Mcabber User Guide

franky

October 11, 2008

Contents

Introduction

This guide was written for users who are new to mcabber[4] which is a small command line client for the Jabber protocol[6] as an introduction. This introduction may become a comprehensive guide to every aspect of mcabber later, but the first versions should only be regarded as an overview of all features. Feel free to bug me with corrections of spelling errors or with contributions to the content.

Conventions used in this guide:

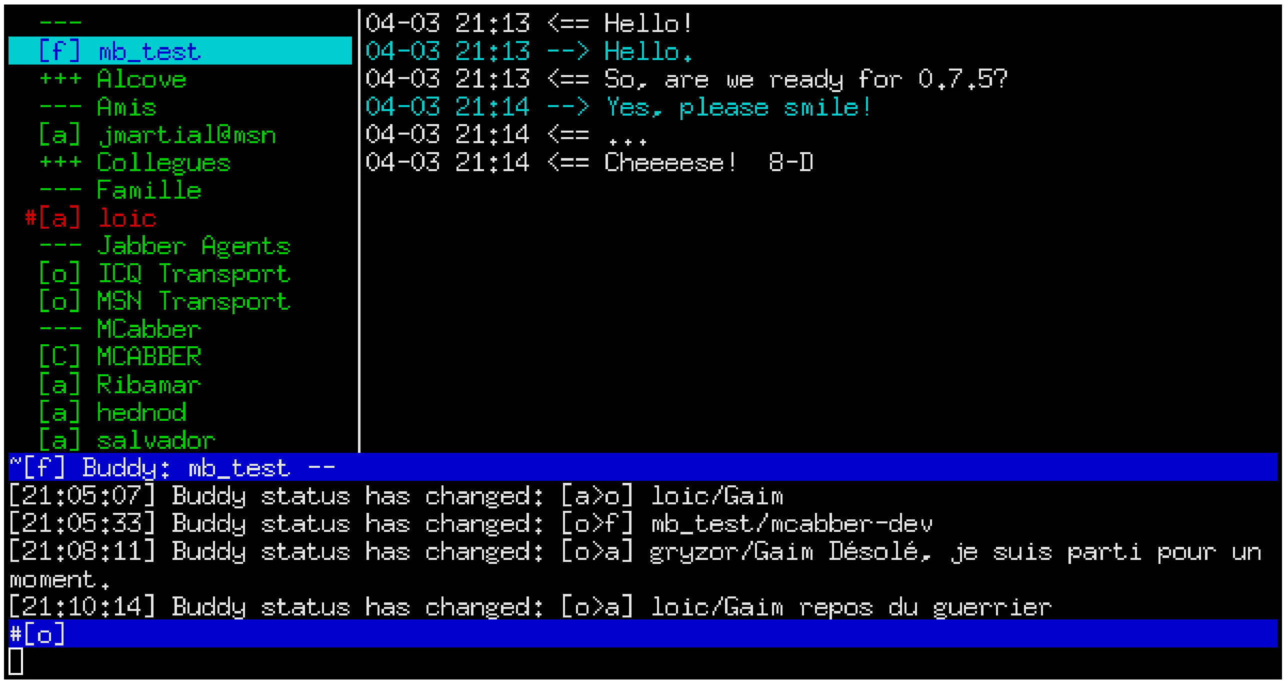
`/command`	This is a command for mcabber which can only be executed during runtime.
`command`	A command for mcabber which can be given in the configuration file, too.
`$command`	The $-sign precedes shell commands.
variable	This is a variable which can be set in the configuration file or during runtime.

Chapter 1

Basic Usage

After installing mcabber, you should create a configuration file first. The easiest way of doing this is to copy the sample configuration file[2] to ~/.mcabber/mcabberrc. Then, you only have to change the variables *username* and *server* before you can, finally, start mcabber. When the Login was successful, mcabber might look similar to the following screenshot:

The screen is divided into 4 regions: The roster(sometimes called buddy list) in the upper left, the chat window/buffer in the upper right, the input line at the bottom and the log window above the input line. Those two blue lines which are below and above the log window are status lines. The lower one displays the general mcabber status and the upper line the status of the selected contact. The screenshot shows mcabber in "chat mode" which can be entered by pressing enter or sending a message when a contact is selected. Leaving chatmode is done by pressing Esc. To move between your contacts, you can use the buttons Pg Up/Pg Down. If you look at the roster of the screenshot, you will notice that the contact "loic" is marked red. That's the default for the "unread" color, so "loic" sent a message to you, but you have not read it, yet. Moving by pressing Pg Down would need 6 key presses, so it might be faster to use `/roster unread_next` to quickly jump to the next unread buffer. Which brings us to the interface of mcabber. Since mcabber has only one input line, there has to be some way of distinguishing mcabber commands from normal text messages. As you might have guessed already, all mcabber commands start with a slash, while any other line which you have entered will be sent as a message to the selected buddy. You won't find a complete list of all commands here, since mcabber has already got a great help system. It's at your fingertips when you enter `/help`. Furthermore, you



```
  ---                   |04-03 21:13 <== Hello!
  [f] mb_test           |04-03 21:13 --> Hello.
  +++ Alcove            |04-03 21:13 <== So, are we ready for 0.7.5?
  --- Amis              |04-03 21:14 --> Yes, please smile!
  [a] jmartial@msn      |04-03 21:14 <== ...
  +++ Collegues         |04-03 21:14 <== Cheeeese!  8-D
  --- Famille           |
 #[a] loic              |
  --- Jabber Agents     |
  [o] ICQ Transport     |
  [o] MSN Transport     |
  --- MCabber           |
  [C] MCABBER           |
  [a] Ribamar           |
  [a] hednod            |
  [a] salvador          |
^[f] Buddy: mb_test --
[21:05:07] Buddy status has changed: [a>o] loic/Gaim
[21:05:33] Buddy status has changed: [o>f] mb_test/mcabber-dev
[21:08:11] Buddy status has changed: [o>a] gryzor/Gaim Désolé, je suis parti pour un
moment.
[21:10:14] Buddy status has changed: [o>a] loic/Gaim repos du guerrier
#[o]
```

Figure 1.1: screenshot of mcabber

can choose the language of the help system by setting *lang* to de, en, fr, it, nl, pl, ru or uk.

1.1 Multi-user Chat

Jabber offers more than other one-on-one chats. There is an extension[10] which adds many-to-many chat, very similar to systems like IRC. Mcabber can participate through the command `/room`. If you happen to find big rooms, you should play a bit with `/color muc` to make the conversation more readable. Last but not least, you might be interested in the channel mcabber@conf.lilotux.net as a new mcabber user. Feel free to drop by and tell us about your experiences with mcabber or this guide!

1.2 Transports

Transports, also known as Gateways, are Jabber server components which are used for translating your Jabber messages for proprietary networks like ICQ, AIM or MSN. At the time of writing, mcabber can't register transports, but, if you've registered a transport with another client, you'll be able to use them within mcabber. Transports are, basically, just another contact on your roster. Thus, using transports is done by sending your status, maybe explicitly with `/status_to`.

1.3 Symbols

Most users may be puzzled when they see all those different characters in the roster, the status bar and in the chat window for the first time. Some of those symbols might be obvious, but others aren't descriptive at all. So, here is a list of all characters and their meanings, structured per window where they were taken from.

Roster	
Symbols	Rosteritem
---	unfolded Group
+++	folded Group
[ ]	buddy or chat room(MUC)
{ }	buddy who can't see your status
+	buddy is typing
.	buddy stopped typing
_	buddy status: offline
o	buddy status: online
a	buddy status: away
d	buddy status: do not disturb
n	buddy status: not available
f	buddy status: free for chat
?	buddy status unknown
x	muc status: not connected
C	muc status: connected

Chat Window	
Symbols	Meaning
-->	sent message
<==	received message
~	message was encrypted with GPG
O	message was encrypted with OTR
***	info message
#<#	error message

Upper Status Line	
Symbols	Meaning
~	chat mode enabled
*	buffer is scroll-locked
[C]	chatstate: composing
[A]	chatstate: active
[I]	chatstate: inactive
[P]	chatstate: paused
[G]	chatstate: gone

Chapter 2

Logging

Some people write their diaries to remember special moments. Others merely save their chat logs, which are often full of personal details, too. Of course, mcabber can keep your chat history and fortunately, the format of these logs is easy enough to read with your favourite text editor. Moreover, searching is possible with standard unix tools like `$grep`.

2.1 Enable Logging

To enable logging, you'll have to add some lines to your configuration file:

```
set logging = 1
set load_logs = 1
```

Those variables tell mcabber to log the chat history and to load old logs when you restart it. Please note that there are a lot of other variables affecting logging. You can e.g. choose whether you want to log Multi-user Chats(*log_muc_conf*) or whether you want to log status changes(*logging_ignore_status*). You'll find them all in the sample configuration file[2] if you search for "History Logging".

2.2 Grouping logs

If you know other instant messengers, you might be used to a concept called "metacontacts". They can be used to group users, which have multiple representations in your roster (e.g. because they have work and home accounts).

Unfortunately, mcabber doesn't support metacontacts yet, but at least you can group the history of different contacts together. This basically works by merging those histories and then creating a symlink frome one to the other. Here's an example with the histories of foo_work@jabber.org ond foo_home@jabber.org:

```
$ cd ~/.mcabber/histo/
$ merge_history.py foo_work@jabber.org foo_home@jabber.org > foo
$ mv foo foo_home@jabber.org
$ rm foo_work@jabber.org
$ ln -sf foo_home@jabber.org foo_work@jabber.org
```

The script merge_history.py is included in the mcabber tarball.

2.3 Converting centerim logs

You've got old centericq[8] or centerim[1] logs and want to keep them in mcabber? There's a script called cicq2mcabber.pl in the contrib directory of the mcabber tarball which can be used to do that. Now you only have to figure out where exactly centerim saves the history of a particular contact(somewhere under ~/.centerim/), convert this file to the mcabber history format and save it to ~/.mcabber/histo/ with the Jabber ID as the filename.

Chapter 3

Scripting

This chapter will introduce you to the scripting facilities of mcabber.

3.1 Aliases

Aliases are used to map complex commands to a word. `/roster search` is one of the commands worth remapping because it will be used every day. If you want to remap it to `/rs`, issue the command `alias rs = roster search`.

3.2 Internal Hooks

Internal Hooks can be used to let mcabber execute commands when a special event occurs. At the time of this writing, only 2 hooks were defined:

- *hook-post-connect*
- *hook-pre-disconnect*

The time of the execution is self-explaining, just set those variables to the command you want to execute.

3.3 External scripts

Mcabber will execute the external script specified in the variable *events_command* whenever a new message is received. It will be called at least with 3 parameters:

1. MSG or STATUS
2. IN, OUT or MUC if the first argument was MSG or one character of "_OFDNAI" if the first argument was STATUS
3. The JID which triggered the event
4. (optional) filename of the file which contains the message body

If you need the message body for your event script, you'll have to set the option *event_log_files* to 1. The script has to delete that file after it processed it. To execute mcabber commands, you'll have to compile mcabber with FIFO support(`$./configure --enable-fifo`). Then activate it by setting the variable *fifo_name* to the filename of the fifo in your config file. There will be some verbose debug output in the logging window if you don't set the variable *fifo_hide_output* to 1.
There is another variable which might be interesting for the developer of external scripts. If *statefile* is set to a filename, mcabber saves the JIDs of all unread buffers to that file, one per line.

Chapter 4

Encryption

Encryption is one of the greatest advantages of Jabber compared to similar "legacy" networks like ICQ, MSN, AIM, etc. With encryption, you'll be able to communicate in hostile network environments with a clear conscience, because almost nobody will be able to spy on you. Practically, every connection which is established when using the Jabber protocol could be encrypted. First of all, you can connect to your Jabber server via ssl, which will prevent anyone between you and the server from sniffing your plaintext messages or your password. In mcabber you could enable this feature by adding

```
set ssl = 1
set ssl_verify = -1
```

to your configuration file. The next step is server to server encryption, which you can't control, unless you're the admin of your server. So, theoretically it would be possible to spy between Jabber servers, or maybe your buddy doesn't use an ssl connection. Of course, if you are truly paranoid, you won't trust the server admins but only your buddy. So, to encrypt messages between you and your buddy, end-to-end encryption can be used. There are two protocols for accomplishing this: The first one is OpenPGP/GnuPG, which is a semi standard[9] in the Jabber world. The other protocol is Off-the-Record(OTR) Messaging, which was designed by cryptographers Ian Goldberg and Nikita Borisov[5]. Although OTR isn't a standard, a lot of developers have implemented it for their instant messaging clients. The main reason might be the availability of libotr[7], an open-source library from the authors of OTR which does all the hard work. GnuPG is considered to be more mature, but it also has a major drawback: You can't encrypt messages between Jabber and legacy networks. Because naturally, the clients of other networks won't support Jabber extensions. OTR can fill that gap, because it only uses the message body as an envelope for the whole protocol.

I will now describe in detail how to use both protocols with mcabber.

4.1 OpenPGP

First of all, I'm assuming that you've already generated a gnupg key which you want to use to encrypt Jabber messages. If you haven't done this before, go to `http://gnupg.org/documentation/` first.

Using PGP encryption with mcabber should be straight-forward, but you have to restart mcabber for configuring it. To enable PGP for mcabber, you must add two lines to your configuration file:

```
set pgp = 1
set pgp_private_key = 23DEADBEEFDEAD42
```

The first one enables OpenPGP and the second line tells mcabber which PGP key it should use. If you didn't notice: This hexadecimal number is the last characters of your PGP key fingerprint, get it with `gpg --fingerprint --list-secret-keys`. If you have no problems writing your passphrase to your hard disk, you could also set the variable *pgp_passphrase* in your config file. Otherwise mcabber will ask you for your passphrase at every start-up. From now on, mcabber will magically use PGP encryption whenever it can. Basically this works because presence messages of users using PGP are signed with their key. If you happen to have a contact whose Jabber client doesn't conform to that part of the standard, you can manually give mcabber their key fingerprint:

```
pgp setkey foo@bar.net D9940C9BB1B92310
```

4.2 OTR

Enabling OTR in mcabber works in almost the same manner as PGP. But instead of generating your encryption key by hand, mcabber will create one for you when it starts. This might take some time and you can speed it up by generating randomness by typing, moving the mouse, writing to hard disk etc. But first of all, you must add the following lines to your configuration file and restart mcabber:

```
set otr = 1
otrpolicy default manual
```

The first line is analogous to PGP. The second one gives the OTR subsystem the hint that you want to be able to start encrypted OTR channels manually(by issuing `/otr start`). If you want that they are also "magically" started, you should change `manual` to `opportunistic`. The opportunistic mode adds some whitespace characters to the first message sent to a buddy. So, his client can find those characters and establish an OTR channel. Please note that "establishing" the channel could take some time, depending on the delay of exchanging messages between both ends. (For the technically inclined: Four messages have to be sent before the channel is fully established. The whole protocol is described in detail at `http://www.cypherpunks.ca/otr/Protocol-v2-3.1.0.html`)

Keep in mind that an established channel with a buddy isn't per se secure. It could be possible that you become a victim of a Man-in-the-middle attack. To make it secure, you'll have to exchange the fingerprints of the keys through a trusted medium (telephone, in person, etc). You can see your own fingerprint by using the command `/otr key` and the fingerprint of your chat buddy by `/otr fingerprint`. Check `/help otr` to find out how to trust the fingerprint now. There is even another way for exchanging fingerprints: The Socialist Millionaires' Protocol[3]. It's basically a cryptographic secure way for two parties with the secret information x and y to check whether x==y without revealing any information about the secrets. Unfortunately there aren't many clients supporting that feature yet. In mcabber it is mapped to the commands `/otr smpq bob@foo.net your_secret` for the initiating party and `/otr smpr alice@foo.net your_secret` for the answering party. If both parties supply the same secret, the fingerprints can be marked trusted.

Bibliography

[1] Centerim. `http://www.centerim.org`.

[2] Sample configuration file for mcabber. `http://www.lilotux.net/~mikael/mcabber/hg/?raw-file/tip/mcabber/mcabberrc.example`.

[3] Chris Alexander and Ian Goldberg. Improved user authentication in off-the-record messaging. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*. ACM, 2007. `http://www.cypherpunks.ca/~iang/pubs/impauth.pdf`.

[4] Mikael Berthe. Mcabber. `http://www.lilotux.net/~mikael/mcabber/`.

[5] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use pgp. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*. ACM, 2004. `http://www.cypherpunks.ca/otr/otr-wpes.pdf`.

[6] XMPP Standards Foundation. Jabber/xmpp protocols. `http://www.xmpp.org/protocols/`.

[7] Ian Goldberg, Chris Alexander, and Nikita Borisov. Off-the-record messaging. `http://www.cypherpunks.ca/otr/`.

[8] Konstantin Klyagin. Centericq. `http://thekonst.net/en/centericq`.

[9] Thomas Muldowney. *XEP-0027: Current Jabber OpenPGP Usage*. XMPP Standards Foundation, 1999-2006. `http://www.xmpp.org/extensions/xep-0027.html`.

[10] Peter Saint-Andre. *XEP-0045: Multi-User Chat*. XMPP Standards Foundation, 1999-2008. `http://www.xmpp.org/extensions/xep-0045.html`.