETURNERRORS `(ERROR ,(GET-REPLY-STAT REPLY-STAT)
                              ,(GET-ACCEPT-STAT ACCEPT-STAT)))
       (OTHERWISE (ERROR "RPC Garbage Arguments"))))

(DEFUN RPC-ERROR-MISMATCH (ERRORFLG REPLY-STAT REJECT-STAT LOW HIGH)
   "NIL"
   (CASE ERRORFLG
       (:NOERRORS NIL)
       (:RETURNERRORS `(ERROR ,(GET-REPLY-STAT REPLY-STAT)
                              ,(GET-REJECT-STAT REJECT-STAT)
                              `(,LOW ,HIGH)))
       (OTHERWISE (ERROR "RPC Mismatch: High: ~A Low: ~A" LOW HIGH))))
```

```
(DEFUN RPC-ERROR-AUTHENTICATION (ERRORFLG REPLY-STAT REJECT-STAT AUTHENTICATION-STAT)
   "NIL"
   (CASE ERRORFLG
       (:NOERRORS NIL)
       (:RETURNERRORS `(ERROR ,(GET-REPLY-STAT REPLY-STAT)
                              ,(GET-REJECT-STAT REJECT-STAT)
                              ,(GET-AUTHENTICATION-STAT AUTHENTICATION-STAT)))
       (OTHERWISE (ERROR "Authorization Error: ~A" (GET-AUTHENTICATION-STAT AUTHENTICATION-STAT)))))
```

;;; Authentication

```
(DEFCONSTANT *AUTHENTICATION-TYPEDEF*
   '(:STRUCT AUTHENTICATION (TYPE (:ENUMERATION (:NULL 0)
                                                (:UNIX 1)
                                                (:SHORT 2)))
          (STRING :STRING))
   "NIL")

(DEFCONSTANT *NULL-AUTHENTICATION* (MAKE-AUTHENTICATION :TYPE :NULL :STRING ""))

(DEFUN CREATE-UNIX-AUTHENTICATION (STAMP MACHINE-NAME UID GID GIDS)
   "
   Given the fields of a Unix authentication, creates an AUTHENTICATION struct with
   these fields encoded as a string.
   "
   (LET ((UNIX-AUTH (MAKE-AUTHENTICATION))
         (TEMPSTREAM (CREATE-STRING-RPC-STREAM)))
        (XDR-UNSIGNED TEMPSTREAM STAMP)
        (XDR-STRING TEMPSTREAM MACHINE-NAME)
        (XDR-UNSIGNED TEMPSTREAM UID)
        (XDR-UNSIGNED TEMPSTREAM GID)
        (XDR-GENCODE-INLINE NIL '(:COUNTED-ARRAY :UNSIGNED)
               'WRITE TEMPSTREAM GIDS)
        (SETF (AUTHENTICATION-TYPE UNIX-AUTH)
              :UNIX)
        (SETF (AUTHENTICATION-STRING UNIX-AUTH)
              (GET-OUTPUT-STREAM-STRING (RPC-STREAM-OUTSTREAM TEMPSTREAM)))
        UNIX-AUTH))

(DEFUN ENCODE-AUTHENTICATION (RPCSTREAM AUTH)
   "
   Given an AUTHENTICATION struct, converts the struct to its XDR encoding and writes it to
   the RPC-STREAM specified.
   "
   (IF (NULL AUTH)
       (SETQ AUTH *NULL-AUTHENTICATION*))
   (CHECK-TYPE AUTH AUTHENTICATION)
   (XDR-GENCODE-INLINE NIL *AUTHENTICATION-TYPEDEF* 'WRITE RPCSTREAM AUTH))

(DEFUN DECODE-AUTHENTICATION (RPCSTREAM)
   "
   Reads an authentication from specified RPC-STREAM and returns it as an AUTHENTICATION
   struct.
   "
   (XDR-GENCODE-INLINE NIL *AUTHENTICATION-TYPEDEF* 'READ RPCSTREAM))

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

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

MACRO INDEX

PROPERTY INDEX