0.1 HIGHER-LEVEL NS PROTOCOL FUNCTIONS

The following is a description of the Interlisp-D facilities for using Xerox SPP and Courier protocols and the services based on them.

0.1.1 SPP Stream Interface

This section describes the stream interface to the Sequenced Packet Protocol.

(SPP.OPEN HOST SOCKET PROBEP NAME) [Function]

This function is used to open an SPP stream. If HOST is specified, an SPP connection is initiated to HOST with remote socket SOCKET. If both HOST and PROBEP are specified, then the connection is probed for a response before returning the stream; NIL is returned if HOST doesn't respond. If HOST is NIL, a passive connection is created which listens for an incoming connection to local socket SOCKET. NAME is a mnemonic name for the connection process, mainly useful for debugging. The function returns an SPP stream, for which the standard stream operations BIN, BOUT, CLOSEF, and EOFP are defined. In particular, COPYBYTES may be used on SPP streams.

The SPP stream that is returned is open for both input and output, since SPP connections are bidirectional. However, the underlying stream I/O functions use only a single buffer. Some care must therefore be exercised to insure that any buffered output data is forced out before any new data is read, and that all data up to a message boundary has been read before any new data is written. Functions described below are used for this purpose. While these restrictions may seem severe, in practice most use of SPP streams is done by the Courier remote procedure call facility, rather than directly by the programmer. Courier conforms to the model of alternating exchanges of messages quite well.

SPP.USER.TIMEOUT [Variable]

Specifies the time, in milliseconds, to wait before deciding that a host isn't responding.

(SPP.FLUSH STREAM) [Function]

This function forces any buffered output data to be transmitted.

(SPP.SENDEOM STREAM) [Function]

This function forces out any buffered data and causes an End of Message indication to be sent.

(SPP.CLOSE STREAM ABORT?) [Function]

This function closes an SPP stream using the reliable termination protocol. If ABORT? is not NIL, the stream is closed even if there is an outstanding bulk data transfer in progress.

(SPP.DSTYPE STREAM DSTYPE) [Function]

This function gets or sets the current datastream type. If DSTYPE is specified, all subsequent packets that are sent will be of this datastream type, until the next call to SPP.DSTYPE. Since this affects the current partially-filled packet, the stream

should probably be ushed (via SPP.FLUSH) before this function is called. If DSTYPE is not speci ed, this function returns the datastream type of the current packet being read.

(SPP.READP STREAM) [Function]
This function returns T or NIL depending on whether or not there is data to be read without waiting.

(SPP.EOFP STREAM) [Function]
This function returns T or NIL depending on whether or not the connection has been closed.

(SPP.EOMP STREAM) [Function]
This function returns T or NIL depending on whether or not an End of Message indication has been reached. This will only be true after the last byte of data in the message has been read.

0.1.2 Courier Remote Procedure Call Protocol

(COURIER.OPEN HOSTNAME SERVER TYPE NOERR ORFL G NAME) [Function]
This function opens a Courier connection to the speci ed HOST and returns an SPP stream. If HOST is a LITATOM, string, or list representation of a Clearinghouse name, SERVER TYPE should specify what type of server HOST is, so that the name may be looked up in the Clearinghouse database. Currently, SERVER TYPE must be one of PRINTSERVER or FILESERVER. Normally, this function will retry the connection \MAXETHERTRIES times before generating an error. If NOERR ORFL G is speci ed, NIL will be returned if the connection fails. The Courier connection will be given NAME , if speci ed.

(COURIERPROGRAM NAME . . .) [NLambda NoSpread Function]
This function is used to de ne Courier programs. The syntax is

```
(COURIERPROGRAM name (programNumber versionNumber)
   TYPES
   ((typeName typeDefinition)
    ...)
   PROCEDURES
   ((procedureName ARGS (argType ...)
                   RESULTS (resultType ...)
                   ERRORS (errorName ...)
                   procedureNumber)
    ...)
   ERRORS
   ((errorName ARGS (argType ...) errorNumber)
    ...))
)
```

Type de nitions are written in the Courier template language, described below.

Courier types may either be type names that are defined in the current Courier program, qualified names of the form `(otherCourierProgram . typeName)`, or explicit definitions in the template language.

0.1.2.1 Courier Template Language

This section describes how Courier types are described in Interlisp, and how corresponding values are represented. (See also the Courier protocol definition.)

Predefined types:

`BOOLEAN` is represented by `T` and `NIL`. `CARDINAL` is represented by integers. `INTEGER` is represented by integers. `LONGCARDINAL` is represented by integers. `LONGINTEGER` is represented by integers. `STRING` is represented by strings. `UNSPECIFIED` is represented by integers.

Constructed types:

`(ENUMERATION (NAME VALUE) ... (NAME VALUE))` `(ARRAY LENGTH TYPE)` `(SEQUENCE TYPE)` `(RECORD (NAME TYPE) ... (NAME TYPE))` `(CHOICE (NAME VALUE TYPE) ... (NAME VALUE TYPE))`

Representation of constructed types in Lisp:

Objects of Courier type `(ENUMERATION (UNKNOWN 0) (RED 1) (BLUE 2))` are represented by the LITATOMs `UNKNOWN`, `RED`, and `BLUE`.

Objects of Courier type `(ARRAY 3 INTEGER)` are represented by lists of three integers, such as `(10 1 59)`.

Objects of Courier type `(SEQUENCE BOOLEAN)` are represented by arbitrary-length lists of `T` and `NIL`, such as `(NIL T T NIL T)`.

Objects of Courier type

```
(RECORD (NETWORK LONGCARDINAL)
        (HOST (ARRAY 3 CARDINAL))
        (SOCKET CARDINAL))
```

are represented by lists like `((NETWORK 174) (HOST (100 24 363)) (SOCKET 20))`.

Objects of Courier type

```
(CHOICE (STATUS 0 (ENUMERATION (BUSY 0) (COMPLETE 1)))
        (MESSAGE 1 STRING))
```

are represented by lists like `(STATUS COMPLETE)` or `(MESSAGE "Your request has completed.")`.

`(COURIER.CALL STREAM PROGRAM PROCEDURE ARG1 ...ARGn NOERRORFLG)`
[NoSpread Function]

This function calls the remote procedure `PROCEDURE` of the Courier program

PROGRAM.STREAM is the SPP stream returned by COURIER.OPEN. The arguments should be Lisp values appropriate for the Courier types of the corresponding formal parameters of the procedure (defined under the ARGS property for the procedure). Returns results of the Courier types defined under the RESULTS property. If there is only a single result, it is returned, otherwise a list of results is returned. The NOERRORFLG argument controls the treatment of remote errors. If NOERRORFLG is NIL, a Lisp error will be generated. If NOERRORFLG is T, NIL will be returned as the result of the call. If NOERRORFLG is RETURNERRORS, the result of the call will be a list consisting of the atom ERROR followed by the Courier name of the error and any arguments.

Examples:

```
(COURIERPROGRAM EXAMPLEPROGRAM (17 1)
   TYPES
   ((PERSON.NAME (RECORD (FIRST.NAME STRING)
                         (MIDDLE (CHOICE
                                   (NAME 0 STRING)
                                   (INITIAL 1 STRING)))
                         (LAST.NAME STRING)))
    (BIRTHDAY (RECORD (YEAR CARDINAL)
                      (MONTH STRING)
                      (DAY CARDINAL))))
   PROCEDURES
   ((GETBIRTHDAY ARGS (PERSON.NAME)
                 RESULTS (BIRTHDAY)
                 3))
)
```

Defines EXAMPLEPROGRAM to be Courier program number 17, version number 1. The example defines two types, PERSON.NAME and BIRTHDAY, and one procedure, GETBIRTHDAY, whose procedure number is 3. The following code could be used to call the remote GETBIRTHDAY procedure on the host with address HOSTADDRESS.

```
(SETQ STREAM (COURIER.OPEN HOSTADDRESS))
(COURIER.CALL STREAM
              (QUOTE EXAMPLEPROGRAM)
              (QUOTE GETBIRTHDAY)
              (QUOTE ((FIRST.NAME "Eric")
                      (MIDDLE (INITIAL "C"))
                      (LAST.NAME "Cooper"))))
```

COURIER.CALL in this example will return a value such as

```
((YEAR 1959) (MONTH "January") (DAY 10))
```

0.1.2.2 Manipulating Courier Representations

Several Courier programs use values of type (SEQUENCE UNSPECIFIED) to handle user-defined or otherwise extensible object types. Often it is necessary to convert between a list of 16 bit words (the sequence of UNSPECIFIEDs) and a Courier value. The following function should be used for this purpose.

(COURIER.READ.REP LIST.OF.WORDS PROGRAM TYPE) [Function]
This function returns the Lisp representation of the Courier object of type TYPE defined in the Courier program PROGRAM whose underlying Courier representation is LIST.OF.WORDS.

0.1.2.3 Using Bulk Data Transfer with Courier

Two Courier types are treated specially when they appear in the argument list of a procedure. They are BULK.DATA.SINK and BULK.DATA.SOURCE. A Courier procedure may have at most one such sink or source parameter. The result of a COURIER.CALL on such a procedure is an SPP stream, open for input or output according to whether the bulk data paramter is a sink or a source. The client uses this stream to receive or send the appropriate bulk data object. If the object consists of bytes, this may be done with the usual stream I/O functions such as COPYBYTES. If the data is a stream of Courier objects, the following function should be used.

(COURIER.READ.BULKDATA STREAM PROGRAM TYPE) [Function]
STREAM is the bulk data stream returned from COURIER.CALL. TYPE is the type of each Courier object in the stream. PROGRAM is the Courier program in which TYPE is defined. A list of objects of Courier type TYPE will be returned.

The observant reader may wonder what happens if the Courier procedure returns one or more results, in addition to taking a bulk data parameter. If a bulk data stream is returned to the caller, what happens to the results? The answer is that the results are collected when the bulk data stream is closed, after the client has transferred the bulk data. The disposition of these results depends on what actual parameter is supplied for the formal bulk data parameter at the time of the call. If it is NIL, the results, if any, will be ignored. Otherwise, the value is assumed to be a function which to be applied to the results. A FUNARG may be used for full generality.

For example, the Courier procedure to print an Interpress master uses a bulk data source to transfer the master, and also returns a request identifier. The Lisp function which performs the COURIER.CALL passes a functional to be called on this request identifier after the stream is closed and printing begins; this functional in turn spawns a process which monitors the progress of the job.

(COURIERTRACE FLG REGION) [Function]
This function controls the tracing of Courier remote procedure calls. It is similar to PUPTRACE and XIPTRACE, but operates at the call/return level rather than the packet level.

0.1.3 NS Printing

This section describes the facilities that are available for printing Interpress masters on NS printservers.

NS.DEFAULT.PRINTER [Variable]

The value of this variable is used whenever no printserver is specified for the functions described below. If its value is a LITATOM, string, or Clearinghouse name, the Clearinghouse is queried to find the address of the printserver with that name. If its value is NIL, it will be set automatically to some printserver in the local Clearinghouse domain. In environments where there is no Clearinghouse, the value of NS.DEFAULT.PRINTER must be an appropriate NSADDRESS record.

(OPEN.NS.PRINTING.STREAM PRINTER DOCUMENT.NAME DOCUMENT.CREATION.DATE SENDER.NAME RECIPIENT.NAME qCOPIES MEDIUM PRIORITY STAPLE? TWOSIDED? NOWATCHDOG?) [Function]

This function returns a stream for printing an Interpress master on PRINTER or on NS.DEFAULT.PRINTER as mentioned above. The caller should write the Interpress data to the stream and then close it using CLOSEF. Printing begins after the stream is closed.

DOCUMENT.NAME is the document name to appear on the header page (a string).

DOCUMENT.CREATION.DATE is the creation date to appear on the header page (a Lisp integer date). The default value is the time of the call.

SENDER.NAME is the name of the sender to appear on the header page (a string). The default value is the name of the user.

RECIPIENT.NAME is the name of the recipient to appear on the header page (a string). The default value is the name of the user.

qCOPIES is the number of copies to be printed. The default value is 1.

MEDIUM is the medium on which the master is to be printed. This must be a Courier value of type MEDIUM, which is a list of the form (PAPER (KNOWN.SIZE NAME)), where NAME is one of the LITATOMs US.LETTER, US.LEGAL, A0 through A10, ISO.B0 through ISO.B10, and JIS.B0 through JIS.B10. The default value is determined by the printer.

PRIORITY is the priority of this print request (LOW, NORMAL, or HIGH). The default value is NORMAL.

STAPLE? is T or NIL depending on whether the document should be stapled. The default value is NIL.

TWOSIDED? is T or NIL depending on whether the document should be printed on two sides. The default value is NIL.

NOWATCHDOG? is non-NIL if the client does not want a watchdog process to monitor the status of the printing job.

(NSPRINT PRINTER FILE.NAME DOCUMENT.NAME DOCUMENT.CREA TION.DATE SENDER.NAME RECIPIENT.NAME q COPIES MEDIUM PRIORITY STAPLE? TWO.SIDED?) [Function]

This function prints an Interpress master on PRINTER or on NS.DEFAULT.PRINTER as mentioned above. FILE.NAME should be the name of an Interpress file to be printed. The remaining arguments are all optional, and are as described for OPEN.NS.PRINTING.STREAM above. DOCUMENT.NAME defaults to the full name of the file, and DOCUMENT.CREA TION.DATE defaults to the creation date of the file.

(NSPRINTER.STATUS PRINTER) [Function]

This function returns the Courier value resulting from the GET.PRINTER.STATUS call.

(NSPRINTER.PROPERTIES PRINTER) [Function]

This function returns the Courier value resulting from the GET.PRINTER.PROPERTIES call.

0.1.4 Clearinghouse

This section describes functions that may be used to access Clearinghouse servers. Note that these functions are used by the NS printing functions if the printserver is specified by name rather than address.

(START.CLEARINGHOUSE REST AR TFL G) [Function]

This function enables Clearinghouse access. It performs an expanding ring broadcast in order to find the first Clearinghouse server. If REST AR TFL G is non-NIL, the cache of Clearinghouse information is invalidated and a new broadcast is done. This may be necessary if the local Clearinghouse server goes down.

CH.NET.HINT [Variable]

Hint as to which network the local Clearinghouse server is on, for use by START.CLEARINGHOUSE above. If CH.NET.HINT is bound to a network number, that network will be tried first, followed by the others in the routing table. If the local Clearinghouse server is not on the directly connected network, setting CH.NET.HINT to the proper network number in the local INIT file will speed up START.CLEARINGHOUSE considerably.

(SHOW.CLEARINGHOUSE) [Function]

This function displays the structure of the cached Clearinghouse information in a window. Once created, it will be redisplayed whenever the cache is updated. The structure is shown using GRAPHER‘‘.

(SHOW.ENTIRE.CLEARINGHOUSE) [Function]

This function attempts to cache information about all the Clearinghouse domains, so that the Clearinghouse structure window will show the entire database.

CH.DEFAULT.DOMAIN [Variable]

This is a string specifying the default Clearinghouse domain. If it is NIL, it will be set automatically by START.CLEARINGHOUSE. Otherwise, it should be set in an INIT file.

CH.DEFAULT.ORGANIZATION [Variable]

This is a string specifying the default Clearinghouse organization. If it is NIL, it will be set automatically by START.CLEARINGHOUSE. Otherwise, it should be set in an INIT file.

(CH.ORGANIZATIONS OR GANIZA TIONP ATTERN) [Function]

This function returns the list of organization names in the Clearinghouse database matching OR GANIZA TIONP ATTERN . The default pattern is "*", which matches anything.

(CH.DOMAINS DOMAINP ATTERN) [Function]

This function returns the list of domain names in the Clearinghouse database matching DOMAINP ATTERN . The default pattern is "*", which matches anything.

(CH.ENUMERATE OBJECTP ATTERN PR OPER TY) [Function]

This function returns the list of object names matching OBJECTP ATTERN and having the property PR OPER TY . Currently, PR OPER TY must be one of USER, PRINTSERVER, FILESERVER, and ALL. For example, (CH.ENUMERATE "*:PARC:Xerox" (QUOTE USER)) will return a list of the names of users at Xerox PARC.

(CH.LOOKUP.USER NAME) [Function]

This function returns the user information for the first user whose name matches NAME .

(LOOKUP.NS.SERVER NAME TYPE) [Function]

This function returns the NSADDRESS for the first server whose name matches NAME and has the property TYPE , which must be PRINTSERVER or FILESERVER.

0.1.5 NS Filing

This section describes functions that may be used to access NS fileservers.

0.1.5.1 Pathnames and NS Fileservers

The NS Filing protocol does not support conventional file system pathnames directly. However, the Interlisp-D software that supports access to NS fileservers uses IFS-style pathnames and does the appropriate mapping in software. One important difference, however, is that fileserver, directory, and file names may have spaces in them, each of which must be preceded by a percent sign. The name of an NS fileserver is required to have a colon in it. Thus, even if the fileserver is in the local Clearinghouse domain, a trailing colon should be appended to the name. Case is not significant. For example, {LISPFILE:}<LISPDRAWER>XYZ;3 is a valid name for a file on the NS fileserver "LispFile:Parc Place:Xerox".

(NSDIRECTORY PATTERN) [Function]

This function returns a list of file names in PATTERN , which must be the NS pathname for a directory. (Any wildcards in the name field of the pathname are ignored.)

(NSCREATEDIRECTORY HOST/DIR) [Function]

This function creates a new directory with pathname HOST/DIR . Top level directories (‘‘ le drawers’’) cannot be created in this way.

(CLOSE.NSFILING.CONNECTIONS) [Function]

This function closes any open connections to NS leservers.

Pathnames and NS Fileservers