Z-Mail Reference Manual

Version 2.0

Z-Code Software Corporation
4340 Redwood Highway, Suite B-50
San Rafael, CA 94903
415-499-8649

Important Notice

Z-Code Software Corporation (Z-Code) and its licensors retain all ownership rights to the Z-Mail computer program and other computer programs (herein after called "Z-Mail"). Z-Code retains all ownership rights to the Z-Mail documentation. Use of Z-Mail is governed by the license agreement accompanying your original media. The Z-Mail source code is a confidential trade secret of Z-Code. You may not attempt to decipher or decompile Z-Mail or develop source code for Z-Mail, or knowingly allow others to do so. You may not develop passwords or codes or otherwise enable the program for equipment that is not authorized for use with Z-Mail. Z-Mail and its documentation may not be sublicensed and may not be transferred without the prior written consent of Z-Code.

Only you and your employees and consultants who have agreed to the above restrictions may use Z-Mail and only on the authorized equipment.

Your right to copy Z-Mail and this manual is limited by copyright law. Making copies, adaptations, compilation works (except copies of Z-Mail for archival purposes or as an essential step in the utilization of the program in conjunction with the equipment), without prior written authorization of Z-Code, is prohibited by law and constitutes a punishable violation of the law.

Z-CODE SOFTWARE CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Z-Code may revise this publication from time to time without notice. Some states or jurisdictions do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

Copyright © 1992 Z-Code Software Corporation, SanRafael, California, U.S.A. All rights reserved.

Z-Mail, Z-Script and Z logo are trademarks of Z-Code Software Corporation.

UNIX and Open Look are registered trademarks of UNIX System Laboratories.

Motif is a trademark of Open Software Foundation.

X Window System is a trademark of Massachusetts Institute of Technology.

PostScript is a trademark of Adobe Systems.

FrameMaker and FrameViewer are trademarks of Frame Technology.

Wingz is a trademark of Informix Software.

Sun is a trademark of Sun Microsystems.

Any provision of Z-Mail to the U.S. Government is with "Restricted Rights" as follows: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7013, and in subparagraphs (a) through (d) of the Commercial Computer-Restricted Rights clause at FAR 52.227-19, and in similar clauses in the NASA FAR Supplement, when applicable. Any provision of Z-Mail documentation to the U.S. Government is with Limited Rights.

Printed in the United States of America.

CONTENTS

Z-Mail Reference Manual

Chapter 1 ***Introducing Z-Mail 1***

Using Z-Mail Documentation 2
Where You Should Start 3

Chapter 2 ***Z-Mail Basics 5***

E-Mail Terminology 6
Z-Mail's User Interfaces 8
Using Help 9
Specifying Message Lists 10
Specifying Text Patterns 11
Specifying Filename Patterns 12
Specifying E-Mail Addresses 13

Chapter 3 ***Starting and Initializing Z-Mail 17***

Starting Z-Mail 18
Z-Mail Options Summary 19
Z-Mail Options Reference 22
Initializing Z-Mail 37
Managing Initialization Files 40

Chapter 4 ***Using Z-Mail's Character Interfaces 45***

Starting Z-Mail's Character Interfaces 46
Command-Line Interface Basics 47
Fullscreen Interface Basics 60
Tilde-Escape Editing Commands 65
Using Macros 75

Chapter 5 ***Z-Script Commands 83***

Command Summary 84
Command Reference 89

? 89
about 89
alias 89
alternates 90
ask 91
await 92
bind 93
bind-macro 94
builtin 94
button 95
cd 96
close 96
cmd 96
copy 97
delete 97
detach 98
dialog 99
display 99
each 100
echo 101
edit 101
error 102
eval 102
exit 103
expand 103
filter 103
flags 105
folder 105
folders 106
foreach 107
from 108
fullscreen 109
function 109
headers 111
help 112
history 113
iconify 113
if...else...endif 113
ignore 118
jobs 118
lpr 119
ls 119
mail 120
map 122
mark 123
merge 124
msg_list 124
my_hdr 125
next 126
pick 126
pipe 126
preserve 127
previous 128
print 128
pwd 128
quit 128
reply 129
replyall 130
resume 130
retain 131
return 131
save 132
saveopts 133
search 133
set 136
sh 137
shift 137
sort 139
source 141
stop 141
top 141
type 142
unalias 142
unbind 142
unbutton 142
uncmd 143
undelete 143
undigest 143
unfilter 144
unfunction 144
un_hdr 145
unignore 145
unmap 145
unmark 146
unset 146
update 147
version 147
write 147

Chapter 6 ***Z-Mail Variables 149***

Types of Variables 150
Setting Variables 151
Storing Variable Settings 153
Using Variables 154
Variable Summary 158
Variable Reference 168

alwaysignore 168
ask 168
askcc 169
attach_types 169
autodismiss 169
autoedit 170
autoiconify 170
autoinclude 170
autoprint 170
auto_route 171
autosign 173
autosign2 174
cdpath 176
cmd_help 176
complete 177
crt 178
crt_win 178
cwd 178
date_received 178
dead 179
deletesave 179
domain_route 179
dot 180
edit_hdrs 180
editor 181
escape 182
fignore 182
folder 183
fortune 183
fortunates 183
fullscreen_help 184
gui_help 184
hangup 184
hdr_format 185
hdrs_only 186
history 186
hold 186
home 186
hostname 187
ignore_bang 187
ignoreeof 187
indent_str 187
input 188
in_reply_to 188
is_fullscreen 189
is_gui 189
is_sending 190
is_shell 190
keepsave 190
known_hosts 190
logfile 191
mail_icon 191
mbox 191
metoo 191
mil_time 192
msg_win 192
newline 192
newmail_icon 192
no_expand 192
no_hdrs 193
no_reverse 193
nonobang 193
nosave 194
output 194
pager 194
picky_mta 195
post_indent_str 195
pre_indent_str 195
printer 196
print_cmd 196
prompt 196
quiet 197
realname 197
record 198
record_users 198
recursive 198
redirect 199
reply_to_hdr 199
save_empty 200
screen 200
screen_win 200
sendmail 200
show_deleted 200
sort 201
squeeze 201
status 201
templates 201
thisfolder 202
title 202
tmpdir 202
toplines 202
unix 202
verbose 203
verify 203
version 204
visual 204
warning 204
window_shell 205
wrap 205
wrapcolumn 205
$$ 206
$* 206
$# 206
$0 207
$1, $2, . . . 207

Chapter 7 ***Integrating Z-Mail with Other Products 209***

Importing Text Files into Messages 210
Defining Attachment Types 211
Launching Applications from Z-Mail 216

Index 219

CHAPTER 1

Introducing Z-Mail

In today's computing environments of networked servers and workstations, electronic mail (e-mail) is the most popular method of communication. In fact, in large organizations it is common for people to receive a hundred or more mail messages per day. Z-Mail is a comprehensive electronic mail system that not only enables you to send and receive those messages, but also helps you manage your mail efficiently.

Unlike most e-mail products, Z-Mail gives you a choice of user interfaces that make you more productive regardless of what UNIX computer or display device you use.

If you have a graphical workstation or X terminal, you can use Z-Mail's point-and-click graphical user interface (GUI). Z-Mail offers both Motif and Open Look GUI versions. If you have a character terminal, you can use Z-Mail's fullscreen character interface. If you're a seasoned user, you may also use Z-Mail's command-line interface to manage your mail. In any case, you have access to all of Z-Mail's features and capabilities.

If you ever need additional information about a feature or function, Z-Mail provides context-sensitive help from any location. Each dialog box includes a help button that provides valuable information should you need it.

For more experienced mail users, Z-Mail offers Z-Script,™ a scripting language for creating message filters, functions, GUI buttons and dialog boxes. You can use Z-Script to customize your Z-Mail installation so it works exactly the way you want.

Using Z-Mail Documentation

Z-Mail User's Guide

The *Z-Mail User's Guide* helps you understand the basic functions of Z-Mail and how to use them. The guide is available in both Motif and Open Look versions.

Z-Mail Reference Manual

This book — The *Z-Mail Reference Manual* — is intended for experienced users and system administrators. It is organized to help you understand basic Z-Mail concepts, administer your Z-Mail installation, configure Z-Mail to meet your specific needs, and use the Z-Script language.

Z-Mail Installation Guide

The *Z-Mail Installation Guide* provides step-by-step instructions for successfully installing Z-Mail. The guide assumes you are familiar with your equipment and the UNIX operating environment you need to run Z-Mail.

The Z-Mail Handbook

Another helpful text is O'Reilly & Associates' book entitled *The Z-Mail Handbook* by Hanna Nelson. It contains a complete discussion of Z-Mail's various interfaces and features as well as many examples of how to use Z-Mail effectively.

You can obtain any of the above documentation by calling Z-Code Software in San Rafael, California at 415-499-8649.

Where You Should Start

Motif and Open Look Version Users

If you are going to be using the Motif or Open Look versions of Z-Mail, you should first read the *Z-Mail User's Guide*. It discusses how you can make the most of Z-Mail's powerful mail management capabilities using its point-and-click interfaces. After that, if you intend to write scripts or customize your Z-Mail installation, you may want to read various sections of the *Z-Mail Reference Manual* as the need arises.

Fullscreen Version Users

If you plan to use Z-Mail's fullscreen interface, you should read the introductory sections of this manual first. After you understand basic Z-Mail concepts, you should read the sections called *Z-Mail Basics* beginning on page 4 and *Fullscreen Interface Basics* on page 57. To understand fullscreen keyboard macros, see the section called *Using Macros* on page 75. If you intend to write scripts or customize your Z-Mail installation, you may want to read various sections of the *Z-Mail Reference Manual* as the need arises.

Command-Line Version Users

If you are an experienced mail user and intend to use the command-line version of Z-Mail, you should read the beginning of this manual first. After you understand basic Z-Mail concepts, you should read the sections called *Using Z-Mail's Character Interfaces* on page 42, and then *Z-Script Commands* on page 83. You can use almost any Z-Script command at Z-Mail's command-line prompt. To understand Z-Mail macros, see the section called *Using Macros* on page 75.

Alternative to *mail* Command

You can also use Z-Mail as a UNIX command to send mail directly from your UNIX shell. You do this by specifying options — command-line modifiers to the *zmail* command — that contain all the information Z-Mail needs to send your message. When you use Z-Mail in this send-only mode, it does not start one of its interfaces. Instead, Z-Mail returns control back to the UNIX shell or the program that called Z-Mail. For more information on Z-Mail's send-only capabilities, see *Z-Mail Options Summary* beginning on page 17, and *Z-Mail Options Reference* beginning on page 20.

CHAPTER 2

Z-Mail Basics

Electronic mail, or *e-mail*, is a form of personal communication used heavily by computing communities. Instead of sending paper letters through the mail, users send electronic *messages* via telecommunication lines. You create messages on your workstation or terminal and send them to other users.

As e-mail users have increased in sophistication, they have demanded more from their e-mail systems. Today, users not only send messages, they also routinely send programs and documents as attachments to mail messages, just as you would enclose a brochure or article in a business letter.

Today's electronic networks allow users all over the world to communicate and exchange electronic documents. As the number of users and messages skyrockets, so does the need to manage all this electronic information.

Z-Mail is designed to handle these demanding needs and more. To properly use all of Z-Mail's capabilities, it is helpful to understand some basic terminology as well as the conceptual building blocks that are used throughout Z-Mail.

E-Mail Terminology

E-Mail Addresses

All e-mail users have a unique *e-mail address* that identifies their name and the computer on which they read mail. This address is all you need to send a message to another user. A typical e-mail address for John Smith at the Justin Case Company is *jsmith@justin.com.* The string *justin.com* identifies the central mail computer at the Justin Case Company. The *.com* suffix denotes that Justin Case is a commercial company. If Justin were an educational organization, it would have an address that ends in *.edu*, such as *mit.edu*. E-mail addresses are assigned to you by your system administrator.

Mail Transport Agent

The operating system of your UNIX computer has a *mail transport agent* (MTA) that functions similarly to your local post office in that it picks up mail you send and delivers mail that others have sent to you. In addition, your computer may be connected to a worldwide network of computers such as the *Internet* that functions like a postal service. To send a message to a user on another computer, you simply address the message and your computer forwards it to the proper remote computer by using the network. The remote computer then delivers it to the addressee.

Mail User Agent

The software program you use to read and write your mail is called a *mail user agent* (MUA). Z-Mail is a mail user agent that not only allows you to read and send messages, but also provides you with extensive capabilities to manage your mail.

Mail Messages

The basic form of e-mail communication is the *message*. An e-mail message is made up of several *headers*, a *body* and an optional *signature*. The *headers* are formatted fields with special names that contain information about whom the message is from, whom it is for, what it is about, how it was created, how it is to be delivered, and more. The *body* contains the text of the actual message. The *signature* is an optional block that a mail user agent can place on the end of the message, in effect "signing" it with the author's name, title, or other information.

Mail Folders

All UNIX mail messages are stored in data files called *folders*. Typically, each message in a folder is appended to the previous message. UNIX delivers messages to your e-mail address by writing them to your *System Folder* where they stay until you start Z-Mail. When you are through reading your mail, Z-Mail stores your mail back to either your System Folder or your *Main Mailbox*. The messages stay in your System Folder or your Main Mailbox until you delete them or move them to another folder you create and name. You can think of the System Folder as the mailbox outside your house and the Main Mailbox as the in-box inside your house on the corner of your desk.

Z-Mail permits you to have more than one folder open at a time. You can switch rapidly among these opened folders to more easily manage your mail. However, only one folder is *active,* or current, at a time. Z-Mail applies commands to the active folder. The messages in each folder are numbered so you can identify specific messages when you issue Z-Mail commands.

Z-Mail's User Interfaces

GUI

Z-Mail offers two different graphical user interfaces (GUIs) for UNIX users: Motif and Open Look. The GUI interfaces use the point-and-click simplicity of a mouse-driven interface to boost the productivity of both novice users and experienced systems professionals. You start Z-Mail's graphical user interfaces with one of these commands:

```
zmail –motif
zmail –openlook
```

Fullscreen

Z-Mail's fullscreen interface makes use of the curses program library that is a part of all UNIX systems. With Z-Mail's fullscreen interface, instead of using a mouse, you use keyboard commands to move a cursor around your screen. Z-Mail's fullscreen interface is designed to help users with character devices such as ASCII terminals manage their mail easily. To start the fullscreen interface, enter:

```
zmail –fullscreen
```

Command Line

Z-Mail also offers a command-line interface for managing your mail. Z-Mail supports mail commands you may already know from using standard UNIX mail or BSD Mail. In addition, Z-Mail has many commands that go way beyond standard UNIX mail capabilities. For more information on Z-Mail's command language, see *Z-Script Commands* beginning on page 83. To start Z-Mail using its command-line interface, enter:

```
zmail
```

Send-Only Mode

You can also use Z-Mail as a mail utility in its send-only mode. This mode does not have a user interface at all. Instead, you can issue the *zmail* command from your UNIX shell prompt and specify a variety of options to send messages to other users. For more information, see the section called *Z-Mail Options Summary* beginning on page 17.

Using Help

Z-Mail's help facility provides information on both a context-sensitive basis as well as through an index of specific help categories.

Command Line

In the command-line mode, you can enter **–?** after a command to receive information about the command. If you type **?** as a command, Z-Mail displays a list of possible commands.

Fullscreen

In fullscreen mode, the **?** key displays a help message that explains how to obtain a list of your current key-to-command bindings. A keystroke or set of keystrokes corresponds directly to a command.

GUI

In GUI mode, Z-Mail provides more extensive and context-sensitive help. Windows with menu bars have a *Help* menu in the upper right corner, and other dialog windows have a *Help* button in the lower right corner. Selecting any help item opens a new window with the appropriate help text that you can keep on your screen as you work.

Using the GUI interface, you can also select the *On Context* option from the *Help* menu. Z-Mail changes the system pointer to a hand with a pointing finger. Click on the item the item for which you want help and Z-Mail opens a window with information about that item.

Customizing Z-Mail's Help Files

You can customize Z-Mail's help files, changing existing text to meet your needs, or adding entries for new functions or buttons you have added to your Z-Mail installation. To tell Z-Mail to look for your custom help files instead of the default help files that come with Z-Mail, you must reset the values of the *gui_help* and *cmd_help* variables. These variables are listed individually in the section called *Variable Reference* beginning on page 168. You may also want to read about the *help* command on page 112.

Specifying Message Lists

At several places in Z-Mail, you specify lists of message numbers that tell Z-Mail what messages you wish to process. In many cases, you tell Z-Mail to perform an action on the current message(s) by entering a command. You can also tell Z-Mail explicitly which messages to process by entering a *message list*.

Z-Mail supports a special set of *metacharacters* that are shorthand notations to specify complex lists of message numbers. When processing message lists, Z-Mail recognizes the following special syntax:

*****	All messages
^	First message
$	Last message
. (period)	Current message
n–*m*	Range beginning with message number *n* and ending with *m*
{*messagelist***}**	Ignore all messages in *messagelist*

Be aware that Z-Mail parses message lists from left to right, and follow these rules when specifying lists:

- You can use spaces or commas to separate arguments.
- You cannot begin a list with a negated {*messagelist*} argument.
- You can turn on negated messages later in argument lists.
- Ranges must be ascending (e.g., 1-7 is a valid range; 5-2 is invalid).

EXAMPLES

The following table illustrates the use of metacharacters in specifying message lists.

Message List:	***Refers to Messages:***
4–7	4 5 6 7
1,3,7–10 12	1 3 7 8 9 10 12
4–$	4 through last message
^–.	First message through current message
^,$	First and last message
*** {4–8}**	All messages except 4 through 8
1-20 {2–6} 5	1 5 7-20
*** { . }**	All messages except current one

Specifying Text Patterns

When you search the text contents of mail messages, you specify *text patterns*. These patterns are strings that contain literal and wildcard characters that Z-Mail interprets. To specify these filenames and patterns, you can use the standard UNIX wildcard characters called *metacharacters*. These characters include, but are not limited to:

.	Any single character except *newline*.
*	Any number of repetitions of the previous character (including none).
^ (as prefix)	Looks at beginning of string for following characters.
$ (as suffix)	Looks at end of string for preceding characters.
\	Prevents interpretation of following metacharacter.
[xyz]	Matches any one of the characters enclosed.

EXAMPLES The following table illustrates the use of metacharacters in matching text patterns.

String:	***Matches:***	***But not:***
mark..	market, mark12	mark, 2mark
mark.*	market, mark, mark12	2mark
mark$	mark, market, mark5	unmark
^mark	unmark, 2mark	market
mark[a–c]	marka, markb, markc	market, markd
mark[ac]	marka, markc	markb
abc	ab, abc, abcc, abccc	abcd, abcabc
abc.	abc, abcd, abcde	ab, cab
.abc	1abc, aabc	ttabc, abc, abcd
g[abc]*	gabc, gabcabc	gabc2, gabcd

Specifying Filename Patterns

In the *folder* command and *Folders* dialog box in Z-Mail, you specify the names of system files and mail folders. Z-Mail allows you to specify these filenames using patterns based on standard UNIX wildcard characters called *metacharacters*. These characters include, but are not limited to:

?	Any single character except *newline*.
*	Any number of previous characters including none.
\	Prevents interpretation of following metacharacter.
[xyz]	Matches any one of the characters enclosed.
{abc,def,...}	Matches one of the patterns enclosed.

EXAMPLES

The following table illustrates the use of metacharacters in matching filename patterns.

String:	***Matches:***	***But not:***
pro??	proga, pro12	pro123, pro
pro?*	prog, prog12, protest	pro, apro
pro*	pro, prog, protest	apro
pro[a–c]	proa, prob, proc	pro2, protest
pro[ac]	proa, proc	prob
pro	unpro, 2pro, protest	pre, apro
?pro	1pro, 2pro, apro	unpro, prog
pro{ab,g,123}	proab, prog, pro123	pro, prob, apro

Directory Notations

Z-Mail also supports the following notations when you specify filenames to identify mail folders:

~ (tilde)	Refers to your home directory.
+	Refers to your folder directory.

The tilde character (**~**) refers to your home directory, and the plus character (**+**) refers to your folder directory (*~/Mail* or the value of the *folder* variable). You can specify another user's *home* directory as *~username*, and for this reason files in your *home* directory must be referenced as *~/filename*. However, the forward slash (*/*) character is optional when you refer to folders. That is, *+filename* and *+/filename* both refer to the same folder in your *folder* directory.

Filename Completion

If you set the *complete* variable to enable filename completion, Z-Mail's command interpreter expands filename metacharacters when it completes filenames. Z-Mail expands metacharacters to all the files matching the pattern you specify when you press the completion key (the default completion key is **Esc**). See the description of the *complete* variable on page 177 for capabilities and limitations of this facility.

Specifying E-Mail Addresses

Just like mailing a letter, to send an e-mail message you must properly address the message for it to reach its destination. This section describes a few basic ways to address your outgoing messages correctly.

Local Addresses

To send a message to users on your local machine, you only need to enter their usernames.

```
mail john pat kfong rgomez
```

If you use blank characters to separate user addresses, Z-Mail automatically converts them to commas.

```
john, pat, kfong, rgomez
```

Remote Addresses

When you send a message to users who do not read their mail on the same computer as you do, you may need to route the message through a series of host computers. To route your message properly, the mail transport agents on those computers look for its *UUCP address*. The address describes the various hosts through which your message must be routed to reach its destination.

SYNTAX

A UUCP address may indicate as many hosts as necessary to route a message to the correct user.

```
host1!host2! ... !hostn!user
```

EXAMPLE

The following address sends your message from your host computer to the host named *utech*, which sends it to *amcorp*, which in turn sends your message to *eastsoft*. The recipient, *psteele*, reads the message on the *eastsoft* host computer.

```
utech!amcorp!eastsoft!psteele
```

Domain Addresses

Fortunately, there is an easier way to route messages to remote users. Instead of specifying the exact path including all the hosts required to route your message, you can use *domain addresses* — also known as *Arpanet, Internet, RFC822,* and *fully qualified* addresses.

S*YNTAX*

The general form of a domain address is:

```
user@host.domain
```

EXAMPLE To send a message to Kyle Dwyer at Ukiah Fiber Company, you might enter the address **kdwyer@ukiahfib.com**. The **@** character separates the username, *kdwyer*, from the domain name, *ukiahfib.com*.

Note: You may also see addresses using **%** characters to separate user and domain names. Some, but not all hosts use this notation. All hosts, however, recognize the **@** notation, so it is safer to use instead of the **%** character.

For more information on network addressing, see your system manager or O'Reilly & Associates text, *A Directory of Electronic Mail Addressing & Networks*.

Address Conventions

Z-Mail understands e-mail addresses that contain a *comment field*. Comment fields do not affect the destination address of mail. These fields are purely for your legibility and may be specified according to the following constraints:

- Anything within angle brackets is an address; whatever is outside of the address is considered a comment:
  Dan Powers <utech!ukiahfib!dpowers>
  Dan Powers <dpowers@ukiahfib.com>
- Anything inside parentheses is a comment; whatever is outside of the parentheses is considered the address:
  dpowers@ukiahfib.com (Dan Powers)
- Double quotes (**" "**) are treated just like parentheses; whatever is inside the quotes is treated as a comment:
  "Dan Powers" utech!ukiahfib!dpowers
  "Dan Powers" dpowers@ukiahfib.com
- If the comment contains a comma, you must use either parentheses or double quotes, or Z-Mail interprets the comma as the start of a new address:
  mary@finance.justin.com (Mary Edwards, VP Finance)
- If the comment contains unbalanced quotes, you can get unpredictable results, and Z-Mail may not deliver the mail.
- Angle brackets have the highest precedence and may be used in conjunction with quotes and parentheses:
  The Boss (Tom Justin) <tom>
  Frank "Frankie" Jones <fjones@justin.com>
- You may enter multiple addresses on a line:
  mary, fjones@justin.com, dpowers@ukiahfib.com

Note: Since mail transport agents are case-insensitive, you may specify e-mail addresses in any mixture of upper- and lowercase letters. In other words, *mary@justin.com* is equivalent to *Mary@Justin.COM*.

Specifying Alias Addresses

An alias is an abbreviation of an e-mail address or group of addresses. You can define mail aliases using the following Z-Script command forms:

```
alias john John Talbott <john@justin.com>
alias mary Mary Edwards <mary@justin.com>
alias sales john, mary
```

The aliases *john* and *mary* refer to single users. The alias *sales* refers to the aliases *john* and *mary*. Once you have defined these three aliases, you can address mail to *john, mary* or *sales*, as well as by entering their real e-mail addresses. Note, however that you cannot specify an alias and a comment on the same address line:

```
To: "Sales Department" <all>          NOT ALLOWED
```

Z-Mail does not retain the comment *Sales Department* when it expands the alias *all*. Z-Mail replaces the entire address and comment with the expansion.

CHAPTER 3

Starting and Initializing Z-Mail

You start Z-Mail using the *zmail* command. The command has many options that allow you to configure your Z-Mail session and choose which of Z-Mail's interfaces you wish to use.

Many of the options to the *zmail* command are helpful only when you are using Z-Mail in its send-only mode. Most users who prefer the Motif, Open Look or fullscreen interfaces use Z-Mail menus to perform the various functions offered by the numerous *zmail* command options.

Starting Z-Mail

You start Z-Mail by typing the *zmail* command at your UNIX shell prompt:

GUI To start Z-Mail using its graphical user interface, type:

zmail –gui

Fullscreen To start Z-Mail using the fullscreen character interface:

zmail –fullscreen

Command Line To start Z-Mail using its command-line interface:

zmail

Send-Only You can also use the *zmail* command at the UNIX shell prompt to send mail just like the regular UNIX *mail* command.

SYNTAX The general syntax of the *zmail* command is a series of options followed by arguments:

zmail [*option1* [*arg_list1*]] [*option2* [*arg_list2*]] . . .

The general syntax for sending messages from the UNIX shell command line is:

zmail [–subject "*subject* "] [–attach [*program*:]*filename*]
[–cc *cc_list*] [–bcc *bcc_list*] [to_list] [< *textfile*]

The syntax for editing and sending an existing message file from the UNIX shell command line is:

zmail [–send[!]] –draft *draftfile*

> **NOTE:** The next several pages explain the various options to the *zmail* command. If you do not plan to use Z-Mail's command-line interface or its send-only mode, you may want to skip this section.

Z-Mail Options Summary

This section summarizes the numerous options to the *zmail* command by category:

- User Interface Options
- Initialization and Startup Options
- Options for Sending Mail
- Options for Managing Folders
- Miscellaneous Options

For details about a specific option to the zmail command, see *Z-Mail Options Reference* beginning on page 20.

User Interface Options

–fullscreen Starts Z-Mail's fullscreen character interface.

–motif Starts Z-Mail's Motif interface.

–gui Starts Z-Mail's Motif or Open Look graphical user interface, depending on your version of Z-Mail.

–openlook Starts Z-Mail's Open Look interface.

–shell Forces Z-Mail to start even if you have no mail waiting for you.

–interact Forces Z-Mail to start in interactive mode (i.e., starts one of Z-Mail's user interfaces).

Initialization and Startup Options

–echo Tells Z-Mail to process input only after carriage returns in command-line mode.

–eval Executes a Z-Mail command you specify at startup.

–headers Previews the contents of a folder by displaying header summaries of its messages.

–init Loads an initialization file you specify.

–lib Specifies the location of Z-Mail's program library.

–mailbox Specifies the location of your system mailbox, or System Folder.

–noheaders Does not display header summary information.

–noinit Does not load initialization files.

–picky Stops Z-Mail from supplying *From:* and *Date:* headers to the mail transport agent.

–source Sources (processes) commands from a file you specify.

–timeout Specifies how often you want Z-Mail to check for new mail.

Options for Sending Mail

Most of these options are for using Z-Mail's send-only mode to send mail without starting a Z-Mail user interface.

–attach Attaches a file whose name you specify.

–blind Sends a blind carbon copy of the message to a list of users you specify.

–carbon Sends a carbon copy of the message to a list of users you specify.

–direct Passes all redirected input to the mail transport agent.

–draft Includes a draft file (a prepared text file), whose name you specify, as the body of the message.

–send Send a message with an included draft file immediately.

–subject Set the subject of the message to a character string you specify.

–template Use a text file whose name you specify as a template for a new message.

–verbose Passes the *verbose* option to the mail transport agent.

Options for Managing Folders

–filter Tells Z-Mail to run message filters at startup.

–folder Opens a folder at startup whose name you specify.

–nofilter Tells Z-Mail to avoid running message filters at startup.

–readonly Does not allow updates to the mail folder opened by Z-Mail.

–user Sets the user name to a name you specify.

Miscellaneous Options

–debug Activates verbose user message output for debugging Z-Mail.

–help Displays information on the various options to the *zmail* command.

–register Registers your Z-Mail host license and its users.

Z-Mail Options Reference

This section explains the various options to the *zmail* command in alphabetical order.

–attach

zmail command option *SEND-ONLY MODE*

SYNTAX **–attach [*type*:]*filename***

SYNONYM **–A**

To mail an attachment, specify the *–attach* option and Z-Mail prompts you for additional information about the attachment, such as its filetype and a short description.

You must specify *filename*, and you may optionally precede *filename* with its *type* and a colon. Z-Mail attaches the file called *filename* to your current message.

If you specify a binary file, Z-Mail automatically encodes the file for transport. If the file you specify is a text file, you do not need to specify an encoding format.

If you are unsure that a file is binary or text, specify *type* as *binary,* since you must encode binary files, and it never hurts to encode text files. Note, however, that encoded text files are larger than standard text files.

For additional information on sending attachments, see *Defining Attachment Types* beginning on page 211 and the *detach* command on page 98.

You cannot specify *–attach* if you specify the *–motif*, *–openlook,* or *–gui* option. The *–attach* option overrides the *–fullscreen* option unless you specify the *–shell* option.

EXAMPLE The following command instructs Z-Mail to send an attachment called *reserve.form* to *robert*. Z-Mail prompts you for the additional information it needs to send the attachment.

```
> zmail –A reserve.form robert
Attachment type (? for list): text
Attachment description: Reservation form
Encoding (? for list) : ?
compress   btoa
Encoding (? for list) : none
To: robert
. . . etc.
```

–blind

zmail command option *SEND-ONLY MODE*

SYNTAX **–blind *bcc_list***

SYNONYMS **–blindcarbon, –bcc, –b**

To send a carbon copy of a message to a list of users without having their names appear in the *Cc:* header, specify the *–blind* option. If you specify more than one e-mail address or an address containing spaces in *bcc_list*, you must enclose the entire list in quotes.

You cannot specify *–blind* if you specify the *–motif*, *–openlook* or *–gui* option. The *–blind* option overrides the *–fullscreen* option unless you specify the *–shell* option.

EXAMPLE The following command starts a message addressed to *robert* with blind carbon copies to *keri* and *marc*. Z-Mail then prompts for a subject and waits for you to enter the text of the message.

```
> zmail –bcc "keri marc" robert
To: robert
Subject: Design Meeting at 10:30
Bcc: keri, marc
. . . etc.
```

–carbon

zmail command option *SEND-ONLY MODE*

SYNTAX **–carbon *cc_list***

SYNONYMS **–copy, –cc, –c**

To send a carbon copy of a message to a list of users, specify the *–carbon* option. The list of names in *cc_list* appear in the *Cc:* header of your outgoing message. If you specify more than one address or an address containing spaces, you must enclose the entire list in quotes.

You cannot specify *–carbon* if you specify the *–motif*, *–openlook* or *–gui* option. The *–carbon* option overrides the *–fullscreen* option unless you specify the *–shell* option.

EXAMPLE The following command starts a message addressed to *danielle* with carbon copies for *jon* and *terry*, and a subject of *Architectural Review Notes*. Z-Mail then waits for you to enter the text of the message.

```
> zmail –cc "jon terry" danielle
To: danielle
Subject: Architectural Design Notes
Cc: jon, terry
. . . etc.
```

–direct

zmail command option *SEND-ONLY MODE*

SYNTAX **–direct**

SYNONYMS **–notempfile, –D**

If you are sending a message file and want Z-Mail to pass its standard input directly to the mail transport agent (MTA), specify the *–direct* option. Z-Mail passes the data on its standard input without buffering it in a temporary file. Z-Mail ignores this option unless you redirect input from a file or a UNIX shell pipeline. The *–direct* option is intended for sending large text files when storage space in the file system is at a premium.

EXAMPLE The following command sends the file *rollup.data* to *joe* with a subject of *Consolidated data*.

```
> zmail –s "Consolidated data" –direct joe < rollup.data
```

–draft

zmail command option *SEND-ONLY MODE*

SYNTAX **–draft *draft_file***

SYNONYMS **–h, –headerfile, –message**

To send a previously prepared message file (a *draft*), specify the *–draft* option. Z-Mail reads an existing message file named *draft_file* for sending as an e-mail message. The *draft_file* may be an unsent message generated by an earlier Z-Mail session.

The *draft_file* must contain all message headers needed to send the message correctly. Z-Mail adds only a new *Date:* and a *From:* header if there is none present. The file must contain at least a To: header for Z-Mail to accept it as a message draft. In other words, if there is no *To:* header in the draft file, Z-Mail cannot send the message.

If you specify a **–** character as the *draft_file* name, Z-Mail reads the standard UNIX input for the draft text. However, you cannot use the *–direct* option in conjunction with *–draft*, because the input must be scanned more than once in order to extract information that Z-Mail uses to connect with the MTA.

You can only specify the *-draft* option when you use Z-Mail in its send-only mode. You therefore cannot specify *–draft* if you specify the *–motif, –openlook* or *–gui* option.

EXAMPLE The following command sends the file named *directions* to *linda* after waiting for you to finish editing the text of the message.

```
> zmail –draft directions linda
27 lines
(continue editing or ^D to send)
```

The next command uses the *–send* option to send the draft file without waiting for further input from you.

```
> zmail –draft directions –send linda
27 lines
```

–eval

zmail command option *ALL INTERFACES*

SYNTAX **–eval *zcommand***

SYNONYMS **–e, –execute, –command**

To have Z-Mail execute a command immediately before it begins to accept input from you, specify the *–eval* option. After loading all initialization files, including any you specify with the *–source* option, Z-Mail executes *zcommand*, then waits for input from you. If *zcommand* contains spaces, you must enclose the entire command in quotes.

If you specify several *–eval* options, Z-Mail executes each in the order in which they appear in your *zmail* command.

Z-Mail ignores the *–eval* option when you use the send-only mode of Z-Mail, unless you specify the *–shell* option.

EXAMPLE The following command starts Z-Mail's Motif interface and opens the *Compose* window so you can send a mail message.

```
> zmail –motif –eval "dialog compose"
```

–echo

zmail command option *COMMAND-LINE ONLY*

SYNTAX **–echo**

SYNONYM **–E**

If you specify the *–echo* option, Z-Mail sets local echo so it remains on. This tells Z-Mail to only process input after you type a carriage return. Normally, Z-Mail runs with local echo off and each character you type is processed individually. This

allows Z-Mail to handle macros and keyboard mappings. Typically, keyboard management is transparent to you, and the use of this option is usually not necessary. If you specify the *–echo* option, key sequences you set with the *map* or *map!* commands do not take effect. Z-Mail ignores this option in its fullscreen and GUI interfaces.

–filter

zmail command option *ALL INTERFACES*

SYNTAX **–filter**

To tell Z-Mail to run your message filters as it loads your active folder, specify the *–filter* option. This option also allows Z-Mail to filter incoming mail before it is delivered to your System Folder (see the *–folder* option below). You define a filter by using the *filter* command in an initialization file. If you have not yet defined any filters, the *–filter* option has no effect.

During the loading of your active folder, Z-Mail does not execute filters defined with the command *filter –n*, even if you specify the *–filter* option.

You use this option most frequently in conjunction with *–folder*, such as when you automatically run Z-Mail from a *.forward* or *.maildelivery* file.

If you specify both the *–filter* and *–nofilter* options, Z-Mail recognizes the latter one you enter on the command line.

–folder

zmail command option *ALL INTERFACES*

SYNTAX **–folder [*foldername*]**

SYNONYM **–f**

To open a file named *foldername* as your active folder instead of opening your System Folder, specify the *–folder* option. If you do not specify *foldername*, you must place the *–folder* option as the last argument on the *zmail* command line. When you do, Z-Mail opens your *Main Mailbox* (*mbox*) in your home directory or the folder you specify with the *mbox* variable.

EXAMPLES The following command starts Z-Mail's Motif interface, opens the folder called *surveys*, and makes it your active folder.

```
> zmail –motif –folder surveys
```

By specifying *foldername* as a single hyphen ("–"), you can run Z-Mail automatically from delivery files such as *.forward* and *.maildelivery*. Instead of starting one of its interfaces, Z-Mail performs the actions you specify with the *–filter*, *–source*, and *–eval* options, and then appends all messages not marked for deletion to your System Folder.

WARNING: When you specify *foldername* as a single hyphen ("–"), any action that causes Z-Mail to exit prematurely — whether in initialization files, in a file read by the *–source* option, or in commands you execute with the *–eval* option — prevents Z-Mail from appending messages to the System Folder. In this case, Z-Mail leaves your messages in a temporary file, subject to automatic deletion by your operating system at reboot or after a certain period of time. For this reason, novice users should avoid the *–source* and *–eval* options. Instead, they should use the *–filter* option, using filters defined in the *.zmailrc* file instead.

–fullscreen

zmail command option *FULLSCREEN ONLY*

SYNTAX **–fullscreen**

SYNONYMS **–visual, –V**

To start Z-Mail's fullscreen interface, specify the *–fullscreen* option.

Z-Mail ignores this option when you are using *zmail* as a UNIX shell command to send a single mail message unless you specify the *–shell* option that tells Z-Mail to start its interface.

EXAMPLE The following command starts Z-Mail's fullscreen interface:

```
> zmail –fullscreen
```

–gui

zmail command option *GUI ONLY*

SYNTAX **–gui**

SYNONYMS **–motif, –openlook**

To start Z-Mail's Motif or Open Look interfaces, specify the *–motif, –openlook* and *–gui* option. List this option as the first one on the command line before you specify any other X Window System or Z-Mail options. You may also use the *–motif, –openlook* and *–gui* options to start Z-Mail GUI interfaces.

EXAMPLE The following command starts Z-Mail's graphical user interface:

```
> zmail –gui
```

–headers

zmail command option

SYNTAX **–headers[:*c*]**

SYNONYM **–H**

To preview the contents of a folder without starting a Z-Mail interface, specify the *–headers* option. Z-Mail displays the message headers for all or a subset of the messages in your System Folder, or another folder you specify with the *–folder* option. To identify the messages for which you want to display headers, specify one of the following header modifiers as the *:c* argument:

a	All messages
d	Deleted messages
f	Forwarded messages
n	New messages
o	Old messages
p	Preserved messages
r	Replied to messages
s	Saved messages
u	Unread messages

If you do not specify *:c*, Z-Mail displays a list of headers for all the messages in the active folder (as if you specified **H:a**).

The format of the information Z-Mail displays in the list of headers depends on the value of the *hdr_format* variable. For more information, see the *hdr_format* variable on page 185.

Z-Mail ignores the *–headers* option if you specify the *–motif, –openlook,* or *–gui* option. Otherwise, the *–headers* option prevents Z-Mail from starting an interface, overriding both the *–shell* and *–fullscreen* options.

EXAMPLE Assume your System Folder has three new messages in it and your personal initialization file sets the value of *hdr_format* with this command:

```
set hdr_format = "%15n %M %–2N %7T %20s"
```

If you enter the following command, Z-Mail lists message headers for the three new messages in your System Folder:

```
> zmail –H:n
51   N   Sean O'Brien    Jul 15   8:49am    Meeting at 11:30
52   N   Jon Yost        Jul 16   12:10pm   Porting standards
53   N   Ian Trainor     Jul 16   1:05pm    Maintenance
```

–help

zmail command option *SEND-ONLY MODE*

SYNTAX **–help**

SYNONYM **–?**

If you specify the *-help* option, Z-Mail displays the general syntax of the *zmail* command as well as a list of all options to the *zmail* command. In other words, it displays a subset of the information covered in this section of this manual.

–init

zmail command option *ALL INTERFACES*

SYNTAX **–init[!]** ***filename***

SYNONYM **–I** (upper-case i)

To force Z-Mail to process a specific initialization file before it performs any other initialization activities, specify the *–init* option. Z-Mail reads and processes the file you specify as *filename* before the default system initialization file.

The optional exclamation point (**!**) modifier prevents Z-Mail from processing *system.zmailrc,* the default system initialization file, so you can use the *–init!* option to specify a substitute default file. When you specify *–init!,* Z-Mail stills processes your personal initialization file normally. For more information, see *Initializing Z-Mail* on page 35.

EXAMPLES The following command starts Z-Mail's command-line interface, processing the initialization file called *new.zmailrc* before processing Z-Mail's default initialization files.

```
> zmail –init new.zmailrc
```

The following command starts Z-Mail's fullscreen interface, processing the initialization file called *test.zmailrc* instead of the default file, *system.zmailrc*.

```
> zmail –init! test.zmailrc
```

–interact

zmail command option *ALL INTERFACES*

SYNTAX **–interact**

SYNONYM **–i**

To force Z-Mail to enter its interactive mode (i.e., start one of its interfaces) even if you redirect input to Z-Mail, specify the *–interact* option.

This option is intended for remote host mail sessions (with the *–echo* option), but it also allows you to redirect input from a *script* of Z-Mail commands. For information on writing scripts, see *Initializing Z-Mail* on page 35 and *Z-Script Commands* beginning on page 83.

This option is different from the *–i* (*ignore*) option of UCB Mail.

–lib

zmail command option *ALL INTERFACES*

SYNTAX **–lib *librarydir***

SYNONYM **–L**

If you specify the *–lib* option, Z-Mail searches the directory *librarydir* for the Z-Mail program library. If Z-Mail is unable to find the directory you specify, it searches the directory you specify in the value of the *ZMLIB* variable, as well as the default directory */usr/lib/Zmail*. If Z-Mail is unable to find the library in any of these directories, Z-Mail cannot run.

–mailbox

zmail command option *ALL INTERFACES*

SYNTAX **–mailbox *mailboxpath***

SYNONYM **–m**

To force Z-Mail to use a specific file as your System Folder, specify the *–mailbox* option. Z-Mail treats *mailboxpath* as your System Folder instead of the default */usr/spool/mail/$USER* mailbox.

The *–mailbox* option is helpful if you have configured your mail transport agent to deposit your mail in various files in your home directory. By specifying this option, you tell Z-Mail to treat these files the same way it normally treats your System Folder.

–motif

zmail command option *MOTIF ONLY*

SYNTAX **–motif**

SYNONYM **–gui**

To start Z-Mail's Motif interface running under the X Window System, specify the *–motif* option. You should place this option as the first option on the command line, before any other X Window System or Z-Mail options.

EXAMPLE The following command starts Z-Mail's Motif interface:

> **zmail –motif**

–nofilter

zmail command option *ALL INTERFACES*

SYNTAX **–nofilter**

To prevent Z-Mail from running message filters as it opens your active folder, specify the *–nofilter* option.

You define filters with the *filter* command in an initialization file. If you do not specify any filters, this option has no effect. If you specify both the *–nofilter* and *–filter* options, Z-Mail recognizes only the last one you specify.

–noheaders

zmail command option *COMMAND-LINE ONLY*

SYNTAX **–noheaders**

SYNONYM **–N**

To prevent Z-Mail from displaying message headers in command-line mode, specify the *–noheaders* option. Z-Mail ignores this option if you specify the *–fullscreen*, *–motif*, *–openlook* or *–gui* option.

–noinit

zmail command option *ALL INTERFACES*

SYNTAX **–noinit[!]**

SYNONYM **–n**

To prevent Z-Mail from processing its system initialization files at startup, specify the *–noinit* option. If you specify the optional exclamation point (**–noinit!**), you also prevent Z-Mail from processing your personal *.zmailrc* file as well. For more information on initialization files, see *Initializing Z-Mail* on page 35.

–openlook

zmail command option *OPEN LOOK ONLY*

SYNTAX **–openlook**

SYNONYM **–ol, –gui**

To start Z-Mail's Open Look interface running under the X Window System, specify the *–openlook* option. You should place this option as the first option on the command line, before any other X Window System or Z-Mail options.

EXAMPLE The following command starts Z-Mail's Open Look interface:

> **zmail –openlook**

–picky

zmail command option *ALL INTERFACES*

SYNTAX **–picky**

SYNONYM **–P**

Some mail transport agents do not accept *From:* and *Date:* headers generated by a user agent. To prevent Z-Mail from passing *From:* and *Date:* headers to your MTA, specify the *–picky* option, For more information, see the *picky_mta* variable on page 195.

–readonly

zmail command option *ALL INTERFACES*

SYNTAX **–readonly**

SYNONYM **–r**

If you specify the *–readonly* option, Z-Mail opens your active folder in read-only mode. If you specify this option, you are not permitted to make any updates to the active folder.

–register

zmail command option *ALL INTERFACES*

SYNTAX **–register**

To enter new license information for your Z-Mail installation, specify the *–register* option. Z-Mail prompts you for your Z-Mail license password. If you enter a valid password, Z-Mail updates your license file with the new password and other pertinent information. Then Z-Mail displays the new information about your Z-Mail license.

To register users, you must have permission to write to the */usr/lib/Zmail/license* directory.

For additional information on recording your license information and registering users, see the *Z-Mail Installation Guide*.

–send

zmail command option *SEND-ONLY MODE*

SYNTAX **–send[!]**

SYNONYM **–U**

To immediately send a draft file without editing it further, specify the *–send* option. If you append the optional exclamation point (**–send!**), Z-Mail does not append *signatures* or *fortunes* to your outgoing message.

You may only use *–send* in conjunction with the *–draft* option. For more information see the *mail* command on page 120.

EXAMPLE The following command immediately sends the draft file called *directions* to *linda* without waiting for further input from you.

```
> zmail –draft directions –send linda
27 lines
```

–shell

zmail command option *NOT FOR GUI MODE*

SYNTAX **–shell**

SYNONYM **–S**

If your System Folder or the folder you specify with the *–folder* option is empty or does not exist, by default, Z-Mail does not start. To force Z-Mail to start its shell in this case, specify the *–shell* option.

EXAMPLE The next command uses the *–shell* option to start Z-Mail even though the folder named *meetings* does not exist:

```
> zmail –folder meetings –shell
```

You can also use the *–shell* option to force Z-Mail to start its user interface after you compose a message. Normally, when you specify options or addresses in your *zmail* command for composing a message, Z-Mail assumes you want to be in send-only mode, and returns you to your UNIX shell when you are finished composing your message. If you specify the *–shell* option, Z-Mail starts its interface after you are finished composing the message.

–source

zmail command option *ALL INTERFACES*

SYNTAX **–source[!]** ***filename***

SYNONYMS **–file, –F**

To force Z-Mail to read commands from a file after it opens your active folder, specify the *–source* option. Z-Mail reads commands from *filename* after it opens your active folder, but before it accepts input from you. The file specified by *filename* must have the same format as the initialization file Z-Mail reads on start-up with the exception that you may specify commands that manipulate or search messages. Normally, those commands may not appear in initialization files because Z-Mail processes the initialization file before it opens the folder.

The optional exclamation point (**–source!**) prevents Z-Mail from starting one of its interfaces after sourcing *filename*. Otherwise, Z-Mail continues and starts the interface you specify.

EXAMPLE The following command opens the folder called *results* and processes the file called *sortmsg* that sorts the messages in the folder:

```
> zmail –folder results –source sortmsg
```

–subject

zmail command option *SEND-ONLY MODE*

SYNTAX **–subject** ***text***

SYNONYM **–s**

To specify a subject for an outgoing message, specify the *–subject* option. Instead of prompting you for the subject of your message, Z-Mail uses *text* as its subject. If *text* has any blank space, you must enclose it in quotes.

You cannot specify –subject if you specify the *–motif*, *–openlook* or *–gui* option. The *–subject* option overrides the –fullscreen option unless you specify the *–shell* option.

EXAMPLE The following command starts a message addressed to *sales* with a subject of *New Vending Machines* and then waits for you to enter the text of the message.

```
> zmail –subject "New Vending Machines" sales
To: sales
Subject: New Vending Machines
. . . etc.
```

–template

zmail command option *SEND-ONLY MODE*

SYNTAX **–template *templatename***

SYNONYM **–p**

To load a message template as the starting point for a new message, specify the *–template* option. Z-Mail loads a message template named *templatename* as a new message for you to send. You then fill in the various fields of the template and send it.

You cannot use the *–template* option when you start Z-Mail's graphical user interface. Instead, start *zmail* with the *–gui*, *–motif* or *–openlook* option and select the *Templates* option from the *Compose* menu.

EXAMPLE The following command starts a phone message addressed to *tfoley* by loading a prepared template:

> **zmail –template phonemsg tfoley**
To: tfoley
Subject: Phone Message

Message from: **Ira Cohen**
Phone Number: **800-555-2041**
. . . etc.

–timeout

zmail command option *GUI ONLY*

SYNTAX **–timeout *nseconds***

SYNONYM **–T**

To specify how often you want Z-Mail to check for new mail, specify the *–timeout* option. Z-Mail then checks for new mail every *nseconds*. For performance reasons, you cannot check for new mail more frequently than every thirty (30) seconds. If you specify this option, Z-Mail enters the GUI mode, even if you do not specify the *–motif*, *–openlook* or *–gui* option.

EXAMPLE The following message starts Z-Mail's Motif interface and tells Z-Mail to check for new mail every two minutes.

> **zmail –motif –timeout 120**

–user

zmail command option *ALL INTERFACES*

SYNTAX **–user [*username*]**

SYNONYM **–u**

If you specify the *–user* option, Z-Mail acts as if another user (specified by *username*) is running Z-Mail. If you do not specify *username*, Z-Mail uses *root*.

Note: Z-Mail does not change your userid or permissions to that of the other user, but Z-Mail does set your *home* directory, System Folder, real name, and login name to that of the user named *username*. Z-Mail does not pass this information to the mail transport agent, preventing malicious users from masquerading as others.

The *–user* option is intended for system administrators who need to read mail directed to *postmaster* or to other pseudo-users.

–verbose

zmail command option *SEND-ONLY MODE*

SYNTAX **–verbose**

SYNONYM **–v**

To force Z-Mail to pass the *verbose* option to your mail transport agent when Z-Mail sends messages to it, specify the *–verbose* option. Some MTAs do not have a *verbose* option, so this feature may not apply to your system. This option applies only when you send mail and cannot be used if you specify the *–motif, –openlook* or *–gui* option.

Setting the *verbose* variable also causes Z-Mail to display additional information about the *send* process. For additional information, see the *verbose* variable on page 203.

Initializing Z-Mail

When Z-Mail starts, it processes one or more initialization files that set variable values, aliases and command-line aliases. This capability lets you customize your Z-Mail installation on an organization-wide or personal basis.

When Z-Mail starts, it:

- Interprets the arguments in your *zmail* command,
- Loads a series of initialization files, and
- Opens your active folder.

Loading System Initialization Files

Automatic Initialization

After evaluating your *zmail* command, Z-Mail configures itself to your precise requirements by loading Z-Mail initialization files. Z-Mail loads the following files, in order.

1. Z-Mail first looks for a *–init* or *–I* option to the *zmail* command that you enter to start your Z-Mail session. If you specify the option and a filename following it, Z-Mail loads the filename you specify.
2. Next, Z-Mail loads the default system initialization file, *system.zmailrc*, typically located in the */usr/lib/Zmail/lib directory.* This file is usually created by your system administrator and contains Z-Mail commands and settings used throughout your organization.
3. Z-Mail then loads your personal system initialization file which is defined to be the first file Z-Mail finds from the following list:
   - The file indicated by the value of the ZMAILRC environment variable.
   - The file indicated by the value of the MAILRC environment variable.
   - The file called *.zmailrc* in your *home* directory (*~/.zmailrc*).
   - The file called *.mushrc* in your *home* directory (*~/.mushrc*).

Note: By specifying the following options to the *zmail* command, you affect the initialization files Z-Mail processes:

–init ***filename*** If you specify the *–init* or *–I* option as an argument to the *zmail* command, Z-Mail processes *filename* before any other system initialization files.

–init! ***filename*** If you specify the *–init!* or *–I!* option, Z-Mail processes *filename* before any other system initialization files, and does not load the default initialization file called *system.zmailrc*.

–noinit If you specify the *–noinit* or *–n* option, Z-Mail processes only *.zmailrc*, your personal initialization file.

–noinit! If you specify the *–noinit!* or *–n!* option, Z-Mail processes no initialization files whatsoever.

Opening an Active Folder

Loading Messages

After Z-Mail loads your initialization files, it reads all the messages waiting for you in your System Folder. You may tell Z-Mail to read another folder instead by specifying the following option to the *zmail* command:

```
zmail –f fname
```

Z-Mail loads all the messages in the *fname* folder and makes it your active folder.

Sorting Messages

If you set the *sort* variable on, then once Z-Mail reads your System Folder's messages, it sorts the messages according to the value of the *sort* variable. For more information, see the *sort* variable on page 201.

Sourcing Other Initialization Files

After Z-Mail read the messages in your active folder, it checks if you specified a –source option in your zmail command. If so, it sources the file you specified in the option. In effect, this file is another initialization file.

Manual Initialization

Once you start Z-Mail, you may load initialization files manually using the *source* command on page 141.

Starting the User Interface

After loading all your initialization files and opening your active folder, Z-Mail starts the user interface you selected in your *zmail* command by looking for the following options:

–motif If you specify the *–motif* option, Z-Mail starts its Motif interface.

–openlook If you specify the *–motif* option, Z-Mail starts its Open Look interface.

–gui If you specify the *–gui* option, Z-Mail starts either its Motif or Open Look interface, depending on your version of Z-Mail.

–fullscreen If you specify the *–fullscreen* options, Z-Mail starts its fullscreen character interface.

(*none*) If you specify none of the above options, Z-Mail starts its command-line character interface.

Managing Initialization Files

An initialization file is a script of Z-Mail commands and settings stored in a standard text file.

Creating Initialization Files

Saving Z-Mail Settings

The easiest way to customize your initialization file is to start with one whose settings you like.

GUI

To create an initialization file in GUI mode:

1. Set the variable to value you want using the Variable dialog box (Choose the *Variable* option from the *Options* menu).
2. Click on the *Save* button in the Variables dialog box to create an initialization file called *.zmailrc*.

The *Save* function asks you if you want to overwrite your *.zmailrc* file. If you have made no manual changes to the file, you should confirm the overwrite.

WARNING: If you have added commands and comments or made other manual changes to the file, the save procedure overwrites those changes and they are lost.

Character

To create an initialization file in character mode:

1. Set variables to values you want.

```
set askcc no
set autosign = "Robert Trimble"
set hold on
set hdr_format = "%22n %M %–2N %5T (%1) %25s"
```

   etc. . . .

2. Use the *saveopts* command to save all current variable settings, aliases, and other Z-Mail settable attributes into an initialization file.

The *saveopts* command prompts you for a name for the new file and asks you if you want to overwrite an existing file, if a file by the name you specify already exists.

WARNING: If you have added commands and comments or made other manual changes to the file, the save procedure overwrites those changes and they are lost.

Editing Initialization Files

You can edit existing initialization files to:

- Change variable settings,
- Add or modify aliases,
- Define functions and buttons,
- Add conditional expressions, and
- Add comments.

Adding Commands

Since Z-Mail treats initialization files as scripts, you can almost any Z-Script command in an initialization file.

Exit

The *exit* command has special meaning in a Z-Mail initialization file. If Z-Mail encounters an *exit* command, it immediately stops reading from that file. The *quit* and *x* commands cause the Z-Mail process to terminate completely.

Comments

When Z-Mail reads the initialization file, it recognizes the pound (**#**) character as a comment character. It may be anywhere on a line in the file. When this character is encountered, Z-Mail treats the remainder of the line as a comment and does not process it. If you enclose the **#** in single or double quotes, Z-Mail does not consider it as the beginning of a comment.

EXAMPLE

For example:

```
# this entire line is a comment
set shell = /bin/csh  # only the end of this line is a comment.
# the # sign on the next line is inside quotes.
set prompt = "Message #%m: "
```

Commands to Avoid

You should avoid using certain kinds of commands in Z-Mail initialization files.

Interactive Commands

You should never use interactive UNIX commands in a Z-Mail initialization file. Z-Mail cannot prevent you from using these commands because it is impossible to trace which commands are interactive.

Message Filters

Since Z-Mail reads initialization files before it loads any messages, you should not use commands that filter messages in an initialization file unless you plan to run the file later using the *source* command.

Conditional Expressions

You may use if conditional expressions in initialization files. For more details on writing conditional expressions, see the *if...else...endif* command on page 113.

Z-Mail Internal Variables

Z-Mail provides several internal variables that help you set up conditional processing in initialization files so they can handle the various ways you can use Z-Mail, including via:

- Graphical user interface mode,
- Fullscreen interface mode,
- Command-line interface mode, and
- Send-only mode (no interface; just like the standard *mail* command in UNIX).

The following chart summarizes Z-Mail's internal variables. For details see each variable's entry in the section called *Z-Mail Variables* beginning on page 149.

Variable	***Evaluates to true if:***
is_gui	You started Z-Mail with the *–motif*, *–openlook* or *–gui* option, starting Z-Mail's graphical user interface.
is_fullscreen	You started Z-Mail with the *–fullscreen*, *–visual* or *–V* options, starting Z-Mail's fullscreen interface.
is_shell	You are in the Z-Mail program, and not just calling Z-Mail from a UNIX command shell.
is_sending	You are sending mail from the UNIX command line and do not plan to start a Z-Mail user interface.
hdrs_only	You started Z-Mail with the –headers option, telling Z-Mail to only process message headers.
redirect	You are redirecting input to Z-Mail.

EXAMPLES

The following commands test the internal Z-Mail variable called *is_gui* and exit the initialization file if you are not running Z-Mail's graphical user interface. If you are running the GUI, these commands set the *autoprint* variable.

```
if !is_gui
      exit
else
set autoprint
endif
```

The following commands test to see if your terminal type is *adm3a*. Z-Mail checks the value of *TERM* shell variable or environment variable. If the terminal type is *adm3a*, the commands set the *pager* variable to the *more* program. Otherwise, they set the *pager* variable to *less*.

```
if $TERM == adm3a
    set pager = more
else
    set pager = less
endif
```

CHAPTER 4

Using Z-Mail's Character Interfaces

In addition to Z-Mail's graphical user interfaces, Z-Mail offers fullscreen and command-line interfaces, both of which are character-based. You can use Z-Mail's character interfaces on workstations, standard ASCII terminals, and terminal emulators on personal computers.

Command-Line Interface

Z-Mail's command-line interface is designed for experienced users. You interact with the command-line interface by typing Z-Mail commands and pressing **Return**. If you don't specify all the information necessary to complete your request, Z-Mail prompts you for additional information.

The command-line interface is a conventional tty-command mode similar to command-line interpreters such as *csh* as well as other mailers, such as the University of California at Berkeley's *Mail* and Bell Labs' UNIX System V *mailx* interfaces.

Fullscreen Interface

Users of all experience levels can use Z-Mail's fullscreen interface to manage their mail. The fullscreen interface presents a list of all the messages in your active folder, as well as a menu of command choices at the bottom of the screen.

You interact with the fullscreen interface by pressing single keystrokes that invoke Z-Mail commands. When you ask Z-Mail to send or display a message, it replaces the screenful of message headers with the text of the message.

Starting Z-Mail's Character Interfaces

To start Z-Mail's fullscreen interface, enter the following command:

```
zmail –fullscreen
```

To start Z-Mail's command-line interface, do not specify an interface option (*–fullscreen*, *–motif*, *–openlook*, etc.) in your *zmail* command:

```
zmail
```

Switching Between Interfaces

Once you are in the command-line interface, you can switch to the fullscreen interface by entering:

```
fullscreen
```

If you are in the fullscreen interface and you want to switch to the command-line interface, type a colon (**:**), then press **Return**. The key binding for the *line-mode* command in your Z-Mail installation may be different, so you may have to type another character. To check your current key bindings, press **b** and then **Return**.

To enter a Z-Mail command from the fullscreen interface without switching to the command-line interface, type a colon followed by the command, then press **Return**.

Command-Line Interface Basics

When you use Z-Mail's command-line interface, you can enter almost any of the over 85 commands explained in the section called *Z-Script Commands* beginning on page 83. But to perform the everyday task of managing your mail, you need only a small subset of the commands.

When you first start Z-Mail's command-line interface, Z-Mail displays:

- The name of your active folder,
- The number of messages in the folder,
- A screenful of message summaries for the messages in the folder, and
- A customizable prompt indicating the current message number or other meaningful information.

EXAMPLE

```
[0]+ "/usr/spool/mail/psteele": Type '?' for help.
>1    r   Sean O'Brien     Jul 15   8:49   Meeting at 11:30
 2        Shawna Wilson    Jul 16  11:58   Tuesday's schedule
 3    p   Jon Yost         Jul 16  12:10   Porting standards
 4    N   Ian Trainor      Jul 16  13:05   Maintenance program
 5    N   Shawna Wilson    Jul 18   8:31   Reports Due Now
 6    N   Terri Johnson    Jul 18  10:02   Version 3.1 is ready
psteele: (#1 of 6) >
```

The layout of your screen could be different because the format of both the header fields and the prompt are customizable. For details, see the *hdr_format* variable on page 185 and the *prompt* variable on page 196.

Current Message

The **>** character next to the first message indicates that it is the *current message*, as does the string *#1 of 6* in the prompt at the bottom of the example.

Message Lists

When you enter commands in Z-Mail, you can optionally include a message list. If you do not specify a message list, Z-Mail performs the command you enter on the *current message* (or messages) as indicated by the value of the *output* variable. Every time you execute a command, Z-Mail resets the value of *output* to the messages output (indicated) by the command. For more information, see *Specifying Message Lists* on page 9 and the *output* variable on page 194.

The remainder of this section explains how you perform basic mail management functions using Z-Mail's command-line interface. For a complete discussion of each command in detail, see the section called *Z-Script Commands* beginning on page 83.

Displaying Messages

To display the current message, enter **display**. To maintain compatibility with UNIX mail, you can also enter **type**, **t**, **print** or **p** to display a message. You can abbreviate **display** as **.** (period). You cannot abbreviate **display** as **d**; as in UNIX mail, **d** is the abbreviation for the **delete** command.

EXAMPLES

```
. . .
>5    !    Shawna Wilson     Jul 18     8:31   Reports Due Now
 6    r    Terri Johnson     Jul 18    10:02   Version 3.1 is ready
psteele: (#5 of 6) > display
Message  #5  (15  lines)
From:  swilson@utech.com  (Shawna Wilson)
Date  Mon,  18 Jul 1991  8:31:19 PDT
To:  research
Subject:  Reports Due Now

The deadline for product status reports is today at noon. If you have yet
to submit your report, please e-mail it to me by the end of the day.
```

You can also display a specific message by entering its number, or by entering **display** and its number:

```
psteele: (#5 of 6) > 3
psteele: (#5 of 6) > display  3
```

To display the next undeleted message in your active folder, enter **next**. To display the previous undeleted message, enter **previous**.

Displaying Header Summaries

To display a new set of header summaries for the messages in your active folder, **headers** or **h**.

EXAMPLE

```
psteele: (#8 of 14) > headers
>1    !    Sean O'Brien      Jul 15     8:49   Meeting at 11:30
 2    !    Shawna Wilson     Jul 16    11:58   Tuesday's schedule
 3    N    Jon Yost          Jul 16    12:10   Porting standards
. . .
14    p    Terri Johnson     Jul 27    14:45   Tech support issues
psteele: (#8 of 14) >
```

For information on how to customize the content and format of the information Z-Mail displays in your header summaries, see the *hdr_format* variable on page 185.

Sending Messages

To send a message, enter **mail**. Z-Mail then prompts you with header names that you can fill in. You must enter at least one user address on the *To:* line. All other headers are optional; to omit them from your outgoing message, simply press **Return**. When Z-Mail finishes prompting you for headers, it displays a blank line and puts the cursor at the left side of the screen. You can then enter the text of your message.

EXAMPLE

```
>5   !   Shawna Wilson    Jul 18    8:31   Reports Due Now
 6   r   Terri Johnson     Jul 18   10:02   Version 3.1 is ready
psteele: (#5 of 6) > mail
To: jyost, tjohnson
Subject: Trip to London

I'll be out of the office for the next 10 days on a trip to London. If you
need to reach me, Lori has the itinerary for my trip. I'll be reading my
mail every few days as well. See you in two weeks.
```

When you are finished, type **Return** to move the cursor to the beginning of a new line. Then type **Ctrl-D** to send your message. Z-Mail may prompt you with **Cc:** for a list of users to whom you want to send carbon copies of your message.

Replying to Messages

When you reply to a message, Z-Mail automatically addresses the reply message and fills in the subject line. To send a reply, you can enter one of these commands:

reply	Reply to sender of the current message.
replyall	Reply to sender and all recipients of the current message.

You can abbreviate **reply** as **r** and **replyall** as **R**. You can also specify the number of the message to which you want to reply.

EXAMPLE This command replies to the sender of message number 6:

```
>5   !   Shawna Wilson    Jul 18    8:31   Reports Due Now
 6   !   Terri Johnson     Jul 18   10:02   Version 3.1 is ready
psteele: (#5 of 6) > reply 6
To: tjohnson@utech (Terri Johnson)
Subject: Re: Version 3.1 is ready

Sounds great. How can I get a copy of Version 3.1 to test?
<Ctrl-D>
```

Forwarding Messages

To forward a copy of a message to other users, enter **mail –f**, and Z-Mail prompts you for the addresses of the users:

EXAMPLE

```
mail –f
To: itrainor tjohnson
```

Z-Mail then lets you enter text that it appends to the end of the forwarded message. When you are finished entering text, type **Ctrl-D**. Z-Mail may prompt you with **Cc:** for a list of users to whom you want to send carbon copies of your message.

Signing Messages

To append a personal signature to the end of your outgoing messages, see the *autosign* variable on page 173, and the **~S** command in *Tilde-Escape Editing Commands* on page 62.

Canceling a Message

To cancel a message you are composing, type **Ctrl-C**. Z-Mail will then display this prompt:

```
** interrupt - - one more to kill letter **
```

Type **Ctrl-C** again to cancel the message.

Adding Messages, Files and Attachments

Z-Mail gives you several ways to add information to the text you enter in the body of your mail messages. You may:

- Include the contents of another message in your message,
- Read text from an external text file into your message,
- Use the contents of an external file as a template, and
- Attach an external file to your message.

Including an Existing Message

When you send a new message to another user, you may want to add your comments into the existing text of the another message. Since forwarding a message doesn't allow you to make changes, Z-Mail allows you to include a copy of the current message in an outgoing message. To do so, enter:

```
mail –i
```

You can also include the text of a message by appending its number to the **mail** command:

```
mail – i 4
```

Z-Mail includes the text of the fourth message in your outgoing message, clearly marks it as an included message, and allows you to edit its contents with an external text editor. To invoke a text editor, enter **~e** and Z-Mail starts the editor indicated by the value of the *editor* variable.

Frequently, when you reply to a message, you must include a copy of the original message to make your reply meaningful to its recipients. To include a copy of the original message in your reply, add the **–i** option to your **reply** or **replyall** command:

reply –i
replyall –i

Z-Mail includes a copy of the original message in your reply, identifying it so the recipients of your reply can distinguish your message from the original message. For details on how Z-Mail identifies the original message, see the variables named *indent_str*, *pre_indent_str* and *post_indent_str*.

EXAMPLE

psteele: (#2 of 10) > **mail –i**
To: **rgomez**
Subject: **Feedback from our meeting**

including message 2 **. . .** (37 lines)
I have assembled a few comments that I heard about our last meeting and included them for your review.
<**Ctrl-D**>

Reading Text Files

Another way of importing text is by reading the entire contents of a text file into the body of an outgoing message. To read the file, enter **mail –H** followed by the pathname of the file. If the text file contains the headers for your outgoing message, enter **mail –h** and Z-Mail looks for the message's subject, carbon copy list, etc., in the text of the file as Z-Mail reads it.

Z-Mail loads the exact text of the file as if you had typed all its contents yourself, then lets you edit the text of the file using an external text editor. To invoke a text editor, enter **~e** and Z-Mail starts the editor indicated by the value of the *editor* variable. If the file is a template, you can fill in its various fields and add comments. When you are done, type **Return** to move the cursor to the beginning of a line, then **Ctrl-D** to send the message.

EXAMPLES

psteele: (#5 of 6) > **mail –H results.txt**
To: **sales**
Subject: **Survey results**

105 lines
(continue editing or ^D to send)
If you have any questions, please call me or send me a message.
<**Ctrl-D**>

Note: If you want to precede the included file with your comments, type the comment and use the *~r* editing command explained on page 66 to read in the text of the file.

If you use the **–H** option, Z-Mail assumes it has all the headers information it needs in the text file and does not prompt you for header information.

```
psteele: (#5 of 6) > mail –h results.txt

105 lines
(continue editing or ^D to send)
Here's a summary of the results from the customer satisfaction
survey.
<Ctrl-D>
```

You can also attach files to a message by entering **~r** at the beginning of a text line. For more information, see the *editor* variable on page 181 and *Tilde-Escape Editing Commands* on page 62.

Using Templates

A message template is a partially composed message — in effect, a form — that you can load as a starting point for composing a new message.

Unlike the *–h* option to the *mail* command, you do not need to specify the entire pathname of the file you want to read. You need only specify the template name and Z-Mail looks for it in the directory indicated by the value of the *templates* variable or in the directory */usr/lib/Zmail/forms*.

Note: The default forms directory for your Z-Mail installation may be different if the value of the *ZMLIB* environment variable is not equal to */usr/lib/Zmail*.

The following command starts a new message for you to compose based on the *phone* template.

```
triley (#4 of 17) > mail –p phone

18 lines
(continue editing or ^D to send)
~e
```

To edit the various lines of the template, use an external text editor. To invoke a text editor, enter **~e** and Z-Mail starts the editor indicated by the value of the *editor* variable.

For more information, see the *templates* variable on page 201, the *editor* variable on page 181, and *Tilde-Escape Editing Commands* beginning on page 62.

Sending Attachments

To send the contents of an external file such as a text file or an applications document (such as a FrameMaker or Wingz file), you need to attach the file to your outgoing message. When you attach a file, Z-Mail does not make it part of your message, but instead sends it like an enclosure in a business letter. By sending the attachment in this manner, Z-Mail allows recipients to detach and save the file in its native format.

To attach a file to an outgoing mail message, enter **mail –A** followed by the pathname of the file you wish to attach to the message. The following Z-Mail command attaches a file called *survey.doc* from your home directory to your outgoing mail message.

EXAMPLE

```
psteele  (#11 of 52) > mail  –A  ~/survey.doc
Attachment type (? for list):  framemaker
Attachment description:  Corporate Benefits Survey
Encoding (? for list):  uuencode
To:  allmgrs
Subject: Draft Copy of Benefits Survey
```

Z-Mail then lets you enter the text of your message. For technical details on specifying attachment types, see *Defining Attachment Types* on page 211.

You can also attach files to a message by entering **~A** at the beginning of a text line. For more information, see *Tilde-Escape Editing Commands* on page 62.

Deleting Messages

To delete a message or list of messages, use the **delete** command. The general syntax of the command is:

delete [*messagelist*]

If you do not specify *messagelist*, Z-Mail deletes the current message.

EXAMPLES

The following commands perform as indicated:

delete	Deletes the current message.
delete 2 4-6	Deletes messages 2 and 4 through 6.
delete 6-$	Deletes messages 6 through the last message.

For more information on message lists, see *Specifying Message Lists* on page 9.

Saving Messages

Z-Mail allows you to save messages in *folders* for future reference. To save a message or list of messages to a folder, use the **save** command. The general syntax of the command is:

save [*messagelist*] [*foldername*]

If you do not specify *messagelist*, Z-Mail saves the current message. If you do not specify *foldername*, Z-Mail saves the message in a special folder, your *Main Mailbox* in your home directory. The pathname for your Main Mailbox is *~/mbox*.

EXAMPLES The following commands perform as indicated:

save	Saves current message in your Main Mailbox.
save 5 7 meetings	Saves messages 5 and 7 in the *meetings* folder.
save 1-. ~/mgr/norm	Saves message 1 through the current message in the folder whose path is *~/mgr/norm*.

By default, Z-Mail does not delete messages from your active folder after saving them to another folder. For details on changing this behavior, see the *hold*, *keepsave* and *deletesave* variables.

Printing Messages

To print a message or list of messages, use the **lpr** command. The general syntax of the command is:

lpr [*messagelist*]

If you do not specify *messagelist*, Z-Mail deletes the current message.

EXAMPLES The following commands perform as indicated:

lpr	Prints the current message.
lpr 1 3 4	Prints messages 1, 3 and 4.
lpr .-$	Prints the current message through the last message.

For more information on message lists, see *Specifying Message Lists* on page 9.

Using Aliases and Distribution Lists

The *alias* command allows you to define a name that stands for a list of mail addresses. For example, if you send mail regularly to two field sales managers, you may want to define the alias *salesmgr* as a shorthand for their e-mail addresses:

alias salesmgr kthomas@east.bigcorp ariley@west.bigcorp

For details about aliases, see the *alias* command on page 89.

Sorting Messages

The *sort* command orders message by author, date, status, priority or subject. For example, to sort your active folder by date and within date, by message length, enter:

```
sort –d –l
```

For more information about sorting, see the *sort* command on page 139.

Searching Messages

The *search* command allows you to select messages that match criteria you specify. For example, the following command finds all the messages that are dated prior to January 1, 1991.

```
search –d –1/1/91
```

The following command finds all messages from *kthomas*.

```
search –f kthomas
```

Using pipelines, you can feed the output of one *search* command into another, executing complex searches for specific sets of messages. For more information about searching messages, see the *search* command on page 133.

Special Command-Line Interface Features

Z-Mail supports several features that you may want to take advantage of as an experienced UNIX mail user:

- A customizable command prompt,
- Command history,
- Command-line aliasing, and
- Pipelines that redirect command input and output.

Customizable Command Prompt

Z-Mail lets you customize the prompt that Z-Mail displays when it is waiting for you to enter a command. When Z-Mail displays its command prompt, it evaluates the *prompt* variable. which by default is set to the string:

```
Msg %m of %t:
```

So if you have 10 messages and your current message is 5, then your Z-Mail prompt looks like:

```
Msg 5 of 10:
```

For more information on customizing your command-line prompt, see the *prompt* variable on page 196.

Command History

You can refer to the history of commands you entered earlier in your Z-Mail session, rather than retyping them at the command-line to re-execute them.

Z-Mail supports a history mechanism similar to that supplied by *csh*. Z-Mail also supports selected *csh* history modifiers so you can refer to commands you previously issued and to extract arguments you specified.

The Z-Mail history mechanism stores a list of your most recent *n* commands, where *n* is the value of the *history* variable. If you do not set this variable, Z-Mail only remembers your most recent command.

EXAMPLES

The following command sets the value of the *history* variable to 25, causing Z-Mail to record your most recent 25 commands.

```
set history = 25
```

The following command displays the most recent 4 commands.

```
psteele: (#5 of 6) > history 4
42    display
43    mail –i
44    delete
46    headers 4
```

For details, see the *history* command on page 113 and the *history* variable on page 186.

Referring to Previous Commands

To reference commands you previously entered, use the exclamation point (**!**). You can select some or all of the arguments in previous commands, as well as print the commands using these special characters. The arguments in every command are numbered beginning with argument 1. The command name itself is argument 0. The general form of a history reference is *command:action* where:

Command	***Refers to the:***
!*n*	Command number *n*.
!–*n*	Command that was *n* commands ago.
!!	Current command (same as **!-1**).
!*pattern*	Most recent command beginning with *pattern.*
!?*pattern*?	Most recent command containing *pattern.*

Action	***Refers to:***
n	Argument number *n* (first argument is 0).
n-m	Arguments numbered *n* through *m*.
$	Last argument.
*	All arguments.
p	Prints (displays) the command without executing it.

EXAMPLES

Using these special characters, you can refer to commands you typed earlier.

!–2	Re-execute the command from two commands ago.
!!	Re-execute the most recent command.
mail !!:1-3	Send a message using arguments 1, 2, and 3 from the most recent command.
!4:p	Print (display) the fourth command and all its arguments.
!sort	Re-execute the most recent *sort* command.
!h	Re-execute the most recent command beginning with the letter h.
!4 *	Execute the fourth command, appending * to it.

Turning Off History References

If you send messages using UUCP notation (which contains exclamation points or *bang* characters), you may want to disable Z-Mail's history capabilities. To do so, set the *ignore_bang* variable. Z-Mail then treats all **!** characters literally instead of trying to expand them as history references.

Note: When you source initialization files (e.g., *.zmailrc*), Z-Mail does not expand **!** characters into previously entered commands.

Command Aliases

You use command aliases to create special names for commands. This feature enables command substitution similar to *csh*. To create a command-line alias, use the *cmd* command whose general syntax is:

SYNTAX

cmd [*name* [*command*]]

When you specify *name* and *command*, Z-Mail creates a command-line alias called *name*. Then when you type *name* at the command line, Z-Mail executes the Z-Mail command called *command*. The following commands define command aliases for the Z-Mail commands that follow them:

```
cmd d delete
cmd t type
cmd dt 'd; t'
cmd – previous
```

For more information, see the *cmd* command on page 96.

Piping Commands

You can *pipe* Z-Mail commands to one another using the vertical bar or *pipe character* (|), so Z-Mail uses the output of one command as input to the next command. When you pipe multiple commands together on the same line they are considered to be in a *pipeline*.

The first command in a pipeline uses either the current message or the messages you indicate in the command. Thereafter, each Z-Mail command in the pipeline passes a *message list* of the messages it affected to the next command in the pipeline. Z-Mail suppresses the screen output of all but a few of its commands when you use pipelines.

The most common use of pipelines in Z-Mail is to connect commands that search or select messages to commands that act on those messages.

Note: Since Z-Mail parses the entire command line, you should place questionable shell variables and metacharacters inside quotes to prevent Z-Mail from expanding them.

EXAMPLES

The following command finds all the messages from *fred* and saves them all in the file named *fred_mail:*

```
search –f fred | save fred_mail
```

The next sends messages 4, 5, 6, 7, and 8 to the printer and then deletes them:

```
lpr 4-8 | delete
```

The following example deletes all old mail (mail you have already read):

```
headers :o | delete
```

The next command prints all the message that are more than two weeks old and from *rthomas*.

```
search –ago 2 weeks | search –f rthomas | lpr
```

Pipelining UNIX Commands

Since Z-Mail pipelines pass message lists from command to command, and UNIX commands expect a character string of arguments, you must precede all UNIX commands with Z-Mail's *pipe* command. The *pipe* command allows you to direct a list of messages to UNIX commands.

You may never specify a UNIX command as the first command in a pipeline, unless you use the *pipe* command explicitly. If you wish to execute UNIX commands that are piped to one another, use Z-Mail's *sh* command.

Specifying Multiple Commands

You can specify more than one Z-Mail command on the same line by separating the commands with a semicolon (;). Z-Mail then processes the commands one-at-a-time as though you typed them on two separate lines.

How Z-Mail Processes Commands

Before executing your commands, Z-Mail preprocesses them in a style similar to the C-shell (*csh*). As you enter characters at the command-line prompt, Z-Mail parses each line, separates words and arguments into an array of strings, and expands history references, command line aliases and variable references. When Z-Mail encounters the end of the line, a pipe (|) or semicolon (**;**), it assumes it has reached the end of a command and executes the command.

After parsing your input, Z-Mail searches for command names and predefined function names. If Z-Mail finds one of these commands or functions, Z-Mail calls that operation and passes the remainder of your input as command arguments.

If Z-Mail does not find a command that matches the one you specify, then Z-Mail checks the *unix* variable. If the *unix* variable is set, Z-Mail attempts to run your input as a UNIX command. If the *unix* variable is not set, or if Z-Mail cannot find the command in the path indicated in the *PATH* environment variable, Z-Mail displays a message that the command was not found.

Fullscreen Interface Basics

Z-Mail's fullscreen interface displays a screenful of information at a time. In this interface, Z-Mail uses:

- The top line of your screen to display status information,
- The bottom line to prompt you for input,
- The two lines above the bottom line to show keyboard commands, and
- The remainder of the screen to display messages in your active folder.

To start Z-Mail's fullscreen interface, type **zmail –fullscreen** from your UNIX shell. Z-Mail clears the screen and you see a list of messages in your active folder with summary header information for each message.

EXAMPLE

```
[0]+ "/usr/spool/mail/psteele":  9 messages, 2 new, 3 unread, 0 deleted
>1   r   Sean O'Brien      Jul 15   8:49  Meeting at 11:30
 2       Shawna Wilson     Jul 16  11:58  Tuesday's schedule
 3   p   Jon Yost          Jul 16  12:10  Porting standards
 4       Ian Trainor       Jul 16  13:05  Maintenance program
 5   r   Shawna Wilson     Jul 18   8:31  Reports Due Now
 6   f   Terri Johnson     Jul 18  10:02  version 3.1 is ready
 7   N   Larry Thornton    Jul 19   9:11  heard a good one...
 8   !   Danielle Landis   Jul 19  10:32  Revenue opportunity
 9   !   Ian Trainor       Jul 19  10:53  QA lab is moving upstairs

(RET) display   (s) save        (m) mail          (r) reply
(j) next-msg    (k) back-msg    (z) screen-next   (Z) screen-back
```

You can change the text Z-Mail displays in the bottom two hint lines. For details, see the *fullscreen_help* variable on page 184.

Using the Fullscreen Interface

When you are using Z-Mail's fullscreen interface, you do not enter standard Z-Mail commands. Instead you interact with Z-Mail by pressing single keys that act as a shorthand for a special set of commands that are only recognized in the fullscreen interface.

When you send compose or reply to a message, Z-Mail changes your screen to a normal scrolling mode. As you type the message, Z-Mail adds lines on the bottom of the screen. When you are finished writing, type **Ctrl-D** (end-of-file) to tell Z-Mail to send your message. Z-Mail then returns to fullscreen mode.

While Z-Mail's fullscreen interface has a very visual look, it does not offer some features supported by the command-line interface. Specifically, the fullscreen interface does ***not***:

- Display the keystroke commands you type on the screen.
- Record a history of the commands you enter.
- Allow command piping — the directing the output of one command into the next command.

Z-Mail Fullscreen Commands

The table below summarizes Z-Mail's fullscreen commands and their default key bindings.

***Note*:** Z-Mail is pre-configured with a default set of key bindings that you or your system administrator may reassign using Z-Mail's *bind* command. For this reason, your key bindings may not be the same as the default key bindings indicated in the table below.

Z-Mail Fullscreen Command List

If you are using Z-Mail and can't remember the key for a particular command, just type **?** in the fullscreen interface and Z-Mail displays the information in this table.

Command Name	*Default Key Binding*	*Description*
alias	**a**	Define aliases.
back-msg	**-**, **k**, **K**, **Ctrl-K**	Go to previous message.
bind	**b**	Bind a key sequence to one of the fullscreen commands in this list.
bind-macro	**&&**	Define a fullscreen macro.
bottom-page	**}**	Go to bottom of screen.
chdir	**%**	Change directory.
copy	**c**	Copy one message.
copy-list	**C**	Copy messages; prompt for list.
delete	**d**	Delete one message.
delete-list	**D**	Delete messages; prompt for list.
display-next	**n**	Display next message.
exit	**x**	*Exit Z-Mail without updating folder, or return to fullscreen mode.
exit!	**X**	*Exit Z-Mail without updating folder.
first-msg	**^**	Go to first message.
folder	**f**	Change folder; prompt for name.
folder-menu	**F**	Display menu of folders operations.
goto-msg	**g**	Go to message; prompt for list.
help	**?**	Display help information.
ignore	**i**	Set headers to ignore.

jobs-menu	**J**	Display menu of actions to perform on suspended messages.
last-msg	**$**	Go to last message.
line-mode	**:** (colon)	Escape to command-line mode.
lpr	| (pipe)	Print message.
mail	**m**	Send mail.
mail-flags	**M**	Send mail; prompt for flags (options).
map	**&:**	Define a command-line macro.
map!	**&!**	Define a composition macro.
mark	*	Temporarily mark message.
my-hdrs	**h**	Set personal headers.
next-msg	**+**, **j**, **Return**	Go to the next message.
preserve	**Ctrl-P**	Preserve messages in System Folder.
quit	**q**	*(1) Update folder, then exit Z-Mail, or (2) Return to fullscreen (no update).
quit!	**Q**	*Update folder, then exit Z-Mail.
redraw	**Ctrl-L**	Redraw the screen.
reply	**r**	Reply to sender.
reply-all	**R**	Reply to sender and all recipients.
reply-menu	**Ctrl-R**	Display menu of reply operations.
retain	**I** (shift-i)	Set headers to retain.
reverse-video	~	Toggle to reverse video mode.
save	**s**	Save message.
save-list	**S**	Save messages; prompt for list.
saveopts	**)**	Save options to file; prompt for name.
screen-back	**Z**	Display previous screen of headers.
screen-next	**z**	Display next screen of headers.
search-again	**Ctrl-N**	Continue search for text pattern.
search-back	**Ctrl-**/ (slash)	**Search backward for text pattern.
search-next	/ (slash)	Search forward for text pattern.
shell-escape	**!** (exclamation)	Invoke shell; prompt for command.
sort	**o**	Sort messages.
sort-reverse	**O** (shift-o)	Sort messages; display sort menu.
source	**(**	Source a file; prompt for name.
top	**T**	Display the top lines of a message.
top-page	**{**	Go to top of screen.
unbind	**B**	Unset bindings.
undelete	**u**	Undelete message.
undelete-list	**U**	Undelete messages; prompt for list.
update	**Ctrl-U**	Update folder.
user-button	\ (backslash)	Display user-button menu bar.
variable	**v**	Modify or list variables.
version	**V**	Display version number.
write	**w**	Write messages.

* See the next section called *Exiting the Fullscreen Interface*.

** Some devices use **Ctrl-underscore** in place of **Ctrl-**/.

***Note*:** When Z-Mail displays a listing of key bindings, it indicates the **Ctrl** key as with the **^** character. For example, Z-Mail shows **^U** instead of **Ctrl-U** as the default key sequence for the *update* command.

How Commands Operate in Fullscreen Mode

When you invoke commands in fullscreen mode, Z-Mail prompts you for the information it needs to complete your request. In this mode, Z-Mail functions almost identically to its behavior in the command-line interface.

If the bottom of your screen looks like this and you type **m** to send a message, Z-Mail prompts you for *To:* and *Subject:* headers and lets you enter the text of your message.

```
 8   !   Danielle Landis    Jul 19   10:32   Revenue opportunity
 9   !   Ian Trainor        Jul 19   10:53   QA lab is moving upstairs

(RET) display   (s) save        (m) mail          (r) reply
(j) next-msg    (k) back-msg    (z) screen-next   (Z) screen-back
To: kthomas
Subject : See you at 11:15
. . . etc. . . .
```

Exiting the Fullscreen Interface

The *quit* and *exit* commands have different effects depending on the status of your Z-Mail session. In most situations, the *exit* (**x**) and *exit!* (**X**) commands terminate your session without updating your active folder with any changes you have made. The quit (**q**) and quit! (**Q**) commands update the current folder and then exits Z-Mail.

However, when Z-Mail finishes displaying multi-screen messages, Z-Mail displays a **. . . continue . . .** prompt. When this occurs:

- The *exit* (**x**) command returns you to fullscreen mode,
- The *quit* (**q**) command returns you to fullscreen mode,
- The *exit!* (**X**) command terminates your Z-Mail session.
- The *quit!* (**Q**) command updates you active folder and terminates your Z-Mail session.

Technical Tips for Fullscreen Users

Reverse Video Issues

Z-Mail's fullscreen interface utilizes the curses library — a set of routines for writing cursor- and screen-oriented programs — that is part of most UNIX systems. The exact functionality of the curses library varies across the various versions of the UNIX operating system. To minimize problems caused by these variations, Z-Mail uses only the most basic functions in the curses library.

One unavoidable issue is how reverse video works. Z-Mail uses reverse video to highlight the current message. Z-Mail cannot control the exact behavior of reverse video since it depends on your display device, the *termcap* you are using, and the version of curses you are running. If you have reverse video problems, set the *no_reverse* variable so Z-Mail does not use reverse video. You can still use the ~ command in the fullscreen interface to turn reverse video on and off.

Setting the Number of Messages Displayed

You set the number of message headers Z-Mail displays with the *screen* variable. If you do not set this variable, Z-Mail checks the baud rate and sets the size of the screen according to optimal refresh time. Usually, 300 baud yields 7 lines, 1200 yields 14, 2400 yields 22 lines and all higher baud rates give the size of the screen, whatever that may be. The top line is reserved for *status* information and the bottom line to prompt you for input.

Key Bindings to Avoid

When you define fullscreen macros, avoid defining **Ctrl-C**, **Ctrl-**, and other keystrokes that generate a signal to the UNIX operating system. If you have problems with a particular key binding, your system administrator may have already reserved that key for another use.

Tilde-Escape Editing Commands

While you compose a message in either fullscreen or command-line modes, if you enter a line that has a ~ character in the first column on your screen, Z-Mail treats the line as a *tilde-escape command.*

Tilde-escape commands are a special set of commands that let you edit and save your outgoing message while you are composing it. Z-Mail's GUI interface provides the capabilities of these tilde-escape commands as menu selections. Tilde-escape commands give you the same capability in Z-Mail's character-based interfaces.

The remainder of this section explains the various tilde-escape commands you can use in Z-Mail's fullscreen and command-line interfaces while you are composing messages.

Modifying Headers

~t [*tolist*] Use **~t** to modify the list of recipients in the *To*: list. If you specify a list of addresses in *tolist*, Z-Mail appends this list to the current *To:* list. If you do not specify *tolist*, Z-Mail displays the current list and places the cursor at the end of the list. You can then backspace over the addresses in the list or append more addresses onto the end of the list as you wish.

EXAMPLE If the following example, you begin composing a message in character mode (you could have also begun by pressing the mail (**m**) key in fullscreen mode), and then realize you wanted to send your message to another person.

swilson (#20 of 22) > **mail**
To: **tjohnson gstewart**
Subject: **Offsite meeting**

The offsite meeting to finalize next quarter's development schedule will be next Wednesday at Provincial Suites starting at 8:30.Terri Johnson will present the status of the A/P system, Kim Fong will
~t
To: tjohnson gstewart **kfong**
(continue editing letter)
bring us up to date on the inventory system, and Guy Stewart . . .

~b [*bcclist*] Use **~b** to modify the list of blind carbon copy recipients in the *Bcc:* header of your outgoing message. If you specify addresses in *bcclist*, Z-Mail appends the addresses to your existing *Bcc:* list. If you do not specify *bcclist*, Z-Mail displays your existing *Bcc:* header so you can edit it.

EXAMPLE The following example displays the blind carbon copy list so you can edit it or add to it as you are entering the text of an outgoing message.

```
. . .
without making the necessary changes. We can modify the other
~b
Bcc: pvolk  bgreene
(continue editing letter)
systems later to reflect the new specifications . . .
```

~c [*cclist*] Use **~c** to modify the list of carbon copy recipients in the *Cc:* header of your outgoing message. If you specify addresses in *cclist*, Z-Mail appends the addresses to your existing *Cc:* list. If you do not specify *cclist*, Z-Mail displays your existing *Cc:* header so you can edit it.

EXAMPLE The following example adds a name to the carbon copy list as you are entering the text of an outgoing message.

```
. . .
the publishing deadline. If you hurry and send in your materials, you
may get them to Rick Gomez in time. Otherwise, you should send
~c  rgomez
(continue editing letter)
them via air courier to the conference in Paris . . .
```

~s [*subject*] Use **~s** to modify the subject of your outgoing message. If you specify *subject*, Z-Mail replaces your existing subject with the new one you specified. If you do not specify *subject*, Z-Mail displays your existing *Subject:* header so you can edit it.

```
swilson (#20 of 22) >  mail
To:  tjohnson  gstewart
Subject:  Offsite meeting

The offsite meeting to finalize next quarter's development schedule
will be next Wednesday at Provincial Suites starting at 8:30. Please
~s
Subject: Offsite Meeting next Wednesday at 8:30
(continue editing letter)
arrange for your own transportation . . .
```

~h Use **~h** to modify all message headers. Z-Mail displays each header one-at-a-time so you can edit it. If you don't want to change a particular header, press **Return** instead of editing it.

swilson (#20 of 22) > **mail**
To: **tjohnson gstewart**
Subject: **Offsite meeting**

The offsite meeting to finalize next quarter's development schedule will be next Wednesday at Provincial Suites starting at 8:30.Terri Johnson will present the status of the A/P system, Kim Fong will
~h
To: tjohnson gstewart **kfong**
Subject: Offsite meeting **is Wednesday at 8:30**
Cc: **<Return>**
Bcc: **<Return>**
(continue editing letter)
bring us up to date on the inventory system, and Guy Stewart . . .

Viewing and Editing Message Text

~e [*editor*] Use **~e** to start the text editor indicated by the value of the *editor* variable, environment *EDITOR*, or *vi.* You may enter **~e** at the beginning of any line in the text of an outgoing message. Z-Mail starts an edit session and loads the text of your message into the editor. When you are finished editing, save your file and Z-Mail returns you back to the outgoing message screen.

~p [*pager*] Use **~p** to display the headers and text of the outgoing mail message. If the message is longer than one screen, Z-Mail pages the message using the program called *pager*. If you do not specify *pager*, Z-Mail uses the pager program indicated by the *pager* variable, the *PAGER* environment variable, or the default pager set up by your system administrator. If Z-Mail can't find any of these pagers, it uses its own internal pager program.

~u As you are composing outgoing messages and you notice a mistake in the previous line, you can edit the line without having to start an external editor by entering **~u** and pressing **Return**. Z-Mail displays the previous line and places the cursor at the end, allowing you edit the line.

Note for UNIX System V users: If the new, edited line is shorter than it was before the **~u** command, Z-Mail pads the line with blanks to preserve the previous length of the file.

EXAMPLE The following example shows how you use ~u to edit the contents of line you most recently entered.

The staff meeting will be Monday at 8:30. <Return>
~u
The staff meeting will be Monday at **9:30.**

~v [*editor*] Use **~v** to start the visual editor indicated by the value of the *visual* variable, environment *VISUAL*, or *vi.* You may enter **~v** at the beginning of any line in the text of an outgoing message. Z-Mail starts an edit session and loads the text of your message into the editor. When you are finished editing, save your file and Z-Mail returns you back to the outgoing message screen.

~E Use **~E** to tell Z-Mail to save the contents of the outgoing message immediately in the dead letter file indicated by the value of the *dead* variable. Z-Mail then starts a new outgoing message using the existing message headers.

~E! Use **~E!** to tell Z-Mail to discard the contents of the outgoing message and start a new message.

Adding Signatures

~S Use **~S** to append a signature to the end of the outgoing message. Z-Mail evaluates the *autosign* and *autosign2* variables to determine the text of the signature to append to the message.

EXAMPLE To add a signature at the end of your message, enter ~S.

~S
Adding signature at end of message.
(continue editing letter)

~S! Use **~S!** to stop Z-Mail from appending a signature to the end of the outgoing message.

For more information on signing messages, see the *autosign* variable on page 173 and the *autosign2* variable on page 174.

~F Use **~F** to append a fortune to the end of the outgoing message.

~F! Use **~F!** to stop Z-Mail from appending a signature to the end of the outgoing message.

For more information on appending fortunes to messages, see the *fortune* variable on page 183.

Including Other Text Files

~i [*messagelist*] Use **~i** to include the body text, but not headers of other message(s) in an outgoing message. If you specify a *messagelist*, Z-Mail includes the bodies of all the messages you list. If you do not specify *messagelist*, Z-Mail includes the current message.

Z-Mail marks the included message by indenting it, as well as preceding and following it with descriptive text. For details on how Z-Mail marks included messages, see the *indent_str* variable on page 187, the *pre_indent_str* variable on page 195, and the *post_indent_str* variable on page 195.

EXAMPLE The following example shows how to include the text of messages 8, 9, 10 and 14 in your outgoing message.

To: **kfong**
Subject: **Survey Responses**

Here are the responses I received from the survey.
~i 8-10 14
Including message 8 . . . (24 lines)
Including message 9 . . . (37 lines)
Including message 10 . . . (18 lines)
Including message 14 . . . (21 lines)
(continue editing letter)

~I [*messagelist*] The **~I** tilde-escape command functions the same as **~i**, but also includes the message headers of included messages in your outgoing message.

~f [*messagelist*] Use **~f** to include entire messages in the body of an outgoing message. If you specify *messagelist*, Z-Mail includes all the messages you list. If you do not specify *messagelist*, Z-Mail includes the current message.

Z-Mail marks forwarded messages by preceding and following them with a line saying *Forwarded mail*. Z-Mail does *not* indent forwarded messages in the body text of the outgoing message.

~r *filename* Use **~r** to read the read a text file called *filename* into your outgoing message. Z-Mail includes the text of the file as if you had typed it all on your keyboard, and does not indicate in any way that the text has been loaded from a file.

EXAMPLE The following example reads the file *surveytext* into your outgoing message.

To: **sales**
Subject: **Competition survey**

Please take five minutes to fill out this survey on your latest competitive situations so we can prioritize our product development and marketing plans. Thanks for the help.

~r surveytext
49 lines
(continue editing letter)

Processing Attachments

~A *filename* Use **~A** to attach files to outgoing messages. The **~A** tilde-escape works similarly to the *–attach* option in the *zmail* command, prompting for additional information such as a short description, encoding and decoding instructions.

EXAMPLE The following example attaches a FrameMaker file to the outgoing message.

To: **khagen**
Subject: **Chapter 7 draft**

Kevin -
I have attached the FrameMaker file of the second draft of Chapter 7. Please call me with any questions or comments. Thanks.
~A chap7.doc
Attachment type (? for list): **frame**
Attachment description: **Chapter 7 second draft**
Encoding (? for list): **btoa**
(continue editing letter)

~A Use **~A** to list the numbers and descriptions of all attachments for the currently outgoing message. You can use this information to check that you have properly attached documents you are sending, or to check message numbers so you can delete a specific attachment (see **~A!** below to delete attachments).

EXAMPLE The following example lists all attachments to the outgoing message.

```
. . . and I've included all the files you need to finish the project.
~A
Part  Name       Type Key  Encoding  Size       Description
1     chap7.doc  frame           btoa  (pending)  Chapter 7 Second
2     hiho.c     text         (none)  (pending)  himem module
3     filler.c   text         (none)  (pending)  buffer module
(continue editing letter)
```

~A! ***part*** Use **~A!** to remove a specific attachment from the currently outgoing message. If you do not specify *part* to indicate which attachment to detach, Z-Mail prompts you for a part number or name.

EXAMPLE The following example deletes an attachment from the outgoing message.

```
. . . and I'll send the himem module later.
~A!
Part  Name       Type Key  Encoding  Size       Description
1     chap7.doc  frame           btoa  (pending)  Chapter 7 Second
2     hiho.c     text         (none)  (pending)  himem module
3     filler.c   text         (none)  (pending)  buffer module
Remove part (number or name): 2
(continue editing letter)
```

Storing Messages

~w ***filename*** Use **~w** to write the outgoing message to a file called *filename*. If the *edit_hdrs* variable is set or if you included the *–E* option in the *zmail* command that started your current Z-Mail session, Z-Mail includes the headers of the outgoing message in the file. Otherwise, Z-Mail writes only the text of the outgoing message to the file.

You can use files that include headers as *draft files* — partially composed messages that you can edit and send later. You can read files without headers into outgoing messages later using the **~r** command explained above.

WARNING: The **~w** command overwrites the existing contents of filename. To append to filename without destroying its contents, use the **~a** command below.

~a ***filename*** Use **~a** to append the text of the outgoing message to the file named *filename*. The ~a command works similarly to the **~w** command, but appends to filename instead of overwriting its contents. You can use the **~a** command to keep a log of selected outgoing messages, or as a clipboard that allows you to retrieve text from messages you composed earlier.

Exiting Compose Mode

~q Use **~q** to quit the outgoing message and save it in the dead letter file specified by the *dead* variable. If the *nosave* variable is set, Z-Mail quits without saving the outgoing message.

~z Use **~z** to suspend the outgoing message and return control to the fullscreen or command-line interface. When you suspend a message, Z-Mail neither sends it nor discards it. To resume the outgoing message later in your Z-Mail session, use the *resume* command. To display a list of suspended messages, use the *jobs* command.

EXAMPLE The following example suspends the outgoing message, sets the value of a variable, then resumes the message.

```
. . . the purpose of the meeting. If we can't come to some conclusion
quickly, we should implement the system as it has been designed.
~z
[1]   Stopped.
psteele: #24 of 31 > set  record
psteele: #24 of 31 > resume
(continue editing letter)
We can always add the reporting module later.
```

For more information on suspending messages, see the *resume* command on page 130 and the *jobs* command on page 118.

~x Use ~x to immediately exit the outgoing message without saving its contents in the dead letter file.

EXAMPLE The following example terminates the outgoing message, and returns you to the Z-Mail prompt.

```
. . . the purpose of the meeting. If we can't come to some conclusion
~x
psteele: #24 of 31 >
```

~:*command* This tilde-escape runs *command* as a Z-Mail command. You may not run any command that sends mail. Use **~z** to suspend the current outgoing message if you wish to compose another message. Be careful changing folders with the **~:folder** command since it invalidates any current lists of message numbers.

EXAMPLE The following example displays a list of new mail messages, and then resumes editing the outgoing message.

```
. . . just in time for the quarterly review meeting. I am planning on
~:headers  :n
  5    !    Shawna Wilson     Jul 18    8:31   Reports Due Now
  6    !    Terri Johnson     Jul 18   10:02   Version 3.1 is ready
(continue editing letter)
presenting our latest designs on the Firefox project. If you have . . .
```

~|*command* **(tilde-vertical bar)** Run the UNIX command specified by *command*. You may not run any command that sends mail. Z-Mail passes the text of the outgoing message to *command* as input, and replaces the outgoing message text with the text output by *command*.

If the edit_hdrs variable is set, Z-Mail includes the headers of the outgoing message as input to command; if edit_hdrs is not set, Z-Mail passes only the text of the outgoing message to command.

EXAMPLE The following example passes the text of the outgoing message to a program that checks for spelling mistakes and passes back a marked-up text file indicating any errors. Z-Mail loads the corrected text as the new outgoing message.

```
. . . and thanks again for all your help.
/ David/
~|spellchk
(continue editing letter)
```

Note: The program called *spellchk* is only an example. Your UNIX installation may have a program that checks spelling by another name.

~~ Use the ~~command to begin a line with a tilde character. When Z-Mail encounters tilde characters in screen columns one and two, it inserts a single ~ into the outgoing message.

~$*variable* Insert the string value of *variable* into message. If you specify a boolean variable, Z-Mail inserts nothing.

EXAMPLE The following example inserts the value of the *hdr_format* variable in the text of the outgoing message, then uses the **~u** tilde-escape command to display the line that Z-Mail inserted in response to the **~$** command.

To: **tjohnson**
Subject: **Formatting your Z-Mail header summaries**

Here's the formatting string I use as the value of the hdr_format variable:
~$hdr_format
(continue editing letter)
~u
%19n %M %–2N %6T %.25s

Using Macros

Z-Mail provides three types of macros that help you manage your mail more quickly:

- Fullscreen macros created with the *bind-macro* command.
- Command-line macros created with the *map* command.
- Composition macros created with the *map!* command.

Fullscreen macros work in the fullscreen interface, command-line macros work in the command-line interface, and composition macros work when you compose messages. No macros, including composition macros, work in the GUi interface. Command-line and composition macros do not work when you start Z-Mail with the *–echo* (*–E*) option.

How Macros Work

Key Sequences and Expansions

Macros consist of two parts: a *key sequence* and an *expansion*. The *key sequence* is the character or string of characters that Z-Mail watches for as a trigger to tell it to run the macro. The *expansion* is the string of keystrokes that Z-Mail actually executes when it recognizes the *key sequence*.

Macros are like an interactive search-and-replace function; if you type a recognized key sequence, Z-Mail substitutes its associated expansion in its place. Thus, if you create a macro whose key sequence is **Ctrl-X Ctrl-W** and whose expansion is the string **save**, then when you hold down the **Ctrl** key and type **x** and **w**, the effect is as if you had actually typed the four characters **s, a, v,** and **e**. This process is called *expanding* the macro.

Key sequences are usually made up of control characters or special strings of characters generated by *function keys*, but may be almost any string you desire. Keys that generate a signal from the keyboard (for example, on BSD systems, **Ctrl-Z** generates a TSTP signal and **Ctrl-C** generates an interrupt) can never appear in key sequences, and macros in line or composition modes cannot begin with a *newline*, **Ctrl-D**, or any of the editing keys (*erase*, *word-erase*, *line-erase*, etc.). Otherwise, there are no restrictions.

Key sequences for command-line and composition macros should be unusual characters or combinations of characters, and not individual lower-case letters. If you use common char-

acters or strings for key sequences, you could inadvertently execute macros while typing the text of commands and messages.

Keystroke Timeouts

As you type in commands or the text of messages, Z-Mail uses a timeout to determine whether the keys you have pressed are a key sequence. In other words, once you type the first character of a key sequence, you must type the succeeding characters relatively quickly, or Z-Mail assumes you didn't mean to invoke a macro with a key sequence. For this reason, you should keep key sequences short, or you should automatically generate them by pressing a special function key on your keyboard.

Macros Are Recursive

Macros are recursive, so if the key sequence that invokes one macro appears in the expansion of another macro (or even of the same macro), Z-Mail recognizes the second key sequence as it expands the first macro, and then Z-Mail expands the second sequence as well. Be careful that you do not unintentionally define recursive macro expansions that continue to call each other.

You can prevent Z-Mail from expanding key sequences by using *literal-next characters* from the keyboard or in expansions. In either case, they prevent the character following them from being interpreted as part of a key sequence. There is no literal-next mechanism for fullscreen mode.

Note for advanced users: Z-Mail recognizes the literal-next character from the *tty* settings of your terminal or workstation, provided you have the new BSD-style device driver. Otherwise, Z-Mail recognizes **Ctrl-V** as the *literal-next* character. If you have a *tty literal-next* character, you need to type two of them sequentially to send one to Z-Mail; this is because the *tty* driver consumes the first one. It is not necessary to use two *literal-nexts* in macro expansions unless you wish to cause the second *literal-next* to be literal. In addition, you can use a backslash as a *literal-next* when you type, and sometimes as a literal-next in expansions; but use it with caution, because it also introduces escape sequences.

Errors Abort Macros

A macro always aborts whenever any command called by the macro returns an error. This includes recursive expansions, so no matter how often a macro has recurred, it is terminated completely when it encounters an error. Errors in fullscreen mode include illegal cursor movements, such as up from the top of the screen or down from the last message.

Specifying Control Characters

Z-Mail supports a special syntax for specifying control characters and other non-printing characters in macro key sequences and expansions. This syntax uses the backslash character and is the same as the syntax for specifying key bindings.

If you want to define a key sequence that includes control characters, specify the **Ctrl** key using the **^** character. For example, to define **Ctrl-K** as a key binding, you must bind it as **^K**.

The table below summarizes the keys you must enter to indicate special keys when you are defining macros.

To indicate a:	*Enter:*
Control	**^**
Return	**\n**
Tab	**\t**
^	**\^**
Esc	**\E**

Thus, to create a command-line macro for **Ctrl-X Ctrl-W**, as in the example above, you enter the command:

```
map '^X^W' save
```

If you have an old terminal that does not have a **^** key, use **\C** instead. Therefore, to indicate **Ctrl-P**, you must enter **\CP**.

Prompting for Input within Macros

Z-Mail also provides a method for executing functions from within macros that request input from the user. Two special functions — *getstr* and *getline* — interrupt expansion of the current macro and wait for the user to enter a string at the keyboard. Z-Mail then inserts this string into the macro expansion.

***getline* function**

The *getline* function retrieves the input string and retains the *newline* character (**Return**) at the end of the input string.

***getstr* function**

The *getstr* function retrieves the input string and strips off the *newline* character (**Return**) at the end of the input string.

You tell Z-Mail to execute these functions by surrounding their names with square brackets. Make sure to include no white space inside the brackets. For example:

```
map '^X^W' save [ getline ]
```

creates a command-line macro that runs when you type **Ctrl-X Ctrl-W**, and which displays **save** followed by a space and then waits for you to type a line of input. Z-Mail then uses the line you type as arguments to the *save* command.

Fullscreen Macros

You can use fullscreen macros to access any of Z-Mail's fullscreen command quickly and easily.

To create fullscreen macros, use the *bind* command. To reference a fullscreen command, you simply type the character that invokes that command.

EXAMPLES

To sort your messages by date and then send the most recent one to the printer, you can use the following command:

```
bind  @  macro  'od$|'
```

When you type @, this macro performs several steps:

1. Sorts (**o** from the default bindings), by date (**d**).
2. Moves the current-message pointer to the last message (**$**).
3. Prints that message (**|**).

The previous macro is somewhat cryptic since it uses key bindings instead of the names of fullscreen functions. Using function names, you can define the same macro with the following command.

```
bind  @  macro  [sort ]d[last-msg][lpr]
```

For a list of the names of fullscreen functions, see *Z-Mail Fullscreen Command List* on page 61. The only function you cannot use inside square brackets is the special *macro* function.

One of the most common uses of fullscreen macros is to quickly perform functions that require an escape to command-line mode. The following macro binds **R** to escape to command-line mode and types the string **reply** followed by a space. You can then indicate option flags for the *reply* command that you want Z-Mail to use. In this case, you provide the *–ei* options as defaults.

```
bind  R  macro  '[ line-mode ] reply  –ei'
```

If you use the *getline* or *getstr* function in an escape to command-line mode, you cannot erase the text you typed before the function. The following macro forces you to use the *–ei* options to the *reply* command.

```
bind  R  macro  '[ line-mode ] reply  –ei  [ getline ]'
```

Command-Line Macros

Command-line macros combine the convenience of single-keystroke commands with the versatility of the command-line interface. You use the *map* command to create command-line macros.

EXAMPLES

If you frequently forward messages to the e-mail address *fred*, you can create a macro to do it with the following command.

```
map '^F' 'mail –f . fred\n'
```

This macro causes the single keystroke **Ctrl-F** to forward the current message to the e-mail address *fred*. The *newline* character **\n** at the end of the expansion causes the command to be executed immediately, without requiring you to press **Return**. Z-Mail displays the expansion part of a command-line mode macro on your screen as Z-Mail expands it, so you can see what the macro is doing. You can therefore use parts of the expansion as a prompt.

In the above example, you can prompt for a message list rather than always forwarding the current message by changing the macro to:

```
map '^F' 'mail –f [ getstr ] fred\n'
```

This macro displays *mail –f* and a space, then waits for a you to enter a string at the keyboard. After you press **Return**, the *getstr* function strips the *newline* character and passes the string as the message list to the *mail* command. The macro also passes the address *fred* to the *mail* command, forwarding the messages to *fred*.

Making Changes to Macro Input

If you want to be able to change your mind after starting a command-line macro, you must leave the **\n** out of the expansion. Without the *newline*, Z-Mail waits until you enter **Return** before executing the macro, so you have a chance to erase or change the command. You can never backspace past the beginning of a string retrieved by the *getline* function, and you can backspace past the beginning of a string retrieved by *getstr* only after the *getstr* function has completed its work.

Note: To use command-line macros you must be in the command-line interface; you cannot use command-line macros while using the line-mode escape function in fullscreen mode. If you are in the fullscreen interface, you must type **Return** at the : prompt in order to enter command-line mode.

Composition Macros

Composition mode macros are essentially typing accelerators that work while you compose messages. You specify composition macros similarly to command-line macros, except you use the *map!* command (note the exclamation point).

EXAMPLES To define a macro that inserts the word **percent** into a message whenever you press the**%** key, enter the following command.

```
map!  % percent
```

Another common use of composition macros is to perform regular indenting at the beginning of message lines. The following macro inserts four spaces into your message whenever you type **Ctrl-T**.

```
map!  '^T'  '    '
```

Invoking Tilde-Escapes with Macros

You can also use composition macros to execute tilde-escape editing commands. For example, you could create a macro to invoke an external text editor:

```
map!  '^E'  '\n~v\n'
```

When you type **Ctrl-E**, this macro types a *newline* (to be sure that the *tilde* is the first character on a line), then types **~v** followed by another *newline*, to start the editor. For more information, see *Tilde-Escape Editing Commands* on page 65.

Mixed-Mode Macros

Normally, you create macros to run in a specific mode such as command-line, fullscreen or composition mode. You can, however specify an expansion in a macro that switches modes. For example, the command:

```
map!  '\CE' '\n~v  emacs\n'
```

defines a macro that starts the *emacs* editor when you enter **Ctrl-E** while composing a message. Once you have defined this macro, you can create a command-line macro that calls it:

```
map '#'  'reply  –i  [ getline ] ~t  [getstr]\n^E'
```

Now when press the **#** key in command-line mode, this macro displays *reply –i* and waits for a message list, then starts composing a reply. The macro displays the *To:* line (the **~t** escape) and waits for you to add other addresses. Finally, it recursively expands the **Ctrl-E** macro, to start editing the message with *emacs*.

Note: As you can see from this example, the Z-Mail macro facility is very powerful. But you should exercise caution to avoid accidentally expanding recursive macros, especially when you use macros that change modes. When you test new macros, it is a good idea to start Z-Mail with your active folder in read-only mode (**zmail –r**) to be sure you do not lose or alter messages unexpectedly.

Deleting Macros

You do not have to delete a macro in order to redefine it. When you enter a command that specifies a new expansion for a previously defined key sequence, the new expansion automatically replaces the old one.

To remove the definition of a macro, use the following commands:

Use the command:	***To delete:***
unbind	Fullscreen macros
unmap	Command-line macros
unmap!	Composition macros

Remember to use a backslash (\) or other *literal-next* character to prevent the expansion of line mode macros when you use these commands, especially *unmap*.

CHAPTER 5

Z-Script Commands

This section describes Z-Script commands you can type at the Z-Mail command prompt, in the command entry field in GUI mode or within functions you create using the Z-Mail GUI function editor or any standard text editor of your choice. You can use these commands to build scripts that operate on messages and folders. Using *function* and *cmd*, you can build your own functions and commands that add new functionality to Z-Mail and perform repetitive actions.

You can also use scripts and functions to customize Z-Mail so it resembles other e-mail products. Z-Mail is pre-configured to provide many commands that are compatible with other line-mode e-mail products, including all the commands in basic UNIX mail. This allows you to leverage your prior knowledge while you become familiar with the advanced capabilities of Z-Mail.

This section provides an alphabetical list of available commands, with syntax summaries and some examples. See the *Z-Mail User's Guide* and *The Z-Mail Handbook* for additional information on writing scripts.

Command Summary

This section summarizes Z-Script commands into functional categories. The categories appear in the following order:

- Help Commands
- Message Commands
- Folder Commands
- Sending and Addressing Commands
- GUI Interface Commands
- Programming Commands
- UNIX and System Commands
- Configuration Commands

For details on individual Z-Script commands, see the section called *Command Reference* beginning on page 16.

Help Commands

?	Displays a list of valid Z-Mail commands.
about	Displays information about the creators of Z-Mail.
help	Displays help information on various following topics.
version	Displays the current version of Z-Mail.

Message Commands

delete	Marks messages for deletion.
detach	Stores one or more message attachments and optionally displays those attachments.
display, type, print	Displays the contents of a message.
edit	Opens messages so you can edit them using either the *editor* you specify.
flags	Examines or sets the status flags of messages.
from	Displays information taken from the headers of your message list.
headers	Displays a screenful of message headers from messages in the active folder.

lpr	Sends the contents of messages to the printer.
mark	Tags messages with priorities and temporary marks.
next	Displays the contents of the next message, or the current message if it has not yet been read.
preserve	Saves messages to your System Folder rather than your Main Mailbox unless you explicitly delete them.
previous	Displays the contents of the previous message.
search, pick	Searches for and selects messages according to specific criteria.
sort	Sorts the messages in the active folder according to a variety of criteria.
top	Displays the top lines of messages so you can review them quickly in character mode.
undelete	Reverses the effect of a *delete* command.
undigest	Bursts multi-message digests into individual messages.
unmark	Removes priorities and temporary marks from messages.
unpreserve	Stops Z-Mail from saving messages to your System Folder rather than your Main Mailbox unless you explicitly delete them.

Folder Commands

copy	Copies messages to another folder.
close	Closes mail folders.
folder	Changes the active folder, opens a folder, or closes a folder.
folders	Lists the folders that are present in the folder directory. In the GUI interface, this is the same as *dialog folders*.
merge	Copies messages from a folder you specify into your active folder.
save	Saves messages to a file or folder.
update	Writes all changes that have been made to a folder (usually the active folder) to disk.
write	Saves the body text from messages to a file or folder, ignoring message headers.

Compose and Address Commands

alias	Creates e-mail address aliases and distribution lists.
alternates	Tells Z-Mail about other e-mail addresses you may have.
expand	Shows the addresses referenced by a mail alias. Z-Mail fully expands nested or hierarchical aliases.
mail	Opens a window so you can compose a message.
reply	Replies to a message you have received, addressing the response only to the sender of that message.
replyall	Replies to the sender of a message and also to everyone who received a copy of the message.
unalias	Deletes aliases and distribution lists.

GUI Interface Commands

dialog	Displays any of Z-Mail's standard dialog boxes.
iconify	Closes Z-Mail window to an icon on your UNIX desktop.

Programming Commands

ask	Asks a question and obtains a response from the user.
builtin	Calls a Z-Mail command having the same name as a user-defined function.
each	Executes a command you specify on each message in a message list.
echo	Re-displays the commands you type, expanding all arguments to show their values.
error	Works just like *echo*, except sets *status* error code to -1.
eval	Re-parses a command's arguments, then executes them.
foreach	The *foreach* command iterates a loop over message numbers or filenames you specify in a list, executing a command on each iteration.
function	Creates user-defined functions.
if ... else ... endif	Defines conditional processing in a script or function.

msg_list	Simplifies a message list in metacharacter form into a simple list of numbers.
return	Terminates a script or function and returns control to the calling routine.
shift	Parses the arguments of a user-defined function.
unbutton	Removes user-defined interface buttons.
uncmd	Deletes command abbreviations.
unfunction	Deletes user-defined functions.

UNIX and Z-Mail Control Commands

await	Instructs Z-Mail to stop functioning interactively and check for new mail periodically.
cd	Changes the current working directory.
exit	Terminates Z-Mail without updating any folders. In user-defined scripts, terminates the script.
fullscreen	Changes Z-Mail interface from command-line to fullscreen.
history	Displays most recent commands you have entered.
jobs	Displays a list of suspended outgoing messages.
ls	Lists the files in your current directory.
pipe	Sends the contents of selected messages as input to standard UNIX commands.
printenv	Displays the set of environment strings that Z-Mail passes to other UNIX processes it creates.
pwd	Displays the name of the current working directory.
quit	Terminates Z-Mail, after updating any open folders that have been modified.
resume	Reactivates a suspended message composition.
sh	Executes a command using the shell (/bin/sh).
stop	In the character interface, suspends Z-Mail session. In GUI mode, iconifies Z-Mail session on the system desktop.
x	Terminates your Z-Mail session immediately without updating any folders.

Configuration Commands

bind, bind_macro	Define shorthand keystroke bindings for fullscreen mode.
button	Creates and lists user-defined interface buttons.
cmd	Creates and lists command abbreviations.
filter	Creates and lists message filters.
ignore	Tells Z-Mail which message header fields to hide when it displays messages.
map	Creates or lists macros for the command-line interface.
my_hdr	Creates and lists custom headers that Z-Mail inserts at the beginning of all your outgoing messages.
retain	Specifies those header fields that Z-Mail should show when it displays the contents of messages.
saveopts	Stores a copy of all of the current Z-Mail settings in a file.
set	Sets the values of Z-Mail variables.
setenv	Sets the values of environment strings that Z-Mail passes to other UNIX processes it creates.
source	Reads and executes commands from a script or initialization file.
unfilter	Removes message filters.
unignore	Tells Z-Mail to stop ignoring selected message headers.
un_hdr	Removes custom headers that Z-Mail inserts at the beginning of all your outgoing messages.
unmap	Removes keyboard macros in the command-line interface.
unretain	Stops Z-Mail from displaying header fields when it displays the contents of messages.
unset	Unsets Z-Mail variables and their values.

Command Reference

The remainder of this section lists all of Z-Mail's scripting commands in alphabetical order.

?

The *?* command displays a list of valid Z-Mail commands, or help information about a specific command you specify.

SYNTAX **? [*command_name*]**

If you enter the *?* command with no arguments, Z-Mail displays a list of valid commands. If you specify a command name as an argument, Z-Mail displays help for that command.

about

The *about* command displays information about Dan Heller and Bart Schaefer of Z-Code Software, the creators of Z-Mail.

SYNTAX **about**

alias

The *alias* command allows you to define a name that stands for a list of mail addresses.

SYNTAX **alias [*aliasname*] [*addresslist*]**

RELATED COMMAND **unalias**

If you specify no arguments, *alias* displays all the current aliases. With one argument, it displays the list associated with that name. If you specify more than one argument, *alias* creates a new alias.

EXAMPLES You can use an alias instead of entering one or more user mail addresses when you are sending a message. For example, if you send mail regularly to Kim Fong at Genisoft Corporation, you may want to define the alias *kim* as a shorthand for her e-mail address:

```
alias kim kfong@genisoft.com
```

The sales manager at Genisoft may want to define an alias for all his direct reports:

alias direct kfong rtaylor ithacker vgomez

The system administrator at Genisoft who has already defined aliases called *sales* and *engineering* may want to define an alias for the entire company, including the president *fkramer*:

alias allhands sales engineering fkramer

Use the *unalias* command to delete previously defined aliases.

alternates

Use the *alternates* command to inform Z-Mail of additional login names you have on various host computers.

SYNTAX **alternates [*hostlist*] [*[*username*]] [*address*] [*!path!user*]**

SYNONYM **alts**

RELATED VARIABLES **metoo**

When you have logins on several computers, other users may send messages to all your e-mail addresses hoping to find you the next time you read mail, regardless of what machine you use. If you reply to one of those messages, you may accidently send your reply to yourself at all your various e-mail addresses. To stop this, you can use the *alternates* command. When you specify your alternate e-mail addresses, Z-Mail does not send a copy of the reply to your login name at any of the hosts listed on the alternates list.

If you specify an asterisk (*) as *username,* then Z-Mail attempts to match your login name with all pathnames and local addresses on the specified hosts. If you specify the asterisk followed by a *username*, Z-Mail looks for that name instead of your login name on the remote systems.

If you issue the *alternates* command with no arguments, Z-Mail displays your current set of alternate names. Names you entered in *user@domain* form are displayed in UUCP format.

EXAMPLES If you have another login name on the local or remote machine, you may specify the address either as *user@domain* or by preceding the login name or its UUCP pathname by an exclamation point (!). For example, you can issue the command:

alts hq1 hq2 jhundley@hub.genisoft.com !john

This command looks for the login name *jhundley* on the hosts *hq1* and *hq2*. The last item, !*john*, blocks mail to the e-mail address *john* only if it is destined for the local machine. Root accounts elsewhere are not considered to be equivalent. The address *jhundley@plan.genisoft.com* indicates that at the machine *plan.genisoft.com*, the user *jhundley* is the same person as the current user on the local machine. You could also specify this in UUCP format:

```
alts !hub.genisoft.com!jhundley
```

You must specify the exclamation points (!) in the address to distinguish the name of the domain from the username that follows it. By separating the names, you tell the mail system to route your message through the machine *hub* to the machine *genisoft*, and to look for the login name *jhundley*.

Note: The effect of an *alternates* command is not cumulative; the arguments you specify in the most recent *alternates* command removes your current list of alternate names and replaces it with the new ones you specify.

ask

Use the *ask* command to tell Z-Mail to prompt you with a question and to wait for a reply.

SYNTAX **ask [-i *variable* [-m]] [-d *default*] *question* [*choices...*]**

In GUI mode, *ask* generates a prompt box, with either *Yes/No/Cancel* buttons or a scroll box of choices, depending on the options you specify. In character mode, *ask* displays a prompt and Z-Mail supplies a default choice. You can accept the default by pressing return, or you can backspace over the default and type another answer. To cancel the prompt, you may type the interrupt character (usually **Ctrl-C**) or erase the default response and type the end-of-file character (usually **Ctrl-D**).

If you only specify *question*, *ask* expects a response of *Yes*, *No*, or *Cancel*. *Ask* sets the variable *status* to 0 (zero) if the response is Yes, 1 (one) for No, and –1 for Cancel. You may indicate the default response using the *–d* option.

If you ask for a text string, you can use the *–i* (input) option to name a variable into which the text is returned. In this case, you can specify a list of possible choices, and *ask* displays the list along with the question. If you specify no default using the *–d* option, the first choice in the list is used as the default.

If you specify the *–m* option, the user's response must match one of the listed choices, or Z-Mail repeats the question until the user either gives a valid response or cancels the command. Z-Mail sets the *status* variable to 0 if the user answers the question and to –1 if the user cancels the operation.

The response to the question you ask must be a single argument, so you must enclose your response in quotes if it contains more than one word.

EXAMPLES

To prompt for a text string and assign it to the *folder* variable:

```
ask –i folder "Enter the name of your folder directory"
```

To ask for text input and assign the input to the *realname* variable:

```
ask –i realname "What is your real name?"
```

To ask a question and specify *yes* as the default answer:

```
ask –d y Continue?
```

To specify a list of choices that appear in a list or GUI scroll box:

```
ask –i history "Number of commands to remember" 10 15 20 25
```

await

The *await* command suspends your Z-Mail session and waits for new mail to arrive.

SYNTAX

await [–T *timeout*]

The await command does not return until new mail arrives or until you type a keyboard interrupt (usually **Ctrl-C**). Z-Mail looks for new mail approximately every 30 seconds, or every *timeout* seconds as specified by the *–T* option.

In GUI mode, *await* returns immediately if new mail has arrived or if you did not specify *timeout*. Otherwise it waits *timeout* seconds before returning control to the screen.

bind

The *bind* command identifies a sequence of keystrokes as a shorthand for a Z-Mail function. The key binding you specify works only in fullscreen mode.

SYNTAX **bind [*string* [*function* [*parameters*]]]**

RELATED COMMAND **unbind**

The *bind* command tells Z-Mail to assign the sequence of keystrokes specified by *string* to the fullscreen-mode function called *function*. Try to limit *string* to one or two characters, or a sequence generated by one of the function keys (e.g., **F6**) on your terminal or workstation.

If you specify both *string* and *function*, Z-Mail creates a new key binding so when you type *string*, Z-Mail executes *function*. If you specify no arguments, Z-Mail displays the current set of fullscreen key bindings. If you only specify *string*, Z-Mail displays the binding for just that string.

You can also specify *parameters* that pass arguments to the function indicated in the *bind* command. The parameters are currently recognized only for the special *macro* function; see the *bind-macro* command for details. For a complete list of fullscreen-mode functions, see *Fullscreen Interface Basics* on page 60.

EXAMPLE To change the action of the **r** key so it presents a menu of reply choices normally bound to **Ctrl-R**, enter the command:

```
bind r reply-menu
```

This example does not remove the Ctrl-R key binding. To remove a current key binding, use the *unbind* command.

When you define fullscreen macros, avoid defining **Ctrl-C**, **Ctrl-**, and other keystrokes that generate a signal to the UNIX operating system. If you have problems with a particular key binding, your system administrator may have already reserved that key for another use.

bind-macro

The *bind-macro* command defines keystroke sequences as shortcuts that run Z-Mail commands in the fullscreen interface.

SYNTAX **bind-macro [*string* [*expansion*]]**

If you specify no arguments, *bind-macro* lists the currently defined macros. Otherwise, *bind-macro* functions identically to the *bind* command.

Once you have defined a macro, if you type the key sequence specified by *string*, Z-Mail expands *string* to the full command specified by *expansion*.

Z-Mail provides a special syntax for including non-printing control characters in both the string and the expansion. See the sections *Fullscreen Interface Basics* on page 60 and *Using Macros* on page 3 for details.

EXAMPLE Once you define this macro for the key sequence **@r**:

```
bind-macro @r 'f +record \n'
```

when you type **@** followed by **r**, Z-Mail opens the folder *+record*. The *f* is the *folder* fullscreen mode function, and *\n* stands for newline (as if you pressed the **Return** key).

builtin

The *builtin* command is useful when you have defined a function with the same name as an internal Z-Mail command. By preceding a command name with *builtin*, you tell Z-Mail to execute the internal command, and not the user-defined function with that name.

SYNTAX **builtin *command* [*arguments...*]**

RELATED COMMANDS **cmd, function**

RELATED VARIABLE **recursive**

If you do not specify *builtin* before *command*, Z-Mail assumes you want to execute the user-defined function called *command*.

EXAMPLE

```
cmd write 'echo "Not saving headers." ; builtin write'
```

Note: If you execute a Z-Mail command from within a user-defined function or command alias of the same name, make sure to precede it with *builtin*. If you do not use *builtin*, the function terminates with an error. Another way to avoid this is

to set the *recursive* variable to true; the function then calls itself recursively. This applies only to functions, not to commands.

button

The *button* command allows you to create GUI buttons and fullscreen menu choices. You can use the *button* command to customize your Z-Mail screen, making Z-Mail functions easy to execute.

SYNTAX

button [–n] [*label* [*function*]]

RELATED COMMANDS

unbutton, function

In GUI mode, you use this command to create a user-defined button labeled *label* in the Action area of Z-Mail's Main Window. When you press the button, Z-Mail attempts to execute *function*. The function argument must be a single word, and is typically a user-defined function.

If you specify no arguments, *button* displays all the current buttons and their associated function names in the order in which they appear in the Main Window. If you specify only *label*, Z-Mail displays the function name associated with that button.

When you press a user-defined button, Z-Mail checks that you have selected one or more messages. Z-Mail then passes the numbers of the selected messages to the function as its arguments.

If you specify the *–n* option, users are not required to select messages first, and Z-Mail calls *function* with no arguments.

You can use the *unbutton* command to remove buttons from your Main Window.

The *button* command has no effect in line mode. In fullscreen mode, Z-Mail displays each "button" as a menu entry in the menu generated by the *user-button* keyboard command. For more details, see *Fullscreen Interface Basics* on page 60.

EXAMPLE

The Read button on Z-Mail's Main Window is created with the command:

```
button Read type
```

Note: Although you can operate Z-Mail's regular fullscreen menus by typing single keystrokes, you must move the cursor to the desired button name by repeatedly pressing the space bar. Once you have moved the cursor to user-defined button you want, press **Return**.

cd

The *cd* command changes your current working directory.

SYNTAX **cd [*directory*]**

RELATED COMMAND **pwd**

If you specify *directory*, Z-Mail changes your current working directory to *directory*. If you specify no arguments, Z-Mail changes the current working directory to your home directory.

To display the name of your current working directory, use the *pwd* command.

close

The *close* command closes an open folder.

SYNTAX **close *foldername***

RELATED COMMAND **folder**

The *close* command is equivalent to issuing the command:

folder –d *foldername*

To open folders, use the *folder* command.

cmd

The *cmd* command defines aliases for Z-Mail command names.

SYNTAX **cmd [*name* [*command*]]**

RELATED COMMAND **uncmd**

When you specify *name* and *command*, Z-Mail creates a command-line alias called *name*. When you type *name* at the command line, Z-Mail executes the Z-Mail command called *command*. You should enclose *command* in quotes if it contains semicolons, metacharacters, or pipe symbols (|).

If you specify no arguments, *cmd* lists the current command-line aliases. If you specify only *name*, *cmd* shows the command associated with that alias.

EXAMPLE This command displays the message *Checking for mail...* and waits for mail.

cmd check 'echo Checking for mail...; await'

See the section called *Command Aliases* on page 57 for more information.

copy

The *copy* command functions identically to the *save* command except that it does not mark messages for deletion, regardless of the value of the *keepsave* variable.

SYNTAX **copy [–f] [–s | –a] [*messagelist*] [*filename* / *directory*]**

SYNONYM **co**

RELATED COMMANDS **save, write**

See the *save* command for more information on using the *copy* command.

delete

The *delete* command marks a message or group of messages for deletion.

SYNTAX **delete [*messagelist*]**

SYNONYM **d**

RELATED COMMAND **undelete**

After you delete a message, Z-Mail does not save it when you update your active folder, nor is it available for processing by several other Z-Mail commands.

If you have marked a message for deletion, but have not yet updated your active folder, you can recover the message with the *undelete* command.

detach

The *detach* command disconnects an attachment from a message and stores it in a file you specify.

SYNTAX

detach [–part *num*] [–name *attachname*] [–encode *key*] [–display | –use *type*] [*messagelist*] [*filename*]

detach [–all | –list | –rehash]

RELATED COMMANDS

mail

RELATED VARIABLES

attach_types

If you specify *messagelist*, *detach* uses only the first message in that list. If you do not specify *messagelist*, *detach* uses the current message. If you specify no arguments at all, *detach* lists the attachments of the current message.

You may specify attachments by name using the *–name* option, or by numeric position in the attachment list using the *–part* option. If you specify both an attachment name and a part number, Z-Mail detaches the part numbered *num* and uses *attachname* as the output file name.

You may also specify an output filename as the last argument on the command line. If you specify *filename,* Z-Mail uses *attachname* only to identify the part to detach. If you specify *num*, *attachname* and *filename*, Z-Mail ignores *attachname* and detaches the part numbered *num* and writes it to a file called *filename*.

Z-Mail automatically attempts to decode attachments which are encoded. You may also use the *–encode* option to tell Z-Mail to use a specific decoding key.

You may optionally display or post-process the output file by specifying the *–display* option. The *filetype* of the attachment determines how it is processed or displayed. You may force Z-Mail to display the attachment as a specific *filetype* by specifying the *–use* option. Use caution when you do this since it may have unpredictable results.

To detach all attachments for a specific message, use the *–all* option. To list all the attachments for a particular message, use the *–list* option.

You or your system administrator should define your attachment types and encoding keys in the file called *attach.types* in the Z-Mail library or the file specified in the value of the a*ttach_types* variable.

When you make changes to the *attach_types* file and you want Z-Mail to recognize your changes without starting Z-Mail, enter the command:

detach –rehash

For details on sending attachments, see the *mail* command.

dialog

The *dialog* command causes Z-Mail to display a specific dialog box. This command has effect only in GUI mode.

SYNTAX **dialog *dialogname***

RELATED COMMANDS **button, function**

The *dialog* command displays the dialog box named *dialogname*. This command is useful when you are creating user-defined buttons (see the *button* and *function* commands). Dialog box names are the same as the labels under the dialog icons in the Z-Mail toolbox:

Aliases	Envelope	Options	Sort
Buttons	Folders	Printer	Templates
Compose	Headers	Save	Toolbox
Dates	Help	Search	

display

The *display* command displays messages on your screen.

SYNTAX **display [*messagelist*]**

SYNONYMS **print, p, type, t**

RELATED COMMANDS **ignore, retain, lpr**

RELATED VARIABLES **alwaysignore**

You may optionally specify a *messagelist*; if you do not, the *display* command displays the current message.

By default, the *display* command does not display headers you have asked Z-Mail to ignore. If you specify the first letter of the command as a capital letter (e.g., **Display**, **Print** or **Type**) then Z-Mail displays *ignored* headers provided you have not set the *alwaysignore* variable to ignore all headers. See the *ignore* and *retain* commands for more information about ignoring message headers.

You can use the **+** and the **–** keys to display the next and previous messages respectively. When you press the **+** key, Z-Mail expects that the message is not paged.

each

The *each* command executes a command on a list of messages, one at a time.

SYNTAX **each *messagelist command***

RELATED COMMANDS **cmd, foreach, function**

The *command* must be a single argument; it should be enclosed in quotes if it contains spaces. The *each* command provides each of the messages in *messagelist* to *command* as arguments one at a time, so you should use complex commands containing pipelines or semicolon separators carefully. It is recommended that you use a *cmd* alias or a user-defined function as *command*.

For example, if you enter:

```
each 1–3 10 reply
```

Z-Mail executes the commands:

```
reply 1
reply 2
reply 3
reply 10
```

If any of the commands in the loop returns a nonzero exit status (e.g., sets the *status* variable to –1) the entire loop terminates.

Note: *cmd* aliases and functions may in some cases include commands that return a nonzero status without themselves returning nonzero. As with filters, certain commands cannot be executed in an *each* loop. See the *filter* command for a list of these commands.

echo

The *echo* command causes Z-Mail to re-display the commands you type, expanding all arguments to show their values.

SYNTAX **echo [–n] [–h | –p] *arguments***

RELATED COMMAND **from, error**

RELATED VARIABLES **hdr_format, prompt**

The *echo* command echoes all arguments entered on the command line. It also expands any history references for variables and history.

If you specify the *–n* option, *echo* does not append a newline. If you specify the *–h* option, *echo* looks for message and folder information strings as if you issued the *from* command for the current message. If you specify the *–p* option, *echo* looks for message and folder information strings as if your prompt were changed temporarily. For more information, see the *hdr_format* variable on page 185, *Message Information Strings* on page 155, and *Folder Information Strings* on page 156.

EXAMPLES **echo –h This is from %a and is dated %d.**

might produce:

This is from genisoft!lfong and is dated Dec 14, 5:47pm.

And:

echo –p There are %n new messages to read in %F.

might produce:

There are 5 new messages to read in /usr/spool/mail/mstrohm.

Note: You cannot specify the *–h* and *–p* options together.

edit

The *edit* command allows you to edit messages using a text editor you choose.

SYNTAX **edit [*messagelist*]]**

RELATED VARIABLES **editor**

If you specify no arguments, Z-Mail launches the text editor indicated by the *editor* variable and loads the text of the current

message. If you specify more than one message, Z-Mail allows you to edit one message at a time. As you finish editing each message, Z-Mail asks if you want to edit the next one. If you answer *yes*, Z-Mail loads the message. This process continues until Z-Mail exhausts the list of messages.

When you edit a message, be careful not to remove any message headers, such as *Date* or *From*, or any others that look important. If you remove or change something you shouldn't have, you are notified of the error and Z-Mail does not save the changes you made to the message.

error

The *error* command causes Z-Mail to re-display the commands you type, expanding all arguments to show their values.

SYNTAX **error [–n] [–h | –p] *arguments***

RELATED COMMANDS **from, echo**

RELATED VARIABLES **hdr_format, prompt, status**

The *error* command functions identically to the *echo* command except that in sets the value of the *status* variable to –1. In GUI mode, the *error* command displays an error dialog box.

eval

The *eval* command re-parses its arguments and re-executes the results as a command.

SYNTAX **eval [–h | –p] *argument1 argument2* . . .**

You can use this command to force multiple levels of variable expansion.

If you specify the *–h* option, *eval* expands header format strings in the argument list before executing the command. Similarly, the *–p* option expands prompt format strings in the argument list before executing. Z-Mail expands history and variables prior to these formats and implicitly quotes them so embedded quotes, spaces, tabs, **!** characters, etc. are handled correctly. Header formats are expanded using the current message.

The following example finds all messages from the same author as the current message.

EXAMPLE **eval –h search –f %a**

If you issue the *search* command above without preceding it with *eval -h*, Z-Mail incorrectly searches for all messages from the author named "*%a*". See the entries for *hdr_format* and *prompt* variables for more information.

Note: You may not specify the *–h* and *–p* options together.

exit

If you type *exit* from the command line, you exit Z-Mail immediately without updating any open folders, including the System Folder.

SYNTAX **exit**

SYNONYM **x, xit**

When Z-Mail encounters an *exit* command in a user-defined function or script, it terminates the processing of the function or script. You must use the *x* or *builtin exit* commands to terminate the entire Z-Mail program from within a function. Z-Mail asks you for confirmation if you are currently composing any messages.

expand

The *expand* command allows you to convert an alias name into its original name.

SYNTAX **expand *aliaslist***

This command takes *aliaslist*, the alias name or names you specify as arguments, and changes them to the actual names of the recipients. The *expand* command is useful for identifying circular definitions before you attempt to use an alias name.

filter

The *filter* command allows you to install a filter that Z-Mail invokes whenever new mail arrives in your System Folder.

SYNTAX **filter [–n] [*filtername* [*command* [*search-options*]]]**

RELATED COMMAND **unfilter**

The *filtername* argument identifies a filter by name so you can modify, replace or delete it later.

The *command* argument indicates the command you want Z-Mail to run when it executes *filtername*. The *command* argument must be a single text string; it must contain quote marks if it has any spaces.

You can use any valid option from the *search* command as a *search-option* in the *filter* command.

EXAMPLES

```
filter expire-old-mail delete –ago 3 months
filter save-announce "save +announce" –i –s announce
filter –n meeting "echo You have a meeting" –i –s meeting
```

You may not use certain Z-Mail commands in filters because they can alter the state of the folder in a way that prevents other filters from functioning correctly. These restricted commands include:

await	previous	sort
filter	print	type
next	quit	update

To open a new folder, you can use the *folder* command within a filter. However, Z-Mail adds the new folder without updating the folder that is currently being filtered.

If you use the *–n* option when you define the filter, it is restricted so that it can only apply to newly arrived mail while you are using Z-Mail interactively. You may apply a filter to any folder at the time it is opened by using the *–f* option of the folder command or the *–filter* option on the Z-Mail command line:

```
folder –f +mbox
zmail –filter –folder +mbox
```

You may explicitly prevent a filter from being executed at the time the folder is opened by using the *–F* option of the *folder* command:

```
folder –F foldername
```

To avoid filtering your mail when you first start Z-Mail, use the *–nofilter* option.

```
zmail –nofilter
```

Otherwise, all filters operate whenever a new message arrives in the System Folder. You can only suppress this filtering by removing the filters.

You can remove previously defined filters using the *unfilter* command.

Note: It is never necessary for a filter to save or copy a message to the System Folder. Z-Mail automatically appends new messages to your System Folder unless they are marked for deletion by a filter or by you.

flags

The *flags* command modifies the flag bits set for selected messages.

SYNTAX **flags [[+ | –] [D f N ! O P p R r S U]] [*messagelist*]**

Z-Mail provides this command so you can to use *cmd* to alter the way certain conditions are recorded. You may set or modify multiple flag bits at one time.

The modifiable flag bits are:

D	Deleted
f	Forwarded
N	New
!	Newly arrived
O	Old
P	Preserved
p	Printed
R	Read
r	Replied-to
S	Saved
U	Unread

If you include a plus sign (**+**) or minus sign (**–**) in the list of bits, the specified bits are turned on or off respectively, without modifying other bits. If you do not specify a plus or minus sign, the list of bits is set explicitly and all other bits are lost.

You can pipe message lists to or from the *flags* command.

folder

The *folder* command changes the active or current folder.

SYNTAX **folder [–a] [–d] [–f | –-F] [–l] [–N] [– n] [–r] [%[*user*] | & | #[*n*] | file**

SYNONYM **fo**

RELATED COMMAND **close**

If you do not include any arguments when your issue a *folder* command, Z-Mail displays the name of the active folder. Valid arguments are:

–a	Add the named folder to those already opened.
–d	Close the named folder (delete from the open list).
–f	Perform filtering when opening the folder.
–F	Do not perform filtering when opening the folder.
–l	List the folders that are open.
–N	Display no headers upon entering new folder.
–n	Do not update the current folder.
–r	Set read-only mode so you can't alter the new folder.
% [*user*]	Change to /usr/spool/mail/[*user*] (default is yours).
&	Change folder to $mbox (Default is **~/mbox**).
#	Switch back to the most recently active folder.
#*n*	Switch to folder number *n* (*n* is an integer).

File names that do not begin with **/** are interpreted relative to the current directory unless they begin with one of the meta-characters **+** or **~**. As in *csh*, the character **~** expands to your home directory (or to some other user's home directory if used as *~username*). The character **+** expands to the name of your folder directory (defaults to ~/**Mail** or the value of the variable *folder*). For compatibility with other mailers, no **/** character needs to appear between the **+** and the name of the folder.

This command can only appear in a pipeline if it is the first command; it returns a list of all the messages in the new folder. You cannot use this command in initialization files before the shell has started.

folders

The *folders* command lists the names of the mail folders in your folder directory.

SYNTAX **folders**

Your folder directory is usually the directory *Mail* in your home directory. However, you can set the *folder* variable to specify another folder directory.

foreach

The *foreach* command iterates a loop over message numbers or filenames you specify in a list, executing a command on each iteration.

SYNTAX

foreach *msgnum messagelist command*
foreach *filename* (*filelist*) *command*

RELATED COMMANDS

cmd, each

If you specify parentheses around the list, Z-Mail assumes it is a list of filenames. If you specify no parentheses, Z-Mail assumes the list is message numbers.

The *command* must be a single argument; it should be enclosed in quotes if it contains spaces.

The arguments *msgnum* and *filename* are strings that are used as variable names in the processing of the *foreach* command. See the use of the *num* argument in the example below.

If any of the commands in the loop returns a nonzero exit status (e.g., sets the *status* variable to –1) the entire loop terminates.

EXAMPLE

For example, to reply to messages 1, 2, 3 and 10, and automatically include their text, enter the command:

```
foreach num 1–3 10 'reply $num –i'
```

Z-Mail then executes the commands:

```
reply 1 –i
reply 2 –i
reply 3 –i
reply 10 –i
```

Note: *cmd* aliases and functions may in some cases include commands that return a nonzero status without themselves returning nonzero. As with filters, certain commands cannot be executed in a *foreach* loop. See the *filter* command for a list.

from

The *from* command displays information taken from the headers of your message list.

SYNTAX **from [+ | –] [*messagelist*] [*pattern*]**

SYNONYM **f**

If you do not specify any arguments, the *from* command displays the current message's header summary. For more information, see the *hdr_format* variable.

If you specify only *messagelist* or if you pipe a list of messages, the *from* command displays the headers of the indicated messages.

If you specify a pattern, the *from* command displays the headers of those messages whose *From:* lines match *pattern*. If you specify both a pattern and a message list (or if you supply a message list via a pipeline), Z-Mail searches for *pattern* in only the messages indicated in the message list.

You can specify the special arguments – and **+** to move the current message pointer to the previous or next message, respectively, while also displaying that message's header. If you specify a message list in addition to – or **+**, then the current message pointer is set to the first or last message in the message list you specify.

EXAMPLES

from + Kim

displays the headers of all messages that contain "Kim" in the author's name and sets the message pointer to the last message that meets the criteria specified.

from – 10–30 {16}

displays the header summaries of messages 10 through 30 except for message 16 and set the current message pointer to 10.

from +

displays the header summary of the message after the current message and increment the current message pointer to the that message.

from $

displays the last message's header summary but does not move the current message pointer.

fullscreen

The *fullscreen* command causes Z-Mail's character interface to change from line-oriented mode to screen-oriented mode.

SYNTAX

fullscreen [off]

You may not use this command when fullscreen mode is already active. When you use the fullscreen command in an initialization file (e.g., *.zmailrc*) this command is the same as giving the *–fullscreen* switch *(–V)* on the Z-Mail command line.

You may only use the *off* argument in initialization files, including those read with *–init (–I)*, and has the effect of turning off the *–fullscreen* switch. Z-Mail ignores the *off* argument at all other times, even in files read using the *–source* option *(–F)*.

function

The *function* command creates user-defined functions.

SYNTAX

function [*name*]
function *name*(){
***commands* ...**
}

RELATED COMMAND

unfunction

If you specify the function command without any arguments, it lists the text of all defined functions. If you specify *name*, the *function* command lists the text of that function.

To define a new function, you must specify a name followed by a set of empty parentheses with no spaces between them, and the last argument must be an open brace ({). Z-Mail reads succeeding lines until it encounters a close brace (}) at the beginning of a line. Z-Mail stores the text enclosed in the braces as the function body, and executes the text when the function name is given as a command. Braces used for negation in message lists do not inadvertently end the function. For more details, see *Specifying Message Lists* on page 10.

You may use indentation and comments (beginning with **#**) just as you do in initialization files and scripts. When you load function text, Z-Mail converts leading tab characters to 8 spaces, but does not change tabs specified elsewhere. This allows you to preserve sensible indentation when Z-Mail writes function text back to a file after a *saveopts* command.

When you use a user-defined function and you specify arguments on the command line, the arguments are available in the function text via a set of special variables. These variables are numbered 1 through the total number of arguments passed, and can be referenced as *$1, $2,* .... The complete argument list can also be referenced as *$**.

EXAMPLE

```
function saveit() {
    if ! $?_filename
        if ! $?mbox
        set mbox = ~/mbox
        endif
    set _filename = $mbox
    endif
    ask –i _filename "Save: $*\nFilename:" $_filename
    if $status == 0
        save $* $_filename
    endif
}
```

The *saveit* function defined above is an example of an interactive user-defined function. The purpose of this function is to prompt the user for a file name to which messages are saved. The *saveit* function remembers the filename you specify so all the filenames you select during your current session appear in the selection list the next time you call the function.

The *saveit* function first checks whether a file name has been defined. If not, it makes sure that the *mbox* variable has been initialized, then sets *_filename* to be the same as *mbox*. Next, it uses the *ask* command to prompt for a filename, providing the name stored in *_filename* as the default response. The function stores the actual response in the *_filename* variable. The question passed to ask shows the list of messages ($*), followed by a newline (represented by \n), and finally prompts with the string "Filename:". Finally, if *ask* returns successfully, the *save* command saves the argument messages to the file.

headers

The headers command displays a screenful of message headers listed in the active folder.

SYNTAX **headers [[-H] [:*c*]] [+ | –]**

SYNONYM **h, z**

If you specify a message number as an argument, the first message for which Z-Mail displays headers it that message number.

You may set the variable *screen* to tell Z-Mail how many headers to display at a time. In GUI mode, the variable *screen_win* contains the number of headers used in the headers subwindow. A typical header may look like:

```
> 5 +Nr dsmith@justin.case.com Feb 19, (10/278) Sales forecast
```

The **>** indicates that the current message pointer is on this message and the **5** indicates that it is message number 5. The next three positions indicate the message status. The first position is only used if the message has been marked or has been tagged with a priority. In the sample line above, the **+** character shows that this message has been marked. See the *mark* command for details on marking and message priorities.

The second position can contain any of the following status indicators:

!	Newly arrived and unread
N	New and unread
U	Old, but still unread
*	Deleted
S	Saved
P	Preserved
(blank)	Already read

The third position may have an *r* if the message has been replied to, an *f* if the message has been forwarded, or a *p* if it has been printed to hardcopy.

The author of the example message header is *dsmith@justin.case.com*, the date is *Feb 19*, the number of lines in the message is *10*, the number of characters is *278* and the subject of the message is *Sales Forecast*. The format of the message header is described by the string variable *hdr_format*. The example given above has a *hdr_format* of:

```
set hdr_format = "%25f %7d (%l/%c) %25s"
```

For more information on specifying *hdr_format* strings, see *Message Information Strings* on page 155.

Header Modifiers

You can display a special subset of message headers by using the –H:*c* option, where *c* is one of:

a	All messages
d	Deleted messages
f	Forwarded messages
m	Marked messages
n	New messages
o	Old messages
p	Preserved messages
r	Replied to messages
s	Saved messages
u	Unread messages

Note that the –H is not required; headers :c is also valid.

Other options to the *headers* command include **+** and –. These characters display the next or previous screenful of message header summaries. The **z** and **z**– synonyms are identical to the **h +** and **h –** commands and provide compatibility with Berkeley *BSD Mail* and *Mush*.

The output of the *headers* command is the message numbers of all the messages it displays, so you may pipe its output to other commands. If you pipe output into the *headers* command, it displays the message headers affected by the previous command. This action is identical to piping to the *from* command, except that the **+** and **–** modifiers do not reset the current message.

help

The help command provides usage information and hints on a variety of Z-Mail topics.

SYNTAX

help [*topic*]

If you want general help, typing **help** provides some general information as to how to get further help and a list of topics suggested for more specific help. To get help on how to use the help facility, type:

help –-?

You can add the *–?* argument to any command to get help on that command. For example, *save –?* displays information about using the *save* command.

If Z-Mail can't find the text for the help file, it displays an error message. You can tell Z-Mail the location of your Z-Mail help files by setting the variables *cmd_help* and *gui_help* to the paths of the files. The *gui_help* variable is recognized by Z-Mail versions using the X Window System, even if GUI mode is not active.

history

The *history* command displays the command history in chronological order.

SYNTAX **history [–h] [–r] [*numlines*]**

RELATED VARIABLE **history**

Z-Mail displays earlier commands first followed by more recent commands. You can suppress the displaying of history event numbers using the *–h* option. If you specify the *–r* option, Z-Mail displays history events in reverse order.

If you specify *numlines*, then Z-Mail displays that number of previous commands rather than the number set by the *history* variable. For more information, see the *history* variable on page 186 and the section called *Command History* on page 56.

iconify

The *iconify* command transforms your Z-Mail GUI session into a Z-Mail icon on your desktop.

SYNTAX **iconify**

When you issue the iconify command, the status of your Z-Mail session is preserved. All Z-Mail windows and dialog boxes disappear from the screen. To resume your Z-Mail session, double click on the Z-Mail icon.

if...else...endif

You can use the *if* command to test for specific conditions before executing another set of commands. You cannot use *if* commands from the Z-Mail command line; you can only use them in scripts and functions.

An *if* command tests an *expression* of the form:

argument1 operator argument2

You must separate all the arguments of an *if* command, including the conditional operator, from each other with blank spaces.

The indentation used in presenting the following syntax and examples is for clarity only; it does not affect the processing or meaning of the commands.

SYNTAX

All conditional statements begin with an *if* expression and end with *endif*. The general form of an *if* command is:

```
if expression
    actions
else
    alternate actions
endif
```

The *else* clause and its associated *alternative actions* are optional. Therefore, the simplest form of an if expression is:

```
if expression
    actions
endif
```

You can specify more than one action per *if* expression, so you can create conditional statements such as:

```
if expression
    action1
    action2
    action3
endif
```

You could also perform multiple actions using this syntax:

```
if expression
    action1; action 2; action 3
endif
```

To create complex conditional processing, you can nest *if* statements. The syntax of a more complex conditional statement is:

```
if expression1
    if expression2
        if expression3
            action
        endif
    endif
endif
```

In the above example, Z-Mail performs *action* when all three expressions evaluate to true.

Specifying Conditional Expressions

When you specify conditional expressions, follow these rules:

- You may specify one conditional expression for each *if* line.
- You must separate arguments in expressions with blank spaces, including comparison operators (e.g. **==**, **>=**, etc.)
- You must have an *endif* statement for every *if* statement.
- You must place the *if* statement on a line by itself.
- You may not use parentheses in conditional expressions.
- You may compare strings to strings, numbers to numbers, but not strings to numbers.
- You must prefix a variable name with a **$** to reference the variable's value. Otherwise, Z-Mail treats the variable name as a character string.

Else Clauses

You use the else clause to tell Z-Mail to perform an alternative action when the conditional expression in your *if* command evaluates to false. You must specify the *else* keyword on a line by itself with the alternative action commands following it.

EXAMPLE

The following statement tells Z-Mail to sign you outgoing message with the contents of the *.signature* file when Z-Mail is in GUI mode and to use the *.alternate* file in character mode.

```
if is_gui
        set autosign = ~/.signature
else
        set autosign2 = ~/.alternate
endif
```

Comparison Operators

When you construct conditional expressions, you may use any of the following operators. Make sure you separate the operators from other arguments by blank spaces.

Operator	*Effect*	*Example*
==	Equivalent	if "$printer" == lpr
!=	Not equivalent	if "$screen" != "$crt"
<	Less than	if "$crt" < 20
<=	Less than or equal	if "$history" <=5
>	Greater than	if "$screen" > 10
>=	Greater than or equal	if "$screen" > 20
=~	Equivalent pattern	if "$TERM" =~ "hp*"
!~	Not equivalent pattern	if "$rep" !~ "report[2,3]"

You may not use **&&** (logical *and*) or **||** (logical *or*) operators to evaluate more than one expression.

Conditional Arguments

The operators in a conditional expression compare two *arguments* that can be:

- Character strings (e.g., **report** or **"sales department"**),
- Numbers, or
- Variable names preceded by a **$** character (e.g., **"$screen**").

Comparison arguments may not contain:

- Numeric expressions (e.g., 4+5), or
- Parentheses (except within quoted strings).

Variables

When you use a variable name as an argument in a conditional expression, you must precede it with a **$** character or Z-Mail interprets the variable name as a text string. If the value of a variable could contain a blank, enclose the variable name in quotes so Z-Mail treats the value as one argument.

EXAMPLE

In the following example, even if the *report* variable evaluates to a multi-word phrase such as *Income Statement* or *Balance Sheet*, the *if* statement still evaluates correctly because the string is enclosed in quotes:

```
if "$report" == "Balance Sheet"
    return
endif
```

Notes: When Z-Mail expands an undefined variable, it evaluates as an empty (null) string. Variables that evaluate to an empty string can cause errors. To test whether a variable is undefined, do not compare it to an empty string. Instead, use the syntax **$?*varname***. This expression evaluates to 1 if *varname* is defined and 0 (zero) if it is not defined.

EXAMPLE

The following *if* command tests if the *printer* variable is defined. If *printer* is defined, the command prints your currently selected messages.

```
if $?printer
    print $*
    return
endif
```

Patterns

You can also use special pattern characters in character strings as the arguments of your conditional expressions. When you specify these arguments, you must use the **=~** operator that compares the two arguments for an equivalent pattern.

Character	*Matches*	*Example*
?	Any single character	**"report3"** matches **"report?"**
*	Any string	**"Prod"** matches **"Pr*"**
[*a,b,c...*]	Any character in list	**"Prod"** matches **"[P,p]rod"**
		"rep2a" matches **"rep[2,3]a"**

EXAMPLES If you set the values of the three variables listed below as indicated, the following table shows how Z-Mail evaluates conditional expressions.

```
set report = 3
set rname = "Product2"
set TERM = "ncd16e"
```

Command	*Evaluates to*
if "$report" == 3	True
if "report" == 3	False (no **$** prefix)
if "$rname" == "Product2"	True
if "$TERM" != 'ncd16e'	False (arguments are **==**)
if "$TERM" =~ "ncd*"	True
if "$TERM" =~ "ncd16?"	True
if "$rname" == "product"	False (lower-case p)
if "$rname" =~ "[P,p]roduct?"	True
if "$rname" == "[P,p]roduct?"	False (needs **=~**)
if "$rname" =~ "[P,p]roduct[2,3]"	True
if "$TERM" =~ ncd1[3,5,6]?"	True

File and Folder Operators

When you reference folders or files, you can test to ensure they exist and contain information before you use them. You use the *–e* option to test for the existence of a folder or file, and the *–z* option to test if an existing folder is of zero length (is empty).

Operator	*Effect*	*Example*
–e	File exists	**if –e $logfile**
–z	File is of zero length	**if –z salesforecast**

EXAMPLES The following *if* command tests for the existence of a folder specified by the *$salesforecast* variable and opens that folder if it does exist.

```
if –e $salesforecast
    folder $salesforecast
endif
```

The next example tests to see if the folder called *tempfile* has any messages in it. If it does, Z-Mail saves your currently selected messages to *tempfile*.

```
if –z tempfile
    save $* tempfile
endif
```

ignore

The *ignore* command allows you to suppress specific headers when you display messages.

SYNTAX **ignore [*headerlist*]**

RELATED COMMANDS **unignore, retain**

When you read messages, all message headers are displayed with the text body of the message. In many cases, these headers are cumbersome and uninteresting. You can filter out unwanted headers by using the *ignore* command. For example,

ignore Received Message-Id Date

ignores the three specified fields. Additional *ignore* commands are cumulative; if you specify additional ignore commands, the fields you specify are added to the fields already ignored. Use the *unignore* is command to reverse the effects of *ignore*.

To specify headers you wish to see, rather than ignore, use the *retain* command.

jobs

The *jobs* command display a list of suspended compositions — composed messages currently in progress.

SYNTAX **jobs**

Z-Mail identifies each suspended composition with a number. The most recently suspended composition is also marked with a **+** sign. By default, this is the composition that is continued by the *resume* command. The **+** sign is followed by the first part of the *To:* line, and then the first part of the *Subject:* line. If the *To:* line is truncated, it ends in ellipses (**...**).

EXAMPLE A typical job listing might look like:

```
[1] + To: kfong, rteller Benchmark results
[2] + To: jgomez         Thanks for the extra effort
[3] + To: managers       June reports are due Friday
```

Note: Suspending compositions is not true job control in the sense used by some shells, because there is no UNIX process associated with a suspended composition. If you are editing headers, the job listing may not accurately reflect the state of the *To:* and *Subject:* lines in the editor file.

lpr

The *lpr* command takes messages from a message list and sends them to the printer, one by one, each separated by page feeds.

SYNTAX **lpr [–h] [–n] [–P *name*] [*messagelist*]**

RELATED COMMANDS **ignore, retain**

RELATED VARIABLES **alwaysignore, print_cmd**

Use the *–h* option to suppress the printing of ignored headers (see the *ignore* and *retain* commands and the variable *alwaysignore*). Specify the *–n* option to suppress all headers. A default printer name is supplied if you do not specify one using the -P*name* option. If you set the *printer* variable, the *lpr* command uses the value of *printer* as the printer name.

If you set the variable *print_cmd* to a value, Z-Mail uses the command described by that value instead of the default system command for printing. In such cases, Z-Mail ignores the *–P* option and the *printer* variable and the command is simply executed as is.

Z-Mail marks each message it prints with a *p* status.

Note: If the *lpr* command produces an error message, your system administrator may have configured Z-Mail to require *–d* in place of the *–P* option.

ls

The *ls* command duplicates the UNIX command */bin/ls* that lists files.

SYNTAX **ls [*flags*]**

By default, the *ls* command always uses the *–C* flag which displays filenames in column format.

mail

The *mail* command sends messages to a list of users. If you do not specify a list of recipients on the command line, Z-Mail requests a list from you by displaying a *To:* prompt.

SYNTAX

mail [*options*] [*recipients ...*]

SYNTAX EXAMPLE

mail *username*, /path/to/*filename*, | *prog_name*, +*folder_name*, ~user/mbox

SYNONYM

m

RELATED COMMAND

reply

You must supply a list of recipients at some point before you send the message. Z-Mail allows you to specify files, mail folders and programs as recipients. If you send a message to a file, the filename must be a full pathname; filenames must start with a slash (/) so Z-Mail can differentiate filenames from user names. The special characters + and ~ may precede pathnames; + expands to the user's folder directory as indicated by the *folder* variable, and ~ expands to the user's home directory. To send a message to a program, precede the program name with the pipe character (|). Since Z-Mail searches the user's path, full pathnames are not required for programs that lie in the user's *PATH* environment variable.

You can specify the following options in the *mail* command:

–A [*type*:]*file*	Attach *file* of indicated *type*
–b *addrlist*	Set list of blind carbon copy recipients.
–c *addrlist*	Set list of carbon copy recipients.
–E	Edit outgoing message headers.
–e	Immediately enter the editor described by the *visual* variable.
–F	Add a random fortune to the end of message.
–f [*msglist*]	Forward messages (not indented).
–H *file*	Read file as prepared text (no headers).
–h *file*	Read file as a draft (text and headers).
–I [*msglist*]	Include messages with headers (indented).
–i [*msglist*]	Include messages in letter (indented).
–p *file*	Read file as a prepared message template.
–s [*subject*]	Prompt for subject.
–U	Send draft unedited (use only with *–h* or *–H*).
–u	Unsigned; no signatures or fortunes added.
–v	Verbose (passed to mail delivery program).

Note: You may not be able to specify the verbose option (–v) depending on the mail transport agent on your system.

The *–E* option causes Z-Mail to place the headers of the outgoing message in the editor file so you can changed them. See the description of the *edit_hdrs* variable for details.

Indenting Included Text

The *–i* option allows you to include the current message in the body of the message you are about to send. By default, the included message is indented by the string **>**. You may change the indent string by setting the *indent_str* variable. If you specify a message list after the *–i* option, then the messages described by that list are included. The *–I* option is identical to the *–i* option except that the headers of the message are also included.

Setting Subject

The *–s* option looks at the next argument and sets the message subject to that string. If the string contains spaces, enclose the entire subject in quotes. If there is no following argument, then Z-Mail prompts you for the subject. This option is useful if:

- You do not have the *ask* variable set, or
- You wish to change the subject used with reply.

Forwarding Messages

Use the *–f* option only for forwarding messages. By default, the *–f* option allows you to edit the forwarded message, by default. In this case, Z-Mail creates a new set of headers for you and places the forwarded message in the body of your outgoing message indicating it as forwarded:

```
--- Forwarded mail from dsmith@justin.case.com

    ( forwarded message here )

--- End of forwarded message
```

If you specify the *–U* (unedited) option in conjunction with the *–f* option, Z-Mail uses the headers of the forwarded message and adds special headers:

```
Resent-From:
Resent-To:
```

and perhaps others, depending on your mail transport agent. *Sendmail*, for example, may add a number of other *Resent* headers.

You can specify an optional message list just as you can with the *–i* option. The subject of the message (if available) is the same as the current message; it is not necessarily the subject of the message being forwarded. The subject of forwarded mail

cannot be changed. Z-Mail does not prompt you for the subject and it is ignored completely if you specify the *–f* option. However, using the *–e* option, you can change the subject once you are in editing mode either by using the escape sequence **~s** or by editing the *Subject:* header.

Overriding Signatures

The *–u* option, meaning *unsigned*, prevents you from appending signatures and fortunes to messages. It overrides the *autosign* and *fortune* variables, and overrides the *–F* option if you specify both *–u* and *–F* on the command line.

Message Templates and Draft Files

The *–h* option indicates that the file you specify is a *message template* — a previously prepared message such as a message, possibly a partial one saved with **~w**. This file is called a *draft*. You must specify the *file* argument and all message headers must either be present or you must add them manually. At a minimum, there must be a *To:* header; Z-Mail automatically adds *From:* and *Date:* headers if needed before sending a message.

If you specify the *–p* option and indicate a filename, Z-Mail starts composing a new message and loads the text from the file so you can edit it.

Importing Text Files

To read a prepared text file that does not contain headers, use the *–H* option. If you also specify the *–U* option, Z-Mail sends the message immediately without further editing. A *message template* is simply a text file that you use for repetitive tasks, such as a phone message.

Attachments

If you specify the *–A* option, Z-Mail appends the file to the message as an attachment. Attachments may be either text or binary, but binary files must be encoded to 7-bit ASCII before they are sent.

map

The *map* command creates or lists macros for the command-line interface, and the *map!* command creates or lists macros for compose mode.

SYNTAX **map[!] [*string* [*expansion*]]**

RELATED S **unmap**

In either line or compose mode, macros act in such a way that, when string is typed, the effect is the same as if you typed *expansion*. The string is usually one or two control characters, or

a sequence sent by one of the function keys of a particular terminal. See the section called *Using Macros* on page 3 for the syntax you use to specify *string* and *expansion*, and for comments on the interactions of macros in the same and in different modes.

If you do not specify any arguments, these commands display the list of macros and expansions in the appropriate mode. If you only specify a string, they display the expansion associated with that string for the appropriate mode. Otherwise, they create a macro, associating the expansion you indicate with the specified string.

You unset line mode macros with the *unmap* command, and you unset composition mode macros with the *unmap!* command.

mark

The *mark* command assigns a priority to messages that you can refer to later when sorting or managing your mail.

SYNTAX **mark [*–priority*] [*messagelist*]** *where priority is A, B, C, D or E.*

RELATED COMMAND **unmark, headers**

EXAMPLE When you specify no *priority*, Z-Mail marks the messages in *messagelist* with a **+** character, indicating a temporary mark that is not saved when you close the current folder or quit Z-Mail.

When you specify a priority such as *–A*, Z-Mail assigns that priority to the messages in *messagelist*. For example, the command **mark –C 7** assigns priority C to message 7.

You can use the *sort* and *search* commands to order or select messages based on their priority. If you mark a message with both a temporary **+** mark and a letter priority, Z-Mail displays only the **+** mark and it takes precedence over the letter priority when you are sorting and searching.

Use the *unmark* command to remove temporary marks only. To remove the alphabetic priority (A-E) of a message, give the command:

mark –

Unless you clear the priority for a message in this way, its priority is saved when Z-Mail updates the active folder.

You can enter the command **headers –H:m** to generate a list of messages that are marked.

merge

The *merge* command allows you to read messages from a specified folder into the active folder.

SYNTAX **merge [–N] *foldername***

RELATED COMMAND **folder**

The merge command copies messages from the folder named *foldername* into the active folder. The header summaries of the merged messages are displayed unless you specify the *–N* option. See the *folder* command for details.

The *merge* command can only appear in a pipeline if it is the first command; it returns a list of all the messages selected from *foldername*. You cannot use the *merge* command in initialization files before Z-Mail has started and opened folders.

WARNING: The merge operation does not currently support combining two open folders. If the folder you want to merge from is one of the open folders, you must manually update it before performing the merge. If you do not do this, the current state of the merged-from folder is not reflected in the merged-in messages.

msg_list

The *msg_list* command parses a message list you specify and set the *output* variable to that message list.

SYNTAX **msg_list [+ | –] [*messagelist*]**

RELATED **output**

This *msg_list* command is a parsing aid that accepts a message list as an argument. The *messagelist* may include metacharacters and negation syntax.

If *messagelist* is syntactically correct, *msg_list* sets the *status* variable to 0 (zero) and sets the *output* variable to the resulting message list, using only ranges and actual numbers, and not other metacharacters. If the list is syntactically correct but produces an empty result, the *output* variable is an empty string. Finally, if the list is not syntactically correct, Z-Mail sets the *status* variable to –1.

As with the *from* command, the arguments – or **+** cause *msg_list* to reset the current message to the first or last message in the

list. Unlike *from*, however, the *msg_list* command never displays any text as output.

my_hdr

The *my_hdr* command lets you create personalized headers in your outgoing messages.

SYNTAX

my_hdr [*header* [: *text*]]

RELATED COMMANDS

un_hdr

COMMAND USAGE

my_hdr	Displays all your headers
my_hdr *header*	Displays value of *header*
my_hdr *header: string*	Sets *header* to *string*
un_hdr *header*:	Unsets *header*

EXAMPLES

my_hdr Phone-Number: 415-555-1234

To set a header, you must specify the first argument as a string containing no spaces or tabs and it must end with a colon (**:**). The rest of the command line is the text to be associated with the header you specify. If you use any quotes in the header and the header itself is not set in quotes, you should precede the quotes with a backslash (\). You should also precede pipe characters, semicolons, or any other metacharacter that Z-Mail might interpret as a command line modifier with a backslash.

If you set the *no_hdrs* variable, Z-Mail does not add your headers to outgoing messages, and unsets no headers.

Z-Mail treats the header *From:* as a special case. If you have set your own *From:* header, Z-Mail performs a simple test to determine whether the address you specify is valid. Any UUCP or domain address that directs mail to your login at the local machine should be acceptable, but certain system configurations may prevent some combinations from being recognized. If the address is valid, your *From:* header is used; otherwise, Z-Mail generates another address known to be valid and uses it instead. Some mail transport agents are "picky" and do not allow Z-Mail to supply a *From:* header; in these cases, Z-Mail removes your *From:* header as you send messages, and replaces it with one generated by the transport agent.

Note: You cannot set the Date: header. If you attempt to do so, Z-Mail does not generate an error message, but overrides your date with the correct system date and time corresponding to when you send your message.

next

The *next* command displays the next new or unread message, beginning with the current message.

SYNTAX **next**

SYNONYM **n**

If no messages are new or unread, the *next* command displays the next message that has not been deleted or saved, beginning with the message after the current message. If no messages match these criteria, *next* displays: *No more messages.*

pick

The *pick* command is an alternate name for the *search* command and is identical to it in all respects.

SYNONYM **search**

pipe

The *pipe* command allows you to direct a list of messages to a UNIX command as input.

SYNTAX **pipe [–p *pattern*] [*messagelist*] *command***

RELATED VARIABLES **alwaysignore**

You may explicitly specify the list of messages or it may come from a Z-Mail *pipeline*. See *Piping Commands* on page 57 for details. If you do not specify or pipe a *messagelist*, Z-Mail uses the current message. All headers are considered part of the message text for purposes of this command unless the value of the *alwaysignore* variable includes the word *pipe*. For example,

EXAMPLE The following example sends the text and headers of messages 3, 5 and 7 to the UNIX *patch* utility:

```
each 3 5 7 patch
```

The example below sends messages 11 to 15 to */bin/sh* to run as shell scripts:

```
each 11–15 pipe
```

If you specify a pattern with the *–p* option, Z-Mail searches each message for a line beginning with that string. Z-Mail directs the matching line and all succeeding lines to *command*. If

you do not specify *–p* and you omit *command*, Z-Mail searches for a line beginning with **#!** and feeds that line and all succeeding lines to */bin/sh*. In other words, a *pipe* command with no arguments treats the current message as a shell script.

An alternative form for specify *pattern* is:

/ pattern1 / , / pattern2 /

in which case displaying begins with the line containing *pattern1* and ends with the line containing *pattern2*, inclusively. Patterns of this form must still match the beginning of a line, and regular expressions are not currently allowed.

You can also invoke the *pipe* command as *Pipe* (note capitalization), in which case only the body of the messages, and none of the message headers, are sent to *command*.

When you set Z-Mail's *unix* variable, UNIX commands can appear anywhere in a Z-Mail pipeline except as the first command in a pipeline without explicitly using *pipe*. However, you must still specify *Pipe* to exclude all headers.

Note: All messages you list as arguments to *pipe* or *Pipe* are sent to the standard input of the same process as a continuous stream, in message number order! This may not be desirable when you extract shell scripts, so be careful. To avoid this behavior, use the *each* or *foreach* commands to pipe messages individually.

preserve

The *preserve* command saves a message list in your System Folder rather than your Main Mailbox unless you explicitly delete it.

SYNTAX **preserve**

SYNONYM **pre**

RELATED VARIABLES **hold, keepsave**

The *hold* variable causes all messages to be held in your spool directory automatically. The *keepsave* variable stops the *save* and *write* commands from marking messages for deletion.

previous

The *previous* command displays the previous new or unread message.

SYNTAX **previous**

If no messages are new or unread, the *previous* command displays the previous message that has not been deleted or saved, beginning with the message before the current message. If no messages match these criteria, the *previous* command displays: *No more messages.*

print

This command is an alternate name for the *display* command and is identical to it in all respects. To print a hard copy messages to a line printer, see the *lpr* command.

SYNONYM **display, type, p, t**

pwd

The *pwd* command displays the name of the current working directory.

SYNTAX **pwd**

RELATED VARIABLE **cwd**

This command also resets the variable *cwd*.

quit

The *quit* command updates all open folders and exits Z-Mail.

SYNTAX **quit**

SYNONYM **q**

RELATED COMMANDS **update, folder**

RELATED VARIABLES **verify, hold**

By default, all messages in your System Folder that are not marked for deletion remain in your System Folder. Z-Mail discards any messages you have marked for deletion (which by default includes saved messages). If you unset the *hold* vari-

able, Z-Mail moves messages you have read from your System Folder to your Main Mailbox. Your Main Mailbox is either *~/mbox*, or the file described by the *mbox* variable.

When updating a folder other than the System Folder, Z-Mail discards messages you have marked for deletion, and leaves all other messages including saved messages in that folder. See the *folder* and *update* commands for more details.

If you try to quit while you still have unfinished messages still being composed, Z-Mail prompts for confirmation that it should abort the compositions before quitting. For more information, see the **~z** escape command for composing messages and the *resume* command.

reply

The *reply* command sends a message to the sender of each message in a messagelist.

SYNTAX **reply [*messagelist*] [–r *path*] [*options*] [*users*]**

SYNONYMS **r, replysender**

RELATED COMMANDS **replyall, mail**

RELATED VARIABLES **reply_to_hdr, auto_route, domain_route**

The *reply* command accepts all options supported by the *mail* command. See the *mail* command for details.

The *–r* option prefixes the address of each recipient in the address list with the indicated path. This overrides the value of the *auto_route* variable, but has the exact same effect.

When you construct a return mail address to the author of a message, the *reply* command searches for special mail headers in the author's message that indicate the most efficient mail path for return mail. By default, Z-Mail searches for the *Reply-To:*, *From:*, and *Return-Path:* headers, in that order. If Z-Mail finds none of these fields in the message, it parses the first line of the message to determine the return path.

If you want Z-Mail to search the headers in a different order, set the *reply_to_hdr* variable to that list of headers in the order you wish them to be searched. If you set *reply_to_hdr* on, but specify no string, Z-Mail searches the first *From* line it encounters regardless of what headers the author's message contains. You may specify the *From* line explicitly as an item in the list of

reply-to headers by specifying the header. See the *Variables* section for more information about *reply_to_hdr*.

replyall

The *replyall* command works identically to the *reply* command, except that it sends a message to the sender *and all recipients* of each message in a messagelist.

SYNTAX **replyall [*messagelist*] [-r *path*] [*options*] [*users*]**

SYNONYM **R**

RELATED COMMANDS **reply**

RELATED VARIABLES **reply_to_hdr, auto_route, domain_route**

See the *reply* command for details on using *replyall*.

When you reply to all recipients of a message using the *replyall* command, Z-Mail determines the original author's address from the message headers. There is no way to determine the best path to the other recipients of the message from message headers aside from taking their addresses directly from the *To:* and *Cc:* lines.

EXAMPLE **replyall 3, 4, 5 –i 4, 5, 7 –s "Meeting Agenda" Research**

In the above example, you reply to messages 3, 4 and 5 and include the text from messages 4, 5 and 7 in the body of the reply. Z-Mail sets the subject to *Meeting Agenda*, and automatically sends the reply to all the authors and recipients of messages 3, 4 and 5, as well as to addresses contained in the alias *Research*.

resume

SYNTAX **resume [*job*]**

RELATED COMMAND **jobs**

The *resume* command allows you to continue composing a message that you suspended with the **~z** escape command. The optional *job* argument is an integer identifying the message you wish to resume. You can obtain a list of the current messages being composed by using the *jobs* command. If you do not specify a job, the most recently suspended composition is resumed.

This command is not valid in the GUI mode because compositions have their own input windows and do not need to be suspended.

Note: To maintain consistency with *csh*, the *fg* command is equivalent to *resume*.

retain

The *retain* command tells Z-Mail which headers to show when displaying messages.

SYNTAX **retain *headerlist***

RELATED COMMAND **ignore**

Enter the names of the headers you want Z-Mail to display in *headerlist*.

EXAMPLE For example, to display only the headers *From:, Date:, Subject:, To:* and *Cc:*

```
retain from date subject to cc
```

return

Use the *return* expression to tell Z-Mail to terminate the current script or function and return control to the script or function that called the script. Z-Mail recognizes *return* commands only within scripts and functions; it does not recognize *return* commands at the Z-Mail command-line prompt.

You can specify an optional status code that sets the value of the *status* variable so the calling script can determine how the script terminated.

SYNTAX **return [*status_code*]**

EXAMPLE This following commands display a message, then they terminate processing and return control to the calling routine with a status code of –1.

```
if $report_no <= 0
    echo Report request canceled by user.
    return –1
endif
```

save

The *save* command saves messages to a file or folder.

SYNTAX **save [–f] [–s | –a] [*messagelist*] [*filename / directory*]**

SYNONYMS **s**

RELATED COMMANDS **copy, write**

RELATED VARIABLES **mbox, keepsave, deletesave**

If you do not specify any arguments, the *save* command writes the current message to the file *mbox* in your home directory or to the file specified by the *mbox* variable. If you specify *messagelist*, Z-Mail saves the messages specified in the list.

If you specify *filename*, that filename is used instead of *mbox*. If the file exists and you are allowed to write to it, Z-Mail appends each message in *messagelist* to the end of the file.

The *–s* option tells Z-Mail to use the subject of the message as *filename*. Similarly, the *–a* option tells Z-Mail to use the author of the message as *filename*. If you are saving more than one message at a time, Z-Mail uses the subject or author name from the message with the lowest message number. Using the *–s* and *–a* options, you may specify a directory name other than the current directory where the files are to be created.

If you specify the *–f* option, the file is overwritten and its current contents are lost. Note that the *–f* option has no effect when used in conjunction with the *–a* or *–s* options. For compatibility with older versions of Z-Mail, the exclamation character **!** may be substituted for *–f.* Do not precede the **!** with a dash.

If the current folder is the System Folder, Z-Mail marks saved messages for deletion. If the *keepsave* variable is set or the current folder is not the System Folder, then messages are not marked for deletion.

The *write* command differs from the *save* and *copy* commands in that the message headers are not saved in the file along with the body of text. The *copy* command is like save except that messages are not marked for deletion, regardless of the value of the *keepsave* variable.

saveopts

The *saveopts* command saves the settings of all variables, aliases, *cmds* and functions in the Z-Mail initialization file. For more information on initialization files, see *Initializing Z-Mail* on page 37.

SYNTAX **saveopts [*filename*]**

If you specify filename, Z-Mail writes a new initialization file to that filename.

WARNING: This overwrites the existing file named *filename.*

The *saveopts* command is intended for novice users who almost always use Z-Mail in the same mode, and who therefore do not wish to learn the Z-Script scripting syntax. Experienced users are encouraged to edit their existing initialization file directly, preserving any changes that they may have previously made manually.

The *saveopts* command cannot override some settings that Z-Mail reads from the *system.zmailrc* file. To change these system settings, you must edit your *.zmailrc* file manually or start Z-Mail with the command **zmail –noinit** to prevent it from reading the *system.zmailrc* file.

search

The *search* command allows you to select messages that match criteria you specify.

SYNTAX **search [*options*] [*pattern*]**

SYNONYM **pick**

The *pattern* argument is a *regular expression* as described by the UNIX *ed* command. You can search for messages from a user, for a particular subject line, between certain dates, and limit searches to a range of messages. You can also find all messages that do not match the same arguments mentioned above.

The search command recognizes the following options:

+*n*	Keep only the first *n* messages matched.
–*n*	Keep only the last *n* messages matched.
–ago *format*	Search for messages relative to today's date.
–d –*date*	Search for messages sent before *date*.
–d +*date*	Search for messages sent after *date.*

–e	Interpret all remaining arguments as *pattern*.
–f	Search for *pattern* in *From:* field only.
–h *header*	Search for *pattern* in specified *header* only.
–i	Ignore case of letters when searching.
–p *priority*	Select messages with the specified *priority* (A-E)
–r *messagelist*	Search only the messages in *messagelist*.
–s	Search for *pattern* in *Subject:* field only.
–t	Search for *pattern* in *To:* field only.
–x	Select messages not matching *pattern*.

You can abbreviate the *–ago* option as *–a*. You can only specify one of the *–a, –d, –f, –h, –p, –s* and *–t* options at a time, but you may specify multiple *–p* options to select several priorities.

Z-Mail searches all messages including those marked for deletion. Z-Mail scans entire messages, for *pattern* unless you specify the *–a, –d, –f, –h, –p, –s* or *–t* options. The *–x* option may not be used in conjunction with the *+n* and *–n* options.

For the *–d* option, *date* is of the form:

month/day/year

You can put a dash (–) prefix on the date you specify to find messages older than that date. If you omit fields in *date*, the fields default to today's values.

EXAMPLES

search –d 4/20	On April 20, this year
search –d –/2/90	On or before the 2nd, this month, 1990
search –d +5/4	On or after May 4 of this year
search –d /	Today only

At least one **/** char must be used in *date*. There is no strong date checking. For example, 2/30 is accepted as a valid date.

The format for the *–ago* option is very simple. Specify the number of days followed by the word *days*, or the number of weeks followed by the word *weeks*, and so on with *months* and *years*. You are allowed to truncate these options since Z-Mail examines only the first character in each keyword. Therefore, all of the following expressions are equivalent:

```
search –ago 1 day, 2 weeks
search –ago 2 Weeks 1Day
search –ago 2 w,1day
search –a 2w1d
```

These examples find all messages that are exactly 2 weeks and 1 day old. All *–ago* dates collapse into day time segments. This means that months are 30.5 days long. If you require more precise date selection, use the *–d* option and indicate specific dates.

Also note that the *–ago* option allows before (–) and after (+) arguments. As a result, you may select all messages older than 1 week with:

```
search –ago –1 week
```

The following command finds all the messages between the dates February 5, 1990, and February 5, 1991, that contain the subject *meeting minutes* and send them to the printer.

```
search –d –2/5/91 | search –d +2/5/90 | search –-s
    "meeting minutes" | lpr
```

The next command deletes messages that have "Re:" in the Subject header.

```
search –s Re: | delete
```

The next command finds all messages from *frank* in the folder named *+project*.

```
folder +project | search –f frank
```

The following command searches all messages that have the header *Return-Path:* and determines if the string "hqvax" is in the header. Note that case sensitivity applies only to the pattern searched, not the header itself.

```
search –h return-path hqvax
```

The next command finds all messages that are a week or less old and saves them in the file called *current*, which is found in the folder indicated by the *folder* variable.

```
search –ago +1w | save +current
```

The following finds the first three messages containing the string "expo".

```
search  +3 expo
```

The next command finds the next message with the same subject as the current message.

```
eval –h "search +2 –r –$ –s %s" | search –1
```

set

The *set* command assigns values to Z-Mail variables.

SYNTAX

set *variable1 variable2 . . .*	FOR BOOLEAN VARIABLES
set *variable* = *value*	FOR SINGLE-VALUE VARIABLES
set *variable* = "*value1 value2*"	FOR MULTI-VALUE VARIABLES
set *?variable*	FOR HELP ABOUT VARIABLES

RELATED COMMANDS

unset

If you do not specify any arguments, the *set* command displays the values of all variables. If you specify *variable*, the *set* command assigns a value to the named variable.

EXAMPLES

When you are setting the value of a boolean variable (turning it "on"), you should not specify an equal sign.

```
set hold
```

You may set more than one boolean variable in a set command by listing the names of all the variables on one line:

```
set hold auto_route fignore
```

You set single-value variables by entering the variable's name on the left side of the equal sign and its value on the right. If a string value contains blanks, make sure you enclose it in quotes:

```
set crt = 20
set editor = vi
set autosign = signature.file
set realname = "Wolfgang Mozart"
```

You set multivalued variables in the same way as single-valued variables. You must enclose the list of values in quotes if you use blanks to separate the values:

```
set autodismiss = "compose print folder buttons"
```

The special command **set ?all** displays all variables internal to Z-Mail and a brief description of what they do. The command **set ?variable_name** displays the same information for one variable instead of all variables.

The *unset* command clears the values of Z-Mail variables. You may unset all variables by issuing the command **unset** *.

You can create your own variables for use in scripts. In the following example, you set a new variable called *reports* to a list of messages returned by another command by using the piping mechanism.

```
search –s Status Reports | set reports
```

The *reports* variable now contains a message list which can be used as the message list argument to any command which accepts a list. For example:

mail –i $reports rhagan

This command sends mail to *rhagan* and includes the text of all the messages indicated by the value of the *reports* variable.

See the section called *Z-Mail Variables* on page 149 for details about internal Z-Mail variables.

sh

The *sh* command invokes an interactive version of the shell.

SYNTAX **sh [*command*]**

The shell that you spawn is described by the *shell* variable. If you specify the optional argument *command*, then the command is executed under the Bourne Shell (*/bin/sh*). If the special character **&** is at the end of any shell command, the command is executed in background.

shift

The *shift* command parses the arguments of a user-defined function.

SYNTAX **shift [–m] [*count*]**

RELATED VARIABLE **output**

When you write a Z-Mail function that requires arguments, you use the *shift* command to parse and identify the individual arguments.

If you specify the numeric argument *count*, the *shift* command shifts that number of parameters. If you do not specify *count*, the *shift* command removes a single argument from the beginning of the argument list.

The *–m* option tells Z-Mail to remove any numeric message list that precedes the current argument list. You can set a variable to this message list for future reference by setting the variable equal *$output* (the value of the *output* variable), or by piping the output of the shift command to the set command.

EXAMPLE 1 The following function called *strip* demonstrates the effect of the *shift* command. This sample script uses the argument variables *$1*, *$2*, *$3*, etc., to refer to the arguments a user enters when they call the *strip* function.

```
function strip () {
        echo "The list of arguments is:" $*.
        echo "The first argument is" $1.
        shift 2
        echo "After shifting, the first argument is" $1.
}
```

If you enter this command:

```
strip aa bb cc dd ee
```

Z-Mail runs the *strip* function, passing it the arguments, and displays:

```
The list of arguments is: aa bb cc dd ee.
The first argument is aa.
After shifting, the first argument is cc.
```

EXAMPLE 2 The next example demonstrates how you can use the *shift* command to strip off any message numbers type at the beginning of an argument list. Essentially, the *shift* command looks for numeric values at the beginning of the argument list and evaluates the numeric values as a message list.

```
function getmsg () {
        echo "The list of arguments is:" $*.
        shift –m | set mlist
        echo "After shifting, the message list is" $mlist.
        echo "After shifting, the argument list is" $*.
}
```

If you enter the command:

```
getmsg 3 4 5 8 xx yy zz
```

Z-Mail runs the *getmsg* function, passing it the argument list, and displays:

```
The list of arguments is: 3 4 5 8 xx yy zz.
After shifting, the message list is: 3-5 8.
After shifting, the argument list is: xx yy zz.
```

EXAMPLE 3 The next example shows how you can use the value of the *output* variable to set the message list:

```
function getalt () {
        echo "The list of arguments is:" $*.
        shift –m
```

```
        set mlist = $output
        echo "After shifting, the message list is:" $mlist.
        echo "After shifting, the argument list is" $*.
}
```

If you enter the command:

```
getalt 7 4 1-3 taxicab
```

Z-Mail runs the *getalt* function, passing it the argument list, and displays:

```
The list of arguments is: 7 4 1-3 taxicab.
After shifting, the message list is: 1-4 7.
After shifting, the argument list is: taxicab.
```

sort

The *sort* command orders message by author, date, status, priority or subject.

SYNTAX **sort [–i] [[–r] –a | –d | –l | –p | –R | –s | –S]**

The sort command recognizes the following options:

–i	Ignore case in alphabetical sorts.
–r	Reverse sort order of next option.
–a	Sort by author alphabetically.
–d	Sort by date.
–l	Sort by length of message.
–p	Sort by message priority.
–R	Sort by subject including *Re:*
–s	Sort by subject alphabetically.
–S	Sort by message status (the default).

If you specify no arguments, the *sort* command orders messages by status. The following table sizes the order in which Z-Mail sorts messages by status.

Indicator	***Status***
N	New
U	Unread
P	Preserved
r	Replied-to
p	Printed
+	Marked
blank	Read
*	Deleted

If messages have the same status, the *sort* command then orders by priority.

If you sort by priority, Z-Mail orders marked messages first, followed by messages of priority A, then B, C, D, E, and finally messages having neither a mark nor a priority. See the *mark* command for information on attaching marks and priorities to messages.

If you specify only the *–r* option, the *sort* command orders by inverse message status.

If you specify the *–d* option, by default Z-Mail orders messages by the date they were sent by the original author. If you set the boolean variable *date_received*, Z-Mail orders messages by the date you received them.

EXAMPLES

If you specify more than one sort option, the *sort* command applies them in left-to-right order. Thus:

```
sort –a –d
```

sorts messages by author and, if the author of any set of messages is the same, sorts within that set by date.

The *–r* option applies to only the next option encountered. Therefore, to sort by both reverse author and reverse date requires:

```
sort –r –a –r –d
```

You can also group options in the *sort* command:

```
sort –ra –rd
sort –rard
```

You can also sort a consecutive sublist of messages by using pipes. You may specify the exact message list by specifying that message list on the command line:

```
10–20 | sort –d
```

sorts messages 10 to 20 by date.

If you set the *sort* variable, the *sort* command orders messages each time Z-Mail activates your System Folder as the current folder.

Note for fullscreen mode users: To specify nested sorting using the fullscreen interface, you must use the temporary line-mode escape key — the colon (**:**) — and the command must be issued at the **:** prompt, as if you are giving an *ex* command to *vi*. When the command is finished, you see a **...continue...** prompt and you may continue or return to the home screen of the fullscreen mode.

Note for previous Z-Mail and Mush users: For compatibility with previous versions of Z-Mail and Mush, sort options are

recognized even if they are not preceded by a dash (–). Also, if you specify a dash separated by spaces from any following arguments, Z-Mail interprets it as *–r*.

source

The *source* command reads Z-Mail commands from a file.

SYNTAX **source [*filename*]**

RELATED **home**

If you do not specify *filename*, Z-Mail searches for a file called *.zmailrc* in the your home directory. If you set either of the environment variables *ZMAILRC* or *MAILRC*, then the file named by the variable is sourced instead. If you specify a filename on the command line, Z-Mail sources that file. For more information, see the section called *Initializing Z-Mail* on page 37 and the *home* variable.

stop

The *stop* command causes Z-Mail to send a SIGTSTP to itself on UNIX systems with job control.

SYNTAX **stop**

RELATED COMMAND **iconify**

The *stop* command facilitates the stop-job action from a complex command line alias rather than requiring users to type stop characters explicitly.

In the GUI mode, stop closes the Z-Mail windows to icon form. This is equivalent to the *iconify* command.

top

The *top* command displays the top few lines of messages.

SYNTAX **top [*messagelist*]**

If you specify *messagelist*, the top command displays the top few lines of each message in *messagelist*. Otherwise, *top* displays the first few lines of the current message.

You can specify the number of lines by setting the *toplines* variable. The value of *toplines* defaults to the value of the *crt* variable crt. The *top* command is ignored in GUI mode.

type

The *type* command is an alternate name for the *display* command and is identical to it in all respects.

SYNONYM **display, print, t**

unalias

The *unalias* command deletes previously defined aliases. Use the *alias* command to define a new alias.

SYNTAX **unalias [*aliasname* | ∗]**

RELATED COMMAND **alias**

You may delete all previously defined aliases with:

unalias ∗

unbind

The *unbind* command deletes previously defined key bindings for the fullscreen interface. Use the *bind* command to define new bindings.

SYNTAX **unbind *string***

RELATED COMMAND **bind**

The backslash character is a literal-next character; it tells Z-Mail to accept the key binding that follows as a string instead of treating it as a macro and trying to expand it.

unbutton

The *unbutton* command deletes previously defined GUI buttons and character-mode menu functions. Use the *button* command to define new buttons.

SYNTAX **unbutton [*label* | ∗]**

RELATED COMMAND **button**

You may delete all previously defined buttons with:

unbutton *

uncmd

The *uncmd* command deletes command aliases previously defined by the *cmd* command.

SYNTAX **uncmd *name***

RELATED COMMAND **cmd**

You may delete all previously defined command aliases with:

uncmd *

undelete

The *undelete* command reverses the effect of a *delete* command.

SYNTAX **undelete [*messagelist* | *]**

RELATED COMMANDS **delete, quit, update**

RELATED **deletesave**

The *undelete* command removes the asterisk (*) status indicator for all messages in *messagelist*.

Note: You can only use *undelete* to recover messages that are marked for deletion, but have not actually been deleted by an *update* or *quit* command. Once you update a mail folder, all messages marked for deletion are lost permanently.

undigest

The *undigest* command bursts multi-message digests into individual messages.

SYNTAX **undigest [–m] [–p *pattern*] [*messagelist*] [*filename*]**

A *digest* is a mail message which is a collection of other mail messages mailed to a *moderator* by other users. The moderator compiles all the messages into a folder and sends the result to all the subscribers of the mailing list.

If you specify the *–m* option, the *undigest* command merges the new messages into your active folder.

If you specify a filename, the *undigest* command creates a new folder named *filename* and stores the new messages in that folder.

If you specify the *–p* option, you can indicate an alternate pattern that Z-Mail should use as a separator when breaking the digest into individual messages. Z-Mail matches *pattern* against character strings at the beginning of each line of the digest. The default pattern is eight hyphens (--------). The hyphen pattern is the de facto USENET standard digest article separator and should work for most digests, but some use another separator. The *–p* option is also useful for bursting forwarded messages out of a wrapper message.

EXAMPLE The following command bursts out messages forwarded by another user of Z-Mail and merges them into the current folder.

```
undigest –m –p "—— Forward"
```

If you specify *messagelist*, each digest in the list is disassembled to the same filename you specify. If you do not specify *filename* or the *–m* (merge) option, then the *undigest* command creates a temporary file. The name of the temporary file is generated from the subject of the digest.

unfilter

The *unfilter* command deletes filters previously defined by the *filter* command.

SYNTAX **unfilter [*name* | *]**

RELATED COMMAND **filter**

You may delete all previously defined filters with:

```
unfilter *
```

unfunction

The *unfunction* command deletes functions previously defined by the *function* command.

SYNTAX **unfunction [*name* | *]**

RELATED COMMAND **function**

You may delete all previously defined functions with:

```
unfunction *
```

un_hdr

The *un_hdr* command stops Z-Mail from including personalized headers created by the *my_hdr* command in your outgoing mail messages.

SYNTAX **un_hdr [*header* | *]**

RELATED COMMAND **my_hdr**

RELATED **no_hdrs**

The *un_hdr* command deletes the definition of the specified header. To remove the definitions for all personalized headers, issue the command:

un_hdr *

If you set the *no_hdrs* variable, Z-Mail does not add your personal headers to outgoing messages. The *no_hdrs* variable does not delete the definitions of personal headers; it merely stops Z-Mail from using them.

To create your own personal headers, see the *my_hdr* command.

unignore

The *unignore* command cancels the suppression of specific headers when Z-Mail displays messages.

SYNTAX **unignore [*header* | *]**

RELATED COMMAND **ignore, headers**

You may tell Z-Mail to display all previously ignored headers by entering:

unignore *

unmap

The *unmap* command deletes previously defined keyboard macros used in Z-Mail's character interface.

SYNTAX **unmap [*macroname* | *]**

RELATED COMMAND **map**

You may tell Z-Mail to delete all previously ignored macros by entering:

unmap *

unmark

The *unmark* command clears temporary marks you placed on messages using the *mark* command.

SYNTAX **unmark [*messagelist*]**

RELATED COMMAND **mark**

A temporary mark appears as a **+** character immediately preceding the message status. Use the *unmark* command to remove these temporary marks.

You may tell Z-Mail to clear all the temporary marks in your active folder by entering:

unmark *

Z-Mail also allows you to assign a permanent priority to a message; this priority is indicated by *A, B, C, D or E*. For information on removing these priority marks, see the *mark* command on page 123.

unset

The *unset* command clears the values of all variables previously defined using the *set* command.

SYNTAX **unset [*varname* | *]**

RELATED COMMAND **set**

You may tell Z-Mail to delete all previously defined variables by entering:

unset *

Some variables are essential to the operation of Z-Mail and it restores them to default values even after you explicitly unset them.

update

The *update* command updates your current folder.

SYNTAX **update [–r]**

RELATED COMMANDS **quit, folder**

The *update* command writes the changed contents of your active folder to disk just as if you had quit, except that it does not change the New (*N*) status of messages to Unread (*U*). Z-Mail does not list headers when the folder is re-activated.

The *–r* option puts the folder into read-only mode after updating it.

See the *folder* command for complete information.

version

The *version* command displays the version number of Z-Mail program you are running.

SYNTAX **version**

write

The *write* command saves the body text from messages to a file or folder, ignoring message headers.

SYNTAX **write [–f] [–s | –a] [*messagelist*] [*filename* / *directory*]**

SYNONYMS **w**

RELATED COMMANDS **save, copy**

The *write* command functions identically to the *save* command except that it does not save message headers in the specified file called *filename* along with the body of text for each of the messages in *messagelist*. See the *save* command for details on using the *write* command.

CHAPTER 6

Z-Mail Variables

As you use Z-Mail and Z-Script, you use *variables* to configure your Z-Mail installation, create scripts and functions, and store strings and numeric values you want to refer to later.

You can use the variables supplied automatically by Z-Mail as well as create variables of your own. Z-Mail provides variables that you use to:

- Tell Z-Mail how you want it to work,
- Define the way Z-Mail works with mail transport agents and other products,
- Tell Z-Mail where to look for data and setup files,
- Provide data about your Z-Mail session, and
- Help you write Z-Mail scripts and functions.

You can also define your own variables that you use in scripts and functions, or to store a value that you want to refer to later.

As you become familiar with Z-Mail, you'll begin using variables to make Z-Mail work exactly the way you want. You can set and manipulate variables using any of Z-Mail's interfaces.

If you are a system administrator, you should familiarize yourself with Z-Mail's extensive list of variables. By setting Z-Mail's variables appropriately, you can configure Z-Mail to meet the exact needs of your organization.

Types of Variables

Z-Mail supports a variety of variables including system, internal, function and user-defined variables.

System Z-Mail provides many *system variables* whose settings you can modify to make Z-Mail function exactly the way you want. The settings of these variables are stored in Z-Mail initialization files.

Internal Some of Z-Mail's system variables are read-only *internal variables* whose values can only be set by Z-Mail. If you try to set the values of these variables, Z-Mail will either warn you that you can't or will reset them to their internal values.

Function Z-Mail provides a special set of internal variables called *function variables* that you use in functions to reference the arguments you specified when you called the functions.

User-Defined You can create your own variables by using the *set* command in scripts, functions or at Z-Mail's command prompt. You can name variables almost anything you want, except for the names of existing Z-Mail system variables. For details on defining your own variables, see the *set* command on page 136.

Types of Variable Values

There are several types of values you can assign to Z-Mail variables:

boolean	Set or unset (*i.e.*, *true* or *false*)
string	Set to a specific character string
numeric	Set to a specific numeric value
multivalued	Set to a list of values

Some of your Z-Mail variables may have attributes that are boolean, string, numeric or multivalued at the same time.

Setting Variables

You can set the values of variables and save their settings using any of Z-Mail's interfaces.

GUI To set variables in GUI mode, select the *Variables* option from the *Options* pull-down menu, and the *Variables* dialog box appears. You can click on a variable name in the scroll box on the left side of the dialog box and Z-Mail displays a description of that variable. Use the buttons and fields displayed next to the scroll box to set the variable to the value you desire.

You may then set the value of additional variables repeating the procedure above. When you are finished setting variables, press the *Save* button to save their settings to a Z-Mail configuration file. If you wish to give the configuration file a name other than *.zmailrc,* be sure to type the full path name in the text box at the top of the *Variables* dialog box.

Fullscreen To set variables in fullscreen mode, you must issue the *variables* fullscreen command which by default is bound to the **v** key. From Z-Mail's main window displaying message summaries, type **v**, and Z-Mail displays:

```
Variables:  set  unset  all  help
```

Type **s** to select the *set* option and Z-Mail displays the prompt:

```
set:
```

Then follow the rules for setting variables in command-line mode as specified below.

Command-Line To set boolean variables to *true*, type **set** followed by the name of the variable. To set boolean variables to *false*, type **unset** followed by the variable name. You can **set** or **unset** more than one variable at a time.

```
set hold
unset nosave deletesave
```

To set the name of a numeric variable, type the name of the variable you wish to set followed by an equal sign, and then the new value for the variable. For example, to set the number of message summaries Z-Mail displays in fullscreen mode to 17, type:

```
crt_win = 17
```

To set the value of a string variable, type the name of the variable followed by an equal sign and then the string. If the string contains blank spaces, make sure the string is enclosed in quotes.

```
set editor = vi
set realname = "Kurt Wilhelm"
```

To set the value of a multivalued variable, type the name of the variable followed by a quoted text string in which you separate the various values with spaces.

```
set quiet = "complete autosign newmail"
```

For more information about setting variables, see the*set* command on page 94.

Note for system administrators: Z-Mail system variables have default values as specified in the file */usr/lib/Zmail/variables*. You can alter the default values for your installation by editing this file or by running initialization files at the startup of Z-Mail to change the defaults to values you desire.

Storing Variable Settings

Z-Mail stores variable settings in initialization files that it reads every time you start Z-Mail.

Your system administrator may set variables to configure your organization's Z-Mail installation to work in a specific manner. The variable settings implemented by your system administrator are stored in a system initialization file.

You can override your organization's variable settings by specifying new values for Z-Mail variables and storing them in your personal initialization file. By default, Z-Mail looks for a file called *.zmailrc* in your home directory and overrides your installation's system defaults with settings it finds in that file.

If you want to configure Z-Mail to work different ways for various groups, you can create several Z-Mail initialization files. For details on using initialization files to configure your Z-Mail installation, see the sections called *Initializing Z-Mail* beginning on page 37 and *Z-Mail Options Summary* beginning on page 19.

Using Variables

When you are writing scripts or using the command-line interface, you can expand a variable name to its value by preceding the name with a **$** character. For example, to set the value of *progname* equal to the value of *editor*, you could enter:

```
set progname = $editor
```

To create a new variable named *formfile* whose value is the pathname of a file called *formlist* in your *home* directory, you can enter:

```
set formfile = $home/formlist
```

Technical Note: If while using Z-Mail, you reference a variable that is not explicitly set, Z-Mail checks your UNIX environment variables for default values. For example is you use *$editor* in a script and the Z-Mail editor variable is undefined, Z-Mail uses the value of the UNIX environment variable named *EDITOR* if it has one.

Variable Modifiers

You can affect the expansion of a variable by putting one of these suffixes called *modifiers* at the end of the variable's name.

:t If you use the :t modifier, Z-Mail treats the variable as a file path name, and returns the name of the file (the tail of the path) as the value of the variable. In other words, Z-Mail returns everything to the right of the last slash (/).

:h If you use the *:h* modifier, Z-Mail treats the variable as a path name, and returns the *head* of the pathname as the value of the variable. In other words, Z-Mail returns everything up to, but not including, the last slash (/).

:l If you use the *:l* modifier, Z-Mail converts all alphabetic characters in the variable's value to lower case.

:u If you use the *:u* modifier, Z-Mail converts all alphabetic characters in the variable's value to upper case.

:num If you use the *:num* modifier, Z-Mail converts the value of the variable to a list of words separated by spaces, numbered from one (1), with the word that is indexed by number selected and returned. The *:num* can be greater than the actual number of words, and in this case, Z-Mail returns an empty string as the variable's value.

Note: Z-Mail allows only one colon (**:**) modifier for each **$** expansion.

EXAMPLES

The following table gives examples of how variable modifiers affect how Z-Mail expands variable references.

If:Equals:	***Then:***	***Evaluates to:***
home/home/dsmith	$home:t	dsmith
home/home/dsmith	$home:h	/home
realnameDerek Taylor	$realname:l	derek taylor
realnameDerek Taylor	$realname:u	DEREK TAYLOR
realnameDerek Taylor	$realname:2	Taylor
realnameDerek Taylor	$realname:3	*<empty string>*

Message Information Variables

When Z-Mail encounters a % character inside square brackets, preceded by a **$** character, it examines the character string that follows as a *message information variable*. The values of message variables depend on the characteristics of the current message.

SYNTAX

$[%*x*] where *%x* is a valid message information string below

When Z-Mail encounters this syntax, it evaluates the string inside the brackets and substitutes the result in the current command.

Message Information Strings

When Z-Mail encounters a % character inside square brackets, it makes the following substitutions:

%a	address of the author
%c	number of characters (bytes) in the message
%d	date and time of the message
%f	entire *From:* field (author)
%i	the message-id (may not be present)
%l	number of lines in the message
%m	month number from the message date
%M	month name from the message date
%n	name of the author
%N	day number of the month of the message
%p	priority of the message
%s	subject of the message
%t	*To:* field (recipients)
%T	time only of the message
%u	user name (login) of the author
%W	day of the week (Sun, Mon, etc.)
%y	last two digits of the year of the date
%Y	year of the message
%Z	time zone of the message
\n	a newline
\t	a tab

You may optionally specify a field width for each information string. For example, **%20f** displays the first 20 characters of the *From:* line. You may also use colon variable modifiers in conjunction with this bracket notation.

EXAMPLE

The following command saves the current message in a file whose name is the first eight characters of the name of the message's author:

```
save $[%8n]
```

The next command sets a user-defined variable called *_subj* equal to the first twenty letters of the subject of the current message, converted to upper case:

```
set _subj = "$[%20s]:u"
```

Note: Evaluating many folder variables in a single command is inefficient. To expand several variables, use the *–h* and *–p* options in the *echo* or *eval* commands which also provide better quoting of the interpolated strings. For more details see the *echo* command on page 101 and the *eval* command on page 102.

Folder Information Variables

When Z-Mail encounters a % character inside parentheses, preceded by a **$** character, it examines the character string that follows as a *folder information variable*. The values of folder variables depend on the characteristics of the current folder.

SYNTAX

$(%*s*) where *%s* is a valid prompt specification

When Z-Mail encounters this syntax in a command, it evaluates the string inside the brackets and substitutes the result.

Folder Information Strings

When Z-Mail encounters a % character inside parentheses, it makes the following substitutions:

%d	number of deleted messages
%F	full path name of the current folder
%f	name of the current folder (tail of %F)
%M	current month
%m	current message number
%N	today's date (Number of the day in the month)
%n	number of *new* messages
%T	current time (hours and minutes)
%t	total number of messages
%u	number of unread messages
%W	weekday name (Sun, Mon, Tue)
%Y	this year
%y	last two digits of %Y
\n	a newline
\t	a tab

You may specify a field width for each variable. For example, **%3M** displays the first three characters of the current month. You may also use colon variable modifiers in conjunction with this parentheses notation.

EXAMPLES

The following command displays the current time:

```
echo $(%T)
```

The following command sets the user-defined variable *_folder* to the first eight characters of the current folder in upper case:

```
set _folder = "$(%8f):u"
```

Note: Evaluation of many folder variables in a single command is inefficient. To expand several variables, use the *–h* and *–p* options in the *echo* or *eval* commands which also provide better quoting of the interpolated strings. For more details see the *echo* command on page 101 and the *eval* command on page 102.

Variable Summary

This section summarizes Z-Mail's variables into functional categories. The categories appear in the following order:

Configuration Variables
- Help Variables
- Header Variables
- Display Variables
- Folder Variables
- Directory and File Variables
- Print Variables
- Editor Variables

Addressing and Sending Variables
- Compose Variables
- Addressing, Routing and Transport Variables
- Signature Variables

Interface Variables
- GUI Variables
- Fullscreen and Command-Line Variables

Programming Variables

UNIX System Variables

For details on individual Z-Mail variables, see the section called *Variable Reference* beginning on page 23.

Help Variables

cmd_help Z-Mail looks for the file indicated by the *cmd_help* variable to find the general help file used by the Z-Mail *help* command.

complete Set the *complete* variable to enable filename completion.

fignore Set the *fignore* variable to have Z-Mail to ignore a list of filename extensions when it performs filename completion.

fullscreen_help When you set the *fullscreen_help* variable to a string of commands separated by spaces, Z-Mail lists those commands at the bottom of the screen all the time.

gui_help Z-Mail looks for the file indicated by the *gui_help* variable to find the help file used by the graphical user interface versions of Z-Mail.

quiet The *quiet* variable tells Z-Mail to suppress warnings and user messages under specific circumstances.

verify The *verify* variable causes Z-Mail to request confirmation of certain actions.

version Z-Mail stores its current version number in the *version* variable.

warning If you set the warning variable, Z-Mail displays warning messages under a variety of situations.

Header Variables

alwaysignore Use the *alwaysignore* variable to tell Z-Mail to ignore specific message headers.

hdr_format The *hdr_format* variable allows you to control the format of the header information Z-Mail displays for each message.

hdrs_only Z-Mail sets the value of the *hdrs_only* variable to true if you specified the *–headers* option in the *zmail* command that started your Z-Mail session.

in_reply_to Set the *in_reply_to* variable to a string that completes the *In-Reply-To:* header.

no_expand If you set the *no_expand* variable, Z-Mail does not expand aliases, so message headers contain aliases exactly as you typed them.

no_hdrs If you set the *no_hdrs* variable, Z-Mail does not include your personalized mail headers in messages.

reply_to_hdr If you set the *reply_to_hdr* variable and provide a list of headers (delimited by spaces or commas), then Z-Mail searches that list for return paths.

Display Variables

autoprint If you set the *autoprint* variable, after you delete a message, Z-Mail automatically displays the next message.

date_received If you set the *date_received* variable, Z-Mail displays the date and time you receive messages rather than the default of when they were sent by the author.

mil_time Set *mil_time* on to display time of day in 24-hour format (hh:mm) instead of 12-hour format (hh:mm am/pm).

show_deleted If you set the *show_deleted* variable on, Z-Mail lets you display deleted messages in command-line mode using any command that displays a message.

squeeze If you set the *squeeze* variable, whenever you read, pipe or save messages, Z-Mail squeezes all consecutive blank lines into one blank line.

toplines The *toplines* variable tells Z-Mail the number of message lines to display when you issue the *top* command in character mode.

wrap If you set the *wrap* variable, when you move beyond the last message in the active folder using the *next* or *delete* commands, Z-Mail moves the pointer to the first undeleted message in the folder.

wrapcolumn Set *wrapcolumn* to a column number where you want Z-Mail to automatically move to the next line as you compose messages.

Folder Variables

deletesave If you set the *deletesave* variable on, when Z-Mail updates your active folder, it deletes messages you have saved.

folder Set the *folder* variable to the path of a directory where you keep your mail folders.

hangup If you set the *hangup* variable, when Z-Mail receives a SIGHUP signal, it updates your folder before exiting.

hold Set the *hold* variable to tell Z-Mail to keep any messages you have read in your System Folder (This is the default behavior).

keepsave If you set the *keepsave* variable, the *save* and *write* commands do not mark messages for deletion.

mbox Set the *mbox* variable to the pathname of the file you want Z-Mail to use as your default folder for saving mail.

save_empty If you set the *save_empty* variable, it prevents Z-Mail from deleting empty folders.

sort Set the value of the *sort* variable to presort the messages in your System Folder when you first start Z-Mail.

thisfolder Z-Mail sets the *thisfolder* variable to the full pathname of the active folder (current mailbox).

Directory and File Variables

attach_types The *attach_types* variable allows you to specify a file where you define encoding keys and file types for attachments.

cdpath Z-Mail uses the *cdpath* variable to search for a directory when you issue the *cd* command.

cwd Z-Mail sets the *cwd* variable to the name of the current working directory automatically.

dead Set the value of the *dead* variable to the name of the file to which you want Z-Mail to send interrupted (incomplete) messages.

folder Set the *folder* variable to the path of a directory where you keep your mail folders.

home Use the *home* variable to define your home directory. Z-Mail initializes the *home* variable to the value of the *$HOME* environment variable, but you can modify it any time during your Z-Mail session.

logfile Set the *logfile* variable to a filename in which Z-Mail logs the headers of outgoing messages.

mbox Set the *mbox* variable to the pathname of the file you want Z-Mail to use as your default folder for saving mail.

record Set the record variable to the full pathname of a file where you want Z-Mail to record all outgoing mail.

record_users The *record_users* variable allows you to record all outgoing messages you send to a specific set of users for which you have log folders.

templates Set the *templates* variable to a list of directories, separated by spaces, where you store prepared message templates.

tmpdir Set the *tmpdir* variable to the pathname of the directory you want Z-Mail to use to write all Z-Mail temporary files.

Print Variables

printer Set the *printer* variable to tell Z-Mail which printer to use as the default printer when you use the *lpr* command.

print_cmd Set the print_cmd variable to describe a UNIX command other than lpr to send messages to the line printer.

Editor Variables

autoedit If you set the *autoedit* variable, Z-Mail automatically starts the editor of your choice whenever you send or reply to mail.

edit_hdrs If you set the *edit_hdrs* variable, you can use an external editor such as *vi* or *emacs* to edit the headers of outgoing messages in fullscreen or command-line mode.

editor Set the *editor* variable to the name of the text editor you want to use in the command-line and fullscreen interfaces.

visual Set the visual variable to the name of the text editor you want to use in the GUI interface.

Compose Variables

ask If you set the *ask* variable, Z-Mail prompts you for a subject for outgoing mail messages in command-line mode.

askcc If you set the *askcc* variable, Z-Mail prompts you for a list of carbon copy recipients when you are finished editing a message you are composing in command-line mode.

autoinclude If you set the *autoinclude* variable, Z-Mail inserts a copy of the message to which you are replying into the body of all reply messages.

dot If you set the *dot* variable, you can type a period (**.**) on a line by itself at the end of a composed message to tell Z-Mail's command-line interface that you have finished writing the message.

escape If you set the *escape* variable, you can compose a mail message and place an escape character as the first character on a line.

indent_str If you set the *indent_str* variable, Z-Mail indents each line of old messages you include in outgoing messages, preceding each line with the text string you specify.

msg_win Set the *msg_win* variable to tell Z-Mail how many lines to display in the Compose Window in GUI mode.

nosave If you set the *nosave* variable, Z-Mail does *not* store a backup copy of terminated messages.

post_indent_str If you set the *post_indent_str* variable and you include the body of a message into an outgoing message or reply, Z-Mail inserts a line of text after the last line of the included message.

pre_indent_str If you set the *pre_indent_str* variable and you include the body of a message into an outgoing message or reply, Z-Mail inserts a line of text before the first line of the included message.

record Set the *record* variable to the full pathname of a file where you want Z-Mail to record all outgoing mail.

record_users The *record_users* variable allows you to record all outgoing messages you send to a specific set of users for which you have log folders.

Addressing, Routing and Transport Variables

auto_route If you set the *auto_route* variable, Z-Mail removes address redundancies from lists of recipients and checks for matches with known_hosts machines.

domain_route If you use the *domain_route* variable in combination with the *auto_route* variable, you can force Z-Mail to speed delivery of mail to certain addresses that contain fully-qualified domains.

known_hosts Z-Mail uses the *known_hosts* variable in conjunction with the *auto_route* variable.

metoo Set the *metoo* variable to tell Z-Mail to keep you on the list of recipients when you reply to a message.

no_expand If you set the *no_expand* variable, Z-Mail does not expand aliases, so message headers contain aliases exactly as you typed them.

picky_mta If you set the *picky_mta* variable on, Z-Mail does not pass *From:* and *Date:* headers to the mail transport agent when you are sending mail.

sendmail Set the *sendmail* variable to the program and arguments you want Z-Mail to execute when it delivers messages to the mail transport agent.

verbose If you set the *verbose* variable on, Z-Mail passes the *verbose* option to the mail transport agent when it sends messages.

Signature Variables

autosign The *autosign* variable allows you to append a signature to outgoing mail.

autosign2 Use the *autosign2* variable to specify an alternate signature to the signature specified in the autosign variable.

fortune If you set the *fortune* variable, Z-Mail appends a random fortune to the end of all outgoing messages using the UNIX command *fortune*.

fortunates If you set the *fortunates* variable, you can specify a selected group of people who are to receive fortunes.

GUI Variables

autodismiss When you set *autodismiss*, Z-Mail automatically removes dialog boxes from the screen once they have completed an operation.

autoiconify The *autoiconify* variable automatically iconifies any windows you select after performing a specific operation.

crt_win If you are using the GUI interface, set the *crt_win* variable to the number of lines you want Z-Mail to display in the Read Window.

gui_help Z-Mail looks for the file indicated by the *gui_help* variable to find the help file used by the graphical user interface versions of Z-Mail.

mail_icon If you set the *mail_icon* variable to a pathname identifying an icon pixmap Z-Mail uses the pixmap as its desktop icon when you close (iconify) the main Z-Mail window.

newmail_icon If you set the *newmail_icon* variable to a pathname identifying an icon pixmap Z-Mail uses the pixmap as its desktop icon

when new mail has arrived and you have closed (iconified) the main Z-Mail window.

screen_win Set the *screen_win* variable to the number of message summaries you want Z-Mail to concurrently display in the GUI interface.

title The *title* variable describes the title you want Z-Mail to display shown at the top of its Main Window in the GUI and fullscreen modes.

window_shell Set the window_shell variable to the name of the program to run when you create shell processes in GUI mode.

Fullscreen and Command-Line Variables

crt If you are using the command-line interface, set the *crt* variable to the number of lines on your display.

history Set the *history* variable to the number of commands you want the shell interface to remember.

ignore_bang If you set the *ignore_bang* variable, Z-Mail ignores exclamation point characters (**!**) as command history references.

nonobang If you set the nonobang variable, Z-Mail avoids expanding history references that do not match anything, rather than generating error messages.

no_reverse If you set the *no_reverse* variable, Z-Mail does not use reverse video in the fullscreen interface to indicate the current message.

pager When Z-Mail displays a message longer than the number of lines you specify with the *crt* variable, Z-Mail executes the program you specify with the *pager* variable.

prompt Set the *prompt* variable customize Z-Mail's command-line prompt.

screen Set the *screen* variable to the number of message summaries you want Z-Mail to concurrently display in the command-line and fullscreen character interfaces.

Programming Variables

input The *input* variable holds the message list that Z-Mail passes to a function in a command pipeline.

is_fullscreen Z-Mail sets the *is_fullscreen* variable to true if Z-Mail is in full-screen mode, even though the interface may not yet be started.

is_gui Z-Mail sets the *is_gui* variable to true if Z-Mail is in graphical user interface mode, even though the interface may not yet be started.

is_sending Z-Mail sets the *is_sending* variable to true if you issue the *zmail* command from the UNIX command line to send a message.

is_shell Z-Mail sets the variable *is_shell* true when it has started the Z-Mail shell.

output Z-Mail sets the *output* variable to a message list representing the output of your last successful command.

realname Set the *realname* variable to your actual name. Z-Mail initializes *realname* to the value of the *NAME* environment variable when you start Z-Mail.

recursive If you set the *recursive* variable, Z-Mail allows user-defined functions to recursively call themselves.

redirect Z-Mail sets *redirect* to *true* if it believes input to Z-Mail is redirected.

status The *status* variable reports the success or failure status of your most recent Z-Mail command.

thisfolder Z-Mail sets the *thisfolder* variable to the full pathname of the active folder (current mailbox).

$* Z-Mail sets $* equal to the a character string containing all the arguments passed to the currently executing function.

$# Z-Mail sets $# equal to the number of arguments passed to the currently executing function.

$0 Z-Mail sets $0 equal to the name of the currently executing function.

$1, $2, ... Z-Mail sets each of $1, $2, ... $*n* equal to the individual arguments passed to the currently executing function.

UNIX System Variables

hostname Set the *hostname* variable to the name of your computer. Z-Mail uses the value contained in the *hostname* variable to compose a correct *From:* line if you use a mail transport agent that does not create this header automatically.

ignoreeof Set the *ignoreeof* variable so when you type Ctrl-D, Z-Mail does not exit.

newline Set the *newline* variable to a string so that the string is executed as a command when you type a newline (carriage return).

realname Set the *realname* variable to your actual name. Z-Mail initializes *realname* to the value of the *NAME* environment variable when you start Z-Mail.

unix If you set the *unix* variable on, Z-Mail treats all commands that are not Z-Mail commands as UNIX commands.

$$ This special Z-Mail variable returns the process id (PID) of the current Z-Mail process.

Variable Reference

The remainder of this section describes all pre-defined Z-Mail variables in alphabetical order.

alwaysignore

BOOLEAN / MULTIVALUE

You use the *alwaysignore* variable to completely stop Z-Mail from showing message headers you have specified in the *ignore* command.

You use Z-Mail's *ignore* command to specify which headers you want Z-Mail to ignore when it displays messages. When you issue a *display*, *type* or *print* command, Z-Mail does not display the selected headers. However, if you capitalize the first letter of the commands (i.e., *Display*, *Type*, *Print*), Z-Mail shows the message headers anyway.

Normally, Z-Mail also displays message headers you set with the *ignore* command when you:

- Send messages to the printer,
- Interpolate messages into letters with *~f* or *~I*,
- Use the *–I* option with the *mail* or *reply* commands.

You can set the value of *alwaysignore* to one or more of the following values, Z-Mail ignores headers as indicated:

forward	Ignore headers when forwarding messages (~f).
include	Ignore headers when including messages (~I).
pipe	The *pipe* command ignores headers.
printer	The *lpr* command ignores headers.

If you set *alwaysignore*, but specify none of the above arguments, it is equivalent to specifying all of the arguments.

Note: Z-Mail never ignores headers during updates or when you use the *save* command, since you might ignore headers required by Z-Mail or any other mail system to read folders.

For more information, see the *ignore* command on page 118.

ask

BOOLEAN

If you set the *ask* variable, Z-Mail prompts you for a subject for outgoing mail messages. Once you are in the message, use the tilde escape *~s* to set the subject or specify the *–s* option in the *mail* command at the Z-Mail prompt.

askcc

BOOLEAN

If you set the *askcc* variable, Z-Mail prompts you for a list of carbon copy recipients when you are finished editing a message you are composing. If you set the *edit_hdrs* variable, Z-Mail does not prompt you for a list of carbon copy recipients, but adds a *Cc:* line to outgoing messages which you may fill in using your text editor.

attach_types

STRING

The *attach_types* variable allows you to specify a file where you define encoding keys and file types for attachments. Z-Mail provides you with a default file containing some of these definitions. The file named by the *attach_types* variable may contain alternate or additional encodings and file types.

You must use the **detach –rehash** command to reload the attachment type definitions if you either change the value of the *attach_types* variable or you modify the file while Z-Mail is running.

autodismiss

BOOLEAN / MULTIVALUE GUI ONLY

When you set *autodismiss*, Z-Mail automatically removes dialog boxes from the screen once they have completed an operation. You may set *autodismiss* to a list of keywords, separated by commas or spaces, which describe the dialog boxes you want to affect. *Autodismiss* closes the following dialog boxes:

Keyword	***Automatically closes:***
buttons	The Buttons dialog, after installing a button.
compose	The Compose window, after sending mail.
folder	The Find Folders dialog, after changing folders.
print	The Print Messages dialog, after printing.
save	The Save Messages dialog, after saving.
sort	The Custom Sort dialog, after sorting.
templates	The Templates dialog, after Use.

If you set *autodismiss* on, but you do not specify a list of dialog box names, Z-Mail closes all dialog boxes automatically after you use them to perform an operation. The autodismiss variable only has an effect in Z-Mail's GUI interface.

autoedit

BOOLEAN / MULTIVALUE

If you set the *autoedit* variable, Z-Mail automatically starts the editor of your choice whenever you send or reply to mail. This variable can also be limited to particular circumstances by setting one of the following keywords:

new	Autoedit only when you compose a new message.
reply	Autoedit only when you compose a reply.

The *reply* argument is helpful when you use it in conjunction with the *autoinclude* variable.

autoiconify

BOOLEAN / MULTIVALUE GUI ONLY

The *autoiconify* variable automatically iconifies any windows you select after performing a specific operation.

If you set both *autodismiss* and *autoiconify* for a particular window, *autoiconify* takes precedence and Z-Mail iconifies the window. Setting *autoiconify* without any specific conditions is equivalent to setting all of the possible conditions. *Autoiconify* affects the following dialog boxes:

Keyword:	***Automatically iconfies:***
compose	The Compose window after sending mail.
message	The Message Display window on delete.

The autoiconify variable only affects Z-Mail's GUI interface.

autoinclude

BOOLEAN

If you set the *autoinclude* variable, Z-Mail inserts a copy of the message to which you are replying into the body of all reply messages. To make it easier for recipients to identify the included message, Z-Mail precedes the included text with the value of the *pre_indent_str* variable, follows the text with the value of *post_indent_str*, and indents the text using the string described by the *indent_str* variable.

autoprint

BOOLEAN

If you set the *autoprint* variable, after you delete a message, Z-Mail automatically displays the next message. This is particularly helpful when you are using the GUI interface.

auto_route

BOOLEAN / STRING

If you set the *auto_route* variable, Z-Mail automatically generates the most efficient address it can for recipients of outgoing messages. When you set *auto_route*, Z-Mail:

- Shortens UUCP addresses whenever possible,
- Takes advantage of implied UUCP connections, and
- Take advantage of hosts to which your organization has direct mail connections.

When you reply to a message, Z-Mail examines the *From:* and *Cc:* headers for information about return paths to the author and other recipients of the original message.

Note: The *–r* flag on the *reply* and *replyall* commands overrides the value of *auto_route*.

EXAMPLES

The following example explains how Z-Mail constructs return paths when you reply to a message.

How Z-Mail Constructs Return Paths

Assume you are Dana Parsons and your e-mail address is *dparsons@utech.com*. Josh Ford — whose e-mail address is *jford@chemcorp.com* — sends you a message and send a copy of the message to Regina Tracy at *rtracy@gasco.com*. On the way to you and Regina, the message passes through several UUCP nodes called *blue*, *green* and *orange*. The exact path the message took to get to you is indicated below in the *From:* header of the example. the path the carbon copy took to get to Regina is in the *Cc:* header.

```
From: orange!green!chemcorp!jford (Josh Ford)
To: dparsons@utech.com (Dana Parsons)
Cc: blue!gasco!rtracy (Regina Tracy)
```

If you reply to Josh and send a copy of the reply to Regina, Z-Mail constructs the following return paths from the header information above:

```
From: dparsons@utech.com (Dana Parsons)
To: orange!green!chemcorp!jford (Josh Ford)
Cc: orange!green!blue!gasco!rtracy (Regina Tracy)
```

Z-Mail knows that since Josh sent mail to Regina via the host called *green*, that his machine can also route your reply via the same path.

Taking Advantage of Known Hosts

Continuing with the same example, assume that your machine, *utech*, has direct network connections to the hosts called *blue* and *black*. You therefore set the value of the *known_hosts* variable with the following command:

```
set known_hosts = "blue orange black"
```

If you set *auto_route*, Z-Mail takes advantage of these known hosts when constructing return paths for replies. Z-Mail scans the return path of each addressee of your reply, and discovers that the carbon copy to Regina is routed through the *blue* host. Since you have a direct connection to both *orange* and *blue*, Z-Mail selects the host farthest to the right (closest to Regina's host) and shortens Regina's address in the *Cc*: header to:

```
Cc: blue!gasco!rtracy (Regina Tracy)
```

For more information on specifying hosts to which you have direct connections, see the *known_hosts* variable on page 45.

Setting *auto_route* to a Pathname

If you set *auto_route* to a specific UUCP pathname (e.g., host names separated by **!** characters), Z-Mail prepends that pathname to all addresses in a list of recipients. This is useful for hosts which have a UUCP connection to a node that connects to the Internet or uunet.

Assume you are on the host named *bigcorp* and it has a direct UUCP connection with the host called *calif*. You can describe your path to the Internet by setting the value of *auto_route* with this command:

```
set auto_route = "bigcorp!calif"
```

If you reply to a message from Alan Roberts with a *From:* header such as:

```
From: west!cosoft!aroberts (Alan Roberts)
```

Z-Mail generates this address:

```
To: bigcorp!calif!west!cosoft!aroberts
```

Eliminating Redundant Hosts

After constructing return paths, Z-Mail eliminates duplicate hosts from each address. Suppose you have defined the value of auto_route with the following statement:

```
set auto_route = "bigcorp!calif"
```

If you receive a message from Alan Roberts with the following *From:* header:

```
From: west!bigcorp!cosoft!aroberts (Alan Roberts)
```

Z-Mail generates this address:

```
To: bigcorp!calif!west!bigcorp!cosoft!aroberts
```

And then, Z-Mail notices that *bigcorp* appears twice in the return path to Alan. There is no need to route the message through the hosts names *calif* and *west* since all they will do is send the message back to *bigcorp*. Therefore, before sending the message, Z-Mail eliminates the redundant hosts and shortens the return path:

```
To: bigcorp!cosoft!aroberts
```

autosign

BOOLEAN / STRING

The *autosign* variable allows you to append a signature to outgoing mail. If you set *autosign* on, but not to a string, Z-Mail appends the text contained in the *~/.signature* file to your messages. Otherwise, Z-Mail interprets the *autosign* variable in one of four ways:

- By default, Z-Mail interprets the string as a pathname relative to the current directory. For this reason, you should always use full pathnames. As usual, the special characters tilde (**~**) and plus (**+**) are expanded for you. If Z-Mail finds the file, it opens the file and reads its contents into the message buffer.
- If you set *autosign* to a string that begins with a backslash (\ ) then Z-Mail uses the string itself as the signature (not including the \ character). Z-Mail performs no expansion and reads no additional files.
- If you set *autosign* to a string that begins with a vertical bar or *pipe* character (|), Z-Mail interprets the rest of the string as a command whose output Z-Mail uses as the signature. Z-Mail runs the command using the */bin/sh* shell without expanding the special characters **~** and **+** in the command name and looks for the command in all directories specified in your *$PATH* environment variable. Z-Mail passes the list of addresses to the command as arguments in the same form they are passed to the mail transport agent (MTA). Depending on your MTA, you may find that the addresses have been separated by commas.

EXAMPLE

In the latter two cases, set the value of *autosign* using single quotes to prevent Z-Mail from interpreting the \ and | characters. For example:

```
set autosign = '\Support support@z-code.com'
```

If you specify redirection of the command's output from the calling shell, Z-Mail ads no signature or fortune to outgoing mail. For example:

```
% zmail –s report michelle < report_file
```

In this command, Z-Mail sends a message to the user *michelle* with a subject of *report*. Z-Mail redirects the contents of the file *report_file*, using the text as the body of the outgoing message. In this case, Z-Mail appends no signature.

autosign2

STRING

You may use the *autosign2* variable to specify an alternate signature to the signature specified in the *autosign* variable. You can use *autosign2* for special cases where:

- You do not want to use your default signature.
- You do not want a signature sent to special addresses.
- You want to sign a specific way for special addresses.

When you set the value of *autosign2*, you specify a list of addresses and/or address domains for which you want to use your alternate signature, as well as the signature itself. If you do not specify a signature, Z-Mail does not sign messages whose addresses match the addresses specified in the value of the *autosign2* variable.

SYNTAX

The format for setting the value of the *autosign2* variable is:

```
set autosign2 = "address1, address2, . . . : <signature>"
```

The colon (**:**) separates the list of addresses from the optional signature description. If you specify no colon or omit the address list, Z-Mail treats the entire string as the signature, excluding the colon.

When you send a message, Z-Mail looks at the addresses of all recipients you specify. Z-Mail expands aliases, removes comments, then scans the list of recipients, matching the following patterns against those addresses (*address1*, *address2*, etc.) you specified in the *autosign2* variable:

address or address@domain
If you specify an *autosign2* address with no special characters, Z-Mail compares each recipient of an outgoing message for an exact character match.

****username***
If you precede an e-mail address in the value of *autosign2* with a * character, Z-Mail performs an exact character match against the username of recipients, ignoring any sites or domains you may specify (e.g. *@hq2.com*).

!host **or** ***!hostname!host1!host2...***
If you specify a path of hostnames beginning with the **!** character, recipients who reside on the last host of the path match.

@domain **or** ***@sub.domain***
If you specify a domain name beginning with the @ character, all users residing in that domain match.

If *all* the recipients you specify in your outgoing mail message match one of the above patterns, Z-Mail uses the signature you specify in the value of the *autosign2* variable. Otherwise, Z-Mail uses the signature you specify in value of the *autosign* variable.

Notes:

- You must set the *autosign* variable for the *autosign2* variable to work.
- If you invoke the *zmail* command using redirection from the calling shell, Z-Mail does not append signatures.

EXAMPLES

The following table gives examples of how Z-Mail matches *autosign2* addresses against the addresses of recipients you specify in outgoing messages.

Autosign2 address	***Recipient:***	***Match?***
rtaylor	**rtaylor**	Yes
rtaylor	**rtaylor@xyz.com**	No
***rtaylor**	**rtaylor**	Yes
***rtaylor**	**rtaylor@xyz.com**	Yes
!xyz	**xyz!rtaylor**	Yes
@xyz.com	**rtaylor**	No**
@xyz.com	**rtaylor@xyz.com**	Yes
@xyz	**rtaylor@xyz.com**	Yes

** Assuming you are not using a host named *xyz.com*

EXAMPLE 1

Using *autosign* and *autosign2*, you can designate separate signatures for local (internal) and remote (external) mail messages.

```
set autosign = "~/external.sig"
set autosign2 = "!research @zcode.com *ian tom: ~/internal.sig"
```

In this example, Z-Mail uses the alternate signature in the file called *internal.sig* whenever you send mail to:

- Anyone on the host named *research*,
- Anyone within the *zcode.com* domain,
- User *ian* at any host anywhere—even locally, and
- User *tom* on your local machine only.

In all other cases, Z-Mail signs your outgoing messages with the signature it finds in the *external.sig* file.

EXAMPLE 2 Similarly, you can designate no signature for internal messages by leaving the signature file off the value of *autosign2* :

```
set autosign = "~/external.sig"
set autosign2 = "!research @zcode.com *ian tom: "
```

EXAMPLE 3 If some of the recipients of your message match the autosign2 address list, and some of the recipients do not, Z-Mail uses the value of *autosign* and not *autosign2* to sign your outgoing message. For example:

```
set autosign = "~/standard.sig"
set autosign2 = "michelle, shawna, johann: /alt.sig"
```

If you send a message to **johann** and **shawna**, Z-Mail signs the outgoing message with the contents of the *alt.sig* file. But if you send a message to **johann**, **shawna** and **kim**, Z-Mail uses the text in the *standard.sig* file since **kim** is not in the list of *autosign2* addresses.

cdpath

STRING

Z-Mail uses the *cdpath* variable to search for a directory when you issue the *cd* command. You set the *cdpath* variable to a string of pathnames separated by spaces.

cmd_help

STRING

Z-Mail looks for the file indicated by the *cmd_help* variable to find the general help file used by the Z-Mail *help* command. Set the value of *cmd_help* to the complete pathname of the help file. Normally, only your system administrator resets this variable in Z-Mail's initialization files, when the default location of the *help* file has changed. In most cases, you do not have to reset the value of the *cmd_help* variable.

complete

STRING FULLSCREEN & COMMAND-LINE ONLY

You set the *complete* variable to enable filename completion. When you type the completion character at the end of a string of characters, Z-Mail interactively completes that filename. If the prefix is not unique, Z-Mail completes the filename as far as possible and sounds a bell. You can configure Z-Mail not to ring the bell by setting the *quiet* variable.

Z-Mail uses the first character of the value of the *complete* variable as the *completion character*. If you set the *complete* variable, but not to a value, Z-Mail uses the **escape** character as the completion character.

Z-Mail uses the second character of the value of the *complete* variable as the *completion listing character*. When you type this character, Z-Mail displays all possible completions and prompts you again with the original prefix. If you set *complete* to only a single character, you Z-Mail disables completion listing; if you set *complete* on but specify no string, Z-Mail defaults to using **Ctrl-D** as the completion listing character.

If the first and second characters you specify in the *complete* variable are the same, Z-Mail completes unique matches as usual. However, partial matches produce a listing of the possible completions before completing as far as possible. See the description of the *fignore* variable for ways to exclude filenames from completion.

If the filename you are attempting to complete contains metacharacters, Z-Mail completes all possible matches. See the section *Specifying Filename Patterns* on page 12 for details on filename metacharacters.

EXAMPLES

This chart demonstrates the use of the *complete* variable.

Command	*Sets completion character to:*	*Sets completion listing character to:*
set complete	**Escape**	undefined
set complete = \E=	**Escape**	**=**
set complete = \E^L	**Escape**	**Ctrl-L**

Notes

- Z-Mail cannot provide filename completion if you specify the *–echo* (*–E*) option in the *zmail* command.
- In the GUI interface, the window manager may use **Ctrl** key sequences and the **Esc** key for windowing commands, so you may not be able to use **Ctrl-D** or **Esc**.

- In the command-line interface, if the list of possible completions overflows the command-line buffer, Z-Mail truncates all characters beyond the end of the buffer.
- If you want to silence the bell that Z-Mail sounds when it performs filename completion, set the *quiet* variable so it includes *complete* as one of its values.

crt

NUMERIC FULLSCREEN & COMMAND-LINE ONLY

If you are using the character interface, set the *crt* variable to the number of lines on your display. If the number of lines in a message exceeds the value of *crt*, Z-Mail invokes a pager so you can view the message in sections.

crt_win

NUMERIC GUI ONLY

If you are using the GUI interface, set the *crt_win* variable to the number of lines you want Z-Mail to display in the Read Window. If a message exceeds that number of lines, use the scroll bars on the side of the window to move the text so you can read the entire message.

cwd

STRING

Z-Mail sets the *cwd* variable to the name of the current working directory automatically. Z-Mail sets the value of *cwd* automatically at start-up and resets it each time you use the *cd* or *pwd* commands. You may change this variable like any other shell variable, but it does *not* cause the working directory to change.

date_received

BOOLEAN

If you set the *date_received* variable, Z-Mail displays the date and time you receive messages rather than the default of when they were sent by the author.

When you sort messages by date, Z-Mail checks this variable to determine whether the messages should be sorted by date sent or date received.

dead

STRING

Set the value of the *dead* variable to the name of the file to which you want Z-Mail to send interrupted (incomplete) messages. If you do not give dead a value, Z-Mail sends interrupted messages to a file called *dead.letter*. You may set the *dead* variable to a UNIX command by prefixing the value with the pipe character (|). For more information, see the *nosave* variable.

deletesave

BOOLEAN

If you set the *deletesave* variable, when Z-Mail updates your active folder, it deletes messages you have saved (and not explicitly preserved) since the folder's last update. Z-Mail does this by default when your active folder is the System Folder (*spool file*), but in other folders, Z-Mail does not normally delete saved messages. Messages you have preserved with the *preserve* command are not affected by *deletesave*, and are not deleted by Z-Mail. If you set both *keepsave* and *deletesave*, *keepsave* takes precedence and Z-Mail does not delete messages you have saved.

domain_route

BOOLEAN / STRING

If you use the *domain_route* variable in combination with the *auto_route* variable, you can force Z-Mail to speed delivery of mail to certain addresses that contain fully-qualified domains.

If your machine already knows how to deal with domains, set *domain_route*, but specify no list of domains. Z-Mail then eliminates all intermediate hosts, but does not add routing information about your path to an Internet machine.

If your machine doesn't handle domains, set *domain_route* to a UUCP path from your machine to an Internet host or other machine that does understand domains.

Note: You can *only* apply domain short-circuiting to addresses containing a fully-qualified domain. Z-Mail short-circuits the route in spite of any path you specify with the *reply –r* option or through the string value of *auto_route*. As a result, Z-Mail is likely to bypass any *auto_route* or *–r* options you have set. For more information, see the *auto_route* variable on page 26.

EXAMPLE Instead of routing mail via all the hosts specified in a routing path, you can tell Z-Mail to deliver directly its final domain via your Internet connection.

For example, suppose that your machine named *thinker* is not directly connected to the Internet. Instead, you must route Internet mail through a host named *research* which is connected to a machine named *wserver* which is on the Internet. You can set *domain_route* on *thinker* to automatically route responses to messages you receive:

```
set domain_route = research!wserver
```

Then, if you received mail with an address like:

```
uunet.UU.NET!west!victoria.vic3.com!sam
```

Z-Mail uses *domain_route* to shorten it to:

```
victoria.vic3.com!sam
```

and then prepend the value of *domain_route*, resulting in:

```
research!wserver!victoria.vic3.com!sam
```

This allows Z-Mail to send the mail through *research* and *wserver* and lets *wserver* decide the best route to *victoria*.

dot

BOOLEAN FULLSCREEN & COMMAND-LINE ONLY

If you set the *dot* variable, you can type a period (**.**) on a line by itself at the end of a composed message to tell Z-Mail you have finished writing the message. Z-Mail continues to accept **Ctrl-D** as a valid way to perform the same action. In the GUI interface, the *dot* variable has no effect; instead, you either select the *Send* option from the *Deliver* menu, or press the *Send* button at the bottom of the Compose dialog box when you are finished composing messages.

edit_hdrs

BOOLEAN

While using the fullscreen or character interfaces, if you set the *edit_hdrs* variable, you can use an external editor such as *vi* or *emacs* to edit the headers of outgoing messages. Z-Mail makes this possible by placing the headers of your outgoing message in the same editing buffer as the text of the letter.

Once you are editing a message, you cannot set or unset the *edit_hdrs* variable to affect the message you are currently composing. You must first end the message, either by sending it or canceling it, and then reset the *edit_hdrs* variable.

You should exercise caution when you edit the headers of an outgoing message, noting these rules:

- Z-Mail requires a *To:* header. Without this header, Z-Mail cannot send your message. Even if you specify a valid *Cc:* header, you must still include a *To:* header. You may have as many *To:* and *Cc:* headers as you like.
- You should not change the *From:* header. If you change this header to an address that Z-Mail is unable to identify as authentic, Z-Mail silently inserts your original *From:* header.
- Z-Mail always replaces the *Date:* header with a more accurate time and date stamp. If you change or remove this header it has no effect.
- You cannot add a *Status:* header; Z-Mail will remove it if you do.
- Z-Mail drops blank headers. For example, if your message contains an empty *Cc:* header, Z-Mail deletes the header from the outgoing message.
- If you specify any *aliases* on the command line, Z-Mail expands them and places them into the message buffer in their expanded form, regardless of the value you set for the *no_expand* variable.
- If you set the *no_expand* variable and you use an external editor to edit the headers of a message, Z-Mail does not expand the aliases you enter in the message headers.
- Z-Mail always removes *Bcc:* headers, but still delivers messages to the addresses you specify.
- You can add an *Fcc:* header to specify a *File Carbon Copy.* This must be a pathname to a file or program. For programs, you must precede the pathname with a pipe character (|). Z-Mail interprets all addresses on the *Fcc:* line that do not begin with a |character as filenames. Do not put any other types of addresses in an *Fcc:* header.

Note: If you set the *edit_hdrs* variable, Z-Mail ignores tilde escapes that allow you to add or set headers. You must instead perform these functions with an external editor.

editor

STRING FULLSCREEN & COMMAND-LINE ONLY

Set the *editor* variable to the name of the text editor you want to use in the command-line and fullscreen interfaces. If you set *editor*, but do not specify the name of an editor, Z-Mail uses the value of the *EDITOR* environment string or the *visual* variable.

To make Z-Mail start the *editor* every time you compose or reply to a message, set the *autoedit* variable. Otherwise, you must manually invoke the editor by typing **~e** while editing messages.

escape

STRING FULLSCREEN & COMMAND-LINE ONLY

If you set the *escape* variable, you can compose a mail message and place an escape character as the first character on a line. This causes Z-Mail to examine the next character and execute a function associated with that *escape* command. See the section called *Tilde-Escape Editing Commands* beginning on page 65 for more information.

fignore

STRING

Set the *fignore* variable to a list of filename extensions to ignore when Z-Mail performs filename completion. You may set the value of *fignore* to a list of filename extensions (e.g., **.c .o**), a list of filename patterns containing metacharacters (*** ? { } [ ]**), or a mixture of the two. You must separate each element in the list must from the others with a space. When you ask Z-Mail to perform filename completion, it omits any filenames with the extensions or patterns specified in the value of *fignore* from the list of possible completions.

Technical Notes:

- Z-Mail checks the value of fignore only when you enter the completion listing character, and not the completion character from your keyboard. For more information, see the *complete* variable on page 32.
- If all files in the current directory match the extensions or patterns, Z-Mail disables *fignore* and performs completion. For this reason, it is usually not a good idea to include * in the list of value for the *fignore* variable.

EXAMPLE The following command causes any filename ending in *.o* or *.s*, and any filename beginning with *Make* or *make*, to be excluded from completions:

```
set fignore = ".o .s [Mm]ake*"
```

folder

STRING

Set the *folder* variable to the path of a directory where you keep your mail folders. Z-Mail uses this path for various commands to expand the plus metacharacter (**+**). By default, Z-Mail expands the **+** metacharacter to the path *~/Mail*.

For more details, see the *folder* command on page 105 and the section called *Specifying Filename Patterns* on page 12.

fortune

BOOLEAN / STRING

If you set the *fortune* variable, Z-Mail appends a random fortune to the end of all outgoing messages using the UNIX command *fortune*. If you set the *fortune* variable to something that starts with a hyphen (e.g. **– o**), Z-Mail interprets it as an option to the *fortune* command. If the value of the *fortune* variable starts with a **/**, then Z-Mail executes the program described by the string instead of looking for the *fortune* program.

By default, Z-Mail uses the command **/usr/games/fortune –s** which adds a short fortune to outgoing mail messages.

Note: The pathname for the *fortune* program can vary, so it may be different on your machine.

See the *fortunates* variable for specifying a list of people who are to receive fortunes.

fortunates

STRING

If you set the *fortunates* variable, you can specify a selected group of people who are to receive fortunes. You set the value of fortunates to a list of addresses or aliases separated by spaces or commas, following the same rules as the *autosign2* variable.

In an outgoing messages, if the *To:* and *Cc:* lines contain only addresses specified in the value of *fortunates*, then Z-Mail appends a fortune to the message. If one or more addresses in the To: and Cc: lines contain addresses that are not in the *fortunates* list, then Z-Mail does not append a fortune.

Once you set *fortunates*, you cannot manually override its effect. In other words, you cannot use the *~F* tilde command to force a fortune to be added unless the individuals on the recipient list are all included in the *fortunates* list.

You must set the *fortune* variable in order for the *fortunates* variable to function.

EXAMPLE An alternate use of the *fortune* and *fortunate* variables is to provide extra information in your signature. For example:

```
set fortunates = @gamma.com
set fortune = '/bin/echo "Vice President, Betadata" '
```

Whenever you send mail to anyone at *gamma.com*, Z-Mail appends the string *Vice President, Betadata* to your messages.

fullscreen_help

STRING FULLSCREEN ONLY

When you set the *fullscreen_help* variable to a string of commands separated by spaces, Z-Mail lists those commands at the bottom of the screen all the time. For an example, see *Fullscreen Interface Basics* on page 60.

If you set the *fullscreen_help* variable on, but specify no string, Z-Mail displays a default list of commands. The *help* display normally shortens the display of headers to save screen space, but you may can scroll the display if you want to see more headers. For a listing of fullscreen commands, see the *Z-Mail Fullscreen Command List* on page 61.

gui_help

STRING GUI ONLY

Z-Mail looks for the file indicated by the *gui_help* variable to find the help file used by the Z-Mail's graphical user interface. Set the value of *gui_help* to the complete pathname of the help file. Normally, only your system administrator resets this variable in Z-Mail's initialization files, when the default location of the *help* file has changed. In most cases, you do not have to reset the value of the *gui_help* variable.

hangup

BOOLEAN

If you set the *hangup* variable, when Z-Mail receives a SIGHUP signal, it updates your folder before exiting. This is useful if you frequently read mail when dialed in over a noisy phone line that often drops carrier. The *hangup* variable also forces an update when the X server shuts down in Z-Mail's GUI mode.

Note: Z-Mail ignores certain errors during this update process because it presumes that it is impossible to ask you for instructions on how to recover from the errors. Z-Mail forces the folder to contain all messages that were not deleted at the time

of the hangup. This means if an error occurs as Z-Mail loads new mail, the new mail is lost. On write errors, Z-Mail preserves both the working tempfile and as much of the active folder as possible.

hdr_format

STRING

The *hdr_format* variable allows you to control the format of Z-Mail displays. This variable affects the main Z-Mail window in GUI mode, as well as the information displayed by the headers command in the fullscreen mode.

When Z-Mail displays header information for a message, it evaluates the *hdr_format* variable and echoes each character in the string unless it encounters a % character identifying the beginning of a message information string. For a list of valid strings, see *Message Information Strings* on page 10.

You may specify a field width for each variable. For example, **%20f** displays the first 20 characters of the *From:* line. You may also use colon variable modifiers in conjunction with this bracket notation. For more information, see *Variable Modifiers* on page 9.

EXAMPLES

For example, if you issue the command:

```
set hdr_format = "%20n %M %-2N %7T  %20s"
```

Z-Mail lists headers like this:

```
  1  !  Sean O'Brien      Jul 15  8:49am   Meeting at 11:30
 >2  N  Jon Yost          Jul 16 12:10pm   Porting standards
  3     Ian Trainor       Jul 16  1:05pm   Maintenance program
```

Note how Z-Mail always displays the message number as the first piece of information on a line.

Z-Mail extracts the address and name of authors from the *From:* field of messages. You may specify the name in parentheses and Z-Mail interprets the rest of the line as the address.

Z-Mail also lets you refer to the contents of headers not shown in the list of *Message Information Strings* on page 10. If you specify *hdr_format* containing a string of the form **%?*hdrname*?**, Z-Mail substitutes the contents of the header named *hdrname*. You must include both the leading and trailing question marks. For example:

```
set hdr_format = "%a %n %?phone?"
```

causes Z-Mail to display the electronic address, name, and if a *Phone:* header is present, the phone number of the sender.

hdrs_only

READ-ONLY BOOLEAN SEND-ONLY MODE

Z-Mail sets the value of the *hdrs_only* variable to true if you specified the *–headers* option in the *zmail* command that started your Z-Mail session. You can use *hdrs_only* in initialization files to see if you started Z-Mail only to display message headers.

You can put the following commands at the beginning of initialization files to check if you are running Z-Mail only to display header summaries. If so, the *exit* command script stops the initialization file after you have set the *hdr_format* variable for displaying the headers. This speeds up Z-Mail's startup time, bypassing lengthy initialization files.

EXAMPLE For example:

```
set hdr_format = "%25f %7d (%l/%c) %25s"
if hdrs_only
      return
endif
```

history

INTEGER COMMAND-LINE ONLY

Set the *history* variable to the number of commands you want the shell interface to remember. This variable functions identically to the *history* variable used in *csh*. If you unset the Z-Mail *history* variable, Z-Mail maintains no command history and you can refer only to the previous Z-Mail command. For more information, see the *history* command on page 113.

hold

BOOLEAN

You set the *hold* variable to tell Z-Mail to keep any messages you have read in your System Folder, usually a file called */usr/spool/mail/username* or */usr/mail/username*. If *hold* is not set, when you update your system folder, Z-Mail saves any messages you have read, except those marked as *preserved*, in your *mbox*. The *hold* variable only applies to the System Folder, and not to other mail folders.

home

STRING

Use the *home* variable to define your home directory. Z-Mail initializes the *home* variable to the value of the *$HOME* environment variable when Z-Mail starts. Your home directory is the directory where Z-Mail stores temporary files for editing

and other tasks. If Z-Mail cannot find or read your home directory, Z-Mail uses the */tmp* directory instead.

hostname

STRING

Set the *hostname* variable to the name of your computer. Z-Mail uses the value contained in the *hostname* variable to compose a correct *From:* line if you use a mail transport agent that does not create this header automatically. This feature aids recipients of your mail when they reply to your messages. Z-Mail also uses the *hostname* variable when it performs auto-routing.

Note: You normally do not have to set the *hostname* variable since it should be set automatically by your system. However, some system's hostnames cannot be queried and you may have to set this variable manually. In most cases, if this is an issue, you system manager will have already set your *hostname* in the Z-Mail's system initialization file *system.zmailrc*.

ignore_bang

BOOLEAN COMMAND-LINE ONLY

If you set the *ignore_bang* variable, Z-Mail ignores exclamation point characters (**!**) as command history references. Note that setting *ignore_bang* severely limits use of the *cmd* command, which depends upon history references for argument substitutions.

ignoreeof

BOOLEAN / STRING FULLSCREEN & COMMAND-LINE ONLY

Set the *ignoreeof* variable so when you type **Ctrl-D** as the first character on a line in command-line mode, Z-Mail does not exit. If you set *ignoreeof* to a string, Z-Mail executes that string as a command when you type **Ctrl-D**. This does not effect termination of messages under the *mail* and *reply* commands.

EXAMPLE

```
set ignoreeof = "echo Please type quit to exit Z-Mail."
```

indent_str

STRING

If you set the *indent_str* variable, Z-Mail indents each line of messages you include in outgoing messages, preceding each line with the text string you specify.

The value of *indent_str* may contain the same formatting characters described in *Message Information Strings* on page 10. These characters are expanded from the headers of the message

that is being indented. If you unset *indent_str*, Z-Mail indents included messages with the **>** character. For more information on identifying messages you include in outgoing mail, see the *pre_indent_str* and *post_indent_str* variables.

EXAMPLE Assume you define the value of *indent_str* with the command:

```
set indent_str = ">>>"
```

If you include a message in an outgoing message, Z-Mail indents it:

```
From: dsmith@bigcorp
To: rwillet
Subj: Directions to offsite meeting

Here's the instructions to the offsite:
>>> From the airport:
>>> Take route 71 south for 5 miles.
>>> Exit at Rolling Road. Go west for 1.3 miles.
>>> Turn left at Moore Avenue. Go 3 blocks.
>>> The Prescott Hotel is on your right.
```

input

READ-ONLY STRING

The *input* variable holds the list of messages that Z-Mail passed to the current function via a pipeline. This variable is most useful in line-mode and in scripts. You can use *input* within a function to test if the function is being used in a pipeline, or to test for the presence of a specific set of messages.

If Z-Mail pipes no messages to a function, *input* contains an empty value.

EXAMPLE

```
function tellinput() {
      if $?input
          echo The messages piped to this function are: $input
      else
          echo This function is not in a pipeline.
      endif
}
```

For more information on piping message lists, see the *output* variable on page 49.

in_reply_to

STRING

Set the *in_reply_to* variable to a string that completes the header *In-Reply-To:*. The format of this string is identical to the options for the *Message Information Strings* on page 10.

EXAMPLE If you respond to a message from Carol Scott that was sent on August 21, 1991, at 10:39pm, with *in_reply_to* set to the string:

```
%n's message of %d.
```

Z-Mail adds this header line to your message:

```
In-Reply-To: Carol Scott's message of Aug 21, 1991, 10:39pm.
```

is_fullscreen

READ-ONLY BOOLEAN

Z-Mail sets the *is_fullscreen* variable to true if Z-Mail is in fullscreen mode, even though the interface may not yet be started. You can test the value of *is_fullscreen* in initialization files to decide what variables to set to configure your Z-Mail session.

EXAMPLE For example:

```
if ! is_fullscreen
        set crt = 24
        set screen = 18
endif
```

This *if* statement checks that you are not running in fullscreen mode (the **!** character means *not*), and if not it sets the *crt* and *screen* sizes. This is useful because the fullscreen mode sets those values automatically by checking the size of the screen.

This variable is read-only; you cannot reset its value. It is particularly useful in initialization files as well as in user-defined functions that work across all of Z-Mail's interfaces.

is_gui

READ-ONLY BOOLEAN

Z-Mail sets the *is_gui* variable to true if Z-Mail is in graphical user interface mode, even though the interface may not yet be started. You can test the value of *is_gui* in initialization files to decide what variables to set or what buttons to install to properly configure your Z-Mail session.

EXAMPLE For example:

```
if is_gui
        button Phone phone_msg
        button "Purch Req" purch_req
        button Room reserve_room
endif
```

This *if* statement checks if you are running in GUI mode and if so, it installs three buttons that run associated functions.

This variable is read-only; you cannot reset its value. It is particularly useful in initialization files as well as in user-defined functions that work across all of Z-Mail's interfaces.

is_sending

READ-ONLY BOOLEAN

Z-Mail sets the *is_sending* variable to true if you issue the *zmail* command from the UNIX command line to send a message. You can test the value of *is_sending* in scripts to determine whether to set specific variables. You can also skip setup procedures since you know that Z-Mail is only being used to send a message and that the Z-Mail interface will not be started.

This variable is read-only; you cannot reset its value. It is particularly useful in initialization files to determine if Z-Mail is running in send-only mode.

is_shell

READ-ONLY BOOLEAN

Z-Mail sets the variable *is_shell* true when it has started the Z-Mail shell. The value of *is_shell* is always *false* at startup when Z-Mail is reading initialization files. It is always *true* after startup. Therefore, *is_shell* is always true when you load initialization files manually using the *source* command.

This variable is read-only; you cannot reset its value. It is particularly useful in initialization files to determine if Z-Mail is running in send-only mode.

keepsave

BOOLEAN

If you set the *keepsave* variable, the *save* and *write* commands do not mark messages for deletion. Z-Mail checks the value of this variable at update time, not at the time the *save* or *write* is performed.

known_hosts

STRING

Z-Mail uses the *known_hosts* variable in conjunction with the *auto_route* variable. You set the *known_hosts* variable to a list of hosts for which your machine shares UUCP connections. You should separate the host names by commas or spaces.

When you reply to mail, the return path that Z-Mail constructs often has a number of hostnames through which it could route your reply. Instead, Z-Mail looks for hosts it knows it can contact directly and routes the messages to those machines.

EXAMPLE If you respond to mail that contains the path:

unicom!pixar!root

but you know your machine already calls *pixar*, then it is unnecessary to send the mail to *unicom* first. If you set your *known_hosts* string to include *pixar*, Z-Mail routes the message to *pixar* automatically:

pixar!root

See *Specifying E-Mail Addresses* beginning on page 13 for more information on addressing messages correctly.

logfile

STRING

Set the *logfile* variable to a filename in which Z-Mail logs the headers of outgoing messages. If you set the *logfile* variable on, but specify no filename, Z-Mail uses the *~/mail.log* file as the default log file.

To log entire messages including their body text, see the *record* variable. If you set both the *logfile* and the *record* variables, then the log file and the record files match closely. You can then use the log file to quickly scan for specific messages and then check the record file for its full text.

mail_icon

STRING

If you set the *mail_icon* variable to a pathname identifying an icon pixmap Z-Mail uses the pixmap as its desktop icon when you close (iconify) the main Z-Mail window. You may either change this variable in GUI mode and save the changes to your Z-Mail initialization file or you must set this variable explicitly in the initialization file. Once you have made the change, the next time you start Z-Mail, it will use the new icon.

mbox

STRING

Set the *mbox* variable to the pathname of the file you want Z-Mail to use as your default folder for saving mail. If you do not set the *mbox* variable, Z-Mail uses *~/mbox*.

metoo

BOOLEAN

Set the *metoo* variable to tell Z-Mail to keep you on the list of recipients when you reply to a message. By default, Z-Mail

deletes your name your name from the list of recipients. See the *alternates* command on page 90 for information on how Z-Mail determines whether you are on the list of recipients.

mil_time

BOOLEAN

Whenever Z-Mail displays time in a message header or in the prompt, it can display it in either 12-hour am/pm or 24 hour format. Set the *mil_time* variable on to tell Z-Mail to display the time in 24-hour (or military) format. Turn the variable off to display time in the default 12-hour format.

msg_win

INTEGER GUI ONLY

Set the *msg_win* variable to tell Z-Mail how many lines to display in the Compose Window in GUI mode.

newline

STRING COMMAND-LINE ONLY

Set the *newline* variable to a string so that Z-Mail executes as a command when you type a newline (**Return**) in command-line mode. The default behavior of Z-Mail is to give you a prompt whenever you type a newline.

Other mail user agents act as if you typed a *next* command, when you type a newline. If you are comfortable with an older mail user agent, you may wish to set Z-Mail to act the same way. You do this by setting the *newline* variable accordingly:

```
set newline = next
```

newmail_icon

STRING GUI ONLY

If you set the *newmail_icon* variable to a pathname identifying an icon pixmap Z-Mail uses the pixmap as its desktop icon when new mail has arrived and you have closed (iconified) the main Z-Mail window. You may either change this variable in GUI mode and save the changes to your Z-Mail initialization file or you must set this variable explicitly in the initialization file. Once you have made the change, the next time you start Z-Mail, it will use the new icon when new mail is waiting.

no_expand

BOOLEAN

If you set the *no_expand* variable, Z-Mail does not expand aliases, so message headers contain aliases exactly as you typed

them. By default, when you use a Z-Mail alias to refer a list of addresses, Z-Mail expands the list on the *To:* and *Cc:* lines to indicate the complete list of all the recipients of the message.

EXAMPLE If you have defined an alias named *all* to contain the addresses of all 250 people in your organization, you don't want Z-Mail to expand the alias so the *To:* header contains 250 addresses. If you set *no_expand* before sending the message, Z-Mail sends the following header on the message:

To: all

WARNING: The *no_expand* variable is intended as a tool for managing mailing lists, not for sending everyday person-to-person messages. When you set *no_expand*, the recipients of your outgoing messages receive the alias names you enter, just as you type them. If the recipients of your messages do not have access to the same alias definitions that you do, they are unable to construct return paths to reply to the people on the mailing list. Therefore, before you set *no_expand*, make sure that either you don't want people to be able to reply to others in your alias lists, or that everyone on the alias lists has access to their definitions.

no_hdrs

BOOLEAN

If you set the *no_hdrs* variable, Z-Mail does not include your personalized mail headers in messages. This does not unset your headers, it just disables them.

no_reverse

BOOLEAN FULLSCREEN ONLY

If you set the *no_reverse* variable, Z-Mail does not use reverse video in the fullscreen interface to indicate the current message. This variable has no affect in other Z-Mail interfaces.

nonobang

BOOLEAN

If you set the *nonobang* variable, Z-Mail avoids expanding history references that do not match anything, rather than generating error messages. This is useful if you want argument referencing in *cmd* expansions but do not want to escape every exclamation point (**!**) you type in UUCP addresses. You should set this variable in the fullscreen mode because Z-Mail does not keep history for line-mode escapes from this interface.

nosave

BOOLEAN

If you set the *nosave* variable, Z-Mail does *not* store a backup copy of terminated messages. When you compose a message, you can terminate it before sending it by using the **~q** tilde escape or by sending two *interrupt* signals.

Normally, when you terminate a message, Z-Mail saves a copy of it to the *dead.letter* file in your home directory or to the file described by the *dead* variable. The **~E** tilde escape command saves the current message you are editing in the dead letter file and clears the message buffer so you can compose a new message. For more information, see *Tilde-Escape Editing Commands* beginning on page 65.

output

READ-ONLY STRING

Z-Mail sets the *output* variable to a message list representing the output of your last successful command. This variable is most useful in line-mode and in scripts. You can use *output* to recover from broken pipes or to capture the output of a command without affecting the information it displays.

Commands which return an error code (see the *status* variable) do not affect the value of *output*. Many fullscreen mode commands return an error status to indicate the display has been altered, even if no error occurred. Successful commands that return no message list clear the value of *output*.

For more information on piping message lists, see the *input* variable on page 43.

pager

STRING FULLSCREEN & COMMAND-LINE ONLY

When Z-Mail displays a message longer than the number of lines you specify with the *crt* variable, Z-Mail executes the program you specify with the *pager* variable.

To tell Z-Mail to use its own internal pager, set the *pager* variable to *internal* or to a null string.

If you do not set the *pager* variable, Z-Mail checks your PAGER environment variable. If this isn't set, Z-Mail uses the default value for *pager* set by your system manager. If Z-Mail cannot find value to use for *pager*, it uses its internal pager.

Z-Mail checks the value of *pager* when you enter a **~p** tilde-escape command and at other times when Z-Mail is displaying lengthy text objects.

picky_mta

BOOLEAN

If you set the *picky_mta* variable on, Z-Mail does not pass *From:* and *Date:* headers to the mail transport agent when you are sending mail. You should set *picky_mta* on if your mail transport agent generates its own *From:* and *Date:* headers.

post_indent_str

STRING

If you set the *post_indent_str* variable and you include the body of a message into an outgoing message or reply, Z-Mail inserts a line of text after the last line of the included message. You can use any of the formatting strings listed in *Message Information Strings* on page 10.

EXAMPLE For example, you can set *post_indent_str* as follows:

```
set post_indent_str = 'End of message "%s" from %n. \n'
```

Now when you include a message in an outgoing message, Z-Mail follows the included message with your newly defined text string:

```
> . . .
> This is a test message to show how
> you can use post_indent_str.

End of message "Design meeting" from Dan Smith.
```

Z-Mail precedes each line of the included message with the **>** character unless you set the value of the *indent_str* variable to something else. For details, see the *indent_str* variable on page 42.

pre_indent_str

STRING

If you set the *pre_indent_str* variable and you include the body of a message into an outgoing message or reply, Z-Mail inserts a line of text before the first line of the included message. You can use any of the formatting strings listed in *Message Information Strings* on page 10.

EXAMPLE For example, you could set *pre_indent_str* as follows:

```
set pre_indent_str = 'Message “%s”, \n on %7d from %n: \n'
```

Now when you include a message in an outgoing message, Z-Mail precedes the included message with your newly defined text string:

```
Message "Design meeting",
on Jan 19 from Dan Smith:

> This is a test message to show how
> how you can use pre_indent_str.
> . . .
```

Z-Mail precedes each line of the included message with the **>** character unless you set the value of the *indent_str* variable to something else. For details, see the *indent_str* variable on page 42.

printer

STRING

Set the *printer* variable to tell Z-Mail which printer to use as the default printer when you use the *lpr* command.

print_cmd

STRING

Set the *print_cmd* variable to describe a UNIX command other than *lpr* to send messages to the printer.

EXAMPLE The following command sends all messages from the fifth message through the last message to the printer using the *ptroff* command:

```
set print_cmd = 'ptroff –ms –Plp'
lpr 5–$
```

prompt

STRING COMMAND_LINE ONLY

In the command-line interface, Z-Mail prompts you to tell you it is waiting for your next command. You can customize this prompt to tell you many different pieces of information.

When Z-Mail displays a prompt, it evaluates the *prompt* variable and echoes each character in the string unless it encounters a **%** character identifying the beginning of a folder formatting string. For a list of valid strings, see *Folder Information Strings* on page 11.

By default, the *prompt* variable is set to the string:

```
Msg %m of %t:
```

So if you have 10 messages and your current message is 5, then your Z-Mail prompt looks like:

```
Msg 5 of 10:
```

To include the values of variables in your prompt string, you can use the format **%$variable**, where *variable* is the name of any variable. If you do not precede the **$** character with a **%** character, Z-Mail treats the **$** as any other alphabetic character. Thus, two equivalent ways of including the name of the current folder in your prompt are:

```
set prompt = '%F> '
set prompt = '%$thisfolder> '
```

The single quotes prevent Z-Mail from expanding the *thisfolder* variable at the time you set the value of the *prompt* variable. Instead, Z-Mail evaluates *thisfolder* each time it displays the prompt. The only difference between these two examples is that *%F* adds the string [*read-only*] if Z-Mail opened the folder in read-only mode.

quiet

BOOLEAN / MULTIVALUE

The *quiet* variable tells Z-Mail to suppress warnings and user messages under specific circumstances. You may set *quiet* to one or more of these keywords separated by spaces or commas:

autosign	Suppress messages when appending signature.
complete	Suppress word completion error bells.
fortune	Suppress messages when appending fortune.
gui	Suppress bell indicating new mail in GUI interface.
newmail	Suppress new mail notification messages.
pick	Suppress descriptions of pick searches.
save	Suppress verbose descriptions of saving.
startup	Suppress the startup message.

If you set quiet on, but specify no list of keywords, it is the same as saying **set quiet = startup**, and Z-Mail suppresses its startup message.

The *newmail* setting does not prevent automatic inclusion of new mail, it only suppresses the indicators of its arrival (such as the bell Z-Mail sounds in the GUI interface).

Z-Mail always reports error conditions for signatures and fortunes. See the *autosign* and *fortune* variables for more details.

realname

STRING

You set the *realname* variable to your actual name. Z-Mail initializes *realname* to the value of the *NAME* environment variable when you start Z-Mail. If *NAME* is not set, Z-Mail looks in the password file, if available, for your name.

If you want to reset this variable after you have started Z-Mail, you can issue a command like this:

```
set realname = "Frank Glantz"
```

record

STRING

Set the *record* variable to the full pathname of a file where you want Z-Mail to record all outgoing mail. The pathname may begin with a plus sign (**+**), indicating your *~/Mail* directory (or the value of the *folder* variable), or with a tilde character (**~**) or *~user*, indicating your home directory.

The record variable tells Z-Mail to log entire messages including their body text. To log just the headers of outgoing messages, see the *logfile* variable. If you set both the *logfile* and the *record* variables, then the log file and the record files match closely. You can then use the log file to quickly scan for specific messages and then check the record file for its full text.

record_users

BOOLEAN

The *record_users* variable allows you to record all outgoing messages you send to a specific set of users. If you set the *record_users* variable, Z-Mail compares folders in the *folder* directory (*~/Mail* or the value of the *folder* variable) against the user name of each message recipient. If any folder matches a recipient's name, Z-Mail appends a copy of the outgoing message to that folder.

Note: The folder must exist before Z-Mail places a copy in it; Z-Mail never creates folders as a result of you setting the value of the *record_users* variable. To create a new folder, save a message to a folder whose name matches the username for which you want to record outgoing messages, or use a UNIX command such as *touch* to create the folder first.

WARNING: If you set *record_users* to a long list of usernames, it can cause Z-Mail to make many copies of outgoing messages. You should therefore use this variable with caution.

recursive

BOOLEAN

If you set the *recursive* variable, Z-Mail allows user-defined functions to recursively call themselves up to 100 times. If you unset *recursive*, Z-Mail does not allow functions to call themselves at all.

WARNING: If you set the *recursive* variable, make sure the functions you call have a way of terminating themselves.

redirect

READ-ONLY BOOLEAN SEND-ONLY MODE

Z-Mail sets *redirect* to *true* if it believes input to Z-Mail is redirected. Any time Z-Mail is running but it is not directly connected to a terminal, it believes that input has been redirected. If you specify the *–interact* option in your *zmail* command to explicitly identify an interactive session, Z-Mail sets *redirect* to false. This variable is read-only; you cannot reset its value, but it is useful for testing in initialization files.

reply_to_hdr

STRING

If you set the *reply_to_hdr* variable and provide a list headers (delimited by spaces or commas), then Z-Mail searches that list for return paths. If none of the headers listed in the *reply_to_hdr* variable exist in the message, then Z-Mail displays a warning message and searches the *Reply-to*, *From*, and *Return-path* headers, in that order.

If Z-Mail finds no return paths, it parses the first line of the message created by the mail transport agent. A special message header is created by most transport agents (*MMDF* does not). This header is called the *From_* header because it has no colon and contains the return address of the message's author. If Z-Mail cannot find the return path address in the message, and the mail transport agent inserted the special *From_* header, Z-Mail uses this as the return address.

EXAMPLE

You can specify the special case *From_* header as one of the headers to search for:

```
set reply_to_hdr = "reply-to return-path From_"
```

In this example, Z-Mail searches (in order) the headers listed in the *reply_to_hdr* variable. If Z-Mail does not find the first header, then it looks for the next in the list. If Z-Mail finds none of the headers, it searches the default headers — *Reply_to*, *From*, and *Return_path*.

For more information, see the *reply* command on page 129.

save_empty

BOOLEAN

If you set the *save_empty* variable, it prevents Z-Mail from deleting empty folders. Normally, when you delete all the messages in a folder and you close the folder or change to a new folder,
Z-Mail deletes the empty folder.

Note: Z-Mail never deletes the System Folder, even when it is empty, because many programs require a System Folder to be present in order to function properly.

screen

NUMERIC FULLSCREEN AND COMMAND-LINE ONLY

Set the *screen* variable to the number of message summaries you want Z-Mail to concurrently display in the command-line and fullscreen character interfaces.

screen_win

NUMERIC GUI ONLY

Set the *screen_win* variable to the number of message summaries you want Z-Mail to concurrently display in the GUI interface.

sendmail

STRING

Set the *sendmail* variable to the program and arguments you want Z-Mail to execute when it delivers messages to the mail transport agent. Normally, if the *sendmail* variable needs to be explicitly on your system, your system administrator will have previously set its value for you.

show_deleted

BOOLEAN

If you set the *show_deleted* variable on, Z-Mail lets you display deleted messages in command-line mode using any command that displays a message. Z-Mail also displays the headers of deleted messages with the *headers* command. In fullscreen and GUI mode, Z-Mail ignores *show_deleted* and displays deleted messages with other messages so that you can *undelete* them easily.

sort

BOOLEAN / STRING

Set the value of the *sort* variable to presort the messages in your System Folder when you first start Z-Mail. You set the variable to a string of arguments as specified in the *sort* command on page 139.

EXAMPLE The following command sorts your System Folder by reverse date and within date, by status:

```
set sort = " –r –d –S "
```

The next time you start Z-Mail, the initialization file sets the *sort* variable and presorts your System Folder automatically.

squeeze

BOOLEAN

If you set the *squeeze* variable, whenever you read, pipe or save messages, Z-Mail squeezes all consecutive blank lines into one blank line.

status

READ-ONLY INTEGER

The *status* variable reports the success or failure status of your most recent Z-Mail command. All line-mode commands return 0 (zero) for success and –1 (negative one) for error. Some fullscreen mode commands return an error status to indicate the display has changed, even if no real error occurred.

This variable is most useful in scripts to test the success of an operation before proceeding:

```
if $status < 0
        return
endif
```

templates

STRING

Set the *templates* variable to a list of directories, separated by spaces, where you store prepared message templates. Z-Mail searches these directories when you use *Templates* option of the *Compose* command in GUI mode or the *–p filename* option of the *mail* command.

If the same name appears in more than one directory, Z-Mail uses the file in the last directory. Z-Mail always searches the *forms* directory in the Z-Mail library first.

thisfolder

READ-ONLY STRING

Z-Mail sets the *thisfolder* variable to the full pathname of the active folder (current mailbox). You cannot modify or display this variable with the *set* command. Z-Mail modifies the value of *thisfolder* whenever you change active folders.

Note: Use caution when referring to *thisfolder* in initialization files since at startup, Z-Mail has not yet activated a folder and *thisfolder* does not yet have a value.

title

STRING GUI ONLY

The *title* variable describes the title you Want Z-Mail to display shown at the top of its Main Window in the GUI and fullscreen modes. You can specify any of valid folder formatting string. For a list of valid arguments, see *Folder Information Strings* on page 11.

tmpdir

STRING

Set the *tmpdir* variable to the pathname of the directory you want Z-Mail to use to write all Z-Mail temporary files. By default, Z-Mail uses your *home* directory. If Z-Mail cannot access your *tmpdir* or *home* directory, it uses a directory writable by all users, typically the */tmp* directory.

toplines

INTEGER FULLSCREEN AND COMMAND-LINE ONLY

The *toplines* variable tells Z-Mail the number of message lines to display when you issue the *top* command in character mode. If you do not assign a value to *toplines*, Z-Mail uses the value of the *crt* variable to determine the number of lines to display.

Z-Mail displays only message bodies when you use the *top* command. Z-Mail does not count message headers as lines since it does not display them.

unix

BOOLEAN FULLSCREEN AND COMMAND-LINE ONLY

If you set the *unix* variable on, Z-Mail treats all commands that are not Z-Mail commands as UNIX commands. If you set *unix*, you can execute UNIX commands simply by typing them instead of using shell escapes. This variable has an effect only in Z-Mail's character interface.

If a Z-Mail command name conflicts with a UNIX command, use the *sh* command to force Z-Mail to execute the command using the shell, or specify the full pathname of the command (e.g. starting with a forward slash **/**).

Note: Due to the lack of real job control, input/output redirection and UNIX command piping, this mode of Z-Mail is not intended to be used as a login shell. For systems that support job control (*SIGTSTP*)**,** the key sequence **Ctrl-Z** usually stops the entire Z-Mail shell as well as the process being executed. When *SIGCONT* (e.g. *fg*) is executed, both processes receive the signal and Z- Mail continues to wait for the job to finish.

WARNING: Z-Mail pipes transmit message lists, not text. You cannot pipe the output of UNIX commands to or from Z-Mail commands or other UNIX commands with the Z-Mail pipe mechanism. You can, however, pipe Z-Mail commands to a final UNIX command. See the *pipe* command for details.

verbose

BOOLEAN

If you set the *verbose* variable on, Z-Mail passes the *verbose* option to the mail transport agent when it sends messages. Setting *verbose* on also causes Z-Mail to display additional information about the *send* process.

verify

BOOLEAN / MULTIVALUE

The *verify* variable causes Z-Mail to request confirmation of certain actions. You can set *verify* to one or more of these keywords, separated by spaces or commas:

mail	Confirm sending of messages.
save	Confirm save-item selections (GUI only).
quit	Confirm folder update before exiting.

If you set *verify* on, but specify no keyword string, Z-Mail treats your request as **set verify = mail** and asks you for confirmation before sending mail.

Z-Mail asks for verification whenever it appends messages to files that are not folders, regardless of the *verify* variable setting.

version

READ-ONLY STRING

Z-Mail stores its current version number in the *version* variable. Z-Mail displays this information when it starts unless you suppress its display by setting the *quiet* variable. Z-Mail also places current version information in the *X-Mailer:* header in outgoing messages.

visual

STRING

Set the *visual* variable to the name of the text editor you want to use. If you set *visual*, but do not specify the name of an editor, Z-Mail uses *vi* or the value of the *VISUAL* environment string.

To make Z-Mail start the visual editor every time you compose or reply to a message, set the *autoedit* variable. Otherwise, you must manually invoke the visual editor by typing **~v** while editing a message.

warning

BOOLEAN

If you set the *warning* variable, Z-Mail displays warning messages under a variety of situations detailed below.

A command line alias (cmd) looks like a command.
For example:

```
cmd mail 'set fortune; \mail'
cmd respond 'unset fortune; \respond;'
```

Z-Mail displays the warning when you use the *cmd* alias, not when you define it.

The date format of a message is unknown.
Z-Mail takes the date of a message from the *Date:* header. If the date on that header is unknown, Z-Mail searches other headers for a valid date format until a legal one is found. Z-Mail displays a warning when it finds no date or an incomprehensible date.

A variable is unset without first being set.
For example, if you give the command:

```
unset metoo
```

and the *metoo* variable is not set, you are notified that the variable is not defined.

No header can be found for a digest article.
This occurs when the *undigest* command encounters what appears to be an article separator but cannot find a *From:* or *Date:* header in the subsequent text.

The purpose of the *warning* variable is to provide users who are comfortable with their own environments a way of detecting changes in other environments, should they be forced to use them. In an integrated environment, you can also receive messages about other processes failing that are not severe enough to stop Z-Mail from functioning properly.

window_shell

STRING GUI ONLY

Set the *window_shell* variable to the name of the program to run when you create shell processes in GUI mode. The default value for *window_shell* is *xterm –e*.

wrap

BOOLEAN

If you set the *wrap* variable, when you move beyond the last message in the active folder using the *next* or *delete* commands, Z-Mail moves the pointer to the first undeleted message in the folder. Normally, when you delete the last message, the message pointer remains at the last message and you are done reviewing your mail.

wrapcolumn

NUMERIC

Set *wrapcolumn* to a column number where you want Z-Mail to automatically move to the next line as you compose messages. If you set wrapcolumn on, but specify no column number Z-Mail uses column 78 as the default. If Z-Mail can determine the number of columns on your screen, it enforces a maximum value for *wrapcolumn* of two less than that number of columns. You can disable line wrapping either by unsetting *wrapcolumn* or by setting it with a value of 0 (zero).

Line wrapping occurs only at white space (spaces or tabs). Z-Mail does not wrap dense lines containing more characters than the value of wrapcolumn with no white space.

Note: If you start Z-Mail with the *–e* (echo) option, Z-Mail cannot perform line wrapping due to I/O incompatibilities.

$$ *READ-ONLY BOOLEAN*

This *$$* variable returns the process id (*PID*) of the current Z-Mail process. Z-Mail does not recognize colon modifiers for this special variable.

EXAMPLE The following command displays the process number of your current Z-Mail process:

```
echo $$
10326
```

$* *READ-ONLY STRING FUNCTIONS ONLY*

When Z-Mail processes a user-defined function, it sets the *$** variable to the entire list of arguments you specified on the command line that called the function. You can use this variable in conjunction with the *shift* command to pick off arguments as you need them. If you know the order in which the arguments were specified, you can use the variables *$1*, *$2*, etc.

EXAMPLE Assume you invoked the function *treport* with the following command indicating the recipients of the report as arguments:

```
treport kwilson pramirez dfrank
```

Z-Mail assigns all the arguments you specified to the *$** variable as one character string.

```
echo $*
kwilson pramirez dfrank
```

For more information, see the *shift* command on page 137 and the *$#* and *$1, $2, ...* variables.

$# *READ-ONLY NUMERIC FUNCTIONS ONLY*

When Z-Mail processes a script, it sets the *$#* variable to the number of arguments you specified on the command line that called the function. You can use this variable in conjunction with the *shift* command or other variables such as *$** and *$1*, *$2*, etc.

EXAMPLE Assume you invoked the function *treport* with the following command:

```
treport kwilson pramirez dfrank
```

Z-Mail assigns the number of arguments to the *$#* variable.

```
echo $# $*
3 kwilson pramirez dfrank
```

For more information, see the *shift* command on page 137 and the *$** and *$1, $2, ...* variables.

$0

READ-ONLY STRING FUNCTIONS ONLY

When Z-Mail processes a script, it sets the *$0* variable to the name of the function.

EXAMPLE Assume you invoked the function *treport* with the following command:

```
treport kwilson pramirez dfrank
```

Z-Mail assigns the name of the function to the *$0* variable.

```
echo $0
treport
```

$1, $2, . . .

READ-ONLY STRINGS FUNCTIONS ONLY

When Z-Mail processes a script, it sets a series of variables named *$1, $2, ... $n* equal to the individual arguments you specified on the command line that called the function. You can use this variable in conjunction with the $# variable whose value is set to the number of arguments you specified.

EXAMPLE Assume you invoked the function *treport* with the following command indicating the recipients of the report as arguments:

```
treport kwilson pramirez dfrank
```

Z-Mail assigns all the arguments you specified to the *$1, $2* and *$3* variables, and the number of arguments (three) to the *$#* variable.

```
echo $3 $#
dfrank 3
```

For more information, see the *$#* variable.

CHAPTER 7

Integrating Z-Mail with Other Products

You can use Z-Mail to build a desktop environment that integrates the office automation and productivity applications you use regularly.

Z-Mail's extensive capabilities allow you to import text, send documents and application files, and even launch third-party products without ever leaving Z-Mail.

You can use Z-Mail's Motif, Open Look and fullscreen interfaces to construct simple, straightforward user desktops. And after you get used to Z-Mail's scripting capabilities, you'll find its open architecture and scripting language invaluable in building powerful messaging environments that allow you to exchange electronic information not only with other people, but also with databases and applications.

Importing Text Files into Messages

You may use text editors, word processors, desktop publishing applications or other products that can export text files. Z-Mail can read these text files directly into messages.

Including the text of a file is not the same as attaching a document to an outgoing mail message. When you include a text file, Z-Mail treats the contents of the file as if it were text typed by you into the body of the message. Once you include a text file, you can edit its contents. An attachment, on the other hand, is a self-contained document that you cannot edit as the body of the message.

The following paragraphs explain how to include text from an external file using Z-Mail's various interfaces.

GUI When you are composing a message using the GUI interface, type in the filename whose text you want to include in your outgoing message. Then pull down the *File* menu and select the *Open / Insert Text* option.

Character When you are composing a message using either the fullscreen or command-line character interfaces, if you type **~r filename** at the beginning of a line, Z-Mail reads the contents of *filename* into your outgoing message.

Send-Only Mode If you use the *zmail* command in send-only mode, you can specify the *–draft* option to send an entire file, including all the necessary message headers for delivering the message, plus the text of the message itself. The message headers may not contain any blank lines, but should be separated from the text of the message by a blank line. Z-Mail adds a new *Date:* and a *From:* header if there is none present. If there is no *To:* header, Z-Mail does not send the draft.

The following command sends the contents of the file *textreport* as a mail message:

```
zmail –draft textreport
```

For more information, see the *–draft* option on page 24.

Defining Attachment Types

Z-Mail allows you to attach files and documents to your messages. You can specify any type of attachment including binary, text, video, audio, graphics, Postscript, even application files such as FrameMaker and Wingz documents.

To learn more about attaching files and documents using Z-Mail's graphical user interface, see the *Z-Mail User's Guide*. In Z-Mail's character interfaces, use the *–attach* option (*–A*) of Z-Mail's *mail* command.

The Attachment Types File

Before you can attach a file or document to your message, Z-Mail must first tell Z-Mail how to recognize and handle that type of attachment. Z-Mail comes pre-configured to recognize a variety of documents including text, graphics, bitmaps, Wingz, FrameMaker, PostScript and other files.

You may want to define your own types of attachments so Z-Mail can recognize, display and send those attachments automatically. You define attachments by making new entries in the *attach.types* file located in the */usr/lib/Zmail* directory.

Normally, you only have one *attach.types* file that contains definitions for all your attachment types. You may, however, choose to create several *attach.types* files that you use in different Z-Mail applications or user environments. You tell Z-Mail which attachment definitions file to use by setting the value of the *attach_types* variable.

File Format

The *attach.types* file contains a series of special statements that define:

- The attachment types that Z-Mail recognizes,
- The programs Z-Mail uses to display the attachments,
- The programs Z-Mail uses to encode and compress attachments before sending them,
- The programs Z-Mail uses to decode and uncompress attachments it receives, and
- The directories to search to find programs.

The general format of the *attach.types* file is:

```
PATH      directory1:directory2:...
. . .
TYPE      attachtype1    "viewerprog1"  "editorprog1"  "comment"
TYPE      attachtype2    "viewerprog2"  "editorprog2"  "comment"
. . .
CODE      codetype1      "encodeprog1"  "decodeprog1"  "comment"
CODE      codetype2      "encodeprog2"  "decodeprog2"  "comment"
. . .
ENCODE    attachtype1    codetype1
ENCODE    attachtype2    codetype2
. . .
DEFAULT   codetype
```

You may continue an entry over more than one line by using a backslash at the end of the preceding line.

EXAMPLES The following example shows a typical *attach.types* file:

```
PATH      /usr/lib/Zmail/bin: /bin:/usr/bin: /usr/local/bin
TYPE      Wingz      "Wingz %s"       None             "Wingz"
TYPE      bitmap     "bitmap %s"      None             "X11 bitmap"
TYPE      xwd        "xwud -in %s"    "xwd -out %s"    "Xwd File"
CODE      compress   "compress | btoa |"  "atob | uncompress |"  \
          "Compressed data"
ENCODE    Binary     compress
```

You can also place a comment at the end of a line or on a line by itself by preceding the comment with a **#** character.

```
#  The following lines define Z-Mail attachment types:
TYPE     Wingz     "Wingz %s"     None    "Wingz"
TYPE     bitmap    "bitmap %s"    None    "X11 bitmap"
```

PATH Statement

The *PATH* statement designates directories where you want Z-Mail to look to find the programs for viewing and encoding attachments.

SYNTAX

```
PATH  directory1:directory2:directory3 . . .
```

You can specify more than one *PATH* statement. If you do, Z-Mail appends the directories you specify in each *PATH* statement to the values you specified in the previous statements. Z-Mail adds the final *PATH* variable value to your operating system path before it processes any attachments. Specify each path, separated by a colon with no blank spaces.

Note: Z-Mail always searches the path */usr/lib/Zmail/bin*, so you do not need to specify it in your PATH statement.

CODE Statement

The *CODE* statement tells Z-Mail the types of encoding and decoding processes you want to use when for processing attachments.

SYNTAX

CODE *codetype* "*encodeprog*" "*decodeprog*" "*comment*"

Coding Type (codetype)
Indicate the name of the coding type as a quoted text string. The *value* cannot include spaces or tabs, but can include non-alphabetic characters.

Encoding Program (encodeprog)
Indicate the name of the program Z-Mail runs to encode attachments of type *codetype*.

Decoding Program (decodeprog)
Indicate the name of the program Z-Mail runs to decode attachments of type *codetype*.

Comment (comment)
Add any comment explaining the coding type as a quoted text string.

The text strings you specify for *encodeprog* and *decodeprog* are treated by Z-Mail as Bourne shell (*/bin/sh*) commands. By enclosing them in quotes, you protect spaces, pipe symbols, etc.

If *encodeprog* or *decodeprog* begin with a pipe character (|), Z-Mail uses the UNIX standard input pipe to find data for the program you indicate in the string. If *encodeprog* or *decodeprog* end with a pipe character, Z-Mail directs the output of the program to the UNIX standard output pipe.

If *encodeprog* or *decodeprog* contain **%s**, Z-Mail inserts a filename at that point so *encodeprog* or *decodeprog* may read data from that file.

You must specify the *CODE* statement with one of the following:

- leading and trailing pipe characters,
- a **%s** and one pipe character, or
- a single **%s**.

Encoding programs must produce seven-bit ASCII text as output. Otherwise the mail transport agent cannot process the message properly.

You can specify as many CODE statements in your *attach.types* file as you need.

TYPE Statement

The *TYPE* statement defines the attachment types that Z-Mail recognizes.

SYNTAX

TYPE ***attachtype*** **"*viewerprog*" "*editorprog*" "*comment*"**

Attachment Type (attachtype)
Indicate the name of the attachment type as a quoted text string. The value of *attachtype* appears as a menu choice in the *Attachments* dialog box in the GUI mode or as a menu selection in the character mode of Z-Mail.The *value* cannot include spaces or tabs, but can include non-alphabetic characters.

Viewer Program (viewerprog)
Enter a quoted text string indicating the name of the program Z-Mail runs to display the attachment.

Editor Program (editorprog)
Enter a quoted text string indicating the name of the program Z-Mail runs to start an environment where you can view and modify the attachment.

Comment (comment)
Add any comment explaining the attachment type as a quoted text string.

The text strings you specify for *viewerprog* and *editorprog* are treated by Z-Mail as Bourne shell (*/bin/sh*) commands. By enclosing them in quotes, you protect spaces, pipe symbols, etc.

If *viewerprog* or *editorprog* begin with a pipe character (|), Z-Mail gets the data from the standard UNIX input pipe and puts the data on the standard input of the viewer or editor. You should never specify *viewerprog* or *editorprog* with a trailing pipe character. You must specify one of the following:

- a leading pipe character (|),
- one **%s** somewhere in the string, or
- neither of the above.

If you specify *viewerprog* or *editorprog* without a leading pipe character or **%s** formatting string, Z-Mail launches the programs indicated by the strings directly without specifying input files.

You may specify an ampersand character **(&**) at the end of *viewerprog* or *editorprog* to tell Z-Mail to run the editor or viewer in background mode. This is only useful for GUI-oriented programs (e.g., FrameMaker and Wingz), or those programs that do not produce output to the screen (e.g., post-processors).

ENCODE Statement

The *ENCODE* statement tell Z-Mail to use specific encoding methods for specific attachment types.

SYNTAX **ENCODE *attachtype codetype***

When you are sending mail, the *ENCODE* statement tells Z-Mail to automatically encode a document of type *attachtype* using the program specified by *encodetype*. You can always manually override the encoding indicated in the *ENCODE* statement during your Z-Mail session. When Z-Mail encodes outgoing messages, it adds two headers to them:

- *X-Zm-Data-Type:* holds the value of *attachtype*.
- *X-Zm-Encoding-Algorithm:* holds the value of *encodetype*.

You can specify as many ENCODE statements in your *attach.types* file as you need.

DEFAULT Statement

The *DEFAULT* statement tells Z-Mail which programs it should use by default for encoding and decoding attachments.

SYNTAX **DEFAULT *codetype***

If Z-Mail determines that an attachment needs to be encoded before Z-Mail sends it, but you have not specified the type of encoding to use, Z-Mail encodes the attachment use the program indicated *codetype* in the corresponding *CODE* statement.

Launching Applications from Z-Mail

You can launch other applications as well as run UNIX commands from within Z-Mail. While there are many reasons for doing this, a few situations occur often:

- You want to create a document using another application that you want to send using Z-Mail.
- You receive an application file as an attachment, and you want to start that application to display or work with the document.
- You want to run a UNIX command or utility.

Entering UNIX Commands from Z-Mail

You can issue a command to start another application or UNIX utility from within Z-Mail using a variety of methods that vary depending on your Z-Mail interface.

GUI Type **sh** followed by the command in the Command type-in field at the bottom of Z-Mail's main window. (e.g., **sh maker**)

Fullscreen Type an exclamation point (**!**) followed by the command. (e.g., **!maker**)

Command Line Type **sh** followed by the command at Z-Mail's command prompt. (e.g., **sh maker**)

Functions You can also include UNIX commands in Z-Mail functions by preceding them with the **sh** command.

Launching Applications from GUI Buttons

GUI To create a function and GUI button that launches applications, follow these steps:

1. Select the *Buttons* option from the *Options* pull-down menu, and the *Buttons* dialog box appears.
2. Click on the *Clear* button at the bottom of the dialog box.
3. Enter the name of the button in the *Button:* type-in field. Make sure you enter the name exactly as you want it to appear in Z-Mail's main window.

4. Enter a name for the function you are about to create in the *Function:* type-in field. Choose any unique name you want, avoiding the names of Z-Mail and UNIX commands.
5. In the large type-in box labeled *Function Text:* in the lower part of the dialog box, enter **sh** followed by a space and the command that starts the application you want to launch. (e.g., **sh maker**)
6. Click on the *Install* button and Z-Mail creates a GUI button in its main window.
7. If you want to save this button so it appears every time you start Z-Mail, click on the *Save* button at the bottom of the dialog box to save your current settings in your personal initialization file. Z-Mail displays a dialog box that prompts you for a filename. To indicate your default personal initialization file, *.zmailrc*, just click on *OK* without typing in another name. Z-Mail may ask you to confirm that you want to overwrite your *.zmailrc* file.
8. Click on the *Done* button, and Z-Mail returns you to its main window.
9. Click on the button you just created to start the application.

Automatically Processing Attachments

Integrated User Desktops

By specifying the appropriate program entries in the TYPE statements of your *attach.types* file, you build your own integrated user desktops. For example, you can create:

- A purchase requisition form in a Wingz spreadsheet,
- A function called *purchase_req* that starts Wingz, loads the purchase req spreadsheet, and waits for users to press a button indicating they are finished with the form.
- A GUI button labeled *Purchase* that runs the *purchase_req* function.
- An *attach.types* entry that automatically starts Wingz when Z-Mail receives a Wingz document.

Using these system components, you can build a purchasing system that all the e-mail users in your organization can use. They fill out purchase reqs electronically and route them to appropriate parties for approval. When the final approval is in place, Z-Mail can submit the purchase reqs to a central purchasing agent or even a Z-Mail userid that processes the forms and updates the purchasing database.

INDEX

Symbols

! character, in addresses 13
$ character
    as text metacharacter 11
    in message lists 10
$# variable 206
$$ variable 206
*$** variable 206
$0 variable 207
$1, $2, ... variables 207
% character
    in addresses 14
    in information strings 155
( ) characters, as variable notation 156
***** character
    as filename metacharacter 12
    as text metacharacter 11
    in message lists 10
+ character
    as filename metacharacter 12
    on suspended messages 118
. character
    as text metacharacter 11
    in message lists 10
: character, switching interfaces with 46
; character, as command terminator 58
? character
    as filename metacharacter 12
    as help command 9
@ character, in addresses 13
[ ] characters
    as filename metacharacters 12
    as text metacharacters 11
    as variable notation 155
\ character
    as filename metacharacter 12
    as indicator for control keys 77
    as text metacharacter 11
^ character
    as control character 63
    as text metacharacter 11
    in message lists 10
~ character
    as command prefix 65
    as filename metacharacter 12

A

active folder
    described 7
    name of as window title 202
    thisfolder variable 202
addresses
    Arpanet 13
    case insensitivity in 14
    comments in 14, 15
    described 6
    domain 13
    fully qualified 13
    local 13
    remote 13
    removing redundant 171
    RFC822 13
    UUCP 13
address-related variables 163
alias command 89
aliases
    command 57, 96
    defining 89
    deleting address 142
    deleting command 143
    disabling expansion of 192
    expanding 103
    expanding circular definitions 103
    in command-line interface 54
    specifying 15
alternate logins 90

alternates command 90
alts command - see *alternates*
alwaysignore variable 168
applications, launching 216
arguments
    echoing 101
    parsing 102
    parsing in function 137
    shifting 137
    string variable holding all 206
    string variables for individual 207
    variable indicating number of 206, 207
Arpanet addresses 13
ask command 91
ask variable 168
askcc variable 169
asking a question 91
asterisk
    as filename metacharacter 12
    as text metacharacter 11
    in message lists 10
atob binary decompression 212
at-sign (@), in addresses 13
attach.types file 211
attach_types variable 169
attachments
    binary files as 22
    compressing 211
    creating 122
    decoding 98
    detaching to files 98
    displaying 98
    encoding 211
    in command-line interface 53
    location of *attach.types* file 169
    specifying types of 211
auto_route variable 171
autodismiss variable 169
autoedit variable 170
autoiconify variable 170
autoinclude variable 170
autoprint 170
autosign variable 173
autosign2 variable 174
await command 92

B

backslash character
    as filename metacharacter 12
    as indicator for control keys 77
    as text metacharacter 11
Bcc - see blind carbon copies
Berkeley *mail* 45
binary files
    attaching 22
bind command
    described 93
    using the 78
bindings, listing fullscreen key 46
bind-macro command 75, 94
blank lines, squeezing out 201
blind carbon copies 120
body of message, described 6
Bourne shell, invoking the 137
braces in message lists 10
brackets
    as filename metacharacter 12
    as text metacharacter 11
    as variable notation 155
btoa binary compression 212
builtin command 94
button command 95
buttons
    creating 95, 216
    deleting 142
    displaying dialog boxes using 99

C

canceling messages
    in command-line interface 50
carbon copies
    prompting for recipients of 169
    sending to copies to files 181
    sending with *mail* command 120
caret character
    as control character 63
    as text metacharacter 11
    in message lists 10
case insensitivity in addresses 14
Cc - see carbon copies
cd command 96
cdpath variable 176
character interfaces
    starting the 46
    switching between 46

circumflex
as control character 63
as text metacharacter 11
in message lists 10
close command 96
closing folders 96
closing Z-Mail window 113
cmd command 96
cmd_help variable 176
co command - see *copy*
CODE statement 213
colon character
switching character interfaces with 46
command aliases
defining 96
deleting 143
using in command-line interface 57
command history, explained 56
command-line interface
appearance of 47
canceling messages 50
command aliases 96
command history 56, 113
command pipelines 57
customizing your prompt 55
deleting macros 145
deleting messages 53
displaying headers 48
displaying messages 48
forwarding messages 50
including text in messages 50
learning the 3
macros 122
printing messages 54
replying to messages 49
saving messages 54
searching messages 55
sending attachments in 53
sending messages 49
similarities to *csh, Mail, mailx* 45
sorting messages 55
starting the 8, 46
switching to fullscreen 46
using aliases 54
command-line macros 79
command-line-related variables 165
commands
echoing 101
entering multiple 58
how Z-Mail processes 59
list of fullscreen 61–62
pipelines 57
reference 89–147
referencing history of 186
referring to previous 56
running UNIX 202
semicolon as terminator for 58
summary of 84–88
tilde-escape 65–74
using with loops 107
comments
in addresses 14, 15
in initialization files 41
complete variable
described 177
use of 12
completion, filename 12, 177, 182
Compose Window
setting number of lines in 192
compose-related variables 162
composing messages
ending a compose with a dot 180
quitting while 129
suspending messages 118
wrapping at end of column 205
compressing attachments 211
conditionals
if...else...endif command 113–117
in initialization files 42
configuration files - see initialization files
configuration-related variables 158
confirming actions with *verify* 203
... continue ... prompt 63
control characters in macros 77
control-D, exiting Z-Mail with 187
copy command 97
copying messages to files 97
crt variable 178
crt_win variable 178
csh - see *C-shell*
C-shell 45
curly braces in message lists 10
curses library 64
cwd variable 178

D

d command - see *delete*
databases
    using Z-Mail as front-end for 217
date
    listing messages by receive 178
    searching message by 134
date_received variable 178
dead letter file 68, 194
dead variable 179
dead.letter file 179
DEFAULT statement 215
delete command 97
delete_save variable 179
deleted messages, displaying 200
deleting messages
    after saving 179
    in command-line interface 53
    with *delete* command 97
desktops, integrated user 217
detach command 98
detaching attachments 98
dialog boxes
    automatically closing 169
    displaying 99
    error 102
dialog command 99
digests
    bursting into messages 143
    merging into active folder 143
directory
    changing your current working 178
    changing your working 96
    defining your home 186
    displaying working 128
    for templates 201
    for temporary files 202
directory-related variables 161
display command 99
displaying headers
    in command-line interface 48
displaying messages
    after deleting messages 170
    displaying deleted 200
    in command-line interface 48
    number of lines 178
    top lines 141, 202
display-related variables 160
distribution lists - see aliases
documentation
    list of available Z-Mail 2
    obtaining 2
dollar sign
    as text metacharacter 11
    in message lists 10
domain addresses 13
domain_route variable 179
dot variable 180
draft files 122

E

each command 100
echo command 101
echo, setting local 25
edit command 101
edit_hdrs variable 180
editor variable 181
editor-related variables 162
editors
    editing headers with 180
    invoking with *edit* command 101
    invoking with tilde-escape 67
    setting visual 204
    using external 170, 181
electronic mail, described 1
emacs editor
    starting with a macro 80
e-mail, see electronic mail
empty folders, opening 33
ENCODE statement 215
encoding attachments 211
error bells, suppressing 197
error command 102
error status codes 201
escape characters, in messages 182
escape commands - see tilde-escape commands
escape variable 182
eval command 102
exclamation point
    as history reference 187
    in addresses 13
exit command 103
exiting
    from the fullscreen interface 63
    from Z-Mail without updating 103
expand command 103
expanding aliases 192
expansions in macros, described 75

F

f command - see *from*
Fcc - see file carbon copy 181
fg command - see *resume*
fignore variable 182
file carbon copies, sending 181
filename completion 12, 182
file-related variables 161
files
    copying messages to 97
    listing 119
    reading commands from 141
    saving messages to 147
    writing messages to 147
filter command 103
filters
    avoiding filters at startup 104
    avoiding when opening folder 104
    creating 103
    deleting 144
    restricted commands in 104
    running at startup 26
    suppressing running of 31
flags command 105
fo command - see *folder*
folder command 105
folder information variables 156
folder variable 183
folder-related variables 160
folders
    changing 105
    closing 96
    deleting when empty 200
    described 7
    listing all in a directory 106
    listing open 105
    merging 124
    name of active 202
    name of as window title 202
    opening 105
    opening empty system 33
    opening multiple 7
    opening read-only at startup 32
    overwriting when saving messages 132
    read-only 106
    saving empty 200
    setting path for 183
    specifying active at startup 26
    updating 147
folders command 106
foreach command 107
fortunates variable 183
fortune variable 183
fortunes
    adding to messages 120
    appending to messages 183
    specifying recipients of 183
forwarding messages 121
    in command-line interface 50
FrameMaker files 53, 70, 71, 211, 214
from command 108
fullscreen command 109
fullscreen interface
    activating 109
    appearance of 60
    avoiding problems in 64
    buttons as menu entries 95
    de-activating 109
    entering Z-Script commands in the 46
    exiting 63
    functionality not supported in 61
    functions 78
    how commands operate in 63
    is_fullscreen variable 189
    key bindings 93, 142
    learning the 3
    list of commands 61–62
    listing key bindings for the 46
    macros for 78, 94
    nested sorting in 140
    starting the 8, 46
    switching to command-line 46
fullscreen macros 78
fullscreen_help variable 184
fullscreen-related variables 165
function command 109
function variables 206, 207
function variables, described 150
functions
    assigning to buttons 95
    creating 109, 216
    deleting 144
    example of 110
    listing 109
    parsing arguments in 137
    putting comments in 109
    recursive 94, 198
    saveit, as an example 110
    using fullscreen 78

using tabs in 109
using with loops 107
with same name as commands 94

G

getline function 77
getstr function 77
GUI buttons, creating 216
GUI interface
buttons 95, 142
closing Z-Mail windows 113
dialog boxes 99
error dialog boxes 102
iconifying Z-Mail in 113, 141
is_gui variable 189
starting the 8
gui_help variable 184
GUI-related variables 164

H

:h variable modifier 154
h command - see *headers*
h+ and *h-* commands 112
hangup variable 184
hdr_format variable 185
hdrs_only variable 186
head of path variable modifier 154
header-related variables 159
headers
avoiding editing 101
creating 125
creating personalized 125
customizing format of 185
deleting personalized 145
described 6
disabling personalized 193
displaying 108, 111, 131
displaying in command-line 48
editing 180
ignoring 118, 145, 168
in send-only mode 28
in_reply_to 188
logging outgoing 191
modifying with tilde-escapes 65
retaining 131
searching 108
headers command 111
help
context-sensitive 9
requesting 9
help command 112
help files
locating 112
name of fullscreen 184
name of general 176
name of GUI 184
help-related variables 158
history command 56, 113
history of commands entered 56
history references, expanding 193
history variable 186
hold variable 186
home directory, defining your 186
home variable 186
hostname variable 187

I

iconify command 113
iconifying dialog boxes automatically 170
icons
new mail 192
Z-Mail GUI 191
if statements
in initialization files 42
if...else...endif command 113–117
ignore command 118
ignore_bang variable 187
ignoreeof variable 187
ignoring headers 145
always ignoring selected 168
when displaying messages 99
importing text files into messages 122, 210
in_reply_to variable 188
including messages 187, 195
incomplete messages
described 179
resuming 130
indent_str variable 187
indenting included messages
described 121
indent string for 187
initialization files
adding commands to 41
adding comments to 41
bypassing 186
conditional statements in 42

creating 40
creating with *saveopts* command 133
editing 41
-init option at startup 29
loading manually 38
order of loading 37
sourcing at startup 34
storing variable values in 153
suppressing loading of 29, 31
initialization variables
hdrs_only 186
is_fullscreen 189
is_gui 189
is_sending 190
is_shell 190
redirect 199
initializing Z-Mail 37
input variable 188
integrated user desktops 217
interfaces
starting Z-Mail 8
switching character 46
internal variables, described 150
Internet 6
is_fullscreen variable 189
is_gui variable 189
is_sending variable 190
is_shell variable 190

J

jobs command 118

K

keepsave variable 190
key bindings
defining 93
deleting 142
listing fullscreen 46
key sequences in macros, described 75
known_hosts variable 190

L

launching applications 216
library, location of Z-Mail program 30
license, registering your Z-Mail 32
lines
number displayed 178
specifying width of 205
listing UNIX files 119
literal-next character
backslash as 11, 12
in macros 76
logfile variable 191
logging messages
by user 198
entire message 198
headers only 191
logins, alternate 90
loops - see *each* and *foreach*
lpr command 119
ls command 119

M

m command - see *mail*
macros
binding 94
bind-macro command 75
command-line 79
creating 122
deleting 81, 145
disabled by *-echo* option 75
errors in 76
expanding 75
expansions 75
fullscreen 78
invoking tilde-escapes with 80
key sequences 75
listing 122
map command 75, 122
mixed mode 80
prompting users from 77
recursive properties 76
specify control characters in 77
mail command 120
mail transport agent
described 6
passing *From* and *Date* headers to 32, 195
passing verbose option to 36
sending files directly to 24
sending message to 200
verbose option 203
mail user agent 6
mail_icon variable 191
mailing lists - see aliases
MAILRC environmental variable 37
mailx program 45

Main Mailbox
described 7
filename of (*mbox*) 191
map command 75, 79, 122
mark command 123
mbox variable 191
merge command 124
merging folders 124
message information variables 155
message lists 47
as input to current command 188
from most recent command 194
parsing 124
piping to UNIX commands 126
specifying 10
using metacharacters to specify 10
message summaries
number in character interface 200
number in GUI interface 200
message templates, directories for 201
messages
appending fortunes to 183
assigning priorities 123
attachments 122
clearing temporary priorities 146
copying to files 97
creating headers 125
deleting 97
displaying 99
displaying in command-line 48
displaying next 126
displaying previous 128
displaying top lines of 141
editing 101
flag bits 105
forwarding 121
importing text files 122
incomplete 179
indenting included 121, 187
merging 124
overriding signatures 122
paging in character mode 194
printing 119
replying to all recipients 130
replying to sender 129
resuming incomplete 130
saving 132
searching 133
sending draft files as 122
sending in command-line 49
sorting 139
suspending composed 118
undeleting 143
unmarking 146
using templates to create 122
metacharacters
filename pattern matching using 12
specifying message lists with 10
text pattern matching using 11
metoo variable 191
mil_time variable 192
military time 192
modifiers, variable 154
msg_list command 124
msg_win variable 192
MTA - see mail transport agent
MUA - see mail user agent
Mush 112, 140
.mushrc file 37
my_hdr command 125

N

n command - see *next*
name, setting your real 197
negation in message lists 10
new mail
checking for 35
filtering 103
waiting for 92
newline variable 192
newmail_icon variable 192
next command 126
no_expand variable 192
no_hdrs variable 193
no_reverse variable 193
nonobang variable 193
nosave variable 194

O

O'Reilly & Associates 2, 14
output variable 194

P

p command - see *print* and *display*
pager variable 194
pager, lines displayed by 178
paging messages 194
parentheses as variable notation 156

parsing
arguments 102
message lists 124
PATH statement 212
pattern matching
in filenames 12
in text 11
percent sign
in addresses 14
in information strings 155
period
as text metacharacter 11
in message lists 10
pick command - see *search*
picky_mta variable 195
pipe command 126
pipelines
command, described 57
UNIX commands in 58
piping
message lists to UNIX commands 126
only message text 127
plus character
as filename metacharacter 12
post_indent_str variable 195
postmaster, reading mail for 36
PostScript files 211
pre command - see *preserve*
pre_indent_str variable 195
preserve command 127
previous command 128
print command - see *display*
print program, specifying name of 196
print_cmd variable 196
printer variable 196
printer, setting the 196
printing messages
in command-line interface 54
specifying print program 196
using *lpr* command 119
print-related variables 162
priorities
assigning 123
removing 123
process id of Z-Mail 206
program library, location of Z-Mail 30
programming-related variables 166
prompt
customizing your command-line 55
prompt variable 196
prompt, customizing your command-line 196
prompting
with a question 91
within macros 77
purchasing system
using Z-Mail to build 217
pwd command 128

Q

q command - see *quit*
question mark
as filename metacharacter 12
as help command 9
question, asking a 91
quiet variable 197
quit command 128
quitting Z-Mail 128

R

r command - see reply
R command - see *replyall*
Read Window, size of 178
read-only folders
opening at startup 32
opening with *folder* command 106
real name, setting your 197
realname variable 197
record variable 198
record_users variable 198
recording messages
by user 198
entire messages 198
headers only 191
recursive functions 94, 198
recursive variable 198
redirect variable 199
registering users 32
remote addresses 13
reply command 129
reply_to_hdr variable 199
replyall command 130
replying
return paths 199
routing replies 190
to all recipients 130
to sender 129
to yourself accidently 191
with inclusion 170

replying to messages
in command-line interface 49
replysender command - see replying
resume command 130
resuming suspended messages 130
retain command 131
return command 131
return paths on replies 199
reverse video
avoiding problems with 64
disabling 193
use of in fullscreen interface 64
RFC822 addresses 13
routing messages
automatically 171
avoiding unnecessary hosts 179
via known hosts 190
routing-related variables 163

S

s command - see *save*
save command
described 132
differences from *write* and *copy* 132
save_empty variable 200
saveit function 110
saveopts command 133
saving initialization files 133
saving messages 132
deleting after saving 190
in command-line interface 54
saving messages to files 147
screen size
setting in character mode 200
setting in GUI mode 200
screen variable 200
screen_win variable 200
scripts
terminating 131
using variables in 154
writing functions in GUI mode 217
scroll boxes, creating 91
search command 133
described 133
linking multiple 58
searching messages 133
by date 134
in command-line interface 55
semicolon, as command terminator 58
sending messages
in command-line mode 49
with *mail* command 120
sendmail variable 200
send-only mode 3
attaching files in 22
defined 8
displaying headers in 28
sending carbon copies in 23
sending drafts in 24
specifying subject in 34
starting 18
set command 136
setting variables 136
sh command 137, 216
shell
invoking UNIX 137
using to launch commands 216
shift command 137
show_deleted variable 200
signature-related variables 164
signatures
adding with tilde-escapes 68
alternate 174
automatic 173
described 6
local and remote 175
overriding 122
signing messages automatically 173
sign-on screen, suppressing 197
sort command 139
sort variable 201
sorting messages
at startup 201
in command-line interface 55
with *sort* command 139
source command 141
sourcing initialization files at startup 34
spool file - See System Folder 127
squeeze variable 201
starting Z-Mail 18
in command-line mode 18
in fullscreen mode 18
in GUI mode 18
in send-only mode 18
status variable 201
stop command 141
subject
prompting for a 168
setting with *-s* option 121

suppressing headers - see ignoring headers
suspended messages
listing with *jobs* command 118
resuming 130
suspending Z-Mail 141
System Folder
described 7
holding messages in 186
opening an empty 33
preserving messages in 127
specifying location of 30
starting up with other than 26
system.zmailrc file 37, 133

T

:t variable modifier 154
t command - see *type* and *display*
tail of path variable modifier 154
templates
directory for message 201
message 122
using from the *zmail* command 35
templates variable 201
temporary files
directory for 202
thisfolder variable 202
tilde character
as command prefix 65
as filename metacharacter 12
tilde-escape commands
aborting messages with 72
adding signatures with 68
attaching files with 70
editing text with 67
including text files with 69
invoking with macros 80
list of 65–74
modifying headers with 65
referring to variables with 74
running commands with 73
starting text editor with 67
storing messages with 71
suspending messages with 72
timeout
checking for new mail 35
in macro keystrokes 76
title variable 202
tmpdir variable 202
top command 141
toplines variable 202
transport-related variables 163
twenty-four hour clock 192
type command - see *display*
TYPE statement 214

U

unalias command 142
unbind command 142
unbutton command 142
uncmd command 143
undelete command 143
undeleting messages 143
undigest command 143
unfilter command 144
unfunction command 144
un_hdr command 145
unignore command 145
UNIX commands
entering 216
invoking shell for 137
pipelining 58
piping message lists to 126
running 202
UNIX *mail* substitute, Z-Mail as a 3
unix variable 202
unmap command 145
unmark command 146
unmarking messages 146
unset command 146
update command 147
updating folders
before hangup or shutdown 184
verification before 203
with *quit* command 128
with *update command* 147
user interfaces, Z-Mail's 8
UUCP addresses 13, 57, 90, 91, 125, 171, 172, 179, 190, 193

V

variables
address-related 163
command-line-related 165
compose-related 162
configuration-related 158
directory-related 161
display-related 160

editor-related 162
example of user-defined 137
expanding 102
file-related 161
folder information 156
folder-related 160
fullscreen-related 165
function 150
GUI-related 164
header-related 159
help-related 158
initialization files and 153
internal, described 150
list of Z-Mail 168–207
message information 155
modifiers 154
print-related 162
programming-related 166
routing-related 163
setting 136, 151
signature-related 164
storing values of 153
summary of Z-Mail 158–167
system 150
transport-related 163
types of 150
types of values for 150
UNIX system-related 167
unsetting values of 146
user-defined 150
using in scripts 154
verbose option, passing to MTA 36
verbose variable 203
verify variable 203
version command 147
version number, displaying Z-Mail 204
version variable 204
visual editor 204
invoking with tilde-escape 67
visual variable 204

W

w command - see *write*
warning variable 204
warnings
displaying 204
suppressing 197
window
closing Z-Mail 113
shelling a new 205
window_shell variable 205
Wingz, integrating Z-Mail and 217
working directory
changing 96
changing your 96
displaying name of 128
wrap variable 205
wrapcolumn variable 205
write command
described 147
differences from *save* and *copy* 132
writing messages to files 147

X

x command - see *exit*
xit command - see *exit*

Z

z command - see headers
z+ and *z-* commands 112
zmail command
as substitute for UNIX *mail* 3
zmail command reference 22–36
zmail command summary 19–21
Z-Mail Handbook, The 2
Z-Mail Installation Guide 2, 33
Z-Mail User's Guide 2
.zmailrc file 37, 133, 141
ZMAILRC environmental variable 37
Z-Script
command reference 89–147
command summary 84–88
described 1