COURIERSERVE

By: Christopher Lane (Lane@Sumex-Aim.Stanford.Edu)

COURIERSERVE implements a Courier server process for Envos Lisp allowing other hosts to make Courier calls into the workstation. The server supports both multiple Courier stream connections as well as *expedited* (single packet) and *broadcast* calls.

STARTING A COURIER SERVER

The Courier server can be started by evaluating:

(COURIER.START.SERVER *[RESTART]*) [Function]

Once the server is running, it can be invoked by a remote host client using COURIER.OPEN for a Courier stream connection or by using COURIER.EXPEDITED.CALL or COURIER.BROADCAST.CALL for expedited calls. The functions for making Courier client calls from Lisp are documented in the *Interlisp-D Reference Manual* (pages 31.15–31.26).

(COURIER.RESET.SOCKET) [Function]

(Re)Opens and closes the Courier socket. Not normally a user routine, this function is called by COURIER.START.SERVER but it can be called directly if "*socket already open!*" errors persist on the Courier socket (5).

DEFINING A COURIER SERVER FUNCTION

Defining a Courier server program is identical to defining a client program except for the additional field IMPLEMENTEDBY in each procedure in the PROCEDURES section of the Courier program definition:

```
PROCEDURES
((LAYOUT 0 (GRAPHNODES ROOTIDS FORMAT FONT MOTHERD PERSONALD FAMILYD)
           RETURNS (GRAPH)
           REPORTS (LAYOUT.ERROR)
           IMPLEMENTEDBY GRAPH.REMOTELAYOUT))
```

The order of the RETURNS, REPORTS and IMPLEMENTEDBY fields is significant and should be maintained.

The *server function*, named in the IMPLEMENTEDBY field, is invoked when a Courier call to the procedure is made. The server function is applied to the Courier stream, the Courier program and the Courier procedure followed by the arguments named in the Courier definition. The arguments for GRAPH.REMOTELAYOUT would be (COURIERSTREAM PROGRAM PROCEDURE GRAPHNODES ROOTIDS FORMAT ...).

Note that the COURIERSTREAM, PROGRAM and PROCEDURE arguments are not necessarily used, they are made available for implementing special servers.

RETURNING VALUES FROM A COURIER PROGRAM

Results or errors can be returned by a Courier server function by one of two different methods. In the usual, simple case, the server function can return as its result a list starting with one of RETURN, ABORT or REJECT followed by the appropriate values.

For the RETURN result, the tail of the list should be the results as defined in the Courier procedure definition, eg. `(RETURN 23 "John")`.

For the ABORT result, the tail of the list should contain the reason for the abnormal termination (as defined in the Courier program), followed by any error arguments, eg. `(ABORT NAME.NOT.FOUND "John")`.

For the REJECT result, the tail of the list should contain the rejection error as defined in the Courier standard. The only rejection that should occur inside a server function should be UNSPECIFIED if the program needs to reject for any reason. The other rejection types are handled by the Courier server.

Alternatively, the server function can return results directly to the Courier stream and return NIL as its result. To return results directly to the Courier stream use:

(COURIER.RETURN *COURIERSTREAM PROGRAM PROCEDURE RESULTLST*) [Function]

(COURIER.ABORT *COURIERSTREAM PROGRAM ERROR RESULTLST*) [Function]

(COURIER.REJECT *COURIERSTREAM ERROR RESULTLST*) [Function]

EXPEDITED AND BROADCAST COURIER CALLS

The Courier server allows expedited and broadcast Courier calls. The only difference the server function would see if invoked due to an expedited call is that the Courier stream it is handed is actually a record containing an XIP packet and a socket. If the server function does not use the Courier stream directly, then this difference is invisible.

If the server function actually needs a Courier stream to operate (eg. an NS CHAT server), then it should probably include an USE.COURIER abort error in its definition. If the server function needs a Courier stream due to *bulk data* arguments, this will be trapped in the Courier server itself, which will reject appropriately and not invoke the server function.

USING BULK DATA IN A SERVER FUNCTION

If a server function takes a bulk data argument (either BULK.DATA.SINK or BULK.DATA.SOURCE), it is handed an open bulk data stream for that argument when invoked. If the server function returns a result by returning one of the RETURN or ABORT forms as its result, the bulk data stream will be closed automatically. If the server function returns results directly to the Courier stream using COURIER.RETURN or COURIER.ABORT, then the server function must first close the bulk data stream using:

(CLOSE.BULK.DATA *STREAM [ABORTFLG]*) [Function]

The CLOSEF function does not work on the bulk data stream argument and using it will hang the Courier connection. Only the *immediate* bulk data transfer type is handled. NULL, ACTIVE or PASSIVE bulk data transfer types will cause a Courier rejection of type UNSPECIFIED.

SIMPLE SERVER DEFINITION

Below is the Courier definition for a simple evaluation server. The two functions EVAL.REMOTE and APPLY.REMOTE are all that would need to be defined to make the server run:

```
((1105 0)

 TYPES       ((SEXPR STRING)
              (FN STRING)
              (ARGS (SEQUENCE SEXPR))
              (ERRORN (RECORD (ERROR.NUMBER CARDINAL)
                              (ERROR.MESSAGE SEXPR))))

 PROCEDURES ((EVAL 0 (SEXPR)
                   RETURNS (SEXPR)
                   REPORTS (REMOTE.EVAL.ERROR REMOTE.READ.ERROR)
                   IMPLEMENTEDBY EVAL.REMOTE)
             (APPLY 1 (FN ARGS)
                   RETURNS (SEXPR)
                   REPORTS (REMOTE.APPLY.ERROR REMOTE.READ.ERROR)
                   IMPLEMENTEDBY APPLY.REMOTE))

 ERRORS      ((REMOTE.EVAL.ERROR 0 (ERRORN))
              (REMOTE.APPLY.ERROR 1 (ERRORN))
              (REMOTE.READ.ERROR 2 (ERRORN)))
)
```

RELATED FILES

The modules CHATSERVER-NS, COURIERDEFS, COURIEREVALSERVE, COURIERIMAGESTREAM, MONITOR, REMOTEPSW and NSTALK all define Courier servers and/or Courier type definitions.