Ring Documentation

Release 1.0

Savoir-faire Linux

April 25, 2016

Contents

Contents:

CHAPTER 1

Getting Started

For now, this manual only covers developers documentation. To get started with Ring, you should visit the download page where there are instructions for installing Ring on your system.

CHAPTER 2

Contributing

Ring loves external contributions but we do not use the Github PR system. Instead, we host a public gerrit server: https://gerrit-ring.savoirfairelinux.com

Before submitting a contribution, you need to register on our Gerrit server either with your Github or Google account.

Head to the settings section to set one of the following:

- http password and username: https://gerrit-ring.savoirfairelinux.com/#/settings/http-password
- ssh key: https://gerrit-ring.savoirfairelinux.com/#/settings/ssh-keys

In each Ring submodule there is a `.gitreview` file. It contains all the necessary info to send a patchset to our gerrit server.

After you commited your changes (one or multiple commits) you can submit them with:

```
git-review
```

More documentation on Gerrit can be found on the official website.

2.1 Making Changes

- [Optionnal] Create a ticket in our Tuleap bug tracker https://tuleap.ring.cx/projects/ring
- Make commits of logical units.
- Check for unnecessary whitespace with *git diff –check* before committing.
- **Make sure your commit messages are in the proper format**
  - 50 chars title
  - 80 chars message width

CHAPTER 3

Compiling and installing

This section covers compiling the different components of Ring.

3.1 Using the make-ring script (recommended)

3.1.1 Dependencies

The Ring installer uses python3. Please make sure it is installed before running it.

3.1.2 Initialize the repositories

```
./make-ring.py --init
```

3.1.3 On Linux

1. Build and install all the dependencies:

```
./make-ring.py --dependencies
```

Your distribution's package manager will be used.

2. Build and install locally under this repository:

```
./make-ring.py --install
```

3. Run daemon and client that were installed locally:

```
./make-ring.py --run
```

You can then stop the processes with CTRL-C.

You can also run them in the background with the `--background` argument and then use the `--stop` command to stop them. Stdout and stderr go to daemon.log and client-gnome.log.

Install globally for all users instead

```
./make-ring.py --install --global-install
```

Run global install:

```
gnome-ring
```

This already starts the daemon automatically for us.

Uninstall the global install:

```
./make-ring.py --uninstall
```

3.1.4 On OSX

You need to setup Homebrew (<http://brew.sh/>) since their is no built-in package manager on OSX.

Build and install all the dependencies:

```
./make-ring.py --dependencies
```

Build and install locally under this repository:

```
./make-ring.py --install
```

Output

You can find the .app file in the `./install/client-macosx` folder.

3.1.5 On Android

Please make sure you have the Android SDK and NDK installed, and that their paths are properly set. For further information, please visit <https://github.com/savoirfairelinux/ring-client-android>

Build and install locally under this repository:

```
./make-ring.py --install --distribution=Android
```

Output

You can find the .apk file in the ./client-android/ring-android/app/build/outputs

3.2 Just the daemon (advanced)

3.2.1 Linux

1. Compile the dependencies

```
cd ../contrib/
mkdir native
cd native
../bootstrap
make
```

2. Compiling dring

```
cd ../../
./autogen.sh
./configure
make
```

3. Installing dring

```
make install
```

Done !

3.2.2 OSX

1. Installing dependencies

Without a package manager

```
cd extras/tools
./bootstrap
make
export PATH=$PATH:/location/of/ring/daemon/extras/tools/build/bin
```

With a package manager (macports or brew)

Install the following:

- automake
- pkg-config
- libtool
- gettext
- yasm

2. Compiling dependencies

```
cd contrib
mkdir native
cd native
../bootstrap
make -j
```

3. Compiling the daemon

```
cd ../../
./autogen.sh
./configure  --without-dbus --prefix=<install_path>
make
```

If you want to link against libringclient and native client easiest way is to add to ./configure: `--prefix=<prefix_path>`

Done!

Common Issues

`autopoint not found`: When using Homebrew, autopoint is not found even when gettext is installed, because symlinks are not created. Run: `brew link --force gettext` to fix it.

Clang compatibility (developers only)

It is possible to compile dring with Clang by setting CC and CXX variables to 'clang' and 'clang++' respectively when calling ./configure.

Currently it is not possible to use the DBus interface mechanism, and the interaction between daemon and client will not work; for each platform where dbus is not available the client should implement all the methods in the *_stub.cpp files.

3.3 Just libRingClient (advanced)

3.3.1 Basic Installation

These are generic installation instructions.

To install the appplication, type the following commands in a console, while in the root directory of this application:

```
mkdir -p build
cd build
cmake ..
make -j3
make install
```

The following options are often usefull to append to the cmake line:

```
-DRING_BUILD_DIR=<daemon install location>
-DCMAKE_INSTALL_PREFIX=<install location>
-DCMAKE_BUILD_TYPE=<Debug to compile with debug symbols>
-DENABLE_VIDEO=<False to disable video support>
```

3.3.2 Explanation

This script will configure and prepare the compilation and installation of the program and correctly link it against Ring daemon.

All needed files will be built in "build" directory. So you have to go to this directory:

```
cd build
```

Then execute the Makefile, to compile the application (src, doc...)

```
make
```

Then install it all using:

```
make install
```

You have to use "sudo" to be able to install the program in a protected directory (which is the case by default and most of the time). Therefore it will ask for your system password.

3.3.3 OS X Install

Install necessary tools:

```
brew install cmake
brew install qt5
export CMAKE_PREFIX_PATH=<path_to_qt5>
```

hint: default install location with HomeBrew is /usr/local/Cellar/qt5

First make sure you have built ring daemon for OS X.

```
mkdir build && cd build
cmake .. -DCMAKE_INSTALL_PREFIX=<install_dir_of_daemon> [-DCMAKE_BUILD_TYPE=Debug for compiling with
make install
```

You can now link and build the OSX client with Ring daemon and LRC library

3.3.4 Internationalization

To regenerate strings for translations we use lupdate (within root of the project)

```
lupdate ./src/ -source-language en -ts translations/lrc_en.ts
```

Hint: On OSX lupdate is installed with Qt in /usr/local/Cellar/qt5/5.5.0/bin/ when installed with HomeBrew

3.4 Just the gnome client (advanced)

3.4.1 Requirements

- Ring daemon
- libRingClient
- GTK+3 (3.10 or higher)
- Qt5 Core
- X11
- gnome-icon-theme-symbolic (certain icons are used which other themes might be missing)
- libebook1.2 / evolution-data-server (3.10 or higher)
- libnotify (optional, if you wish to receive desktop notifications of incoming calls, etc)
- gettext (optional to compile translations)

On Debian/Ubuntu these can be installed with:

```
sudo apt-get install g++ cmake libgtk-3-dev qtbase5-dev libclutter-gtk-1.0-dev gnome-icon-theme-symbo
```

On Fedora:

```
sudo yum install gcc-c++ cmake gtk3-devel qt5-qtbase-devel clutter-gtk-devel gnome-icon-theme-symboli
```

3.4.2 Compiling

Run the following from the project root directory:

```
mkdir build
cd build
cmake ..
make
```

You can then simply run `./gnome-ring` from the build directory

3.4.3 Installing

If you're building the client for use (rather than testing of packaging), it is recommended that you install it on your system, eg: in '/usr', '/usr/local', or '/opt', depending on your distro's preference to get full functionality such as desktop integration. In this case you should perform a 'make install' after building the client.

3.4.4 Building without installing Ring daemon and libRingClient

It is possible to build ring-client-gnome without installing the daemon and libRingClient on your system (eg: in /usr or /usr/local):

1. build the daemon
2. when building libRingClient, specify the location of the daemon lib in the cmake options with `-DRING_BUILD_DIR=`, eg: `-DRING_BUILD_DIR=/home/user/ring/daemon/src`
3. to get the proper headers, we still need to `make install` libRingClient, but we don't have to install it in /usr, so just specify another location for the install prefix in the cmake options, eg: `-DCMAKE_INSTALL_PREFIX=/home/user/ringinstall`
4. compile libRingClient and do 'make install', everything will be installed in the dir specified by the prefix
4. point the client to the libRingClient cmake module during configuration: `-DLibRingClient_DIR=/home/user/ringinstall/lib/cmake/LibRingClient`

3.4.5 Debugging

For now, the build type of the client is "Debug" by default, however it is useful to also have the debug symbols of libRingClient. To do this, specify this when compiling libRingClient with `-DCMAKE_BUILD_TYPE=Debug` in the cmake options.

CHAPTER 4

Ring release process

This page explains the process of making a new Ring release. It can be used as checklist of things to remember when making a new release. It was written to clarify the release process between the Ring dev team and the distribution maintainers.

4.1 Release tarball

Ring is released in the form of a tarball. They are hosted here:

- https://gpl.savoirfairelinux.net/ring-download/ring-release/tarballs/

Tarballs are generated from the integration branch of the ring-project repository with a job on our Jenkins server. They include a copy of all contrib libraries configured in `daemon/contrib/src`. If you are a Savoir-faire Linux employee, you may trigger the job from this page.

4.1.1 Naming scheme

Tarballs respect the following naming scheme `ring_<date>_<number_of_commits>.<commit_id>.tar.gz` where:

- **date** is the current date, for example 20160422
- **number_of_commits** represents the number of commits that day
- **commit_id** is the commit id of the last ring-project commit

4.2 Packaging

4.2.1 Distribution packaging

Distribution packages should be generated from the release tarballs. It is best that distributions exclude as much embedded libraries as possible from the tarballs and use their packaged versions instead.

Debian

The Debian package is maintained by Alexandre Viau <aviau@debian.org> as part of the Debian VoIP Team <pkg-voip-maintainers@lists.alioth.debian.org>.

The packaging is maintained using git-buildpackage and can be found in the following Alioth repository:

- git.debian.org:/git/pkg-voip/ring.git

The repository contains a Debian.README file explaining the process of importing a new version. To upload a new version of Ring, manual action is required by Alexandre. If he is unavailable, other members of the Debian VoIP team may do the upload.

4.2.2 Upstream packaging

The Ring dev team builds packages for popular Linux distributions. Those packages are built weekly. Instructions on installing the repositories can be found on ring.cx/download.

4.3 Stable releases

At this moment Ring is still considered in beta and does not support stable releases. This may or may not change when the beta period is over. The most secure and stable version of ring is the tip of the master branch.