Freehoo version 3.5.2

User/Developers guide, 1 May 2008

K. Viswanathan gnuvisu@yahoo.com
"Anand Babu" Periasamy ab@zresearch.com

This is the first edition of the Freehoo documentation.

Table of Contents

1 Overview

Freehoo is a free console based messenger for Yahoo IM Service with GNU Bash like tab completion / editing and GNU Emacs like extensibility.

- Highly extensible through '`Scheme`' language. (see Chapter 4 [Customization], page 14) (see Chapter 6 [Extension language], page 17)
- Console based client with Readline interface featuring command line editing, history, etc ... (see Chapter 5 [Tips and Tricks], page 16).
- Most of the features in Freehoo are fully customizable, either through command line arguments (see Chapter 2 [Invoking], page 2), or startup file (see Section 4.1 [freehoo.scm], page 14)
- Consists of almost all the features as the conventional Yahoo messenger for example email alert, conference, ignore etc ...
- Additional features like alias, bell, forward, cc, eval, freehoo, fh-conf, load, ping, date, sh etc., (see Chapter 3 [Commands], page 3)
- With a new concept called '`dynamic-commands`', a command can appear and disappear dynamically, based on the context.
- AUTO-INSERT feature magically decides and inserts the target buddy name each time you press *`RET`* during a session. (see Section 3.36 [AUTO-INSERT mode HOW-TO], page 11)
- `history` feature records all your conversions in '`~/.freehoo/history/`*`login-id/buddy-name`*'.
- Finally, Freehoo is free software. This means that everyone may use it, redistribute it and/or modify it under the terms of the GNU General Public License, as published by the Free Software Foundation (see Chapter 16 [License], page 33)

2 Invoking

Invoking Freehoo at command prompt is very simple. The following are the possible command-line arguments supported,

freehoo [*options*]
where *options* are,

- -u=*yahoo-id* | —user=*yahoo-id*
  yahoo-id is your yahoo account name.
- -s=*status* | —status=*status*
  status can be one of the following numbers,

```
  0:  I'm Available
  1:  Be Right Back
  2:  Busy
  3:  Not at Home
  4:  Not at my Desk
  5:  Not in the Office
  6:  On the Phone
  7:  On Vacation
  8:  Out to Lunch
  9:  Stepped Out
 12:  Invisible
999:  Idle
```

- -v | —version
  Gives the current version of Freehoo. This option does not accept any argument.
- -h | —help
  Gives a brief help on the above options. This option does not accept any argument.

3 Freehoo commands

3.1 Freehoo command - *

`*` *message* [command]

Broadcast *message* to all buddies in the list. `*` is a virtual buddy in your buddy list which means everybody in the list.

```
~qp~> * Hello, World!
~qp~> *
```

The message `Hello, World!` will be sent to all the buddies in your buddy list.

3.2 Freehoo command - <buddy>

`<buddy>` *message* [command]

Send *message* to *buddy*. You can use ***TAB*** key to fill the *buddy* name automatically. On conflict press ***TAB*** twice to list the conflicting *buddy* names.

Example: A sample Freehoo session.

```
~qp~> /who

[Friends]
* markus [Busy Hacking]
  rms
* thomas [Idle]

[Team]
  balugi
* kvisu2000

~qp~> kvisu2000 Hi, How are you
kvisu2000 -> I'm fine
~qp~> balugi Hi, Are you there
Offline message sent to [balugi]
```

3.3 Freehoo command - add

`/add` *buddy* [*group*] [*message*] [command]

Add a buddy to your buddy list. The following are the possible arguments for this command,

a. *buddy* must be a Yahoo ID.

b. Optional *group* under which the *buddy* is added. If the *group* does not exists it will be created newly. The default *group* is `Friends`.

c. *message* is optional message.

3.4 Freehoo command - alias

`/alias` *name buddy1* [*buddy2 buddy3 . . .*] [command]

This command expands *name* to *buddy1*, *buddy2* etc. `/alias` accepts *name* and at least one *buddy* as arguments. Aliases can be recursive.

Using Guile interface, you can add permanent aliases to startup file (see Section 4.1 [freehoo.scm], page 14) like,

```
(define alias '((helpdesk . (abindian balugi kvisu2000))
                (mridul . (gnuindian))
                (bala . (balugi))
                (nags . (nagappanal))
                (visu . (kvisu2000))))
```

3.5 Freehoo command - bell

`/bell` [command]

This command switches bell sound between ON and OFF. By default '`bell`' is ON. A better interface to this command is `/toggle` (see [/toggle bell], page 10)

Using Guile interface, you can disable bell during startup (see Section 4.1 [freehoo.scm], page 14) like,

```
(fh-bell!)
```

3.6 Freehoo command - broadcast

`/broadcast` *message* [command]

Broadcast *message* to all buddies in the list. This is same as `*` (see Section 3.1 [<star>], page 3).

```
~qp~> /broadcast Hello, World!
~qp~>
```

The message `Hello, World!` will be sent to all the buddies in your buddy list.

3.7 Freehoo command - burst-of-romance

`/burst-of-romance` *buddy count message* [command]

Simulate sending of *message*s *count* times to *buddy* as if you are typing by hand (random delays in between). Usually used for sending multiple roses or kisses ;)

```
~qp~> /burst-of-romance mypuchki 16 :*
~qp~>
```

3.8 Freehoo command - burst

`/burst` *buddy message* [command]

Explode chars in *message* with random count as if you typed with hand so intensely. Calling your girl friend's name or saying "I Love You", this command is very useful. Also IRC guys greet this way.

```
~qp~> /burst mypuchki puchki
~qp~>
```

3.9 Freehoo command - buzz

`/buzz` *buddy* [command]

send a BUZZ! to *buddy*

```
~qp~> /buzz marcus
~qp~>
```

3.10 Freehoo command - forward

`/forward` *from-buddy to-buddy1* [*to-buddy2 to-buddy3 ...*] [command]

Messages received from *from-buddy* are forwarded to *to-buddy1*, *to-buddy2* etc. `/forward` accepts *name* and at least one *to-buddy* as arguments.

Using Guile interface, you can add permanent forwards to startup file (see Section 4.1 [freehoo.scm], page 14) like,

```
(define forward '((gnubot . (ramyog_2000 nagappanal balugi))
                  (gopal_narayanan . (parag_mehta))))
```

3.11 Freehoo command - cc

`/cc` *buddy cc-buddy1* [*cc-buddy2 cc-buddy3 ...*] [command]

Message sent to *buddy* is CC'ed to *cc-buddy1*, *cc-buddy2* etc. `/cc` accepts *buddy* and at least one *cc-buddy* as arguments.

Using Guile interface, you can add permanent CCs to startup file (see Section 4.1 [freehoo.scm], page 14) like,

```
(define cc '((rms . (markus roland thomas))
             (gopal_narayanan . (parag_mehta))))
```

3.12 Freehoo command - color-*

`/color-on` [command]

Enables color themes mode.

Using Guile interface, you can permanently enable color themes in startup file (see Section 4.1 [freehoo.scm], page 14) like,[1]

```
(fh-enable-colors)
```

`/color-off` [command]

Disables color themes mode.

Using Guile interface, you can permanently disable color themes in startup file (see Section 4.1 [freehoo.scm], page 14) like,

```
(fh-disable-colors)
```

[1] by default color themes is enabled

`/color-buddy` *buddy color* [command]
Displays all messages from *buddy* in the specified *color*. Possible *color* values are [red, blue, yellow, magenta, green, cyan, white].

Using Guile interface, you can permanently set color for a buddy message in startup file (see Section 4.1 [freehoo.scm], page 14) like,

```
(fh-set-buddy-color! "nirranjan" "green")
(fh-set-buddy-color! "rms" "red")
(fh-set-buddy-color! "marcus" "magenta")
```

3.13 FreeHoo virtual conference commands

This is standard yahoo conference this feature is virtual.

`/conf-start` *room buddy1* [*buddy2 buddy3 . . .*] [command]
This command starts a conference with room name *room* with buddies *buddy1*, *buddy2* etc.

`/conf-add` *room buddy* [command]
This command add a buddy, to a existing conference room.

`/conf-list` [command]
This command lists all the conference rooms associated with you currently.

`/conf-end` *room* [command]
This command quits you from a conference rooms.

`/conf-decline` *room* [command]
This command send your decline to a received invitation from a conference rooms.

`/conf-send` *room* [command]
This command sends message to a conference room.

3.14 FreeHoo command - date

`/date` [*arguments*] [command]
This command displays the system date. Try '`--help`' for complete list of *arguments*.

3.15 FreeHoo command - dbg-backtrace

`/dbg-backtrace` [*arguments*] [command]
Produce scheme backtrace for the last error.

3.16 FreeHoo command - dict

`dict` [*arguments*] [command]
Ask to dictionary buddy. Try '`--help`' for complete list of *arguments*.

3.17 FreeHoo command - eval

`/eval` *exp* [command]

Evaluate *exp*, a list representing a Scheme expression. You have complete access to fh-guile internals, including Scheme extensions through this command.

Example: To send message to yourself

```
~qp~> /eval (fh-send-message (fh-get-default-login-id) "Hello GNU")
abindian -> Hello GNU
~qp~>
```

3.18 freehoo command - exec

`/exec` [*command*] [*args . . .*] [command]

This command is an alias to `shell`.

Example:

```
See /shell example
```

3.19 FreeHoo command - freehoo

`/hoo` *buddy* [command]

This command checks if *buddy* is using FreeHoo.

Example: To check if '`kvisu2000`' is using FreeHoo,

```
~qp~> /freehoo kvisu2000
Yes [kvisu2000] is using FreeHoo
~qp~>
```

3.20 freehoo command - help

`/help` [*command*] [command]

You can always ask FreeHoo itself for information on its commands, using the command `/help`. If *command* is ignored then help on all the commands will be listed.[2]

3.21 freehoo command - history

`/history` [*buddy*] [command]

Display history page by page for *BUDDY*. With no arguments mentioned, this command displays all messages that belongs to current history session. This session is flushed upon every successful login.

[2] If you want talk to the author of freehoo, message *gnubot* buddy

3.22 freehoo command - ignore*

`/ignore` *buddy* [command]
Adds this BUDDY to Yahoo ignore list. This *buddy* will be removed from the buddy list and you will get no messages from him/her.

`/unignore` *buddy* [command]
This command unignores *buddy* from the Yahoo ignore list.

`/ignore-list` [command]
Displays the Yahoo *ignore-list*

3.23 freehoo command - load

`/load` *scheme-file* [command]
`/load` command loads and evaluates Scheme extensions. *scheme-file* argument is a must.

Using Guile interface, you can load other Scheme files from startup file (see Section 4.1 [freehoo.scm], page 14) like,

```
(load "/home/gnu/hello.scm")
```

(see [fh-load], page 20)

3.24 freehoo command - ping

`/ping` *buddy count* [command]
Send ping messages to a *buddy*, *count* times.

3.25 freehoo command - pipe

`/pipe` *buddy command* [*args* . . .] [command]
Pipe the output of *command* to *buddy*.

`pipe` *buddy command* [*args* . . .] [command]
Pipe the output of *command* to *buddy* and also set current target buddy.

3.26 freehoo command - quit

`/quit` [command]
Logout and exit from freehoo.

3.27 freehoo command - refresh

`/refresh` [command]
This command refreshes the buddy list and the status information.

3.28 freehoo command - reject

`/reject` [command]

This command rejects the buddy for adding you in his/her buddy list and removes your name from his/her buddylist.

3.29 freehoo command - remove

`/remove` *buddy* [*message*] [command]

This command permanently removes the BUDDY from buddy list. *message* argument is optional.

Example:

```
~qp~> /remove balugi Poda Kuppa!
```

3.30 freehoo command - restart

`/restart` [command]

This command restarts freehoo from inside freehoo.

3.31 freehoo command - send

`/send` *buddy message* [command]

This command sends *message* to the *buddy*.

Example:

```
~qp~> /send balugi Hi, how are you/
balugi -> Fine and you/
```

3.32 freehoo command - send-file

`/send-file` *buddy filepath* [*message*] [command]

This command sends *file* to the *buddy*.

Example:

```
~qp~> /send-file balugi /etc/passwd my passwd file
```

3.33 freehoo command - shell

`/shell` [*command*] [*args* . . .] [command]

`/shell` command executes the specified *command* with its *args*. With no arguments, `/shell` escapes to shell. You can also chat with shell as if shell is your buddy. Just type `shell` without `/` prefix.

Example:

```
~qp~> /shell ls -lh /tmp
total 12k
drwxr-xr-x    3 root   root    4.0k Jan  1 00:53 emacs-terminfo
```

```
-rw-r--r--    1 root   root       1 Jan  1 05:04 emacsOdVut8
drwx------    2 root   root    4.0k Jan  1 00:13 xdvi7GIKqr
~qp~> /sh
press C-d to return to freehoo
$ rm -f /tmp/xdvi7GIKqr
C-d RET
~qp~>
```

3.34 freehoo command - status

`/status` [*status-number*] [*custom-message*] [command]

Using `/status` command, you can view or set your buddy-status.

a. *status-number* should denote one of the following.

```
  0:  I'm Available
  1:  Be Right Back
  2:  Busy
  3:  Not at Home
  4:  Not at my Desk
  5:  Not in the Office
  6:  On the Phone
  7:  On Vacation
  8:  Out to Lunch
  9:  Stepped Out
 12:  Invisible
 99:  [Custom message]
999:  Idle
```

b. Optionally you can mention *custom-message*, if *status-number* is 99.

c. When no arguments are supplied, `/status` command displays your current status.

3.35 freehoo command - times

`/times` *buddy count message* [command]

Send *count* number of times, the *message* to *buddy*.

Example:

```
~qp~> my_sweetheart_16 sweetheart, i am busy hacking, i cannot take
you to the party tonight
~qp~> my_sweetheart_16 will you please forgive me this time only...
~qp~> /times my_sweetheart_16 64 pleaseeeee
```

3.36 freehoo command - toggle

`/toggle` *state* [command]

This command toggles the following *state*s.

- `bell`
  Toggle *bell* sound between ON and OFF. Default is ON.

Example:

```
~qp~> /toggle bell
Bell sound - [OFF]
~qp~> /toggle bell
Bell sound - [ON]
~qp~>
```

- `session`
  Toggle *session* mode between VANILLA and AUTO-INSERT. Default is AUTO-INSERT.

  VANILLA mode lets the user to type the buddy name manually. How ever user can use *TAB* interface to auto-fill. (see Chapter 5 [Tips and Tricks], page 16)

  AUTO-INSERT mode intelligently selects the buddy name during chat session. (see Section 3.36 [AUTO-INSERT mode HOW-TO], page 11).

  [3]

- `status`
  Toggle display of *status* change notifications between SHOW and HIDE. Default SHOW.

- `who`
  Toggle *who* mode between ONLINE-ONLY and SHOW-ALL. Default is ONLINE-ONLY. (see Section 3.39 [who], page 12).

3.36.0.1 AUTO-INSERT mode HOW-TO

AUTO-INSERT mode makes freehoo intelligent by automatically selecting the buddy name, every time when the user types the message. The following is a small HOW-TO on AUTO-INSERT mode,

- How AUTO-INSERT guesses the buddy name?

  AUTO-INSERT mode guesses the buddy name from the previously sent/received message.

- How to change the target buddy selected by AUTO-INSERT mode?

  Just move the cursor back and edit the buddy name to your choice.

- How to switch the buddy name to reply to the last received message?

  Pressing *RET* without typing any message switches to last received buddy.

3.37 freehoo virtual conference commands

Unlike standard yahoo conference this feature is virtual. Its actually a combination of `cc` and `forward` extensions. All the messages you send to a virtual buddy named `cf` are despatched to the conference members. Similarly the messages received from any of the conference members are forwarded back to other conference members.

[3] Currently typing notifications will be sent remote buddy only in AUTO-INSERT mode. VANNILA mode has no concept of current target buddy.

`/vconf-start` *buddy1* [*buddy2 buddy3 . . .*] [command]
This command starts a virtual conference with *buddy1*, *buddy2* etc. On conference start `/vconf-start` disappears and `/vconf-who`, `/vconf-end` appears. `/vconf-start` accepts at least one *buddy* as its argument.

`/vconf-who` [command]
This command lists all the conference members.

`/vconf-end` [command]
This command ends the virtual conference. On conference end `/vconf-who`, `/vconf-end` disappears and `/vconf-start` appears.

3.38 freehoo command - version

`/version` [command]
This command displays version information.
Example:

```
~qp~> /version
freehoo (Freehoo) 3.5.2
Copyright (C) 2003, 2004, 2008 Freehoo Core Team.
This is free software; see the source for copying conditions.  There
is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
~qp~>
```

3.39 freehoo command - who

`/who` [command]
This command displays the buddy list as well as their current buddy status.
Example:

```
~qp~> /who

[Friends]
* markus [Busy Hacking]
  rms
* thomas [Idle]

[Team]
  balugi
* kvisu2000

~qp~>
```

3.40 freehoo command - xmessage

`/xmessage` *buddy message* [command]
Send a X popup *message* to *buddy*.

Example:

```
~qp~> /xmessage bala Hi Bala, are you there
```

4 Customizing freehoo

Hoo can be customized to a great extent using Guile interface. User can himself customize or extend new features in freehoo using Scheme as extension language. Most of the features are already written in Scheme.

If you want to extend freehoo yourself, you can further explore Hoo Extension Developer Guide. (see Chapter 6 [Extension language], page 17)

4.1 freehoo.scm

Hoo loads the startup options from '`~/.freehoo/freehoo.scm`'. Right from custom settings like username, password . . . to complete Scheme programming can be done in this file.

```
;   this is comment
;;; sample freehoo.scm
;;; login-id is equal to my gnu/linux account
;; (fh-set-default-login-id! (getlogin))

;;; default login-id for yahoo service
(fh-set-default-login-id! "gnu_india")
;; (fh-set-default-login-id! "abindian")
;; (fh-set-default-login-id! "gnubot")

;;; default global password
(fh-set-default-password! "nopassword")
;; (fh-set-default-password! "pressenter")

;;; by default session mode is AUTO-INSERT. switch it to VANILLA mode
;; (fh-toggle! "session")

;;; by default display of status message is SHOW. switch it HIDE
;; (fh-toggle! "status")

;;; if login-id is "abindian"
(and (string=? (fh-get-default-login-id) "abindian")
     ;;; default password
     (fh-set-default-password! "pressenter")
     ;;; switch OFF bell
     (fh-toggle! "bell")
     ;;; show ALL buddies
     (fh-toggle! "who")
     ;;; login in invisible mode
     (fh-set-default-status! 12))

;;; if login-id is "gnubot"
(and (string=? (fh-get-default-login-id) "gnubot")
     ;;; default password
     (fh-set-default-password! "pressescape")
```

```
    ;;; switch OFF bell
    (fh-toggle! "bell"))

;;; create aliases
(define alias '((helpdesk . (abindian balugi kvisu2000))
                (mridul . (gnuindian))
                (bala . (balugi))
                (nags . (nagappanal))
                (visu . (kvisu2000))))

;;; create CC lists
(define cc '((rms . (markus roland thomas))
             (gopal_narayanan . (parag_mehta))))

;;; create forward lists
(define forward '((gnubot . (ramyog_2000 nagappanal balugi))
                  (gopal_narayanan . (parag_mehta))))

;;; my own dict words
(fh-dict-add-word! "acomplexneword")
(fh-dict-add-word! "myfriendsnick")
```

All entries in this '`freehoo.scm`' file are optional. How ever there is no limit in customizing or extending freehoo through Guile interface. Explaining all the possibilities are beyond the scope of this document.

(see Chapter 6 [Extension language], page 17)

4.2 init.scm

All system wide policy settings and extensions are loaded through '`i.scm`'. By default you can find '`init.scm`' at '`/usr/share/freehoo/extensions/`'. To override this system wide '`init.scm`' file, copy it to '`~/.freehoo/extensions/init.scm`'. You must be aware of what you are doing, before you mess up anything here.

(see Chapter 6 [Extension language], page 17)

4.3 Default Scheme extensions

Most of the freehoo features are available through Scheme extensions. To override these extensions, copy them from '`/usr/share/freehoo/extensions/`' to '`~/.freehoo/extensions/`' and edit them.

(see Chapter 6 [Extension language], page 17)

5 Tips and Tricks

You are free to use complete Readline keys inside freehoo. Frequently used Readline keys inside freehoo are,

5.0.1 Cursor motion

character	C-b	C-f
word	M-b	M-f
line up/down	C-p	C-n
line start/end	C-a	C-e

5.0.2 Editing

delete char	C-d
delete char backwards	C-h
delete word	M-d
delete word backwards	C-w
kill line	C-k
kill line backwards	C-u
character swap	C-t
word swap	M-w
paste	C-y
undo	C-_
repeat prefix	M-number

5.0.3 Case change

uppercase word	M-u
lowercase word	M-l
capitalize word	M-c

If you want to do further stunts, jump to Readline manual, See section "Readline" in *Readline*.

[1]

[1] When you press `TAB` twice at freehoo prompt you can see all the possible commands and buddy names.

6 Extension language

An *extension language* is a programming language interpreter offered by an application program, so that users can write macros or even full-fledged programs to extend the original application. Extension languages have a C interface (it is usually C, but it could be any other compiled language), and can be given access to the C data structures. Likewise, there are C routines to access the extension language data structures.

Hoo uses GNU extension language - *Guile* (which can stand for _GNU Ubiquitous Intelligent Language Extension_). Guile started out as an embeddable Scheme interpreter, and has rapidly evolved into a kitchen-sink package including a standalone Scheme interpreter, an embeddable Scheme interpreter, several graphics options, other languages that can be used along with Scheme (for now just _ctax_ and _Tcl_), and hooks for much more.

7 hello.scm extension

Learn how to write a simple extension by yourself.

7.1 Writing hello.scm

This example extension creates a dynamic command `/hello`, which on invocation sends a message *Hello GNU* to yourself.

```
;; hello.scm
(fh-register-command! "/hello" "/hello\n\t- Hello to myself.\n")
(define (/hello args)
   "send me hello message"
   (fh-send-message (fh-get-default-login-id) "Hello GNU"))
```

7.2 Loading hello.scm

Copy '`hello.scm`' to '`~/.freehoo/extensions/`' and add this entry in your '`~/.freehoo/freehoo.scm`'

```
(fh-load "hello.scm")
```

8 Variables

Currently no variables are exported to Scheme environment from Freehoo. Instead we have solved such needs using procedure interface to get/set variables.

9 Procedures

The following are the list of freehoo procedures that are exported to Scheme. Now you are able to call the procedures from Scheme that are written in C.

9.1 General procedures

`fh-load` *filepath* [primitive]

Loads and evaluates *filepath*.scm from mentioned path or from '`~/.freehoo/extensions/`' or from '`/usr/share/freehoo/extensions/`'.

Example:

```
(fh-load "aliases.scm")
```

`fh-add-buddy` *buddy group* [primitive]

Adds *buddy* to *group* in your contact list.

Example:

```
(fh-add-buddy "rms" "GNU")
```

`fh-send-message` *buddy message* [primitive]

Sends *message* to the *buddy*.

Example:

```
(fh-send-message "rms" "hello GNU")
```

`fh-send-message-no-hook` *buddy message* [primitive]

Sends *message* to the *buddy*. This procedure does not run any hooks. You will have to use this procedure while sending messages from inside `fh-message-send-hook` otherwise it will lead to an endless recursion.

Example:

```
(fh-send-message-no-hook "rms" "hello GNU")
```

`fh-set-current-target-buddy!` *buddy* [primitive]

Sets the target *buddy* name in AUTO-INSERT mode. This call makes meaning only when you are in AUTO-INSERT mode.

Example:

```
(fh-set-current-target-buddy! "richard")
```

`fh-register-command!` *command documentation* [primitive]

Registers a dynamic *command* and its *documentation* with Freehoo command interpreter.

Example:

```
(fh-register-command! "/date" "/date\n\t- display current date")
```

`fh-unregister-command!` *command* [primitive]

Un-registers *command* from freehoo's command interpreter.

Example:

```
(fh-unregister-command!  "/date")
```

`fh-version` [primitive]

Return the current version string of freehoo.

Example:

```
(display (fh-version))
```

`fh-display` *message* [primitive]

Prints the *message* in the console. Unlike the `display` primitive, this procedure takes care of printing *messages* asynchronously.

Example:

```
(fh-display ("I am proud of freehoo"))
```

`fh-dict-add-word!` *newword* [primitive]

Adds *newword* to the english dictionary for autocompletion.

Example:

```
(fh-dict-add-word! "acomplexneword")
(fh-dict-add-word! "myfriendsnick")
```

`fh-dict-add-word-sorted!` *newword* [primitive]

Adds *newword* to the english dictionary for autocompletion. Assumes *newword* newword will be lexographically the last word in the dictionary list. Used to load words from huge dictionary files which are already sorted.

Example:

```
(fh-dict-add-word-sorted! "aaa")
(fh-dict-add-word-sorted! "aab")
(fh-dict-add-word-sorted! "abc")
```

`fh-dict-del-word` *unusedword* [primitive]

Removes *unusedword* from the english dictionary for autocompletion.

Example:

```
(fh-dict-del-word "acomplexneword")
(fh-dict-del-word "myfriendsnick")
```

9.2 Configuration procedures

`fh-bell` [primitive]

Switches the bell sound between ON and OFF.

Example:

```
(fh-bell)
```

`fh-toggle!` *state* [primitive]

Switches *state* between two different states.

The value of *state* can be,

state bell Switches bell sound between ON and OFF.

Example:

```
(fh-toggle! bell)
```

state session
: Switches freehoo session between AUTO-INSERT and VANILLA modes.
Example:
```
(fh-toggle! session)
```

state status
: Switches display of status messages between SHOW and HIDE.
Example:
```
(fh-toggle! status)
```

state who
: Switches who mode between ONLINE-ONLY and SHOW-ALL.
Example:
```
(fh-toggle! who)
```

`fh-get-home-dir` [primitive]
: Return the home directory of current user.
Example:
```
(chdir (fh-get-home-dir))
```

`fh-get-config-dir` [primitive]
: Return the directory containing configuration files.
Example:
```
(display (fh-get-config-dir))
```

`fh-get-config-filename` [primitive]
: Return the configuration filepath.
Example:
```
(display (fh-get-config-filename))
```

`fh-get-download-filename` [primitive]
: Return the filename containing URL downloads.
Example:
```
(display (fh-get-download-filename))
```

`fh-get-global-extensions-directory` [primitive]
: Return the directory containing global extensions.
Example:
```
(chdir (fh-get-global-extensions-directory))
```

`fh-get-local-extensions-directory` [primitive]
: Return the directory containing local extensions.
Example:
```
(chdir (fh-get-local-extensions-directory))
```

`fh-get-default-login-id` [primitive]
: Return the yahoo-id of currently logged-in user.
Example:
```
(display (fh-get-default-login-id))
```

`fh-set-default-login-id!` *yahoo-id* [primitive]
Sets the *yahoo-id*.
Example:

```
(fh-set-default-login-id! "kvisu2000")
```

`fh-set-default-password!` *password* [primitive]
Sets the yahoo account *password*.
Example:

```
(fh-set-default-password! "pressenter")
```

`fh-set-default-status!` *number* [primitive]
Sets the yahoo status *number*.
Example:

```
;;; go invisible
(fh-set-default-status! 12)
```

For complete list of status number definitions refer (see Section 3.34 [status], page 10)

`fh-set-prompt!` *prompt-string* [primitive]
Sets the yahoo prompt. This call will be effective only when called during freehoo startup.
Example:

```
;;; set freehoo prompt
(fh-set-prompt! "~qp~> ")
```

`fh-logoff` [primitive]
Logoff from the freehoo connection.
Example:

```
;;; logoff connection
(fh-logoff)
```

`fh-quit` [primitive]
Logoff from the connection and quit freehoo
Example:

```
;;; logoff and quit freehoo
(fh-quit)
```

9.3 Hook related procedures

[1]

`fh-hook-return` [primitive]
Makes the calling procedure return immedieltly after running the hooks.
Example:

[1] The following primitives can be called from procedures that are hooked to freehoo exported hooks. Also check the list of supported primitives for each hook.

```
(define (alias to message)
  "alias nags to nagappanal"
  (and (string=? to "nags")
       ;; send message to actual name
       (fh-send-message-no-hook "nagappanal" message)
       ;; "nags" does'nt exist. so let send return immediately
       (fh-hook-return)))
(add-hook! fh-message-send-hook alias)
```

9.4 Utility procedures

These are general purpose utility procedures written completly in Scheme.
[2]

`symbolrnumber->symbol` *num* [procedure]

Converts *num* to its corresponding Scheme symbol. *num* is any number atom in Scheme.

Example:

```
(symbolrnumber->symbol 5)
```

will return 5 which is a Scheme symbol and not a number.

`any->symbol` *num* [procedure]

Converts string or number or symbol to its corresponding Scheme symbol.

Example:

```
(any->symbol 5)
```

will return 5 which is a Scheme symbol and not a number.

`list->asv` *list delimiter* [procedure]

Converts *list* to vector delimitted by *delimiter*. *list* stands for `list to any seperated vector`.

Example:

```
(list->asv ("gnuindian" "nagappanal" "abindian" "balugi") ", ")
```

will return "gnuindian, nagappanal, abindian, balugi"

`list->csv` *list* [procedure]

Converts *list* to a comma seperated vector delimitted by '`,` '.

Example:

```
(list->csv ("gnuindian" "nagappanal" "abindian" "balugi"))
```

will return "gnuindian, nagappanal, abindian, balugi"

`sentence->words` *sentence* [procedure]

Converts a string of sentence to a list of symbols.

Example:

```
(sentence->words "Free as in Freedom")
=> (Free as in Freedom)
```

[2] utility procedures are loaded through /DATADIR/freehoo/extensions/util.scm and you are free to hack for cool undocumented procedures

`list->symlist` *list* [procedure]

Converts *list* of numbers/symbols into a list of symbols.

Example:

```
(list->symlist (5 a gnu 100 10.5))
```

will return (5 a gnu 100 10.5) where all items in the list are symbols and not numbers. List already containing symbols are not alterted.

`list->strlist` *list* [procedure]

Converts *list* of numbers/symbols into a list of strings.

Example:

```
(list->strlist (5 a gnu 100 10.5))
```

will return ("5" "a" "gnu" "100" "10.5") where all items in the list are symbols and not numbers. List already containing symbols are not alterted.

`send-message-to-group` *group message* [procedure]

Send *message* to a *group*.

Example:

```
(send-message-to-group '(gnuindian nagappanal abindian balugi) "Hello GNU")
```

`local-date-time` [procedure]

Returns the local date and time

Example:

```
(display (local-date-time))
```

`ignore-message!` *message-pattern* [procedure]

ignores messages matching the regex *message-pattern*

Example:

```
(ignore-message! "^PING$")
```

`ignored-message?` *message* [procedure]

Return #t if this *message* is ignored, else #f.

Example:

```
(ignored-message? "PING")
```

10 Hooks

Through Hooks facility FreeHoo lets you steal its control at various important junctures during execution.

`fh-message-send-hook` *buddy message* [hook]

Hook procedure is called with *buddy* and *message* as arguments on every send message operation.

Supporting primitives:

see [fh-hook-return], page 23

Example:

```
(define (cc-proc to message)
  "hook procedure for CCing messages"
  (and (string=? to "rms")
       (fh-send-message-no-hook "thomas" message)
       (fh-send-message-no-hook "roland" message)
       (fh-send-message-no-hook "gord" message)))
(add-hook! fh-message-send-hook cc-proc)
```

`fh-message-receive-hook` *buddy message* [hook]

Hook procedure is called with *buddy* and *message* as arguments on every receive message operation.

Supporting primitives:

see [fh-hook-return], page 23

Example:

```
(define (forward-proc from message)
  "hook procedure for bouncing messages"
  (and (string=? from "rms")
       (fh-send-message-no-hook "thomas" message)
       (fh-send-message-no-hook "roland" message)
       (fh-send-message-no-hook "gord" message)))
(add-hook! fh-message-receive-hook forward-proc)
```

`fh-message-receive-offline-hook` *buddy message timestamp* [hook]

Hook procedure is called with *buddy*, *message* and *timestamp* as arguments on every receive offline message operation.

Supporting primitives:

see [fh-hook-return], page 23

Example:

```
(define (ack-proc from message timestamp)
  "hook procedure for acknowledging offline messages"
  (fh-send-message-no-hook from "Received ur offline message"))
(add-hook! fh-message-receive-offline-hook ack-proc)
```

`fh-mail-notify-hook` *from subject* [hook]

Hook procedure is called with *from* and *subject* as arguments upon every new yahoo mail.

Example:

```
(define (mail-notify-proc from subject)
  "hook procedure for mail notification"
  (system "ogg123 ~/themes/mail-notify.ogg&"))

(add-hook! fh-mail-notify-hook mail-notify-proc)
```

`fh-login-post-hook` [hook]

Hook procedure is called after completion of login operation.

Supporting primitives:

see [fh-hook-return], page 23

Example:

```
(define (history-rotate-proc)
  "hook procedure for flushing the old history messages"
  (history-rotate))
(add-hook! fh-login-post-hook history-rotate-proc)
```

`fh-contact-added-hook` [hook]

Hook procedure is called after a buddy adds you in his/her contact list.

Supporting primitives:

see [fh-hook-return], page 23

Example:

```
(define (auto-add-buddy from message)
  "hook procedure for automatically adding buddy to your contact list"
  (fh-add-buddy from "GNU")) ;; GNU - group name
(add-hook! fh-contact-added-hook auto-add-buddy)
```

11 Learning further

The following are the URLs where you can find useful manuals for Guile and Scheme.

http://www.gnu.org/software/guile/
http://www.schemers.org/
ftp://ftp.cs.utexas.edu/pub/garbage/cs345/schintro-v14/schintro_toc.html
http://www.informatik.uni-kiel.de/~scheme/
http://freespace.virgin.net/david.drysdale/guile/tutorial.html
http://nis-www.lanl.gov/~rosalia/gnudl-doc/learn_libguile_toc.html
http://theoryx5.uwinnipeg.ca/gnu/guile/guile-user.html#SEC_Top
http://www.nada.kth.se/~mdj/guile-ref/guile-ref_toc.html
http://www.red-bean.com/guile/guile/old/3540.html
http://nis-www.lanl.gov/~rosalia/mydocs/guile-user.html
http://www.cs.utexas.edu/users/lavender/courses/scheme/
http://www.cstr.ed.ac.uk/projects/festival/manual/festival_8.html#SEC24
http://www.cs.ccu.edu.tw/~dan/tutorials.html
http://www.wcug.wwu.edu/~randyman/COMPUTERS/SCHEME/start.htm
http://www.dmoz.org/Computers/Programming/Languages/Lisp/Scheme/Teaching/
http://www.cs.caltech.edu/~cs181/doc/
http://cis.csuohio.edu/~hysockel/Links/Documents.html

12 Authors

We believe in Software Freedom and Ethics, the GNU's way.

1.	A. Balamurugan	Developer	bala@zresearch.com
2.	A. Nagappan	Developer	nagappanal@yahoo.com
3.	Anand Avati	Developer	avati@zresearch.com
4.	Anand Babu Periasamy	Project Manager	ab@zresearch.com
5.	Gopal Narayanan	Developer	gopal@debian.org
6.	Harshavardhana Ranganath	Developer	harsha@zresearch.com
7.	H.E. Ramesh	Developer	ramyog_2000@yahoo.com
8.	K. Nirranjan	Developer	nirranjan@yahoo.com
9.	K. Viswanathan	Developer	gnuvisu@yahoo.com
10.	Mridul Jain	Developer	gnuindian@yahoo.com
11.	Parag Mehta	Developer	pm@gnuos.org
12.	Ray Van Dolson	Developer	rayvd@bludgeon.org

Texi documentation written by K. Viswanathan gnuvisu@yahoo.com and revised by Anand Babu Periasamy ab@zresearch.com

13 URLs

`Homepage` [URL]
`http://www.gnu.org/non-gnu/freehoo/`

`Download` [URL]
`http://savannah.nongnu.org/download/freehoo/`

`CVS` [URL]
To know project information like Bugs, Updates, Support, Patches, Tasks, News, Development Status, Activity Percentile, Project Activity visit `http://www.gnu.org/non-gnu/freehoo/`

`Mailing` *list* [URL]
Freehoo has its own mailing list. The mailing list is for support, reporting bugs, mailing announcements. You are welcome to subscribe.

To Subscribe/Unsubscribe visit, `https://savannah.nongnu.org/mail/?group=freehoo`.

`Bugs` [URL]
You are welcome to send bug reports about freehoo to `https://savannah.nongnu.org/bugs/?group=fr`
The bugs that you think are of the interest to the public (i.e. more people should be informed about them) can be Cc-ed to the above mailing lists.

Before actually submitting a bug report, please try to follow a few simple guidelines.

1. Please try to ascertain that the behavior you see really is a bug. If Freehoo crashes, it's a bug. If freehoo does not behave as documented, it's a bug. If things work strange, but you are not sure about the way they are supposed to work, it might well be a bug.
2. Try to repeat the bug in as simple circumstances as possible.
3. If freehoo has crashed, try to run it in a debugger, e.g. 'gdb 'which hoo' core' and type `where` to get the backtrace.
4. Find where the bug is, fix it and send the patches. :) (see Chapter 14 [Guidelines for submitting a patch], page 31)

Send your specific queries to Anand Babu Periasamy `ab@zresearch.com`

14 Guidelines for submitting a patch

1. Copy the latest CVS version of `freehoo` directory as `freehoo-hack`.
2. Make changes in your `freehoo-hack` directory.
3. Create patch using
   # diff -pruN freehoo freehoo-hack > freehoo-patch-title
4. Mail the patch file '`freehoo-patch-title`' to the mailing list with subject prefixed with '`PATCH:`'.
   Please send only text mails with patch as a part of the message body. Don't update '`ChangeLog`' file, instead add your comments at the beginning of the body.

15 Portability

Since freehoo uses GNU Autoconf for building and configuring, and avoids using '`special`' ultra–mega–cool features of any particular Unix, it should compile (and work) on all common Unix flavors.

Various freehoo versions have been compiled and tested under GNU/Hurd and GNU/Linux. However freehoo can be easily ported to any POSIX complaint platform with Guile and Readline ports. If you compile it on an architecture not listed here, please let us know so that we can update it. (see Chapter 13 [URLs], page 30)

16 License

The license of freehoo consists of the GNU GPL plus a special statement giving blanket permission to link with non-free software. This is the license statement as found in any individual file that it applies to:

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2, or (at your option)
any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this software; see the file COPYING.  If not, write to the
Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
02111-1307 USA

As a special exception, the Free Software Foundation gives permission
for additional uses of the text contained in its release of freehoo.

The exception is that, if you link the Freehoo with other files to
produce an executable, this does not by itself cause the resulting
executable to be covered by the GNU General Public License.  Your use
of that executable is in no way restricted on account of linking the
Freehoo code into it.

This exception does not however invalidate any other reasons why the
executable file might be covered by the GNU General Public License.

This exception applies only to the code released by the Free Software
Foundation under the name freehoo.  If you copy code from other Free
Software Foundation releases into a copy of freehoo, as the General
Public License permits, the exception does not apply to the code that
you add in this way.  To avoid misleading anyone as to the status of
such modified files, you must delete this exception notice from them.

If you write modifications of your own for freehoo, it is your choice
whether to permit this exception to apply to your modifications.  If
you do not wish that, delete this exception notice.
```

Concept Index

Command Index

Procedure Index