Binc IMAP manual (DRAFT)

Andreas Aardal Hanssen

May 25, 2003

Contents

Chapter 1

Introduction

> "As an alternative to existing similar IMAP servers, Binc IMAP strives to be very easy to use, but robust, stable and secure."

Welcome to Binc IMAP, a project started December 2003 by Andreas Aardal Hanssen. Binc IMAP is an open source IMAP project which differs from existing IMAP projects. Here is a list of the project goals:

1. Binc IMAP will always have a helpful, hospitable community.
    - Although it is expected that users of Binc IMAP do their homework before posting to the mailing list, the server author and community of the Binc IMAP project will be friendly and will approach everyone with respect. The same behavior is expected from those who post to the list.
    - There will be no RTFM[1] on the project's mailing list. Flaming and personal insults on the project's mailing list will result in banning of the originator.
    - The community is encouraged to pay back to the project's contributors by sharing their own experience and contributions to Binc IMAP through the GPL license.
2. Binc IMAP will not compete with other IMAP projects
    - Under no circumstance will this project be in market driven competition with other IMAP servers.
    - Binc IMAP will first and foremost be a quality driven project.
    - This project is meant to influence the community of authors of similar network protocols and servers, and hopes to increase the general quality of software that is used all over the globe and beyond.

[1] According to acronymfinder.com, RTFM stands, among many other suggestions, for "Read The Fscking Manual" (fsck is a Unix filesystem checker)

3. Binc IMAP provides security through good design
    - A well designed server is less exposed to bugs than a poorly designed server.
    - The server will strive to use every kind of security enhancing feature, while keeping the implementation details as good and simple as possible.
    - The source is open and downloadable. Potential bugs and/or nasty pieces of code are easily uncovered when the whole community is able to study every line of code in detail. Bugs should always be reported to the project's mailing list.
4. Binc IMAP is modular
    - Where possible and practical from both a usage and design stand point, modules will be seperated and replaceable through *pluggable extension support*. Examples of future replaceables and enhancements include:
        - authentication modules (shadow, PAM, LDAP)
        - search modules
        - protocol extensions modules (namespaces, ACL, shared folders)
        - mailbox formats (Maildir, mbox)
    - With a modular and good object oriented design, it will be easy to quickly understand what every method and function does. This will increase third party developers' ability to write extensions and modifications fast.
5. Binc IMAP favors quality over quantity
    - Binc IMAP's releases are milestones. We strive for perfection.
    - Work on improving the existing design and extensibility will always go ahead of adding new features.
    - Through extensive module support, the community is encouraged to contribute to the adding and testing of new features.
    - Core design and implementation will always focus on quality.

This document serves as the server's main documentation.

Chapter 2

Installation

The installation procedure for Binc IMAP is designed to be quick and easy, and in most cases you will get what you want at your first attempt. The source code is designed to be able to compile on most UNIX-like platforms.

If you experience problems with compiling and installing this package on your own platform, don't hesitate to post your problem to the project's mailing list.

2.1 Downloading the package

Binc IMAP is available for a number of platforms and distributions. Download the package most suitable for your distribution from one of the following locations:

- Red Hat Linux:
  - `http://www.bincimap.org/`
  - `http://www.bincimap.andreas.hanssen.name/`
  - `http://bincimap.argonsoft.de/`
- Mandrake Linux:
  - `ftp://sunsite.uio.no/linux/Mandrake/Mandrake-devel/contrib`
- Debian Linux:
  - `http://packages.debian.org/testing/mail/bincimap.html`
  - `http://packages.debian.org/unstable/mail/bincimap.html`
- FreeBSD:
  - `http://www.freebsd.org/cgi/ports.cgi?query=bincimap&stype=all`

- Other (tarball):
  - http://www.bincimap.org/
  - http://www.bincimap.andreas.hanssen.name/
  - http://bincimap.argonsoft.de/

2.2 Building from the tarball

This section describes how you can build the Binc IMAP server using the compressed tar archive (*tarball*). This approach should work on all supported platforms.

The tarball creates a seperate directory in which it installs all source files. The installation procedure is typical for open source C and C++ software that uses the `autoconf` and `automake` tools. First, unpack the tarball:

```
# zcat bincimap-1.1.6.tar.gz | tar xvf -
<outputs a list of files>
```

Now, enter the directory that was created. In this directory, you will find the `configure` script.

```
# cd bincimap-1.1.6
# ls configure
configure
#
```

Run the configure script with all options you find appropriate. The `configure` options are listed in section 2.2.1 on the facing page. Most users will be ok with these common options:

```
# ./configure --prefix=/opt/bincimap --sysconfdir=/etc/opt/bincimap
```

The `configure` script will produce many lines of output, and finally, if all goes well, it will create all the `Makefile`s we need to build this projectm We can now run `make`:

```
# make
```

If the compile finishes with no problems, you can install your new software:

```
# make install
```

If the `prefix` option is `/opt/bincimap`, `make install` will create the following files:

```
/opt/bincimap/bin/bincimap-up            # Startup stub
/opt/bincimap/bin/bincimapd              # Main server
/opt/bincimap/man/man1/bincimap-up.1     # Man page
/opt/bincimap/man/man1/bincimapd.1       # Man page
/opt/bincimap/man/man5/bincimap.conf.5   # Man page
```

The `sysconfdir` option tells Binc IMAP where to install the configuration files. If `sysconfdir` is set to `/etc/opt/bincimap`, `make install` will create the following files:

```
/etc/opt/bincimap/bincimap.conf             # Main conf file
/etc/opt/bincimap/xinetd/imap               # xinetd conf file
/etc/opt/bincimap/xinetd/imaps              # xinetd conf file
/etc/opt/bincimap/supervise/imap/run        # supervise conf file
/etc/opt/bincimap/supervise/imap/log/run    # supervise conf file
/etc/opt/bincimap/supervise/imaps/run       # supervise conf file
/etc/opt/bincimap/supervise/imaps/log/run   # supervise conf file
```

You are now ready to configure Binc IMAP. Read more about this in section 3 on page 11.

2.2.1 Options to configure

- `--prefix=<dir>`
  Where to install binary files and man pages. Binary files will be installed under `<dir>/bin` and man pages go under `<dir>`/man.

- `--sysconfdir=<dir>`
  Where to install configuration files. The main conf file goes under `<dir>/bincimap.conf` and other conf files go under `<dir>/<path>/...`, depending on the type of conf files.

  Note: The bundled supervise files must **not** be stored on an NFS partition or any partition that does not support file locking.

- `--disable-ssl`
  Compile Binc IMAP with no SSL code, and without the OpenSSL requirement. Also removes the STARTTLS capability.

- `--enable-ssl` (default)
  Compile Binc IMAP with SSL code, and with the OpenSSL requirement. Enables the STARTTLS capability.

- `--enable-static`
  Compile Binc IMAP with static linkage.

2.3 Building from the src.rpm

The `src.rpm` packages are designed for the Red Hat Linux distribution. If you are not running Red Hat Linux or similar distributions and this procedure fails, then you should build Binc IMAP from the tarball (section 2.2 on page 8).

- Make sure you have the `rpmbuild` program installed. In Red Hat, it is a part of the `rpm-build` package.
- As root[1], run the following command:

  ```
  rpmbuild --rebuild bincimap-1.1.6-1.src.rpm
  ```

- If the command succeeds, you will have generated a binary rpm package that you can install.

  ```
  rpm -i bincimap-1.1.6-1.i386.rpm
  ```

2.4 After building and installing the package

When the service has been installed successfully, you should take your time to configure it properly. Please refer to section 3 on the next page.

[1] It is possible to build src.rpm packages as non-root. There is info available on http://www.rpm.org/ about this.

Chapter 3

Configuration

With most services, the hardest thing to get right is the configuration file. With Binc IMAP, this should not be a problem for the administrator. There are few options to set, and the package also provides a default configuration file whose default settings are fine for most uses.

This section will first give an overview of the format of Binc IMAP's configuration file, and it will then explain every option that you can set.

3.1 Introduction to `Binc::Storage`

Binc IMAP uses a special file format, called `Binc::Storage` after its C++ class name. This file format is human readable and editable, it's fast to parse programmatically, and it has very little overhead when representing data. It also groups options together in sections, supports comments (c-style) and arbitrary whitespace usage.

The structure of the file is as follows:

1. Aliases
2. Section1
   (a) key1=value1
   (b) key2=value2
   (c) ...
3. Section2
   (a) key1=value1
   (b) key2=value2
   (c) ...
4. SectionN...

Moving on to the actual format, the file starts with an optional list of aliases. These aliases start with a question mark (?), followed by the alias, a colon (:), the replacement text, and finally a semicolon (;). This example defines the alias "m" to represent the text "myfirstsection":

```
?m:MyFirstSection;
```

Then comes sections. A section starts with an identifier — the section name. The name contains only alphanumeric (a to z, A to Z and 0-9) characters, and it's case sensitive ("hello" and "HELLO" could be two different sections). The section content itself is enclosed in braces, as in the following example.

```
MyFirstSection {

}
```

Inside a section we have a comma seperated list of key=value pairs. The keys and values consist of a sequence of one or more alphanumeric words. The value can also contain quoted strings:

```
MyFirstSection {
    certificate path = "/etc/path/certificate.txt",
    optional argument = yes
}
```

The aliases from the start of the configuration file are useful if we wish to appreviate long words in a key or section name in order to save space. Normally this is only used in generated files, but they can also make a manually edited conf file more compact. Here we have replaced "MyFirstSection" with its alias "m":

```
m {
    certificate path = "/etc/path/certificate.txt",
    optional argument = "yes"
}
```

When accessed by Binc IMAP, the two sections are treated equally, as if there was no alias subsitution.

The following sections describe the `Binc::Storage` sections that Binc IMAP recognizes.

3.2 Sections in `bincimap.conf`

Now followes a brief description of all recognized sections in `bincimap.conf`.

3.2.1 Authentication

The authentication details in Binc IMAP are set in a section called `Authentication`:

```
Authentication {

}
```

The recognized options are:

- `allow plain auth in non ssl = yes|no`

  `yes` Clients are allowed to authenticate using plain text passwords when transport is not encrypted with SSL.

  `no` Clients are *not* allowed to authenticate using plain text passwords when transport is not encrypted with SSL.

- `auth penalty = <n>`

  If a user tries to authenticate but the password or username is wrong, the server will sleep for `<n>` seconds before allowing the user to try again.

This is an example of an `Authentication` section in `bincimap.conf`:

```
Authentication {
    allow plain auth in non ssl = no,
    auth penalty = 4
}
```

3.2.2 Logging

The logging details in Binc IMAP are set in a section called `Log`:

```
Log {

}
```

The recognized options are:

- `type = syslog|multilog`

`syslog` Binc IMAP will log using syslog.

`multilog` Binc IMAP will log to *stderr*, which is the default input for `multilog`.

- `environment ip variable = <var>`

If configured to log using `multilog`, Binc IMAP attempts to find the remote host's IP address in the environment variables. This option allows the administrator to suggest which variable the IP address is stored in.

- `syslog facility = <facility>`

  If Binc IMAP is configured to log using syslog, it will log using the syslog facility described by the alias `<facility>`.

- `syslog facility number = <facilitynr>`

  If Binc IMAP is configured to log using syslog, it will log using the syslog facility number `<facilitynr>`. This is typically used if the textual facility name is not recognized by Binc IMAP.

This is an example of a `Log` section in `bincimap.conf`:

```
Log {
    type = syslog,
    syslog facility = LOG_DAEMON
}
```

3.2.3 Security

The security details in Binc IMAP are set in a section called `Security`:

```
Security {

}
```

The recognized options are:

- `jail path = <path>`

  Binc IMAP's invocation stub, `bincimap-up`, enters a "chroot jail" path after starting the main server. The path is given in this setting and it must exist for Binc IMAP to function properly.

- `jail user = <userid>`

  Binc IMAP's invocation stub is by default run as the privileged user `root`. Before the stub enters its "chroot jail", it changes to the `<userid>` user, which should be a user with no privileges.

- `jail group = <groupid>`

  As with `jail user`, but sets the *group* that the invocation stub changes to before entering the jail.

This is an example of a `Security` section in `bincimap.conf`:

```
Security {
    jail path = "/opt/bincimap/bin",
    jail user = nobody,
    jail group = nobody
}
```

3.2.4 Mailbox settings

The mailbox settings in Binc IMAP are set in a section called `Mailbox`:

```
Mailbox {

}
```

The recognized options are:

- `type = <type>`

  Sets the type of mailbox that Binc IMAP should use as default. Currently, the only supported format is `Maildir`.

- `depot = <depottype>`

  Sets the type of depot that Binc IMAP should use. The depot decides how the contents of an IMAP user's mail path is interpreted. Currenty, the two supported depot types are `IMAPdir` and `Maildir++`..

- `path = <path>`

  When a user authenticates, Binc IMAP can optionally change to a seperate mail directory within the user's home area. This directory is set with the `path` setting. For mail storage configurations where there is no seperate area for mail, set `path=""`.

- `auto create inbox = yes|no`

  yes Binc IMAP will create the INBOX mailbox if the IMAP user does not already have one.

  no The server will *not* create INBOX when the user logs in for the first time.

- `auto subscribe mailboxes = <mailboxlist>`

When logging onto the IMAP service for the first time, the user will automatically be subscribed to the mailboxes in the comma seperated list `<mailboxlist>`. The content of the list if for example "INBOX,Trash,Drafts,Sent".

- `umask = <number>`

  All operations in a user's mail directory will be performed using this umask.

This is an example of a `Mailbox` section in `bincimap.conf`:

```
Mailbox {
    path = "Maildir",
    type = "Maildir",
    depot = "IMAPdir",
    auto create inbox = yes,
    auto subscribe mailboxes = "INBOX,Work,Work/Sent",
    umask = 0777
}
```

3.2.5 Session

The session details in Binc IMAP are set in a section called `Session`:

```
Session {

}
```

The recognized options are:

- `auth timeout = <seconds>`

  Before authenticating, Binc IMAP will allow clients to idle (stay connected with no activity) for `<second>` seconds before automatically closing the IMAP session.

- `idle timeout = <seconds>`

  After authenticating, Binc IMAP will allow clients to idle (stay connected with no activity) for `<second>` seconds before automatically closing the IMAP session.

- `transfer timeout = <seconds>`

  Every chunk of data that is sent by Binc IMAP to the IMAP client has `<seconds>` seconds to complete. If this timeout is exceeded, the IMAP session is closed.

- `transfer buffer size = <nrofbytes>`

  To save network roundtrips, Binc IMAP buffers up to `<nrofbytes>` bytes in memory before it is shipped off to the client. Greatly affects performance on high latency links.

This is an example of a `Session` section in `bincimap.conf`:

```
Session {
  auth timeout = 30,
  idle timeout = 1800,
  transfer timeout = 1200,
  transfer buffer size = 1024
}
```

3.2.6 SSL/TLS

The SSL details in Binc IMAP are set in a section called `SSL`:

```
SSL {

}
```

The recognized options are:

- `pem file = <file>`

  Binc IMAP reads a PEM encoded SSL certificate from this location. This file can often be generated with scripts under `/usr/share/ssl/certs`.

- `ca file = <file>`

  Gives Binc IMAP a file that contains certificate authorities. The purpose of this file is specific to the SSL protocol and it is used in the SSL hand shake.

- `ca path = <path>`

  Gives Binc IMAP a path in which the SSL library can search for files containing certificate authorities. The purpose of this path is specific to the SSL protocol and it is used in the SSL hand shake.

- `cipher list = <ciphers>`

  The string provided in `<ciphers>` informs the SSL library of which ciphers are available.

- `verify peer = yes|no`

Informs Binc IMAP wether or not the server should attempt to verify the peer's certificate.

This is an example of a `SSL` section in `bincimap.conf`:

```
SSL {
  pem file = "/usr/share/ssl/certs/server.pem",
  ca file = "",
  ca path = "",
  cipher list = "!ADH:RC4+RSA:HIGH:MEDIUM:LOW:EXP:+SSLv2:+EXP",
  verify peer = no
}
```

Chapter 4

The Binc IMAP Depot

The main role of an IMAP server is to give email clients an interface through which they can authenticate and then access email and mailboxes located on a (remote) mail server. The mails are stored on the mail servers in a special structure, and this structure varies from system to system.

Most often we distinguish between the mailbox *structure* (or *hierarchy structure*) and its *format*. The structure determines how the different mailboxes and submailboxes are stored, and the format decides how each and every email is stored within one mailbox.

Binc IMAP uses the generic container object *Depot* to describe the map between the hierarchy structure and its translation to selectable mailboxes in IMAP. The Depot has two specializations: one for IMAPdir and one for Maildir++.

4.1 Maildir++

Courier-IMAP defines the hierarchy structure Maildir++, which provides a way for existing Maildir users (Maildir is a mailbox format) to have multiple mailboxes and submailboxes inside the directory that contains the default mailbox "INBOX" (which often resides in `~/Maildir` for each UNIX user).

Read more about the Maildir++ format at the following location:

`http://www.inter7.com/courierimap/README.maildirquota.html`

Binc IMAP supports this mailbox structure with a few limitations and a few enhancements:

- The *maildirfolder* file is not created inside each Maildir submailbox. The reason for this is that this only works with mailbox formats that store a mailbox inside a directory, such as Maildir.
- Maildir++ quotas are not supported

- No Maildir++ restrictions to mailbox names apply (such as `.Trash`)
- Mailboxes inside a Maildir++ structure can be of any format, not just Maildir.

4.2 IMAPdir

IMAPdir is Binc IMAP's native mailbox structure. It is open and usable for most existing local mail clients. For a full description, look up the following web site, or search for the `bincimap-imapdir.html` file in the packaged documentation.

http://www.bincimap.org/bincimap-imapdir.html

IMAPdir does not extend/change any mailbox formats; it merely defines a way to describe mailboxes and submailboxes in a way that is suitable for an IMAP server, with as few restrictions as necessary.