0.1 CHAT

CHAT is a ‘‘remote terminal’’ facility, that allows one to communicate with other machines while inside Interlisp- D. The function CHAT sets up a ‘‘Chat connection’’ to a remote machine, so that everything you type is sent to the a remote machine, and everything the remote machine prints is displayed in a ‘‘Chat window’’. The remote machine must support the Pup Telnet protocol.

Multiple simultaneous Chat connections are possible. To switch between typing to di erent Chat connections, simply button within the Chat window you want to use. CHAT prompts for a new window for each new connection, except that it saves the rst window to reuse once the connection in that window is closed (other windows just go away when their connections are closed).

CHAT behaves as if its Chat window is a Datamedia- 2500 terminal of the dimensions determined by the size of the window. Hence, you can talk to hosts that supply Datamedia service and expect something reasonable to happen. If the host does not pay attention to the CHAT terminal speci cation protocol, or you go through that host to another host, you may need to inform the host of the dimensions of your ‘‘screen’’; these are given in the title bar of the chat window. The font should be Gacha10 or other xed- width font for proper Datamedia emulation.

(CHAT HOST LOGOPTION INITSTREAM WINDO W _) [Function]

Opens a Chat connection to HOST , or to the value of DEFAULTCHATHOST. If HOST requires login, as determined by whether it responds to the ‘‘where is user’’ protocol, CHAT supplies a login sequence, or if it determines that you have a single detached job, an attach sequence. If you have more than one detached job, it simply performs a WHEREIS command for you and allows you to select the job. You may alternatively specify one of the following values for LOGOPTION :

LOGIN	Always perform a login.
ATTACH	Always perform an attach. This will fail if you do not have exactly one detached job.
GUEST	Login as user GUEST, password GUEST.
NONE	Do not attempt to login or attach.

If INITSTREAM is supplied, it is either a string or the name of a le whose contents will be read as typein. When the string/ le is exhausted, input is taken from T.

If WINDO W is supplied, it is a window to use for the connection; otherwise, the user is prompted for a window.

While CHAT is in control, all Lisp interrupts are turned o , so that control characters can be transmitted to the remote host.

Commands can be given to an active Chat connection by bugging the MIDDLE button in the Chat window to get a command menu. Current commands are:

Close	Close this connection. Once the connection is closed, control is handed over to the main tty window. Closes the window unless this is the primary Chat window.

Suspend	Same as Close, but always leaves the window open.
New	Closes the current connection and prompts for a new host to which to open a connection in the same window.
Freeze	Hold typeout from this Chat window. Bugging the window in any way releases the hold. This is most useful if you want to switch to another, overlapping window and there is typeout in this window that would compete for screen space.
Dribble	Open a typescript le for this Chat connection (closing any previous dribble le for the window). The user is prompted for a le name; a name of `NIL` just closes the old dribble le.
Input	Prompts for a le to take input from. When the end of the le is reached, input reverts to `T`.
Clear	Clears the window and resets the simulated terminal to its default state. This is useful if undesired terminal commands have been received from the remote host that place the simulated terminal into a funny state.

In an inactive Chat window, the `MIDDLE` button brings up a menu of one item, `ReConnect`, whose selection reopens a connection to the same host as was last in the window. This is the primary motivation for the Suspend menu command. A new Chat connection can also be opened from the Background menu.

The mouse button `LEFT`, when inside `CHAT`, holds output as long as the button is down. Holding down `MIDDLE` coincidentally does this, too, but not on purpose: since the menu handler does not yield control to other processes, it is possible to kill the connection by keeping the menu up too long.

Chat windows are a little bit knowledgable about window operations. If you reshape a Chat window, Chat informs your partner of the new dimensions. And if you close the window, the connection is also closed.

The following variables control aspects of Chat's behavior:

`CHAT.DISPLAYTYPE` [Variable]

The type of display (a number) that Chat should tell the remote host the user is on. If Datamedia emulation is desired, this variable should be set to the number corresponding to the terminal type Datamedia for the remote host. If the remote host does not respond to the terminal type protocol in Pup Telnet, this variable is irrelevant.

`CHAT.ALLHOSTS` [Variable]

A list of host names, as uppercase litatoms, that the user desires to Chat to. Chatting to a host not on the list adds it to the list. These names are placed in the menu that the background Chat command prompts with.

`CLOSECHATWINDOWFLG` [Variable]

If true, every Chat window is closed on exit. If `NIL`, the initial setting, then the primary Chat window is not closed.

`DEFAULTCHATHOST` [Variable]

The host to which `CHAT` connects when it is called with no `HOST` argument.

`CHAT.FONT` [Variable]

If non-`NIL`, the font that Chat windows are created with. If `CHAT.FONT` is `NIL`, Chat windows are created with `(DEFAULTFONT ’DISPLAY)`.

CHAT