The GgrTF v0.7.4.1 User's Guide

Matti Hämäläinen
Jarkko Vääräniemi

The GgrTF v0.7.4.1 User's Guide

by Matti Hämäläinen and Jarkko Vääräniemi

Publication date 2021

Table of Contents

List of Tables

Chapter 1. Preface

What is GgrTF?

GgrTF is a *framework* of AND for triggers to ease playing of BatMUD [http://www.bat.org/] with TinyFugue [http://tinyfugue.sf.net/] MUD-client. GgrTF is designed for modularity and easy development. Our primary development goals are good overall quality of code and maintainability. These are also some of the key elements, that differentiate GgrTF from most other publicly available scripts.

GgrTF is being developed for TF5 of TinyFugue [http://tinyfugue.sf.net/] MUD client under Linux and Solaris platforms. Users of GgrTF are mostly users of UNIX-like systems, but there are several who run TF under Microsoft Windows operating system. Version 0.6.9.15 was the last that supported TF4.

The current version (as of v0.7.4.1 release) has modules for following character classes:

- Mages (The Brotherhood of Wizardry [http://www.bat.org/help/guilds?str=The+Brotherhood+of+Wizardry])
- Channellers (The Guild of Channellers [http://www.bat.org/help/guilds?str=The+Guild+of+Channellers])
- Merchants (The Master Merchants [http://www.bat.org/help/guilds?str=The+Master+Merchants])
- Barbarians (Barbarian Guild [http://www.bat.org/help/guilds?str=Barbarian+Guild])
- LoCs (Lords of Chaos [http://www.bat.org/help/guilds?str=Lords+of+Chaos])
- Tarmalens (The Followers of Tarmalen [http://www.bat.org/help/guilds?str=The+Followers+of+Tarmalen])
- Druids (The Humble Druids [http://www.bat.org/help/guilds?str=The+Humble+Druids])
- Nuns (Sisters of Las [http://www.bat.org/help/guilds?str=Sisters+of+Las])
- Spiders
- Tigers
- Alchemists

In non-guild related sense, GgrTF has most of the basics expected from such triggerset. There is a prot-management and reporting system, skill- and spell-handling, way to easily run commands on each battle round (and more), plus following special modules:

- 'pss'-output mangler
- Hit statistics counter
- Spellchants to names translator
- Party member autoplacer
- Party prot tracker
- Reagent pouch helper

- Identify spell beautifier

And more ...

Is GgrTF for Me?

It is impossible to tell for certain, really. If you are averse of doing some digging of your own, or adjusting some trigger code, it may not be. GgrTF is meant to work as a "platform" and a base, not as a complete bells and whistles-type solution. In the end, you may have just to give it a try.

When I started working on GgrTF, I was just "scratching an itch", and that itch still continues, though few others have joined my efforts on scratching. During the two years I have been developing GgrTF, most of that time I've been scratching my own back, doing stuff mostly for myself. Thus, if you like the design decisions and features I like, play the same guilds as I (though some guilds which I haven't ever touched are supported through efforts of other developers) it is likely that you will like GgrTF.

Also GgrTF is not all that well documented (although that is one thing we are trying remedy) and *requires basic knowledge of how TinyFugue [http://tinyfugue.sf.net/]* and its macro language works. As such GgrTF is not a ready out-of-the-box experience for some users and never will be. *However, if you are able to crank it up and working, and are content with what is readily offered, then it might be something for you.*

All in all, GgrTF *might* be something for you, especially if you like things the way I like them. But on the other hand GgrTF might not be for you, and there are several alternatives out there for those who do not find it satisfactory. Below I have tried to list some pros and cons of using GgrTF.

- Pros:
  - Good general quality of code. As somebody put it, "you do not instantly get a headache from just looking at the code."
  - Modularity, easily extendable for most part.
  - Updated regularly, most bugs get fixed in upstream.
  - User-level documentation exists.
- Cons:
  - Only small number of BatMUD [http://www.bat.org/]'s guilds are specifically supported.
  - Requires bit more programming knowledge than some other solutions.
  - Due to nature of the developer's reincs, is biased towards caster-type functionality than tanks. Thus the support for tank guilds is not very good, although some guilds are supported.

History of GgrTF

As told by Ggr

Way back in march of 2004, I made one of the worst mistakes in my life - I created a character in BatMUD [http://www.bat.org/]. I never thought of myself to be much of a gamer, I've always been more interested in more technical things, programming and a little bit of digital electronics and hardware, so mostly my gaming experience consisted of some c64 games, Nethack and some CRPGs like Fallout. I guess the RPG statmania was one reason that got me hooked in the first place, plus the multiplayer interaction and social aspects.

Started off playing with plain telnet for the first month or so, then installed TinyFugue [http://tinyfugue.sf.net/], because it seemed to be the most popular of the clients available for UNIXish platforms. First I didn't use any triggers at all, TF was just there to work as better terminal and separating the input from output .. and the game was confusing enough to start with, triggers would have just complicated things. At least I felt so back then.

At some point, however, my playing style advanced and need for automating certain things became apparent. I researched some of the available scripts, but most of them were either badly designed, full of very apparent mistakes and bugs, or simply not maintained anymore. Some people (highbies mainly) seemed to have some good stuff, but as usual, they only shared inside small circles of their own.

Fancying myself as a programmer of some sort, off I went, developing my own piece of turf, starting out very simple, expanding and improving when needed. That continued for about a year or a bit over, then the situation got out of hand - I started "releasing" GgrTF periodically on my BatMUD [http://www.bat.org/] related web-page.

The first ever released version of GgrTF was v0.3.0, which was extremely simple and hacky. It supported just channeller/mage stuff and had a broken prots management system. The 0.3.x-series continued up to 0.3.13 or so, after which some bigger changes were made and 0.4.0 was released.

At some point of 0.4.x-series development, more people started noticing GgrTF. I believe some even started using it, or at least used parts of it in their own code. v0.4.x was somewhat short-lived, soon major restructuring was done in the prot management system and it was time for 0.5.0. As the development progressed, it soon again became apparent that bigger changes were needed ... v0.5.5 was the last stable release in that series.

Development of the new features took somewhat longer time than originally expected, there were literally dozens of internal version landmarks (v0.5.6.x, v0.5.9.x, v0.5.10.x, v0.5.11.x and finally even v0.6.0-preX series), which preceeded the final v0.6.0-release in early august 2006. This new release marked a major improvement in code quality and number of features, and also this user's manual was included for the first time. It was also the first time when other people joined up and did some work on few features.

GgrTF License

Chapter 2. Installation

This chapter assumes that you have already succesfully installed TinyFugue [http://tinyfugue.sf.net/] in your system and you know how to use it (basic commands, loading of macros, etc.) It is also assumed that you know how to use your operating system of choice, be it some flavour of UNIX or Windows, although some parts of the instructions are given in step-by-step manner.

Note

Throughout this manual, when GgrTF files are referenced in configuration examples etc., I have used "*ggrtf/*" -directory as path where the script files are located relative to your "home directory". You may need to substitute it with whatever you have installed your copy of GgrTF files in.

There are basically two ways how to get GgrTF, release packages and the Mercurial-repository. Releases are considered as stable snapshots, which should be relatively bug-free, but releases are done somewhat infrequently and at least in this phase we don't backport bugfixes to release versions.

If you are unsure which version to choose, use the latest packaged release.

Releases

To get the latest stable version, head to the downloads-section of GgrTF's homepage [http://tnsp.org/~ccr/ggrtf/] and pick either the newest tarball (*.tar.gz) or zip-archive (*.zip):

- *.tar.gz packages are for UNIX-like systems, such as Linux, *BSD, etc. Please note that the ZIP-packages are meant for Windows only and WILL NOT WORK under UNIX or OS X version of TinyFugue!
- *.zip packages are for Windows version of TinyFugue (the files have been converted to CRLF line endings.)

Both package types also have equivalent PGP/GnuPG [http://www.gnupg.org/] signature files (*.asc), which can be used to cryptographically verify the authenticity of files.

To "install" the package, you simply unpack it to appropriate directory, typically under your home directory. Under UNIX-style system:

```
cd $HOMEtar xzvf ggrtf-0.7.4.1.tar.gz
```

After that, you should have directory $HOME/ggrtf/ with all the *.tf files in it. If so, you can continue to the configuration part.

Development version

Another way to get GgrTF is to go to the very source, the development source repository. This repository contains the latest bleeding edge features, but is also a fast moving target. Changes are directly committed by developers almost in real time, sometimes what you download may be severely broken. Also the documentation will not be updated for every little change, so you may have to figure out and work around possible backward incompatibilities yourself.

New features and bugfixes may sometimes be worth taking the risk. On the other hand, it is all up to what you want and whether you can manage to handle the possibly arising problems.

We utilize Mercurial [http://www.mercurial-scm.org/] (Hg) for GgrTF's version management. Mercurial is a freely available distributed version control system, for which there are clients available for most common platforms, including Windows, Linux and other flavours of UNIX.

GgrTF's Mercurial repository is hosted at tnsp.org, and is available from following location: https://tnsp.org/hg/batmud/ggrtf/. The same URL can be used with a web browser to browse version history.

Checking out

You can download (aka "clone") the GgrTF repository with command line Mercurial client with following command: **`hg clone https://tnsp.org/hg/batmud/ggrtf/ dest_dir`** In which *"clone"* means making a local copy of the repository under directory *dest_dir*.

You will most probably want to download the repository to some specific place, personally I prefer to use ~/ggrtf/ (aka "ggrtf/" under user's home directory) under UNIX. Thus typically you would use following command: **`hg clone https://tnsp.org/hg/batmud/ggrtf/ ~/ggrtf`** If you are using Windows, and wish to use the development version, you can use the regular Mercurial command line client, or a GUI interface such as TortoiseHg [http://tortoisehg.bitbucket.org/].

The procedures for cloning and updating your local copy of the repository are otherwise similar, except the directory/folder path is different, under Windows XP the path would be *C:\Documents and Settings\<username>\ggrtf*. For example, if your username is "Ggr", you could use the following command: **`hg clone https://tnsp.org/hg/batmud/ggrtf/ "C:\Documents and Settings\Ggr\ggrtf\"`** Under Windows Vista and 7, the path is of the format *C:\Users\<username>*

Keeping up to date

After you have checked out your own copy of GgrTF's code repository, you usually wish to periodically update it. This is done easily with the following commands: **`cd ~/ggrtf`** **`hg pull`** **`hg update`** Notice that since Mercurial is a distributed version management system, you can easily keep your own local changes by committing them (*'hg ci'*) and then, instead of using 'pull' and 'update', you can use 'hg fetch' to merge your local changesets automatically. Refer to Mercurial documentation.

If you plan on making modifications to GgrTF, it is recommended that you familiarize yourself with the basics of Mercurial workflow, making commits, fetching, etc. It will help you with maintaining your changes locally or sending patches to us.

Chapter 3. Configuration

After extracting GgrTF files into the appropriate directory folder, you will need to create (or change) configuration for TinyFugue [http://tinyfugue.sf.net/] to load GgrTF modules and possibly change some of your settings in BatMUD [http://www.bat.org/].

If you are upgrading GgrTF from a previous version, it is possible that you do not have to change your configuration. But it is neverthless recommended that you skim through this section, the ChangeLog and module-specific sections of this manual in case of any significant, backwards incompatible changes.

TF configuration file

Typical way to use GgrTF is to load the script modules at startup. This is accomplished via TinyFugue's configuration file, commonly referred as "tfrc", location of which depends on your operating system and environment.

- UNIX-like systems, like Linux: *~/.tfrc* or *$HOME/.tfrc* (aka a file named ".tfrc" in your user home directory.)
- DruWare Win32 port of TF: Windows XP/2k/NT - *C:\Documents and Settings\username\tfrc*, Windows 7/Vista - *C:\Users\username\tfrc*

As a basis for your TF configuration, you can use the example-tfrc.txt provided with GgrTF. You will need to edit it to suit your guilds and certain settings. However, the example-tfrc is only a suggested layout of configuration, mostly to show the order GgrTF requires modules and settings to be loaded. It does not contain everything you can possibly do with TF.

Take note of the order of how different parts of GgrTF are loaded and where certain variables are set. Important thing is to have the order right, certain modules depend on other modules, and while GgrTF will usually print warning message(s) if the depencies are not met, sometimes this is not possible and erratic behaviour will occur.

```
;; Turn logging off, while we set things up
/log off

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Personal settings
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Set player name before loading scripts
/eval /set set_plrname=Ggr

;; Savefiles should be stored under $HOME/.ggrtf/
/eval /set set_datapath=%{HOME}/.ggrtf/

;; We want savefiles to use filename prefix "bat-"
/eval /set set_saveprefix=bat-


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Character set / locale
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; You MAY need to set your character set encoding/locale to have
;; non-ASCII characters work properly in BatMUD. This is outside of
```

```
;; the scope of GgrTF, but it is discussed in more depth in the
;; user's manual.

;/setenv LANG=en_US.iso88591

;; You may also try following settings if ISO-8859-1 does not work:
; /setenv LANG=en_US.iso885915
; /setenv LANG=en_US.latin1


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Load GgrTF
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; We define a helper macro to load modules from
;; under user's home directory $HOME/ggrtf/
/def -i gloadmod = /eval /load %{HOME}/ggrtf/%{*}

;; Preloader module MUST be loaded first
/gloadmod ggrtf-pre.tf

;; And the main module after that ...
/gloadmod ggrtf.tf

;; Some useful modules: mangler, placer, spellname translator
/gloadmod gm-pssmangle.tf
/gloadmod gm-magical.tf
/gloadmod gm-tgtcast.tf
/gloadmod gm-pplacer.tf
/gloadmod gm-spellnames.tf
/gloadmod gm-tf5.tf

;; Load additional modules now
/gloadmod gm-rais.tf
/gloadmod gm-tarma.tf
/gloadmod gm-nun.tf


;; Load previously saved settings after all modules are initialized
/gload


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Load keyboard support files
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Bind some keys
/gloadmod kbd-xterm.tf
/gloadmod kbd-tf5def.tf

;; Some personal keybinds here ...
;/def -i -b'^[Om' = @smode
;/def -i -b'^[Ok' = @cmode


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;; Re-start logging (examples below, uncomment one)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Log to tiny.log in CWD (current working directory)
; /log on

;; Log to tiny.log in user's home directory
; /eval /log %{HOME}/tiny.log

;; Log to tiny-YYYY-MM-DD.log in user's home directory
; /eval /log %{HOME}/tiny-$[ftime("%Y-%m-%d")].log
```

Character set encoding and locales

One thing that is out of scope of GgrTF itself, is the issue of character set translation and locales. This basically means the way different characters are encoded and interpreted, how different values map to characters and vice versa, for example a simple encoding might be A=1, B=2, C=3, etc.

One of the first and most widely used character encodings on computer systems is known as 7-bit ASCII [http://en.wikipedia.org/wiki/ASCII]. This encoding allows 128 symbols or characters, and was commonly used for decades. Unfortunately, 128 symbols is nearly not sufficient enough to represent all the different alphabets or symbols used around the world (scandinavian characters, cyrillic alphabet, etc.) Thus other encodings were invented, most of them incompatible with each other, until introduction of Unicode standard and especially Unicode UTF-8 [http://en.wikipedia.org/wiki/UTF-8].

The de-facto character set used in BatMUD is called "ISO-8859-1" which is a 8-bit encoding that allows 7-bit ASCII characters and 8-bit encoding of scandinavian characters (A and O with dots, etc). This is mostly due to BatMUD's origin in Finland, where ISO-8859-1 has been most deployed.

However, increasing number of systems are starting to use Unicode "UTF-8" encoding, and other encodings are also still used. Unicode is a good thing in itself, and UTF-8 allows compatible 7-bit ASCII characters, but 8-bit and over are not compatible with ISO-8859-1. This is why if your system uses UTF-8, other people in BatMUD may see your non-ASCII characters incorrectly.

Now that we know the issue, what can we do? Sometimes the solution may be very simple, but in many cases rather complex and very dependant on your setup - e.g. where you run TF, is it a shell or running locally, etc. Below is a list of tips and pointers, which may be of some help.

UNIX-likes / Linux

For Linux/UNIX, you need to have the ISO-8859-1 locale installed/configured. How this is done depends on your OS distribution. For example in Debian, you need to use **`dpkg-reconfigure locales`** and add some ISO-8859-1 locale, for example *en_US.iso88591*.

On Ubuntu, things get a bit more complicated:

1. Edit /var/lib/locales/supported.d/local with your favorite editor, and on the last line add: *en_US.ISO-8859-1 ISO-8859-1*. It should look something like this:

   ```
   en_US.UTF-8 UTF-8
   en_US.ISO-8859-1 ISO-8859-1
   ```

2. Then, run **`sudo dpkg-reconfigure locales`**

On other Linux distributions and UNIX-like platforms the methods for installing locales vary. Your best bet is to Use the Google [https://www.google.com/search?q=linux+install+locale+iso-8859-1].

If your system is using some locale other than ISO-8859-1 such as UTF-8, you will need to make the terminal use ISO-8859-1 despite the system-global setting, or alternatively use some software like GNU Screen [https://www.gnu.org/software/screen/] to "translate" between your system's and BatMUD's ISO-8859-1. Some information about how to make GNU Screen do that, can be found from this website: http://aperiodic.net/screen/terminal.

However, if you choose not to use 'screen', you will have to either start your terminal program (inside which TinyFugue will run) with the locale set to ISO-8859-1, OR if your terminal supports the feature, use the following shell script wrapper which will use terminal control codes to change the effective character translation. *This feature is not supported by all terminals, XTerm and Rxvt are known to support it, however.* You will need to copy+paste the script, or download it from here [http://tnsp.org/~ccr/ggrtf/tf5.sh].

```
#!/bin/sh
SAVE_LANG="$LANG"
export LANG="en_US.ISO-8859-1"
printf '\33]701;%s\007' $LANG

/usr/bin/tf5 $*

export LANG="$SAVE_LANG"
printf '\33]701;%s\007' $LANG
```

You should place the script in some directory in your $PATH. You will also need to modify it to point to the correct TinyFugye executable, (e.g. change "/usr/bin/tf5" if needed.) Name the script as something like "mytf" or rename the real TinyFugue executable as "tf5-bin" and the script as "tf5", changing the script to point to tf5-bin.

Remember to set the script executable by changing its permissions, '*chmod 0755 /some/where/scriptfile*' should do it.

The idea is that you run the script instead of TF directly. The script changes your current locale and forces the terminal via special terminal control code to use it, and changes it back after TF exits.

Terminal/keyboard issues

You need to make sure that you have the correct keyboard module(s) loaded. For most Linux/UNIX terminals you want to load EITHER or BOTH *kbd-xterm.tf* and *kbd-tf5def.tf*. For TF on Windows and Mac OSX, you may wish to use *kbd-tf5def.tf* only. You may need to experiment.

Additionally, it may be necessary to enable the so-called "application mode" in your terminal.

- In OSX terminal app, the setting can be found in *Terminal / Preferences / Setting / Advanced* and turn on "Allow VT100 application keypad mode".
- In PuTTY, the application keypad mode should be enabled by default, but if it is not, see the Features panel of PuTTY's configuration, see PuTTY documentation here [http://the.earth.li/~sgtatham/putty/0.60/htmldoc/Chapter4.html#config-appkeypad].

BatMUD settings

Next, log on in BatMUD and change following settings. *It is probably best, if you use direct copy & paste to set these, as most of them are required to be set exactly as shown here for GgrTF to work correctly.*

- Line 'cutter' setting: **`cutter 9999`** Theoretically, '*cutter off*' should be better, but in practice I have noticed that in certain things it simply does not work as expected.

- Short score format for regular BatMUD: **@@sc set H:{colorhp}/<maxhp> [{diffhp}] S:{colorsp}/<maxsp> [{diffsp}] E:{colorep}/<maxep> [{diffep}] $:<cash> [{diffcash}] exp:<exp> [{diffexp}]**
- Short score format for HardCore BatMUD (HCBat): **@@sc set H:{colorhp}/<maxhp> S:{colorsp}/<maxsp> E:{colorep}/<maxep> $:<cash> exp:<exp>**
- Enable automatic short score (required for automatic updating of HP/SP/EP values on statusline): **@@sc on**

  ## Note

  If you do not wish to not see the 'sc' lines, you can turn on '/gagsc' option.

- Prompt: **@@prompt PROMPT:>** If you wish, you can add any BatMUD specific data to the prompt between ":" and ">", this substring will be stored to "status_prompt" TF variable, which can be used in /gprompt. Read more in prompt setup section.
- Listen to battle round flags: **@@battle rounds** This setting is a toggle, make sure that you set it ON ("Listening to round flags.") instead of off!
- Battle listen-level: **@@battle listen all 2** Listen level should be 2 or 3, if you are using hitstats. Otherwise it can be 1 or 0, if you prefer silence.

In case you do NOT want to change your settings to the way GgrTF needs them to be, your only option is to change the regex patterns in GgrTF's code to match your preferences (and redo the changes each time you upgrade GgrTF.)

Chapter 4. Usage

This chapter describes the structure of GgrTF, what the different modules included in GgrTF are and what functionality they provide. There are basically three flavours of modules: generic modules that are either required or optional and provide functionality unrelated to specific guilds; then there are guild-related modules, containing guild-specialties.

Each module section has short introduction about what is provided and section with macro commands and variables available. Some modules also have additional information about usage and configuration with examples.

GgrTF core (ggrtf.tf)

This is the main module and core of GgrTF. It contains helper macro functions used by all the other modules and is thus always required to be loaded. It also provides much of the basic functionality, such as:

- Statusline. Provides nice, realtimeish view of your character's current condition, applied prots and other such information.
- Prot management. Keeps note of what prots have been cast at you, and provides this information in various ways.
- Curse/degen/etc tracking. Tracks what handicaps have been cast at enemies (non-partymembers). This information is most useful in eq-parties.
- Battle round handling. Enables commands to be executed each battle round and autopss functionality.
- Heartbeat and tick prediction. Display prediction of next "tick" via a heartbeat counter.
- Functions for binding command strings to macros, or to cast spells and use skills, with or without reporting to party channel. See bindings section for more information.
- Support for state-saving. Most GgrTF settings can be saved to files and reloaded later (for example at startup) with /gsave and /gload commands.
- Prompt handling and mangling.
- Automatic "ripaction", which is performed when your opponent (monster) dies. This way you can automate small inconveniencies like looting and digging of graves.
- Keyboard movement handling and mapping. Bind your numpad keys to move around, either by walking, walking with peering in adjacent rooms (useful for merchants) or guiding your ship. Additional modes may be provided by other optional modules.
- Enemy shape string highlighting and reformatting. You can make GgrTF mangle 'scan' command's output into more readable and convenient form.
- Keep and display statistics about skills, spells and whatnots. ("/stats" command)
- *Consider-skill reporting*: Takes output of 'consider' skill and compresses the information into one prettyprinted line, which is reported. Output is name of the target, estimated experience worth and final estimation of target's toughness.
- *Combat Damage Analysis reporting*: Triggers for reporting results of CDA-skill to party report-channel.
- *Purse contents prettyprinting*: Highlites and colorizes the output of 'look at purse', and calculates total sum of money (in gold) contained in the purse.

- *Camping status handling and reporting*: Keeps note of whether you can use 'camping' skill. Reports the hp/sp/ep gained from resting.
- *Ceremony status*: Tracks status of ceremony, and provides macro "/ceremony", which executes ceremony skill only if ceremony is not active currently.
- *Path compression for map.tf*: TinyFugue distribution comes with map.tf, which enables easy creation of walkpaths (See "/help map" in TF). GgrTF adds an special purpose RLE compressor function, which reduces the length of the paths. Additional command to output the paths in format that is compatible with BatMUD's "command"-aliases is also provided.
- Plus numerous miscellaneous reporting- and helper-triggers.

Macro commands

<> = required argument, [] = optional argument

Table 4.1. GgrTF core macro commands

Command	Description
/autopss	Toggle autopss-functionality on/off. If enabled, /pss macro is executed on each battle round flag. By default, /pss is 'party short status', but some modules (like PSS-mangler) override this to provide additional functionality.
/balance	Perform Alexandra sleeves 'balance', but only if the current HP - SP difference is favorable.
/binds	List all currently defined GgrTF command bindings. Refer to bindings section for more information and example output.
/ceremony	Perform skill 'ceremony', but only if ceremony is not already "active".
/chkbalance	Check and report the status of Alexandra sleeve 'balance' without actually performing 'balance'.
/cprots	This command clears all prots on you. It is meant for those cases where GgrTF is either bugging and does not notice a prot dropping, or any other
/cruise	Toggle cruise mode if movement mode is ship (/move ship).
/gagsc	Toggle gagging of short score ('sc') messages.
/gload	Load GgrTF settings. Refer to state saving section for more information.
/gprompt [prompt string]	Set or change GgrTF's displayed prompt. The setting can contain any TinyFugue expressions, such as variable substitutions. Refer to prompt settings section for details.
/greset	Reset all skill/spell counters and statistics. Notice that issuing this command also requires executing of "/gsave" if you want to save the zeroed statistics, otherwise the old saved statistics will be loaded on next /gload.

Command	Description
/gsave	Save all GgrTF settings. Refer to state saving section for more information.
/gver	Prints (or returns, if called as function) a short version string of GgrTF.
/gversion	Prints (or returns, if called as function) a long version string of GgrTF with copyright- and TinyFugue version information.
/lichaction <action>	Sets action taken after a Lich performs "soul sucking".
/move <type>	Change the meaning of keyboard movement hooks.
/opts	Lists all the run-time changeable settings of GgrTF, with short descriptions and current values.
/peer [regexp string]	View or set regular expression used with autopeering movement mode (/move peer).
/prots	Show any currently active prots on you. The output is only echoed locally, use BatMUD 'tweak me' emote to list prots to party report-channel.
/rcda	Toggle reporting of Combat Damage Analysis. If set 'off', CDA reports are only displayed locally to you, if set 'on', reporting is done to party report channel.
/ripaction <action>	Set the action performed at opponent RIP. Possible additional settings may be provided by other loaded modules. Functions provided by base GgrTF are: *off* (no special action performed), *dig* (dig grave for corpse), *eat* (get and eat corpse), *get* (get corpse) and *cmd* (execute mud command(s) specified with /ripcommand setting, see /ripcommand)
/ripcommand [commands]	Sets the MUD command(s) to be executed if /ripaction is set to "cmd". This string is sent "as is" to the MUD at opponent R.I.P, if /ripaction is "cmd". Sets action taken after a Lord of Chaos performs "blood corpse". This is useful for automating corpse handling, if you are a LoC yourself, or are partying with one.
/rmisc	Toggle miscellaneous reporting features.
/round [commands]	Sets the BatMUD command(s) to be executed on each battle round. The string of commands is sent to the MUD when battle round flag is received.
/roundmin <value>	Maximum amount of spell rounds left before reporting number of rounds. See /rrounds setting.
/rrounds	Report spell rounds to 'party report' channel. This functionality has some minor "intelligence" to report only relevant information, but it may be considered spammy and annoying by many people.

Command	Description
/shape	Reports the last catched shape of a opponent (monster) in battle. This does not work too well if you were fighting more than one opponents.
/shipmove <off|view|map>	Set what action is performed when ship movement is detected.
/skspam	Toggle skill/spell start/end liting spam. If disabled, normal BatMUD skill and spell start and finish lines are let through unmangled.
/stats	Display miscellaneous statistics about skills, spells, etc.
/verbose	Toggle in-MUD verbosity on and off. Off means that some things are echoed to client only, aka you. On means that those things are reported on party channel, etc.

Table 4.2. GgrTF core 'general' type command bindings

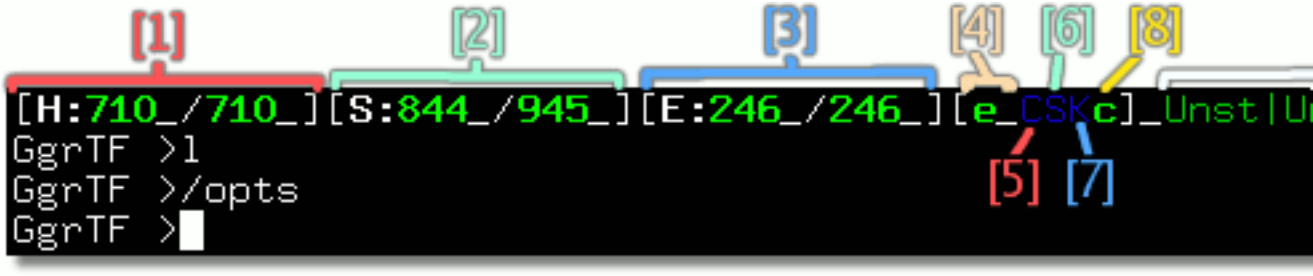
Command	Quiet	NoTarget	Description
cere		X	/ceremony
prots			/prots
curses			/curses
cbalance			/chkbalance
balance			/balance

Statusline

Probably one of the first things that you notice in GgrTF is the statusline (or "statusbar" as some prefer). In GgrTF the TinyFugue [http://tinyfugue.sf.net/] statusline is used to display information that we've considered to be important enough for nearly real-time view.

The basic one-row statusline of GgrTF is meant for compatibility, TF5 support module provides another version, enhanced to take advantage of newer TF5 beta features (TF 5.0 beta 7 or later required).

Statusline is updated synchronously at ticks, skill- and spell castings and certain other events. *This depends on your BatMUD settings being correctly set up.* If HP/SP/EP etc. information does not appear, most likely your 'sc' setting is "wrong", and thus does not get parsed by GgrTF.



Example of typical basic GgrTF statusline.

1. Current hit points / hit points maximum.

2. Current spell points / spell points maximum.
3. Current endurance points / endurance points maximum.
4. [2 light green characters] Last moved direction.
5. ["C"] Ceremony status. (Dark blue = inactive/no ceremony; White = ceremony active)
6. ["S"] Spell status. (Dark blue = no spell going; White = spellcasting)
7. ["K"] Skill status. (Dark blue = no skill going; White = using a skill)
8. ["c"] Camping status. (Light green = ready to camp; Red = camping active/skill ongoing; Yellow = camping not available, or recovering from previous camping.)
9. Currently active prots in short format.

First three elements of the statusbar describe your character's current hit-, spell- and endurance-points. These values are color-coded for better visual clarity. One element shows current value and maximum value, as shown in screenshot below:

Example of how endurance points are shown. "E:" denotes Endurance, "134" is the current value (color-coded) and "246" is the maximum endurance your character has.

Prompt configuration

In addition to basic prompt handling, GgrTF offers very flexible and extensible way of presenting the game prompt. Internal TF variables can be shown, even simple TF macro snippets executed to show the prompt as the user wishes. Also, in GgrTF v0.6.11.3 and later, it is possible to have custom BatMUD data in the prompt (see 'help prompt' in BatMUD).

To get started, you need to have a basic prompt setting in BatMUD, as it is required to have GgrTF handle it properly. The basic setting is presented below, and we'll get to the more complex ones later. **`prompt PROMPT:>`** After setting the BatMUD prompt, you can start configuring how GgrTF "mangles" your prompt. This is done by using "/gprompt" command macro (or alternatively editing the saved configuration files and "set_gprompt" setting.) There are myriad of ways of how and what things can be done, mostly limited by your imagination. Some examples are presented below:

1. **`/gprompt >`** Set prompt to show only ">" and nothing more.
2. **`/gprompt %{status_cast}>`** Show currently in-progress skill or spell in prompt.
3. **`/gprompt @{BCred}%{status_cast}@{n}@{BCwhite}>@{n}`** TinyFugue color attributes can be used also.
4. **`/gprompt @{$[prgetnlite(status_hp,status_hpm)]}%{status_hp}@{n}@{BCwhite}>@{n}`** Above would print your hp, lited similarly to the statusbar HP/SP/EP display. It is possible to use almost any kind of substitutions and call TF functions.

Note

Using TF attribute formatting (e.g. colours) in prompt only works with TF 5.0 beta 8 and later. If running under earlier version, GgrTF disables this feature.

As you can see, lots of things can be done, and only fraction of ideas are represented here. However, in addition to what GgrTF and TF offer, it is also possible to add BatMUD specific information into the prompt. As you remember, we set up the basic prompt above to "PROMPT:>". The user definable data can be put between the colon (":") and greater than (">") sign, and it gets parsed by GgrTF. Observe following example: **`prompt PROMPT:<hp>/<sp>/<ep>|<eqset>>`** Above prompt setting would make the BatMUD prompt show something like: **`PROMPT:663/636/333|spr>`** Where the numbers represent your hp, sp and ep, and 'spr' is the work eqset, as described by 'help prompt'. This string gets parsed into a TF variable called "status_prompt", which can be then used in GgrTF's prompt. Examples follow:

1. **`/gprompt %{status_prompt}>`** This simple example would just prompt the grabbed BatMUD data with greater than sign in the end.

2. **`/gprompt @{BCgreen}%{status_prompt}@{n}>`** Same as above, but coloured bright green.

The prompt can contain any BatMUD data, as long as it is formatted as "PROMPT:your data here>".

State-saving

State-saving is an awfully awkward name for GgrTF's functionality for saving your settings and other interesting data for later retrieval. Settings get saved and thus restored later, but this functionality is not automatic by default, you have to set it up so if you wish - GgrTF only provides you the functions for loading and saving, making it to happen automagically is up to you. How to do this is discussed further below.

What is saved?

Almost all settings (hidden or visible) get saved, but there are certain global settings that you have to set yourself in your TF configuration. These settings are discussed in sub-section below.

It should also be noted, that there are actually *two different types of data (or variables) that is saved into two different files*. In GgrTF, these two types of data are called *pre- and post-init settings*, and the two savefiles are named accordingly. Pre-init settings are variables that need to be defined before the main modules of GgrTF are loaded in TF's startup configuration (certain functionality depends on this). Post-init variables can (and sometimes *must*) be set after loading of all modules.

Special, non-saved variables

There are currently three special GgrTF variables, which you have to set yourself in your TF's startup configuration (.tfrc). These variables *MUST* be set before loading of ANY GgrTF modules, otherwise things start breaking and the state-saving system does not work correctly.

Table 4.3. Special, non-saved variables

Variable	Description	Example
set_plrname	Your character's name, written in correct case.	/set set_plrname=Ggr
set_datapath	Path to the directory where you want your savefiles to reside. If left empty, current working directory is used (the directory you started TF from.)	/set set_datapath=

Variable	Description	Example
set_saveprefix	This string is used as prefix of the two savefiles, so the actual savefiles are "<*prefix*>*pre.tf*" and "<*prefix*>*post.tf*". This is useful, if you play regular Bat-MUD and HCBat, so you can use different values for either.	/set set_saveprefix=bat-

How to setup TF for state-saving

Getting GgrTF's state-saving to work requires some changes to your TF configuration, and possibly setting up a savefile directory

A simplified example configuration layout is presented below with only the state-saving related parts, please refer to setup-section of this manual for a more detailed example. Pay close attention to the order how things are done, it is very important!

```
;; Set directory path where savefiles will be stored
/eval /set set_datapath=%{HOME}/.ggrtf/

;; Set prefix string for savefile names
/eval /set set_saveprefix=bat-

;; Load the special pre-init module which loads pre-init
;; settings from the savefiles. This needs to be done
;; before loading any other GgrTF modules.
/gloadmod ggrtf-pre.tf

;; Load GgrTF core module
/gloadmod ggrtf.tf

;; Load other additional modules
/gloadmod gm-magical.tf
; etc etc.

;; Load and restore other previously saved settings.
;; This should be done after loading any GgrTF modules.
/gload
```

When you have edited your TF configuration, you need to (re-)start it, change settings the way you want them to be and finally issue "/gsave" command to get settings saved. *After saving, you may also want to examine the pre-init -savefile, because it contains certain settings that cannot be changed run-time* . So, here are the steps in short again:

1. *Edit TF configuration (see example above)*: Add *set_datapath*, *set_saveprefix* settings and add commands to load *ggrtf-pre.tf* module and */gload* command.

2. *Re-start TF*: In order to create the initial savefiles with default settings, you need to (re-)start TF, so that the save-state system is enabled.

3. *Change settings*: Change GgrTF settings (listed via /opts etc.) to accommodate your desires.

4. *Save initial settings*: Issue "/gsave" to get settings saved.

5. *Optionally edit pre-init settings*: Certain settings are saved into the pre-init savefile (actual filename is dependant on what you set "set_saveprefix" to). GgrTF does not offer any special interface to change these settings, you have to edit this file by hand, if you wish to change the defaults.

How to make state-saving automatic

In previous example, I only described how settings get restored (as TF loads and initializes GgrTF), but how to get settings saved automatically too?

The answer lies in TinyFugue [http://tinyfugue.sf.net/]'s event hooks. It is possible to define a hook, which is executed when TF disconnects from a world. By setting this hook to perform "/gsave", we can automatize the process of saving settings. Add following line to your TF configuration: **`/def -hDISCONNECT mydisconnect = /gsave`** There is one gotcha: if you don't want to keep saving the statistical values, but only the settings, you may want to add a "/greset" in the soup: **`/def -hDISCONNECT mydisconnect = /greset%;/gsave`**

Macro commands

<> = required argument, [] = optional argument

Table 4.4. Macro commands

Command	Description
/gsave	Save all GgrTF settings and certain other TF variables.
/gload	Load GgrTF settings. Only post-init settings get loaded by this command, to get pre-init settings in effect, you will have to restart TF.
/greset	Reset all skill/spell counters and statistics. Notice that issuing this command also requires executing of "/gsave" if you want to save the zeroed statistics, otherwise the currently saved statistics will be restored on next /gload.

Command bindings

GgrTF's core provides few macros for defining "bindings", which are a way to create commands that look like normal MUD commands but are actually intercepted by TinyFugue [http://tinyfugue.sf.net/] and executed appropriately. These bindings can be used to easily define shorthands for casting spells, using skills, executing macros, etc. GgrTF bindings offer an easy way to define skill/spell commands with automatic party reporting (or without it). *Simply put, bindings are similar to BatMUD "command" aliases, but client-side and with extra abilities (but also with some limitations.)*

In GgrTF, there are *optional pre-defined bindings*, which can be enabled at your consideration. These pre-defined bindings are defined in the relevant modules, and can be enabled at load-time with a setting ("opt_bindings"). See below section for enabling bindings.

Now, you might be asking why would anyone need bindings? Indeed, you can already use BatMUD's command aliases for such things, but the number of command aliases in BatMUD is limited and if you run out of space, you have to remove some less used ones. Sometimes you have to redefine lots just because you are reincarnating... and you can not execute macros with command aliases.

There are many benefits to using bindings, but they do have some drawbacks:

- *Limitations of TF's parsing*: Because of the nature how bindings are defined and parsed, you can only use one binded command per line. For example, let's assume "aoa" is binding to cast Armour of Aether and "clair" is a binding to use clairvoyance. Thus, someone might want to command: **`aoa ggr%;clair`** Unfortunately, this does not work as might be expected. Because TF does not evaluate binding as an expression, the "ggr%;clair"-part is used as a literal argument to "aoa" binding. In effect the command would try to cast Armour of Aether at player "ggr%;clair".

- *BatMUD commands and GgrTF bindings cannot be mixed*: Because the bindings are actually intercepted by TF before the line gets passed to BatMUD's parser, you can't combine multiple bindings or BatMUD commands on the line. For example, assuming "aoa" is again a binding and "hop" is an emote: **`hop jeskko;aoa ggr`** One might easily assume, that the result would be to first hop around player or thing named Jeskko, and then cast Armour of Aether at Ggr. Unfortunately the latter would just produce an error, as it would be instead passed to BatMUD's interpreter and not handled as a binding by TF!

- *Limitations of GgrTF's binding system*: There are certain limitations what kind of bindings you can define with the current functionality provided by GgrTF. Bindings that require complex parameters (more than one optional argument), are not possible - you are better off using BatMUD's command aliases for those.

Defining bindings

Now we know the limitations of this system - let's see how to define some basic command bindings!

```
/prdefgbind -s"cere"    -c"/ceremony" -n
/prdefcbind -s"nf"      -c"Neutralize Field" -n
/prdefsbind -s"fire"    -c"Fire Building" -n
```

Above we have have listed all three (3) classes of binding defination macros supported by GgrTF. First defines a generic command binding to execute "/ceremony" macro when user commands "cere". Second binding defines a spell casting command to cast "Neutralize Field" when user commands "nf". And the third binding defines a skill usage command to use "Fire Building". So basically we have three types of bindings:

- */prdefgbind*: Defines a generic binding, in which the action ("-c" parameter) can be almost anything, like a TF macro, or BatMUD command.

- */prdefcbind*: Defines a casting binding, the action is the name of the spell to be casted.

- */prdefsbind*: Defines a skill usage binding, action argument being name of the skill to be executed.

Each of these defination macros require at least two arguments: the name for the binding command ("-s" option) and the action ("-c" option). There are other optional arguments, which affect aspects of the command binding to be defined. The argument options and their meanings are listed below:

Table 4.5. Arguments common to all binding defination macros

Option	Description
-s"name"	Specifies the name of the command binding. This is basically the string you type to use the command, after which come the optional arguments.
-n	Tells GgrTF that the binding does NOT accept arguments. Anything following the command binding invocation is ignored. This also affects the possible reporting output of the binding.

Table 4.6. Arguments specific to generic bindings (/prdefgbind)

Option	Description
-c"action"	Specifies the action, aka what the binding *does*. For generic bindings, the action is a literal, so it can be a TF macro or BatMUD command.

Table 4.7. Arguments specific to spell casting and skill usage bindings (/prdefcbind and /prdefsbind)

Option	Description
-c"action"	Specifies name of the spell to be cast or skill to be used.
-d"message"	This option overrides the default message string used for reporting (see also option "-q" below).
-q	Makes the binding "quiet", suppresses any reporting. By default, spell casting binding reports the action on party report-channel, but specifying this option disables it.

More examples, now with short explanations:

Table 4.8. Binding defination examples

Definition	Description
/prdefgbind -s"repu" -c"/showrep" -n	Generic binding, which executes /showrep macro and takes no arguments.
/prdefsbind -s"er" -c"Enrage" -n	Skill binding for barbarian "Enrage" skill. Also takes no arguments.
/prdefcbind -s"shelter" -c"Shelter" -n -d"Sheltering ..."	Spell casting binding, for casting conjurer spell "Shelter". Takes no arguments and uses specific reporting string (default would be "Shelter ...", but since it sounds bit stupid we want to use "Sheltering ..." instead.)
/prdefcbind -s"dmp" -c"Dispel Magical Protection"	Another spell casting binding for a conjurer spell, which takes (and requires) a player's name as argument.
/prdefcbind -s"sl" -c"Star Light" -q	Yet another spell casting bind, this time for druid spell "Star Light". Since this is a blasting spell, it is not really useful to report that we are casting it, so we suppress any reporting. Argument is accepted.

As of now, you should be able to define bindings of your own. You can define them in your .tfrc or make another file for them.

Enabling bindings

A special variable called "opt_bindings" controls whether the various /prdef*bind macros will actually define any binding at all. If this variable is set to value "on", bindings will be defined, otherwise not.

This is also the way how pre-defined GgrTF bindings are made optional. By defining "opt_bindings" before loading GgrTF modules, user can choose to enable or disable the default bindings, even "per-module" by changing the value of opt_bindings before each module loading in .tfrc.

It should be noted, that "opt_bindings" is a saveable setting, that gets stored by /gsave and restored by ggrtf-pre.tf Please refer to setup- and state saving- sections for more information.

Bindings listing

All defined GgrTF bindings can be listed via the */binds* command. Below is represented an commented example of such output.

```
,----------------.
| GgrTF Bindings |
+----------------+-------------------------------------------------------.
| pprots         | G | /pprots                                           |   ? |
| rpouch         | G | /rpouch                                           |   ? |
| ad             | C | Aura Detection                                    | Yes |
| invis          | C | Invisibility                                      | Yes |
| float          | C | Floating                                          | Yes |
| ww             | C | Water Walking                                     | Yes |
| seeinvis       | C | See Invisible                                     | Yes |
| seemagic       | C | See Magic                                         | Yes |
| cere           | G | /ceremony                                         |   ? |
| curses         | G | /curses                                           |   ? |
| prots          | G | /prots                                            |   ? |
`-------------------------------------------------------------------------'
```

- 1st column lists the binded command name
- 2nd column shows the type of the binding: "G" = general (for example a macro invocation); "C" = cast/spell; "K" = skill.
- 3rd column is the performed action, which has varying meaning depending on the binding type. For skill and spell bindings, corresponding skill/spell name is shown. Generic bindings show the actual command executed.
- 4th and last column shows whether the binding produces any report output; for generic bindings this is not applicable and shows question mark ("?") instead.

Keyboard numpad hooks

GgrTF provides a mechanism for routing keyboard numpad bindings so that different functionality can be swapped in as needed. Core module offers three modes: walk (normal walking around); peer (walking around with automatic peering to adjacent directions, useful for merchants; ship (control ship movement). Certain other modules provide additional modes (see targetted cast module, utilized by tarmalen and mage modules, for example.)

Note

Keyboard hooks require working numpad, etc. key bindings. Modules are provided for two most common cases in *kbd-*.tf* files. Some terminals may require several kbd-* modules to be loaded, you may have to experiment. It is also possible that your terminal is not supported by those modules and you have to write your own, in such case you may try contacting Matti Hämäläinen

[http://tnsp.org/~ccr/] (Ggr [http://www.bat.org/char/Ggr]) for assistance. However, it is also entirely possible that your terminal does not provide the required 'application mode' at all, which will make things difficult.

Macro commands

<> = required argument, [] = optional argument

Table 4.9. Macro commands

Command	Description
/move [setting]	Set the mode used for numpad bindings. Valid values are shown if command is given without arguments. Core GgrTF module provides following three modes: *move*, *peer* and *ship*
/peer [regexp]	Set the regular expression used for grepping the output of 'peer' command used in "/move peer" mode.
/cruise	Toggle ship cruising speed on/off (off = sail)

Prot management

One very important aspect of any script is the prot management. GgrTF contains a flexible and easy to extend prot management system, which supports basic, stacking and renewable prots. Special cases are also supported, like unstun and conjurer typeprots.

For actually displaying the active prots information, GgrTF offers various methods. The statusline shows the currently active prots in short form (without duration times). Macro command "/prots" can be invoked to locally echo the active prots in long form. Finally, a special BatMUD emote, 'tweak', can be used to trigger reporting of active prots to party report-channel (the 'tweak' emote was chosen for this purpose due to it being the de-facto standard for prot reporting triggers.)

Classes of prots supported by GgrTF are roughly the following:

- Harmful effects/curses/handicaps
- Basic utility prots (infravision, water walking, floating, invisibility, etc.)
- Conjurer prots (minor- and major typeprots, stickyness)
- Evil priest (protection from good, paranoia)
- Tarmalen (unpain, blessing of tarmalen, lifelink, enhanced vitality)
- Nun (protection from evil, soul shield, heavenly protection, mana shield)
- Templar (shield of faith, blessing of faerwon)
- Psionicist (levitation, force shield)
- Druid (flex shield, earth power, earth skin, earth blood, vine mantle, regeneration)
- etc...

The string describing active prots consists of entries with abbreviated name of the prot, current duration (from casting), possible number of stackings and optional stickiness-indicator for sticky conjuprots. The

entries are separated with vertical line character (sometimes called the pipe). Some examples of prot entries with explanations are shown in table below.

Table 4.10. Prot entries

Entry	Description
Unst[1m9s]	Unstun with duration of 1 minutes, 9 seconds.
+AoA[7m19s]	Sticky Armour of Aether.
Infra(2)[0m2s]	Infravision stacked at 2 levels.
Unst<2>[7m19s]	Unstun that has been weakened 2 times.
-Forget[3m5s]	'Forget' curse that has been active for 3 minutes and 5 seconds.

Simple example session featuring some protting action in GgrTF:

```
tweak jeskko

You tweak Jeskko's nose mischievously.
Jeskko [report]: Unst[1h1m13s]

unp jeskko

Ggr [report]: Unpain -> jeskko
GgrTF:  ---- SPELL START ----  (393)
GgrTF: 'unpain' -> jeskko
Unpain: ##### [5]
You skillfully cast the spell with greater haste.
Unpain: ## [2]
Unpain: # [1]
GgrTF:  ---- SPELL DONE ----  in [3] rounds! [0m9s]
You utter the magic words (unpain)
You cast the spell successfully.
H:506/506 [] S:805/941 [-136] E:246/246 [] $:6499 [] exp:68202 []
Jeskko [report]: Unpain ON!

tweak jeskko

You tweak Jeskko's nose mischievously.
Jeskko [report]: Unst[1h4m25s] | Unp[2m35s]
```

Macro commands

<> = required argument, [] = optional argument

Table 4.11. Macro commands

Command	Description
/prots	Show any currently active prots on you. The output is only echoed locally, use BatMUD 'tweak me' emote to list prots to party report-channel.

Command	Description
/cprots	This command clears all prots on you. It is meant for those cases where GgrTF is either bugging and does not notice a prot dropping, or any other reason when you need to remove all prots at your discretion.
tweak <target>	BatMUD's 'tweak' emote is used to report prots to party report-channel.

Curse management

Often an important aspect while bashing those nasty eq-monsters dead, is to keep track of any handicapping spells or skills cast against the said monster. GgrTF's curse handling is made just for that, it tracks most common handicaps cast at non-partymembers.

Notice! To work fully, this feature requires PSS-mangler module to be loaded and in use. While curse tracking does work without it, curses cast by other entities than party members will be counted also, which can be counterproductive.

Macro commands

<> = required argument, [] = optional argument

Table 4.12. Macro commands

Command	Description
/curses	Echo status of tracked curses to yourself (e.g. not shown to others.)
twirl <target>	BatMUD's 'twirl' emote is used to report curses to party report-channel.
/cursewarn	Toggles informing about curses that are soon about to expire. The warning occurse when 20% of the "maximum" curse duration is left. (These maximum durations are set in GgrTF's main module.)

Heartbeat and tick prediction

Since version 0.6.14, GgrTF provides an *experimental* feature for predicting heartbeat duration and ticks. This can be useful if you wish to time your regen breaks nicely, or when to enter combat again (heartbeat factors in with the so-called initial battle round, or so called "zero round").

The usage of this feature is simple: There is one new field displayed on the statusline, the value labeled as "T" shows number of heartbeats from last tick, next tick usually occurs at 9 or 10. It should always be 10, but due to how we "force" simulated HBs and interaction with fast metabolism, they will not always match the real ones. *First tick after leaving battle will not usually be predicted correctly, but the situation should normalize after it.*

Note

You can also utilize a stethoscope, and it should, at least theoretically make the HB and tick prediction accurate, but who knows... fastmeta is a bitch.

Stethoscope is an item, available from Arelium's tinkerer's toolshop. You can use it by applying, 'apply stethoscope to ggr'. Easiest way is to add that to your eqset swapping commands.

Technical details

One might ask, how does this all work? This is where it gets a little bit murky, because we are largely working on assumptions on how things like heartbeat work in BatMUD. So it might not be correct, there may be things that have been overlooked, etc.

First of all, here are some things that the predictor code assumes:

- Heartbeat duration (delta) is 3.0 seconds to start with, we adjust slowly towards the player's "real" average heartbeat frequency.
- One tick is always 10 heartbeats.
- Battle rounds are always around ~3s, not withstanding +/- lag and slight variation due to driver issues.
- Spell rounds, cast and skill finishes occur on heartbeat. Thus we can use these to time heartbeats, when filtering out fastmeta.

Basically what we do in the code is as follows:

1. Start a TF timer that runs gheartbeat_timer() at ~20Hz.
2. In gheartbeat_sc (ran on each 'sc') we check if we seem to have ticked by checking sp/ep deltas against threshold values. We also try to discard 'sc' events caused by things that may not be heartbeats (fires, crystals, heal alls, etc.)
3. In gheartbeat_timer() we check for:
   - Ongoing battle, check when last round went off, disable battle if it was "too long" ago.
   - Check when last hb was, simulate / force a hb if it was too long ago.
   - Check if we need to "force" a tick, and do that.

Based on these, we keep a running average of heartbeat duration to predict incoming heartbeats and ticks. The average starts at 3s, which is the basic BatMUD heartbeat without fast metabolism or such.

Note

If you are interested in some debug-spam produced by the predictor, you can enable it via */set hb_debug=1*

Skill handling

Magical guilds support (gm-magical.tf)

Base module for all things magical. This module is *required* for supporting spells/casting. It is not necessary for being the receiving end of prots or such. Several modules are dependant on this module, most importantly those guild-specific modules that use magic.

Even in the case your guild does not use magic spells (nomad guilds), having this module loaded is not harmful, so you can keep it in your configuration at all times.

- Basic code for blast hit and resistance reporting (some types, such as channeller guild blasts, are added by guild-specific modules).
- Spell round reporting. Enables you to optionally spam your fellow players with information how many rounds are left until your spell goes off.

PSS-mangler (gm-pssmangle.tf)

This module provides support for filtering the output of *pss* (*party status short*) -command. It captures the status and prettyprints it out as a 3x3 table (as 9-man party formation) with player names and their status information in compact format.

The HP/SP/EP values are coloured in accordance to what percentage the player currently has of his maximum. Additionally varying colouring of player's name is used to indicate possible states of *rest, unconsciousness, stun, formation, etc.*.

There is also an number indicator for *number of rescuers* when in stunned or unconscious state, but this is normally hidden. It appears on the left side of player's name, when available.

This module hooks to GgrTF's autopss-functionality (toggleable with /autopss option). A normally issued 'pss' command does not trigger the mangling functionality, but you can use */pss* macro or bind that to 'pss'. (See Bindings-section for more information.)

Typical output of PSS-mangler in a party with first row filled. Toivo and Jeskko are near their maximum hp/sp/ep, but Ggr has taken some damage (yellow HP) and is almost out of endurance points (red EP).

In this screenshot example, Jeskko has become stunned (his name is coloured in bright red). Also, Jeskko has lost some hitpoints (dark green HP) and has only moderate amount of endurance left (light yellow EP). *(No Jeskkos were actually harmed in producing this simulated situation.)*

This screenshot shows how Toivo has taken some serious damage from a blast to the backrow, and has become unconscious. White on bright red background is used to highlight a unconscious player's name, hopefully catching your attention.

HP/SP/EP colours

List of possible indicator colours and their meanings. Colours described here may deviate from what you actually see due to differences in terminal emulation programs and settings.

Table 4.13. List of HP/SP/EP indicator colours

Colour	Background	Explanation
Bright White	Bright Red	Negative value (x < 0)
Bright Red	-	Less than 16% of max
Red (dark)	-	Less than 33% of max
Yellow (dark)	-	Less than 49% of max
Bright Yellow	-	Less than 66% of max
Green (dark)	-	Less than 85% of max
Bright Green	-	More than 85% of max

Player name colours

List of possible name colours and their meanings. Colours described here may deviate from what you actually see due to differences in terminal emulation programs and settings.

Table 4.14. List of possible player name lite colours

Colour	Background	State description
Bright White	-	Normal
Bright Black	-	Idle
Bright White	Bright Red	Unconscious (unc)
Bright Red	-	Stunned (stun)
Magenta	-	In form (form), or member (mbr)
Cyan	-	Resting/camping/meditating (rest)
Bright White	Blue	"Glaced" (NPC special)

Macro commands

<> = required argument, [] = optional argument

Table 4.15. Macro commands

Command	Description
/prettypss	Toggle 'pss' output mangling/prettifying. If set off, normal plain 'pss' output is passed through unfiltered.
/pss	Show party status.
/autopss	Toggle automatic party status reporting. (Defined elsewhere)

Command	Description
/fullparty	If enabled, mangled output always shows full 3x3 party formation, even if all rows aren't filled. When disabled, only the first used rows are shown.
/diffpss	Shows diff-values of hp/sp/ep if enabled.

Party Placer (gm-pplacer.tf)

This module provides functionality for saving and restoring party formation (placement of members). Two simple to use macros can be issued to save the current formation and restore it as needed. Current flower BatMUD provides an in-game implementation via 'party store' and 'party restore' commands, which may be preferable over this module for some people.

However, the functionality of this module is somewhat better than the stock BatMUD commands, and will work under certain situations when the 'party restore' command does not, for example when some party members are missing from the room.

NOTICE! This module requires PSS-mangler module to be loaded!

Macro commands

<> = required argument, [] = optional argument

Table 4.16. Macro commands

Command	Description
/ppsave	Save current party formation. Positions of party members in current formation are stored.
/ppreset	Resets or restores the party formation to the one saved previously. Restoring occurs fully automatically.

Party Prots Tracker (gm-ptracker.tf)

Note

This module is practically obsolete and deprecated in flower BatMUD, because of the in-game 'party prots' command that does essentially the same thing. It is possible that this module will be removed in future.

Module for showing prots (and similar effects) affecting your party members. Tracking these effects can be done in two separate ways: 1) Old way (compatible with HCBat also) is keeping account of different prot up/down messages on the party and report channels. 2) New way is via 'show effects' command available in the normal BatMUD. The 'show effects' method is recommended, because of generally better accuracy. *It should also be noted that there is a difference in the output between these two methods. "Manually" tracked prots show elapsed time of the effect, but with 'show effects' mode, the remaining time is shown.*

Note

Due to the way the up/down message tracking works, it depends on your party members to use *working, non-broken* triggers which use up/down messages that are recognizable to the ptracker module. However, this does not affect 'show effects' mode in any way.

```
.------------+-----+-----+-----+-----+-----+-----+-----+-----+
|            | Flex|WarEs|EMelo|  AoA|MFire|  AoH|  Uns|  Unp|
+------------+-----+-----+-----+-----+-----+-----+-----+-----+
|Blayke      | 0:07| 1:25| 0:49| 1:55| 1:43| 2:03|     |     |
|Yar         |     | 1:25|     | 1:40| 0:07|     |21:34|     |
|Kheldor     |     | 1:24| 0:48| 1:12| 0:41| 2:17|24:40|     |
|Alorn       |     | 1:25| 0:49|     |     |     |     | 6:49|
|Acentaja    |     | 1:25| 0:49|     |     | 1:31|     |     |
|Malcom      |     | 1:25| 0:49|     |     | 1:10|     |     |
|Jrj         |     |     |     |     |     | 6:22|     |     |
|Daria       |     |     |     |     |     |     |     |     |
|Ggr         |     | 1:25| 0:49|     |     |20:13|     |     |
`------------+-----+-----+-----+-----+-----+-----+-----+-----+
```

NOTICE! This module requires PSS-mangler module to be loaded! Also remember to issue at least one "/pss" command after the party is formed, to make the party members known to PSS-mangler and prot tracker.

Note

It should be noted that the prot tracker only supports a limited set of effects, e.g. it only lists effects/prots that are widely considered to be of some interest in eq parties. Showing everything would just waste screen estate and make it less readable.

Table 4.17. Party Prots Tracker macro commands

Command	Description
/cpprots	Clears ALL the currently tracked prots and all other related data. Using this may be useful if some prots erraneously "linger".
/effects	Toggle support for using output of 'show effects' command in party prots tracker. Enabling this will disable tracking of prot up/down messages reported by party members and makes the tracker to use 'show effects' output only. This also means that the shown times in this mode are "time remaining", not "time elapsed". Requires /gsave and restart of TF to work. (If you are not using the GgrTF statesaving system, you can /set opt_effects=on in your .tfrc before loading this module.)
/pprots [command]	Shows currently tracked prots in the party. Optional macro or MUD command argument can be given, to feed the output through it. For example "/pprots emote" would use BatMUD emote to show the data people present in the same room.
/protform	Save current party formation (members and their order) into separate prot formation. This is useful, because the current captured party formation can become jumbled due to fleeing, deaths etc. and thus would change the prot tracker's output making things more confusing. NOTICE! This saved set-

Command	Description
	ting is completely SEPARATE from saved form of the party placer module.

Table 4.18. Party Prots Tracker 'general' type command bindings

Command	Quiet	NoTarget	Description
pprots			/pprots

Spellwords translator (gm-spellnames.tf)

This module adds in translating of spellwords to actual spell names. It is very much based on original work of Cutter [http://www.bat.org/char/Cutter] and Dazzt [http://www.bat.org/char/Dazzt], well known highbie BatMUD players. Their code is used and re-licensed under GPL with permission.

The translator works by printing the spell name in parentheses on the line where the spellwords are. The translated name can optionally replace the spellwords or it can be printed after them.

The spell name can be, if enabled, optionally printed in special colour formatting, which is based on the general class/type of the spell. This visual hinting can help you to notice if dangerous or otherwise interesting spells are cast.

Uncolorized examples of spellwords translator's output:

```
You utter the magic words (unstun)

You utter the magic words 'Paxus' (unstun)

Acolyte waves his index finger while uttering (flame arrow)

Murrough throws a pinch of magic dust in air and chants (lightning storm)
```

Table 4.19. Spellwords translator macro commands

Command	Description
/spcolorize	Toggles colorization of spell name on/off.
/spwords	Toggles showing of actual spell words on/off.

Numpad targetted casting (gm-tgtcast.tf)

Note

This module requires the PSS-mangler module.

HCBat support (gm-hcbat.tf)

Due to the age of Hard Core BatMUD's current codebase and conceptual differences, several exceptions and workarounds are needed to improve GgrTF's compatibility with the HCBat environment. This module tries to provide the glue to make it happen.

Purpose of this module is dissimilar to others, as it is meant to contain almost everything related to peculiarities and differences of HCBat. We do not have separate guild-specific modules for HCBat, the code needed to catch the differences is either in this module or in some special cases in the guild-specific modules.

At the moment this module is rather simple, only replacing GgrTF's output macros to work around the lack of 'party report' -channel in HCBat. In future more functionality may be added, feel free to report bugs/incompatibilities to us.

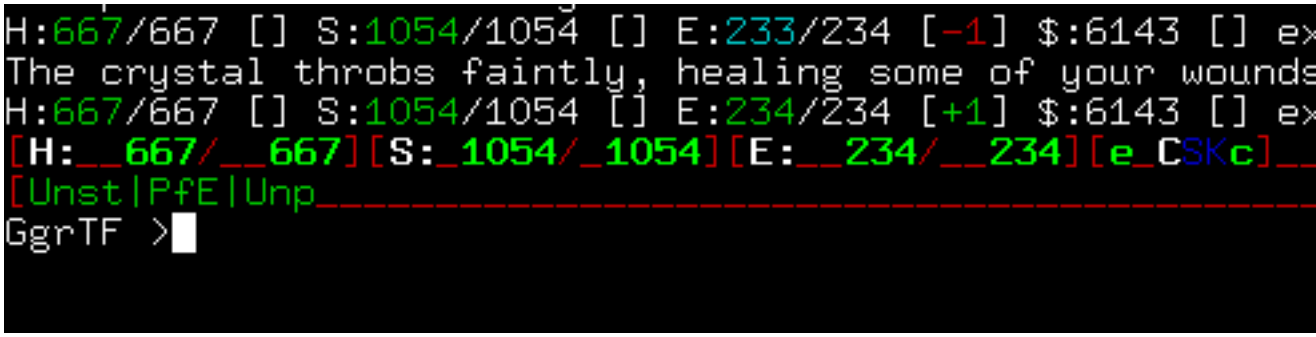
TinyFugue 5 support (gm-tf5.tf)

This is a special enhancement module for those who use GgrTF with TinyFugue [http://tinyfugue.sf.net/] version 5.0 (beta 7 and LATER). Although GgrTF should work with TF5 without this module, you may wish to take benefit of the enhanced features designed for TF5 this module provides.

Note

The code in this module uses features available on TF 5.0 beta 7 and later, it will NOT work correctly in older betas!

The functionality of the statusline depends on your BatMUD settings being correctly set up. If HP/SP/EP etc. information does not appear, most likely your 'sc' setting is "wrong", and thus does not get parsed by GgrTF.



Example of typical basic GgrTF statusline in TF5.

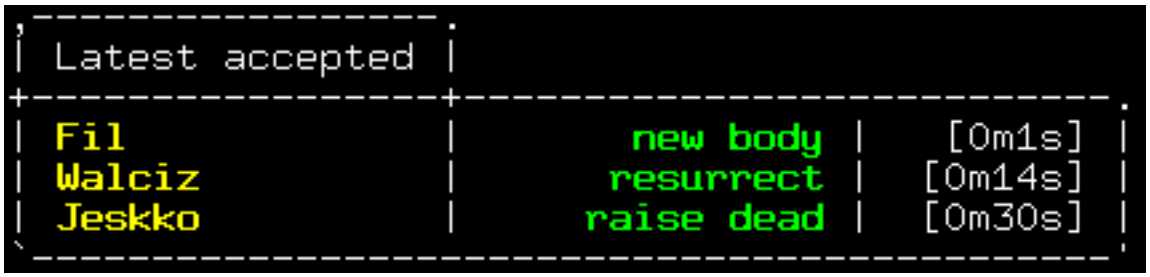
- Enhanced 2-row statusline: First line contains HP/SP/EP- and skill/spell/etc- information as usual. Prot status has been moved to the second line.

Raise/Resurrect/New Body/etc. (gm-rais.tf)

This module contains utility functionality for everyone who casts different types of dead raising spells, including "raise dead", "resurrect", "new body", "reincarnation", etc.

Automatic list is kept of people who have accepted some such service from you, along with information what spell was requested. List can be viewed, and you can launch a cast at the latest accept or by specifying a player's name, cast his/her request.

You can either directly use the macros or use command bindings to have shorthand commands instead.



An example output of "/aclist" macro, listing the players who have accepted services from you and respective requested spells for each. The time field shows how much time has passed from the accept.

Table 4.20. Macro commands

Command	Description
/aclist	View list of people who have accepted something from you.
/accast [name]	Cast Last-type functionality. Without arguments casts the LATEST requested spell. Optionally you can specify a name of the player and the spell requested by him/her is cast.
/acclear	Clears the list. Useful, if the automatic removal malfunctions for some reason. (For example, if target uses 'pray' to become alive again, there is no message provided to the caster.)

Table 4.21. Raise/Resurrect/New Body/etc. 'general' type command bindings

Command	Quiet	NoTarget	Description
cpurge			/acpurge
clast			/accast
acc			/aclist
cclear			/acclear

Hit Statistics (gm-hitstats.tf)

Provides triggers for automatic gathering of melee hit statistics. This functionality is not without limitations, in some cases hits may become erraneously counted (such as Templar guild 'holy strike' skill being counted as bludgeon strike) but it gives you a general impression about what kinds of hits you mostly deal out.

Only certain weapon/hit types are currently supported by the code. In order to get any hit counts, you need to define what weapon types you use. Currently following general hit/weapon classes are supported:

- *none* - This is just a clear/off setting.
- *bash* - Blunt weapons (bludgeons, etc.)
- *pierce* - Impaling weapons (some polearms, some short blades.)
- *slash* - Slashing weapons (long blades, axes, some short blades, some polearms.)

- *shield* - Shield bash.
- *whip* - Whip hits.
- *tiger* - Tiger style martial arts.
- *monk* - Monk style martial arts.
- *unarmed* - Untrained unarmed attacks.
- *claw* - Natural unarmed claw attacks.
- *bite* - Biting attacks.

Note

It should be noted, that the code does not differentiate between wielded weapons or limbs. *This means, that if you are wielding multiple weapons of same type, hits from those are counted as one weapon. Same applies to unarmed hits too, of course.*

Example of /hstats output:

```
,----------------------.
| GgrTF Hit Statistics |
+----------------------------------------------.
+-| Axes/Long blades     |----------------------|
|horribly shred      :        (  0):     2 ( 11)|
|shred               :        (  0):     7 ( 41)|
|incise              :        (  0):     8 ( 47)|
|tear                :     21 (  3):       (  0)|
|cut                 :    462 ( 72):       (  0)|
|lightly cut         :    157 ( 24):       (  0)|
|                                              |
+----------------------------------------------'
+-| Totals |-------------------------------------------+
| Hits..: 640        ( 76%) | Crits: 17         (  2%) |
| Misses: 192        ( 23%) | Total hit types: 6       |
+-------------------------------------------+-------'
| Dodges..: 3          | Parries..:            |
| Tumbles.:            | Stuns....:            |
| Ripostes:            | Stun mano:            |
`-------------------------------------------'
```

Format of hits described above is: | Hit name : number of hits of this type (percentage): number of crits of this type (percentage) |

Note

Hit stats REQUIRES the TF terminal window width to be 95 columns or more! If you are using TF in smaller smaller window and can't make it wider, then you are out of luck.

Table 4.22. Hit Statistics macro commands

Command	Description
/hstats	Print out hit statistics in a pretty table (see the warning above about the terminal width, though.)

Command	Description
/hstreset	Reset and clear current hit-statistics.
/mhits <off|gag|short>	Change hit message mangling: *off* = no mangling, pass hit message through unaltered; *short* = use short messages, collecting ALL your hit messages into one line like "You jab, dodge, parry, CRUELLY TATTER."; *gag* = gag messages completely.
/weapon1 <type>	Set the weapon types you are using. Currently only two concurrent types are supported. Notice, that if you are using several weapons of SAME type/class, you only need to set one (separate weapons of same type are counted as one.) Use "/weapon1" without arguments to see supported types. Use /weapon[2-4] to set the other weapon types, if any.

Identify output beautifier (gm-identify.tf)

Provides a simpler and more readable view of 'identify' spell's output. Not all information can be caught gracefully, however, and thus you should refer to the raw output for details such as "tales" and other special information.

The module also supports merchant belt's 'weigh' command for catching the item's weight, if the belt is worn and */havebelt* setting of the merchant module is enabled.

Example of output:

```
+-| General |-------------------------------------------------------------+
| Item....: a pair of stylish blue breeches                                |
| Name....: blue breeches                                                  |
| Handles.: 'breeches', 'stylish blue breeches', 'silk breeches' and       |
|           'pants'                                                        |
| Names...: Molotov, Belse and Ekkivekki                                   |
| Slots...: leg and leg                                                    |
| Material: silk                                                           |
+--------------------------------------------------------------------------+
| In game.: 2y, 104d, 16h, 13min and 58s   | Condition: great              |
| Size....: somewhat small                | Maxcond..: AWESOME            |
| Worth...: 8851                          | Quality..: superb             |
| Weight..: almost weightless             | Weight/kg: 0.050              |
+-| Stats |------------------------------------------------------------------+
| insignificantly improve your wis                                         |
+-| Misc |-------------------------------------------------------------------+
| Worn....: Ekkivekki for 48d, 17h, 13min and 9s, Prince charming for 2h,  |
|           5min and 44s and Slobber for 157d, 14h, 48min and 52s           |
| Wielded.:                                                                |
+--------------------------------------------------------------------------+
```

Reagent Pouch handler (gm-rpouch.tf)

This module provides utilities for reagent pouch handling, especially useful for mages and merchants. It also has a reagent pouch mangler/beautifier, which attempts to make the pouch contents more readable.

Example output (uncoloured) of looking at a reagent pouch:

```
This is a large pouch for the storing of reagents for spells. You can store
reagents over reboots in this. It cannot hold anything else.
Syntax: store <item> in <this>
        extract [amount] <item> from <this> take <power/standard/poor>
        set_default_reagent <this> to <power/standard/poor>
transfer <number> <reagent> from <here> to <there> [take <power level>]
Some commands allow the use of 'all' in places.
It is labeled as 'pussi'.
The label can be set with 'label' command.
  578 | Acid Blast                ( 578,   0,   0) (Olivine powder)
  477 | Acid Storm                ( 477,   0,   0) (Interlocked rings)
  436 | Blast Vacuum              ( 436,   0,   0) (Bronze marble)
  417 | Lightning Shield          ( 417,   0,   0) (Iron rod)
  397 | Armour of Aether          ( 397,   0,   0) (Highsteel disc)
  388 | Magic Eruption            ( 388,   0,   0) (Platinum hammer)
  341 | Lava Blast                ( 341,   0,   0) (Granite sphere)
  272 | Electrocution             ( 272,   0,   0) (Electrum wire)
  263 | Aura of Wind              ( 263,   0,   0) (Leather bag)
   91 | Golden Arrow              (  91,   0,   0) (Copper rod)
   71 | Shield of Detoxification  (  71,   0,   0) (Amethyst crystal)
It is surrounded by a yellow glow.
This item is in superb condition.
It looks very very heavy.
```

Note

Currently only POWER reagents are handled by the /rpouch commands! Any standard or poor reagents will not be touched by the code in this module.

Table 4.23. Reagent Pouch handler macro commands

Command	Description
/rpouch list	Prints a short help text with basic syntaxes of the commands. Checks and shows a list of reagents to be stocked into "dst pouch" from "src pouch" with current settings. Source pouch can optionally be in container (such as a chest). Like 'check' but instead of a list, moves reagents to "dst pouch" to match the current settings. Moves ALL power reagents from "src pouch" to "dst pouch". Set amount of reagents to stock for given spell. (Like "rset", but takes spell name instead.) Set amount of given reagent to be stocked. (Like "set", but takes reagent name instead.) ??? Unconditionally sets number of ALL reagents to be stocked to given amount. Shows a list of current settings (e.g. how many of each reagent to stock.)

Guild: Channellers (gm-chann.tf)

Game-elements specific to channellers guild are supported and enhanced by this module. At the moment, no special commands are provided, all things are automagic. This module requires the generic magical guilds support module.

- *Channeller aura*: Keeps note of your aura status and duration. If aura is up, it is notified in GgrTF's statusline. If your aura is weakening, that also is noted, reminding you to recharge.
- *Blast resistance reporting*: Adds support for channeller blasts into blast resistance reporting feature of magical guilds module.
- And other miscellaneous lesser features, such as fail and fumble handling, mana transfer cap reporting, etc.

Guild: Mages (gm-mage.tf)

Provides command bindings for all mage blast types (singles and areas) and optional key bindings for blasting and conjurer prot casting.

Note

This module requires the generic magical guilds support module. The keybinds require targetted cast module.

Table 4.24. Guild: Mages 'cast' type command bindings

Command	Quiet	NoTarget	Description
p4	X		Power Blast
p3	X		Venom Strike
p2	X		Poison Blast
p1	X		Thorn Spray
aa1	X		Acid Rain
a5	X		Acid Blast
a3	X		Acid Arrow
pa1	X		Poison Spray
p5	X		Summon Carnal Spores
a2	X		Acid Wind
aa2	X		Acid Storm
ca1	X		Cone of Cold
ca2	X		Hailstorm
s1	X		Vacuumbolt
s2	X		Suffocation
s3	X		Chaos Bolt
s4	X		Strangulation
s5	X		Blast Vacuum

Command	Quiet	NoTarget	Description
sa1	X		Vacuum Ball
sa2	X		Vacuum Globe
f1	X		Flame Arrow
f2	X		Firebolt
f3	X		Fire Blast
f4	X		Meteor Blast
f5	X		Lava Blast
fa1	X		Meteor Swarm
fa2	X		Lava Storm
e1	X		Shocking Grasp
e2	X		Lightning Bolt
e3	X		Blast Lightning
e4	X		Forked Lightning
e5	X		Electrocution
ea1	X		Chain Lightning
ea2	X		Lightning Storm
a1	X		Disruption
a4	X		Acid Ray
pa2	X		Killing Cloud
m1	X		Magic Missile
m2	X		Summon Lesser Spores
m3	X		Levin Bolt
m4	X		Summon Greater Spores
m5	X		Golden Arrow
ma1	X		Magic Wave
ma2	X		Magic Eruption
c1	X		Chill Touch
c2	X		Flaming Ice
c3	X		Darkfire
c4	X		Icebolt
c5	X		Cold Ray

Table 4.25. Guild: Mages keybindings

Key(s)	Function
Meta/Alt + [asdfghj]	Cast blasts (from smallest to biggest, then areas)
Meta/Alt + [qwertyu]	Blast type selection (acid, asphyxiation, cold, electricity, fire, magical, poison)
Meta/Alt + [xcvbn]	Cast conjurer prots

Key(s)	Function
Meta/Alt + z	Stop casting
Numpad Ins	/pss (show party status)

Guild: Merchants (gm-merchant.tf)

This module provides, in addition to basic skill/spell fail/fumble handling, several helper macros, which may ease your work as merchant. Many of the command macros support "autotargetting", which basically means that you can walk in outerworld and just type "/lj" for lumberjacking and "/mine" for mining and GgrTF usually knows what to do. Some macros require bit more specific arguments.

There is also support for merchant belt, which can be enabled if you have one available. In case your belt is not complete or is totally non-existent, the system has semi-automatic tool wielding functionality: if you use /lj and then /mine, the macros will automagically change to proper tools (saw -> hammer/pick). The system is not perfect and sometimes it may fail, however.

Hint: Another useful module for merchants is the reagent pouch handler. It provides several useful functions for handling reagents.

Notice that you probably MUST redefine some of these macros and default settings in your TF configuration file! (See Settings-section below.)

Note

In below table: (*) = autotargets, (!) = moves items to item target

Table 4.26. Guild: Merchants macro commands

Command	Description
/alloy <material1,material2[,material3...]>	Alloy specified materials. Example: /alloy illumium,nullium
/amal <material>	Amalgamate given material. Workbench and tools are automatically selected.
/boxname	Name of the mineral box to use with /mbox operation mode. Default value is 'collect'.
/gcut [material]	Use gem cutting at material. (!) Workbench and tools are automatically selected. If material is not specified, material from previous /gcut command is used.
/havebelt	Toggle support for merchant belt functionality on/off.
/lj [target]	Use lumberjacking (at optional target, if no target given, autotargetting or previous specified target is used.) (*) (!)
/mbox	Change item move target to box (labeled as 'collect' by default, use /boxname command to change this setting.)
/mcut <material> [number] [/amount in grams]	Mineral cut material. Optional number/index and amount/size can be given in grams. If no amount/

Command	Description
	size is specified, material chunk is cut in half. Use "/mcut" without arguments to get some examples.
/mdisc	Change item move target to your floating disc.
/mdrop	Change item move target to dropping of the item.
/mine [target]	Use mining (at optional target, if no target given, autotargetting or previous specified target is used.) (*) (!)
/mr [material]	Make reagent from material. Tools and spell are automatically selected based on the material. If no material is specified, material of previous /mr command is used. Invalid material will print a list of reagents, spells and require materials for them.
/pouchname	Set name of the reagent pouch to use with /mr skill. Used only if /usepouch option has been enabled.
/refine <material> [number]	Refine specified material, or numbered chunk of material.
/usepouch	Toggle whether to use a reagent pouch with /mr (make reagent) command's functionality. See also /pouchname.
/wbgrep <workbench> <mineral regexp>	Performs a "grep '<mineral regexp>' look at <workbench>" with workbench mangling enabled and filters results. This is useful for quick searches to see if given material exists in the bench and how many pieces there are.
/wbmang	Toggles pro-workbench beautifying / output list mangling. This mangling mode tries to make the pro-workbench output more readable, and adds mineral number indexes for easier accesss. Especially helpful when you need to work on specific # of mineral.

Table 4.27. Guild: Merchants 'general' type command bindings

Command	Quiet	NoTarget	Description
wbl			/wbgrep

Settings and user-replaceable macros

Variables and user-replaceable macros definedby this module are described in a table below. They have specific defaults, which you can override in your configuration if you re-set/re-define them after loading the module. Please refer to Setup-section of this manual or see the example configuration for this module.

Table 4.28.

Variable / macro name	Description
mtool_mine	Tool(s) for mining. Example: */set mtool_mine=pick 1,pick 2*

Variable / macro name	Description
mtool_lj	Tool(s) for lumberjacking. Example: */set mtool_lj=saw*
mtool_bs	Tool(s) for blacksmithing.
mtool_cp	Tool(s) for carpentry.
mtool_gc	Tool(s) for gem cutting.
mtool_gb	Tool(s) for glassblowing.
mtool_ma	Tool(s) for masonry.
mtool_sw	Tool(s) for sewing.
mtool_sc	Tool(s) for sculpture skill.
mforge_*	Forges/workbenches for skills (similar to mtool_* variables)

Guild: Alchemists (gm-alchemist.tf)

Alchemist module provides several macros that can be very helpful with potion making and other alchemist activities. Commands for easier handling of herb jars and organ cans, command macro that automates potion mixing to great degree and additional helper for potion research.

Potion research is a big part of alchemist guild activities, but also one of the more tedious ones. It requires juggling with organs, herbs and minerals, keeping book of findings, etc. GgrTF alchemist module will come to help, as it automates getting the herb and organ from your containers, submitting the finished potion concoction flask to authenticator and stores results in a text file.

Furthermore, the said text file is written in a specific format, that can be used with an external utility called AlchTool, developed by Jeskko and yours truly (Ggr). This PHP-based web utility can greatly increase your potion research efficiency. Currently (as of March 2011) AlchTool is in "beta"-phase of version 2, but the code can be retrieved from Pupunen Mercurial repository at https://tnsp.org/hg/batmud/alchtool/.

Default location of the results text file is user's home directory, in file called *alch_results.txt*, but this can be changed by altering *galch_file* TF variable, for example: **`/eval /set galch_file=%{HOME}/someotherfile.txt`** You should put this, if you wish to, in your tfrc file AFTER the loading of alchemist module.

Table 4.29. Guild: Alchemists macro commands

Command	Description
/dolabels	Automatically re-label any jars and cans you have, to match the naming scheme of GgrTF Alchemist module for those containers.
/mix <mineral> <organ> <herb>	Mix a potion from given materials. This command will automatically retrieve organ and herb components from their specific containers. The other logic in the module will also return them into those containers if the skill is interrupted for some reason. Also, organ and herb names are checked for sanity.
/tmix <mineral> <organ> <herb>	Same as /mix, will mix a potion from given materials, but also performs functions for potion re-

Command	Description
	search. Finished potion will be submitted to authenticator, so only use this command when you are doing potion research and reside in the authenticator room of the alchemist guild.
/wconget <herb or organ>	Get specified herb or organ from correct container.
/wconlook <herb or organ name>	Automatically look into a container containing specified type of herb or organ. This is basically the same as 'look at jar x' or 'look at can y', except "/wconlook blueberry" would automatically look into the correct container.
/wconput <herb or organ>	Place specified herb or organ into correct container.

Table 4.30. Guild: Alchemists 'general' type command bindings

Command	Quiet	NoTarget	Description
wla			/wconlook
wput			/wconput
wget			/wconget

Guild: Barbarian (gm-barb.tf)

Provides a translator for barbarian reputation bar, reputation difference calculator and handler for automated looting and burning.

Macro commands

<> = required argument, [] = optional argument

Table 4.31. Macro commands

Binding	Macro	Description
repu	/showrep	Prints your current reputation and reputation difference from previous invoking of the command.
lb	/lootburn	Starts looting and burning skill, lights a torch, waits for couple of seconds and drops corpses from inventory. After burning the torch is extinguished and current reputation printed out. Then action determined by "/burnaction"-setting (see below) is executed. If a skillbreak is detected, corpse dropping will be cancelled immediately.
	/burnaction <action>	Sets the action done after finished burning. Available op-

Binding	Macro	Description
		tions are: *none*, *cash* (drop small coins) and *noeq* (drop low coins and noeq).

Table 4.32. Guild: Barbarian 'general' type command bindings

Command	Quiet	NoTarget	Description
repu		X	/showrep
lb		X	/lootburn
burn		X	/lootburn

Guild: Spiders (gm-spider.tf)

Provides very basic management help for spider demons, plus lites for spider servants.

Table 4.33. Guild: Spiders macro commands

Command	Description
/spidstatus	Prints time elapsed since last demon control, demon drains (and possibly other demon status information in future).

Guild: Tigers (gm-tiger.tf)

Reporting of succesful and failed 'tiger claw' hits, lites for claw and mak reputation gains.

Table 4.34. Guild: Tigers macro commands

Command	Description
/rtiger	Toggle reporting of tiger-module related things.

Guild: Tarmalens (gm-tarma.tf)

Provides command bindings for most common tarmalen spells. Triggers for tracking latest heal alls cast (of course, only those that have 'hit' you are counted.) Optional keybindings for heals, party heals, etc.

Note

This module requires the generic magical guilds support module. The keybinds require targetted cast module.

Table 4.35. Guild: Tarmalens macro commands

Command	Description
/lastheal	Show latest detected heal all.

Table 4.36. Guild: Tarmalens 'cast' type command bindings

Command	Quiet	NoTarget	Description
uns			Unstun
unp			Unpain
bot			Blessing of Tarmalen
rp			Remove Poison
da			Detect Alignment
cp			Cure Player
clw	X		Cure Light Wounds
csw	X		Cure Serious Wounds
ccw	X		Cure Critical Wounds
mih	X		Minor Heal
mah	X		Major Heal
th	X		True Heal
miph	X	X	Minor Party Heal
maph	X	X	Major Party Heal
tph	X	X	True Party Heal

Table 4.37. Guild: Tarmalens keybindings

Key(s)	Function
Meta/Alt + [asd]	Party heals (minor, major, true)
Meta/Alt + [qwer]	Single, targetted heals (cure light, serious, critical, major heal, true heal)
Meta/Alt + [zxc]	Cast deaths door, unstun, unpain
Meta/Alt + f	Stop casting/skills
Numpad Ins	/pss (show party status)

Guild: Druids (gm-druid.tf)

Barebones module with default bindings for some druid spells and support for fails/fumbles. You will probably also want to use the raise module, which provides help for casting reincarnations (along with raises, resurrects, etc.)

Table 4.38. Guild: Druids 'cast' type command bindings

Command	Quiet	NoTarget	Description
eblood			Earth Blood
epower			Earth Power
vmant			Vine Mantle
eskin			Earth Skin
wf	X		Wither Flesh

Command	Quiet	NoTarget	Description
sl	X		Star Light
gf	X		Gem Fire
inquiry			Inquiry to Ahm
cflex			Flex Shield
wdl			Wilderness Location

Table 4.39. Guild: Druids 'general' type command bindings

Command	Quiet	NoTarget	Description
cstaff			@@cast charge staff at amount

Guild: Nuns (gm-nun.tf)

Includes several useful functions for nuns, such as relic identification messages translator, useful command macros, prayer hour reminder (acts as a "prot"), simple task status keeper, counter+lite for accomplished tasks, turn message translator, etc.

Table 4.40. Guild: Nuns macro commands

Command	Description
/beemaint [hive number]	Performs beehive maintenance, extract honey and wax from honeycombs and clean up the beehive.
/np	Use 'npray' on every member of your party. This requires GgrTF::PSSMangle module.
/ntask	Show currently active/started nun task.

Table 4.41. Guild: Nuns 'cast' type command bindings

Command	Quiet	NoTarget	Description
sshield			Soul Shield
st	X		Saintly Touch
hh	X		Holy Hand
de	X		Dispel Evil
du	X		Dispel Undead
how	X		Holy Wind
flames	X		Flames of Righteousness
haven		X	Celestial Haven
pfe			Protection from Evil

Guild: (Old) Lords of Chaos (gm-oldloc.tf)

Very barebones module with support for "blood" ripaction and fail/fumble handling.

Table 4.42. Guild: (Old) Lords of Chaos macro commands

Command	Description
/locaction <action>	This setting is related to LoCs, but defined in main GgrTF module as also other than LoCs have use for this, if partying with a LoC.
/ripaction <action>	LoC module adds option "blood" to /ripaction, which automagically runs "lord_chaos blood corpse" on monster RIP.

Appendix A. Support

- There is no support. GgrTF development has been in maintenance mode for several years. You can still report bugs via tells or e-mail, but do not expect that they will be necessarily acted upon.
  - Take your time to *think* while writing, and before submitting your report. Good rule of thumb is to include all information that you think might be relevant, but no more than that!
  - Remember to include information about your system and environment, (*NIX/Linux/Windows), version of TinyFugue [http://tinyfugue.sf.net/] ("/ver" command in TF), and version of GgrTF ("/gver" command).
- We do not provide support for usage, please refer to the user's manual and only in case the issue is not documented, you might be eligible to ask about it.
- Neither we provide technical support for extending GgrTF, nor documentation of the internals. You'll have to figure out the workings yourself.
- Be ready to be contacted by developers, in case help and/or more information is required with debugging your issue(s).
- Remember not to bother the developers via tells too much, we want to play the game too, instead of just answering your questions and resolving problems.

Sending patches/corrections

If you have prepared a fix for some bug or a new feature, you can send us a patch. *A changeset patch made against current Mercurial repository head/tip is highly preferred*, short instructions on how to do that can be found below. You will probably need to refer to Mercurial guide [http://www.mercurial-scm.org/guide/] for generic Mercurial usage instructions and how to set it up. For making proper commits, you'll need to at the very least set up your username, etc.

```
# 1) Make changes to files
$EDITOR ggrtf.tf

# 2) Commit related changes to a changeset your local repository
hg ci [filenames of changed files or nothing, if you wish to commit everything]

# 3) Repeat steps 1 to 2 until satisfied

# 4) Export your local changes to a bundle
hg bundle mychanges.bz2 https://tnsp.org/hg/batmud/ggrtf/

# 5) Send mychanges.bz2 to us (Ggr), as e-mail attachment ccr@tnsp.org
```

Appendix B. Frequently Asked Questions (FAQ)

B.1. Is "Grizzt" included in the documentation?

Unfortunately no. Grizzt may be included in some future version, though.

B.2. When will next version of GgrTF be released?

"When it's done."

B.3. How to get rid of the battle round flag/marker?

The BatMUD battle round marker (long line of asterisks "*") is required for GgrTF's internal functionality, but you can gag it with following piece of code (add it to your .tfrc or similar):

```
/test prlist_insert("event_battle_round", "myroundgag")
/def -i myroundgag = /substitute -ag
```

The first line adds macro "myroundgag" to be executed on each round marker, the second line defines that macro and sets it to gag the round marker.

B.4. My statusline is not updating, why? (HP/SP/EP values not updating)

Maybe you haven't set up your 'short score' or 'prompt' in BatMUD or there is some typo in the setting. See setup-section for how to do that properly. If you wish that 'sc' line to not be visible, you can turn on '/gagsc' option.

B.5. How can I get my settings saved automatically? Or can I?

Automatic saving and loading is possible, please refer to state-saving and setup -sections of this manual for more information.

B.6. A spell/prot cast by others at me registers in prot-reporting, but does not work when I cast it on myself!

This is most likely a bug, please report the issue via e-mail. Remember to include the lines you get when you cast the spell on yourself (aka the "prot goes up"-line).

Also see the next question below.

B.7. Sometimes the spellword translator does not work, what gives?

Your 'cutter'-setting might be too small, the default in BatMUD is 80, which adds hard linefeeds too early on a line, thus breaking many triggers of GgrTF, including GgrTF::Spellnames module. 'cutter 9999' or 'cutter off' should fix this problem.

Please refer to BatMUD settings -section of this manual for correct in-game settings.

B.8. The TF5 module does not work! I am using TF 5.0 beta 6 or older.

The functionality used by the TF5 module was introduced in TF 5.0 beta 7, so you need to have that or a later version.

B.9. The keypad/numpad functionality (movement, targetting) is not working.

Make sure you have configured your terminal correctly, refer to terminal configuration section for more information.

Appendix C. How to verify package signatures via GnuPG

If you wish, you may verify the tarball package signatures with GnuPG, in case you want to be sure that you are getting unmodified files directly from the upstream. To do that, use the following commands:

- Retrieve Matti Hämäläinen's public key from a keyserver: **`gpg --keyserver wwwkeys.pgp.net --recv-keys 0x307BAAE3`**
- Verify the package signature (assuming you have downloaded both the file and signature): **`gpg --verify ggrtf-0.7.4.1.tar.gz.asc`** If the data is untampered, you should get a result like 'gpg: Good signature from "Matti Hamalainen (ccr) <ccr@tnsp.org>"'

Please refer to GnuPG or PGP documentation for more information about their usage. A great introduction to PKI, cryptography and public key signing, the GNU Privacy Handbook [http://www.gnupg.org/gph/en/manual.html] is available from GnuPG project. (Versions in other languages and formats, including PDF, are available from http://www.gnupg.org/documentation/guides.en.html [http://www.gnupg.org/documentation/guides.en.html].)