Galosh User's Manual

Michael Clarke

Table of Contents

1 Introduction

Galosh is a suite of related utilities to ease the lives of amateur radio operators. Currently, Galosh provides applications to manage your log and deal with the QSLing process. As time goes on, the suite will be expanded to support an increasing range of activities within the hobby.

Galosh has a strong focus on efficient offline operation, and is aimed at users who are comfortable with, and prefer to work using, the command-line. The suite is designed to be user-modifiable at every level, and you are encouraged to tinker and make Galosh do what *you* want. The full source-code is, of course, available, but beyond that, individual applications offer a hook system called *mixins*. This allows you to attach your own code to applications and modify their behaviour.

1.1 Get Involved!

Want to get involved? Galosh is mostly written in Common Lisp, with a bit of Perl in places. Code, documentation, and some graphical design work are all required, as are suggestions of different use-cases. To get started, fork the project on Github (`http://github.com/clarkema/galosh`), where you can also open issues or feature requests. Alternatively, e-mail the author at `mike@galosh.org.uk`.

2 Installation

At this stage in its development, Galosh is not intended to be *installed* as such, but run directly from a `git` checkout. To get up and running, you only need to have `git` and `make` installed.

```
$ git clone git://github.com/clarkema/galosh.git
$ cd galosh
$ sudo make install-debian-packages
$ make install-quicklisp
```

Provided that you are running Debian GNU/Linux, this will:

1. install any packages you are currently missing; and
2. download Quicklisp and set it up to manage Lisp systems within the checkout directory.

If you're not running Debian, you'll have to perform the 'make install-debian-packages' stage manually by inspecting the Makefile to work out what you need and installing the equivalent packages on your system. Don't forget to run `make install-quicklisp` afterwards.

Once all the dependencies are installed, ensure that the main `galosh` script is in your `PATH`, either by adding the checkout directory to your `PATH` directly, or by symlinking the `galosh` command into a directory that is already in your `PATH`.

Galosh makes use of various sources of data from the Internet. Once `galosh` is in your path, the final step is to download the most recent versions of these data:

```
$ galosh update
```

2.1 Bash Completion (Optional)

Adding Bash completion support is fully optional, but offers you the convenience of being able to tab-complete Galosh command-names and options within your shell. For example, with completion enabled, you can simply type `galosh` and press tab twice to get a list of all available Galosh commands.

Galosh provides a Bash completion file in '`contrib/galosh-bash-completion`'. To have completion work for you, this file needs to be sourced, one way or another, when `bash` starts. There are many ways of achieving this. One possibility is make a directory such as '`$HOME/local/etc/bash_completion.d`' to hold your personal Bash completion files, and then add something like this to your '`.bashrc`':

```
#####################
# Enable completion if possible
#####################
if [ -f /etc/bash_completion ]; then
    . /etc/bash_completion

    LOCAL_COMPLETION_DIR=$HOME/local/etc/bash_completion.d

    if [ -d $LOCAL_COMPLETION_DIR -a -r $LOCAL_COMPLETION_DIR -a \
            -x $LOCAL_COMPLETION_DIR ]; then
        for i in $LOCAL_COMPLETION_DIR/*; do
            [[ ${i##*/} != @(*~|*.bak|*.swp|\#*\#|*.dpkg*|.rpm*) ]] &&
                [ \( -f $i -o -h $i \) -a -r $i ] && . $i
        done
    fi

    unset LOCAL_COMPLETION_DIR
    unset i
fi
```

With this in place, symlink ‘`contrib/galosh-bash-completion`’ into ‘`$HOME/local/etc/bash_completion.d`’. Next time you restart your shell, you should have completion of `galosh` commands.

3 Configuration

Galosh is designed to be as flexible as possible and is configurable on many levels. Unfortunately, the cost of this rich configurability is that is can sometimes be difficult to know where and how to make changes. This chapter gives an overview of Galosh's configuration options.

3.1 Global Configuration

Galosh will read its global configuration from '`$HOME/.galosh/config`'. Any configuration options that you would like to apply to all (or nearly all) of your repositories, such as your callsign, you should set in this file. The configuration file uses an INI-style format, with section headers in square brackets on a line of their own. Options are specified one per line, with an equals sign between the name and value, and no whitespace at the start of the line. To start with, edit your global configuration file to specify your name and call.

```
[user]
name = Mike Clarke
call = VP8DMH
```

Configuration options are often referred to in the rest of the documentation in *dotted form*. An option written in dotted form is written on a single line, with a period between the section and option names. For example, we might refer to the `call` option above by saying "Set `user.call` to your callsign."

Notice I said above that you should use the global configuration file for any options that you would like to apply to *nearly* all of your repositories. Each repository also has its own configuration file, which follows exactly the same format as the global file. This allows you to override options on a per-repository basis by entering only those options that you need to change in the repository configuration file. So, just what is a 'repository' anyway? This brings us neatly on to the next section...

3.2 The Repository

Galosh is designed as a suite of separate but related commands. As such, many of the Galosh commands might need access to shared data; from configuration up to your full QSO log. However, you might well want to maintain multiple logs for different purposes; perhaps an 'everyday' log and a 'contest' log, or a distinct log for a special event station. It would be inconvenient if dealing with different logs, rigs, or configurations for any other purpose entailed passing explicit details to every Galosh command you wanted to run. To avoid this, Galosh uses the concept of *repositories*, in a similar fashion to version control systems like `git`, upon which Galosh is in many ways modelled.

Most uses of Galosh require you to be in a repository.

A repository is nothing more than a special directory on the file system that contains configuration files, your log, and other related information. To create one, simply create a new directory, change into it, and run `galosh init`.

```
$ mkdir log
$ cd log
$ galosh init
```

We can explore the new repository with `tree`:

```
$ tree -a
.
`-- .galosh
    |-- config
    |-- log.db
    `-- tmp
```

'`.galosh/log.db`' is the new (currently empty) SQLite database which will contain your logs. '`.galosh/tmp`' is used for temporary files created in the course of running various commands such as `galosh journal`, and can be ignored. Finally, '`.galosh/config`' is the configuration file for this repository.

4 Logging

QSO logging under Galosh is provided by the `log` command. Once you have initialized a log with `galosh init`, QSO logging under Galosh is intended to be as fast as possible; allowing efficient logging under busy run conditions, while also offering sufficient flexibility to allow the operator to enter additional information for a QSO as required.

4.1 Your First QSO

To get started, run `galosh log` within your repository directory. Once the logger has started, simply type in a callsign and hit enter twice. A QSO will be added to the log using all of the default information: 59 both ways on 14260kHz.

4.2 Commands

Normally, anything you type in the logger is taken as the start of a log entry. The first word is taken as the callsign you are logging, and characters are automatically converted to upper-case as you type.

If you start a new entry with a colon, your entry will be interpreted as a command rather than a log entry. Command mode allows you to set information such as your frequency and mode, and to quit the application.

4.2.1 Setting A Different Frequency

Galosh does not currently get frequency information directly from your radio; when you want to log on a new frequency, you must tell the logger manually. To do this, type:

```
:set qrg 7125000
```

Note that the frequency is in Hertz, not kHz.

4.2.2 Setting Mode And IOTA Information

Mode and IOTA information can be changed in much the same way as the frequency:

```
:set mode SSB
:set mode CW
:set iota AN-001
```

4.3 Lesser-used Options

4.3.1 Time Adjustment

Galosh will log using your computer's idea of UTC. Should this be wrong, ideally it should be corrected at the operating system level. However, in some cases that might not be possible, and so Galosh provides the `core.time_fudge` option. This allows you to specify an integer number of seconds to be added to the operating system's idea of UTC. For example, to log a time of 3 hours earlier than would otherwise be the case set `core.time_fudge` to `-10800`.

5 QRZ

Galosh includes several facilities to help you discover more about the stations you're working. This aids in accurate logging, as well as helping you spot stations that are useful for awards you're working towards, and being of general interest.

All of the current lookup features rely on callsign data from `qrz.com`. `qrz.com` offers two forms of access to its database: an XML API for online lookup, and a downloadable database for offline searches. Each has advantages and disadvantages.

The online lookup is always up-to-date with the latest data, and offers far more information than the offline database. However, if you don't have the luxury of an always-on Internet connection, it's of no use. Even if you *do* have an always-on connection, a high latency connection such as a satellite link can introduce an unacceptable delay into the lookup process.

The offline database requires you to download data files from `qrz.com` and import them using `galosh import-qrz-db`, but once the data are imported you can perform lookups much more quickly, and without an Internet connection. The disadvantage is that the offline database provides far less information for each callsign; essentially just name and postal address, along with some license metadata. In particular, the offline database doesn't contain location (latitude and longitude, or grid) or IOTA information.

Galosh supports both `qrz.com`'s XML API and its downloadable database. To get started with either of these, you'll need a subscription from `qrz.com`.

5.1 Getting Started With The qrz.com XML API

If you don't already have an XML subscriber account with `qrz.com`, go to `http://www.qrz.com/XML/index.html` and sign up. Once you have your username and password, you'll need to configure Galosh (see Chapter 3 [Configuration], page 4). Add the following to your configuration file:

```
[qrz]
user = yourcall
password = yourpassword
```

Once Galosh is configured with your account details, you can perform QRZ lookups from the command-line:

```
$ galosh qrz vp8dmh
CLARKE, Mike
Rothera Research Station
Adelaide Island
ANTARCTICA

Lat/Long: -67.566667 -68.133333
Grid: FC52wk
IOTA: AN-001

QSL Via: VIA G0VGS
QSL Methods: eQSL
```

5.2 Getting Started With The qrz.com Offline Database

If you don't already have a database download account with `qrz.com`, go to `http://www.qrz.com/pd/` and sign up. Once you've signed up, download and unpack the archive described as 'Single database file callbkc.dat, with index - sorted by callsign'. In the unpacked files, you should see '`callbkc.dat`' and '`country.dat`'. These contain the raw callsign and country data; for them to be useful to Galosh they must first be imported, using `galosh import-qrz-db`.

```
$ galosh import-qrz-db callbkc.dat country.dat
```

Once the import is complete, you can perform offline QRZ lookups from the command-line:

```
$ galosh qrz --offline vp8dmh
CLARKE, Mike
Rothera Research Station
Adelaide Island
ANTARCTICA
```

As can clearly be seen from the output, less information is available when running offline. However, running both commands under `time` demonstrates another difference:

```
$ time galosh qrz vp8dmh
CLARKE, Mike
Rothera Research Station
Adelaide Island
ANTARCTICA

Lat/Long: -67.566667 -68.133333
Grid: FC52wk
IOTA: AN-001

QSL Via: VIA G0VGS
QSL Methods: eQSL

real    0m6.816s
user    0m0.660s
sys     0m0.788s

$ time galosh qrz --offline vp8dmh
CLARKE, Mike
Rothera Research Station
Adelaide Island
ANTARCTICA

real    0m1.237s
user    0m0.552s
sys     0m0.556s
```

On the satellite connection I was using when I wrote this, there is a 5.5-second difference in response time.