CHAT

Chat is a remote terminal facility that allows you to communicate with other machines while inside Lisp. Chat sets up a Chat connection to a remote machine, so that everything you type is sent to the remote machine, and everything the remote machine prints is displayed in a Chat window.

Chat is an extensible terminal emulation facility. Its core supplies both terminal- and network-protocol- independent functionality; new terminal types and new Chat protocols, based on this core, can be added to Chat at any time. You can choose any terminal type to be used with any network protocol type.

There are currently terminal emulators for the following terminals:

Datamedia 2500

DEC VT100

TEdit (this is actually a TEdit-based Chat window, supporting scrolling and copy-select operations as in standard TEdit).

A number of different network protocol interfaces can be used with Chat. The following protocols are available:

PUP Chat

NS Chat (using the GAP protocol)

`TCP (ARPANET) TELNET`

RS232 Chat (using either the RS232 or TTY ports of the 1108 and 1186 processors)

Each of these is available by loading the corresponding module.

Requirements

DMCHAT

CHATTERMINAL

One of the network protocols: PUPCHAT or NSCHAT or RS232CHAT or TTYCHAT or TCPCHAT.

One of the terminal emulators: DMCHAT or VTCHAT or TEDITCHAT.

The applicable file dependencies enumerated in the Introduction of this manual.

Installation

Load `CHAT.LCOM` from the library.

In addition, you must load at least one of the Chat network protocol modules.

If you want a terminal emulator different from the default DM2500, you must also load it.

User Interface

Chat prompts for a new window for each new connection. It saves the first window to reuse once the connection in that window is closed (other windows just go away when their connections are closed).

Multiple, simultaneous Chat connections are possible. To switch between typing to different Chat connections, press the left button within the Chat window you want to use.

Opening a Chat Connection

The simplest way to open a Chat connection is to select the CHAT option of the right-button (background) menu. The first time you do this, you are prompted in the system's prompt window for the name of a host to which to connect. Subsequently, you are prompted with a menu of all hosts to which you have opened Chat connections; the last entry in this menu is OTHER, and provides a way for you to connect to new Chat hosts.

The other method of opening a Chat connection is to call the CHAT function directly:

(CHAT *HOST LOGOPTION INITSTREAM WINDOW*) [Function]

Opens a Chat connection to *HOST*, or to the value of DEFAULTCHATHOST. If *HOST* requires login, Chat supplies a login sequence.

You may alternatively specify one of the following values for *LOGOPTION*:

Login — Always perform a login.

Attach — Always perform an attach (this is likely to be useful only when opening Chat connections to hosts running the Tops-20 or Tenex operating systems). This fails if you do not have exactly one detached job.

None — Do not attempt to log in or attach.

Note: It is important that you supply information about the types of hosts to which you chat by setting the variable NETWORKOSTYPES (see IRM) or DEFAULT.OSTYPE (see *Lisp Release Notes)*, as CHAT uses that information to determine whether and how to log in. An incorrect login sequence can inadvertantly expose your password.

If *INITSTREAM* is supplied, it is either a string or the name of a file whose contents are read as type-in. When the string/file is exhausted, input is taken from the keyboard.

If *WINDOW* is supplied, it is the window to use for the connection; otherwise, you are prompted for a window.

While Chat is in control, all Lisp interrupts are turned off, so that control characters can be transmitted to the remote host. Chat does not turn off interrupt characters until after creating the Chat window, so you can abort the call to Chat by typing Control-E while specifying the Chat window region.

If you press the left button in an Executive window, the system's focus-of-attention is switched to that window. At the same time, keyboard interrupts, such as Control-E, are reenabled. Whenever you select an open Chat window, the focus-of-attention is returned to the Chat window, and keyboard interrupts are disabled.

Chat Menu

Commands can be given to an active Chat connection by pressing the middle mouse button in the Chat window to get a command menu.

Note: The left mouse button, when pressed inside an active Chat window, holds output as long as the button is down. Holding down the middle button 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.

`CLOSE` Closes this connection. Once the connection is closed, control is handed over to the main Lisp Executive window. Closes the Chat window unless it 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` Holds type-out from this Chat window. Pressing a mouse button in the window in any way releases the hold. This is most useful if you want to switch to another, overlapping window and there is type-out in this window that would compete for screen space.

`DRIBBLE` Opens a typescript file for this Chat connection (closing any previous dribble file for the window). You are prompted for a file name. If you want to close an open dribble file (without opening a new one), just type a carriage return.

`INPUT` Prompts for a file from which to take input. When the end of the file is reached, input reverts to the keyboard.

`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 an indeterminate state.

`EMACS` Turns on or off the Chat EMACS feature, which provides a convenient way to use the workstation's mouse to move the cursor on the remote machine when using the EMACS text editor. When this feature is turned on, pressing the left mouse button in the Chat window causes a sequence of commands to be sent to the remote machine that cause EMACS to move its cursor to the mouse location.

Use of this feature assumes you know the keystrokes to perform cursor-moving commands; see `CHAT.EMACSCOMMANDS` if your EMACS does not use the standard ones. Also, it assumes that you are pointing where there is actually text in your document (not white space beyond the end of a line) and that there are no tabs in your text; otherwise, the cursor position may not be where you expect.

`RECONNECT` In an inactive Chat window, pressing the middle mouse 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.

<EMULATOR>MODE — The Chat menu also contains a command of this form for each terminal emulator that you have loaded. The <EMULATOR>MODE commands are intended to let you dynamically switch between terminal emulators.

However, this feature is currently defective and should not be used. You must also choose your emulator type, by setting CHAT.DISPLAYTYPES, before opening the Chat connection.

Customizing Chat

CHAT.DISPLAYTYPES [Variable]

This variable contains a list that assigns the terminal emulators to be used with the hosts. Each entry on the list is of the form:

```
(<HostName><TerminalTypeNumber><TerminalEmulator>)
```

HostName — When Chat opens a connection, it scans CHAT.DISPLAYTYPES to find an entry whose HostName field matches the name of the Chat host. If no matching entry is found, it scans the list again, looking for an entry whose HostName field is NIL.

TerminalTypeNumber — Is only important when the Chat protocol in use is PUP Chat. This number identifies the terminal type to the Chat host's operating system. Currently, only Tops-20 and Tenex hosts make use of this facility; if the Chat host does not support this feature, the number in the TerminalTypeNumber field is ignored.

TerminalEmulator — Chat uses this field of the entry it finds to choose which terminal type to emulate. Typical terminal emulator names are DM2500, VT100, and TEDIT.

CHAT.KEYACTIONS [Variable]

This variable controls the remapping of the keyboard when the system's focus-of-attention is an active Chat window. The format of this list is:

```
((KEYNAME . ACTIONS) (KEYNAME . ACTIONS) ... )
```

For example, if you prefer the backspace key to send the rubout character (octal 177), you would set CHAT.KEYACTIONS to be:

```
((BS (177Q 177Q NOLOCKSHIFT) . IGNORE))
```

The key actions are assigned when a Chat process is initiated; i.e., changing CHAT.KEYACTIONS only affects new Chat connections.

CHAT.INTERRUPTS [Variable]

A list of interrupts to pass to INTERRUPTCHAR to assign keyboard interrupts; e.g., ((177Q. HELP)) causes the DELETE character (code 177) to run the HELP interrupt.

Like CHAT.KEYACTIONS, this variable only affects new Chat connections.

CHAT.ALLHOSTS [Variable]

A list of host names, as uppercase symbols, to which you want to chat. Chatting to a host not on the list adds it to the list. These names are placed in the menu used by the background Chat command prompts.

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 used to create Chat windows. If CHAT.FONT is NIL, Chat windows are created with (DEFAULTFONT 'DISPLAY).

Note: To work well with the DM2500 and VT100 terminal emulators, you should use fixed–width fonts (e.g., Gacha or Terminal).

CHAT.WINDOW.SIZE [Variable]

This variable is either NIL or a dotted pair of (WIDTH . HEIGHT). The value of the WIDTH field indicates the desired width of the Chat window, in pixels. The value of the HEIGHT field indicates the desired HEIGHT of the window, also in pixels.

Note: Before a new value of CHAT.WINDOW.SIZE is used, CHAT.WINDOW must be set to NIL or NOBIND. If CHAT.WINDOW.SIZE is changed after chat has already been called, and chat is then called, the window is not changed because the information is cached. CHAT.WINDOW must be set to NIL and the window recreated anew before this takes place.

CHAT.WINDOW.REGION [Variable]

This variable is either NIL or an instance of a REGION. When CHAT.WINDOW.REGION is non-NIL, its value is used as the region in which to create the first Chat window.

Subsequent windows are created by prompting for the position of a window of CHAT.WINDOW.SIZE dimensions, or, if that variable is NIL, for an arbitrary window region.

CHAT.TTY.PROCESS [Variable]

When you start up CHAT, it takes the TTY immediately if the value is T. (The initial value is T.)

CHAT.EMACSCOMMANDS [Variable]

A list of five character codes; initially the value of (CHARCODE (^U ^P ^N ^F ^A)). These character codes are used by the EMACS Argument command in changing the position of the cursor:

Up one line
Down one line
Forward one character

Backward one character
Beginning of line

CHAT.IN.EMACS? [Variable]

The initial state of the EMACS feature when a Chat connection is started. Initially NIL, meaning the feature is off.

CHAT.PROTOCOLTYPES [Variable]

Each Chat emulator (TTYCHAT, RS232CHAT, PUPCHAT ...) adds an entry onto CHAT.PROTOCOLTYPES which recognizes host names for the appropriate protocol.

For example, loading PUPCHAT adds an entry (PUP . PUPCHAT.HOST.FILTER) and TCPCHAT adds an entry (TCP . TCP.HOST.FILTER).

Site administrators of complex networks may want to reorganize these entries when there are hosts which are running multiple servers, each running different protocols.

Network Protocols

For the most part, you should not notice too many differences in the behavior of Chat when using one network protocol versus another. The following are unique features of each of the Chat network protocols.

PUP Chat

PUP Chat is in the file PUPCHAT.LCOM. Implementations of PUP Chat servers exist for Tops-20, Tenex, VAX/UNIX, and VAX/VMS operating systems. The PUP Chat protocol contains provisions for automatically setting your terminal type, width, and height whenever you establish a connection or reshape your Chat window.

NS Chat

The NS Chat protocol (also known as GAP, or Gateway Access Protocol) is used to communicate with hosts running GapTelnet service, including VAX/UNIX and the VAX/VMS service XNS/DEC VAX, and also with Xerox 8000-series network services such as 8040 print servers or 8030 file servers. This protocol is contained on the file NSCHAT.LCOM. The NS Chat protocol differentiates among a number of virtual terminal services. When you chat to an NS host, the NS Chat module queries the Clearinghouse for information about the specified host. This information permits the NS Chat module to determine which of the following virtual terminal services are appropriate for the host.

The NS Chat module uses a small set of heuristics to choose which virtual terminal service to invoke, based on information returned by the Clearinghouse. If the Clearinghouse information indicates that only one service type is possible, NS Chat opens a connection to the Chat host and invokes the proper virtual terminal service.

If the Clearinghouse returns information indicating that more than one virtual terminal service is supported by the specified host, you are prompted to choose a service from a menu of the possible service types.

If NS Chat guesses an incorrect service type, or you choose an incorrect service type, you are prompted to choose a service from a menu of all known virtual service types. If this fails, NS Chat abandons its attempts to connect to the specified host.

Remote System Administration

This service lets you log onto print servers and clearinghouse servers, and issue appropriate commands. NS Chat automatically chooses this service when the specified host is registered in the Clearinghouse as any type of server machine.

Remote System Executive

This service is currently supported by VAX/VMS systems running XNS/DEC VAX, by UNIX systems running GapTelnet service, by Lisp workstations running `CHATSERVER` from the library, and by XDE workstations.

Interactive Terminal Service

The ITS is a TTY-based interface to NS mail.

External Communication Service

The External Communication Service (ECS) enables Chat connections to external hosts accessible only by use of a modem. When you open a Chat connection to an ECS, you are prompted for a telephone number; the ECS dials that number and completes the connection if a compatible modem answers.

ECS hosts typically support a variety of modem connection characteristics (specific combinations of parity, character length, baud rate, and flow control settings). Each connection type is known by a different Chat host name; check with your system administrator to determine the Chat host name you should use to connect to a particular external host.

TCP Chat

`TCPCHAT.LCOM` is the interface to the TCP-based TELNET protocol, which is the protocol in use throughout the ARPANET. It loads and initializes the TCP-IP module, if necessary. Read the TCP-IP module in this manual for more information.

RS232 Chat

RS232 Chat is contained on the files `RS232CHAT.LCOM` and `TTYCHAT.LCOM`. RS232 Chat enables use of the 1108, 1185, and 1186 RS232 ports; TTY Chat enables use of the 1108, 1185, and 1186 TTY ports. Read the RS232 module in this manual for more information.

Terminal Emulators

DM2500 Chat

The Datamedia 2500 terminal emulator is contained in `DMCHAT.LCOM`. To use it, load `DMCHAT.LCOM` and add entries to `CHAT.DISPLAYTYPES` in the form:

```
(<HOSTNAME> <TERMINALTYPENUMBER> DM2500)
```

VT100 Chat

The VT100 emulator is contained in VTCHAT. To use it, load VTCHAT.DFASL and add entries to CHAT.DISPLAYTYPES in the form:

```
(<HOSTNAME> <TERMINALTYPENUMBER> VT100)
```

Currently, the VT100 emulator does not emulate the following features of the actual Digital VT100 terminal:

Dual-width or dual-height characters

Graphics character set

Remotely initiated switching between 80- and 132-column mode

TEdit Chat

TEdit Chat supplies a "glass TTY" terminal emulator with a TEdit stream storing all characters received during the Chat session. As a result, you can scroll back and forth through a transcript of your session, and you can use the standard TEdit copy-select command to copy blocks of characters from the Chat window to another TEdit window, a Lisp Executive, etc.

To use TEdit Chat, load TEDITCHAT.LCOM, and add entries to CHAT.DISPLAYTYPES in the form:

```
(<HOSTNAME> <TERMINALTYPENUMBER> TEDIT)
```

Note that since TEdit already uses the middle mouse button, you must click in the window's title bar in order to get the usual Chat menu.

[This page intentionally left blank]