The Leading Open Source Backup Solution

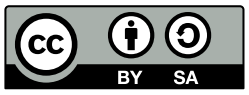
# Bacula® Miscellaneous Guide

Kern Sibbald

February 12, 2024

This manual documents Bacula Community Edition 13.0.4 (12 February 2024)







2024-02-12

version 13.0.4

Bacula Community

# Contents

# List of Tables

# Chapter 1

# Manually Installing Bacula

In general, you will need the Bacula source release, and if you want to run a Windows client, you will need the Bacula Windows binary release. However, Bacula needs certain third party packages (such as `MySQL`, `PostgreSQL`, or `SQLite` to build and run properly depending on the options you specify. Normally, `MySQL` and `PostgreSQL` are packages that can be installed on your distribution. However, if you do not have them, to simplify your task, we have combined a number of these packages into three **depkgs** releases (Dependency Packages). This can vastly simplify your life by providing you with all the necessary packages rather than requiring you to find them on the Web, load them, and install them.

## 1.1 Source Release Files

The Bacula source code has been broken into four separate tar files each corresponding to a different module in the Bacula SVN. The released files are:

`bacula-5.2.0.tar.gz` This is the primary source code release for Bacula. On each release the version number (5.2.0) will be updated.

`bacula-docs-5.2.0.tar.bz2` This file contains a copy of the docs directory with the documents prebuild. English HTML directory, single HTML file, and PDF file. The French, German, Spanish translations are in progress, but are not built.

`bacula-gui-5.2.0.tar.gz` This file contains the non-core GUI programs. Currently, it contains bacula-web, a PHP program for producing management viewing of your Bacula job status in a browser; and `bimagemgr` a browser program for burning CDROM images with Bacula Volumes.

`bacula-rescue-5.2.0.tar.gz` This is the Bacula Rescue USB key code. Note, the version number of this package is not always tied to the Bacula release version, so it may be different. Using this code, you can create a USB key with your system configuration and containing a statically linked version of the File daemon. This can permit you to easily repartition and reformat your hard disks and reload your system with Bacula in the case of a hard disk failure.

`win32bacula-5.2.0.exe` This file is the 32 bit Windows installer for installing the Windows client (File daemon) on a Windows machine. This client will also run on 64 bit Windows machines, but VSS support is not available if you are running a 64 bit version of Windows. This installer installs only the FD, the Director and Storage daemon are not included.

`win64bacula-5.2.0.exe` This file is the 64 bit Windows installer for installing the Windows client (File daemon) on a Windows machine. This client will only run on 64 bit Windows OS machines. It will not run on 32 bit machines or 32 bit Windows OSes. The win64bacula

release is necessary for Volume Shadow Copy (VSS) to work on Win64 OSes. This installer installs only the FD, the Director and Storage daemon are not included.

## 1.2 Building Bacula from Source

The basic installation is rather simple.

1. Install and build any **depkgs** as noted above. This should be unnecessary on most modern Operating Systems.
2. Configure and install MySQL or PostgreSQL (if desired). **Installing and Configuring MySQL Phase I** chapter (chapter 50 page 555) or **Installing and Configuring PostgreSQL Phase I** chapter (chapter 51 page 561) of the Bacula Community Edition Main manual.

   If you are installing from rpms, and are using MySQL, please be sure to install **mysql-devel**, so that the MySQL header files are available while compiling Bacula. In addition, the MySQL client library **mysqlclient** requires the gzip compression library **libz.a** or **libz.so**. If you are using rpm packages, these libraries are in the **libz-devel** package. On Debian systems, you will need to load the **zlib1g-dev** package. If you are not using rpms or debs, you will need to find the appropriate package for your system.

   Note, if you already have a running MySQL or PostgreSQL on your system, you can skip this phase provided that you have built the thread safe libraries. And you have already installed the additional rpms noted above.

   SQLite is not supported on Solaris. This is because it frequently fails with bus errors. However SQLite 3 may work.
3. Detar the Bacula source code preferably into the **bacula** directory discussed above.
4. cd to the directory containing the source code.
5. ./configure (with appropriate options as described below). Any path names you specify as options on the ./configure command line must be absolute paths and not relative.
6. Check the output of ./configure very carefully, especially the Install binaries and Install config directories. If they are not correct, please rerun ./configure until they are. The output from ./configure is stored in config.out and can be re-displayed at any time without rerunning the ./configure by doing cat config.out.
7. If after running ./configure once, you decide to change options and re-run it, that is perfectly fine, but before re-running it, you should run:

   ```
   make distclean
   ```

   so that you are sure to start from scratch and not have a mixture of the two options. This is because ./configure caches much of the information. The make distclean is also critical if you move the source directory from one machine to another. If the make distclean fails, just ignore it and continue on.
8. make If you get errors while linking in the Storage daemon directory (src/stored), it is probably because you have not loaded the static libraries on your system. I noticed this problem on a Solaris system. To correct it, make sure that you have not added **--enable-static-tools** to the ./configure command.

   If you skip this step (make) and proceed immediately to the make install you are making two serious errors:

   1. your install will fail because Bacula requires a make before a make install.
   2. you are depriving yourself of the chance to make sure there are no errors before beginning to write files to your system directories.

9. `make install` Please be sure you have done a `make` before entering this command, and that everything has properly compiled and linked without errors.

10. If you are new to Bacula, we **strongly** recommend that you skip the next step and use the default configuration files, then run the example program in the next chapter, then come back and modify your configuration files to suit your particular needs.

11. Customize the configuration files for each of the three daemons (Directory, File, Storage) and for the Console program. For the details of how to do this, please see **Setting Up Bacula Configuration Files** chapter (chapter 21 page 207) of the Bacula Community Edition Main manual. We recommend that you start by modifying the default configuration files supplied, making the minimum changes necessary. Complete customization can be done after you have Bacula up and running. Please take care when modifying passwords, which were randomly generated, and the **Name**s as the passwords and names must agree between the configuration files for security reasons.

12. Create the Bacula `MySQL` database and tables (if using `MySQL`) **Installing and Configuring `MySQL` Phase II** chapter (chapter 50.1 page 556) or create the Bacula `PostgreSQL` database and tables **Configuring `PostgreSQL` II** chapter (chapter 51.1 page 562) or alternatively if you are using `SQLite` Installing and Configuring `SQLite` Phase II.

13. Start Bacula (`./bacula start`) Note. the next chapter shows you how to do this in detail.

14. Interface with Bacula using the Console program

15. For the previous two items, please follow the instructions in the Running Bacula chapter of this manual, where you will run a simple backup and do a restore. Do this before you make heavy modifications to the configuration files so that you are sure that Bacula works and are familiar with it. After that changing the conf files will be easier.

16. If after installing Bacula, you decide to "move it", that is to install it in a different set of directories, proceed as follows:

```
make uninstall
make distclean
./configure <your-new-options>
make
make install
```

If all goes well, the `./configure` will correctly determine which operating system you are running and configure the source code appropriately. Currently, FreeBSD, Linux (Red Hat), and Solaris are supported. The Bacula client (File daemon) is reported to work with MacOS X 10.3 is if readline support is not enabled (default) when building the client.

If you install Bacula on more than one system, and they are identical, you can simply transfer the source tree to that other system and do a `make install`. However, if there are differences in the libraries or OS versions, or you wish to install on a different OS, you should start from the original compress tar file. If you do transfer the source tree, and you have previously done a `./configure` command, you **must** do:

```
make distclean
```

prior to doing your new `./configure`. This is because the GNU `autoconf` tools cache the configuration, and if you re-use a configuration for a Linux machine on a Solaris, you can be sure your build will fail. To avoid this, as mentioned above, either start from the tar file, or do a `make distclean`.

In general, you will probably want to supply a more complicated `configure` statement to ensure that the modules you want are built and that everything is placed into the correct directories.

For example, on Fedora, Red Hat, or SuSE one could use the following:

```
CFLAGS="-g -Wall" \
     ./configure \
    --sbindir=/opt/bacula/bin \
    --sysconfdir=/opt/bacula/etc \
    --with-pid-dir=/var/run \
    --with-subsys-dir=/var/run \
    --with-mysql \
    --with-working-dir=/opt/bacula/working \
    --with-dump-email=$USER
```

The advantage of using the above configuration to start is that everything will be put into a single directory, which you can later delete once you have run the examples in the next chapter and learned how Bacula works. In addition, the above can be installed and run as non-root.

For the developer's convenience, I have added a `defaultconfig` script to the `examples` directory. This script contains the statements that you would normally use, and each developer/user may modify them to suit his needs. You should find additional useful examples in this directory as well.

The **--enable-conio** or **--enable-readline** options are useful because they provide a command line history, editing capability for the Console program and tab completion on various option. If you have included either option in the build, either the **termcap** or the **ncurses** package will be needed to link. On most systems, including Red Hat and SuSE, you should include the `ncurses` package. If Bacula's configure process finds the `ncurses` libraries, it will use those rather than the `termcap` library. On some systems, such as SuSE, the `termcap` library is not in the standard library directory. As a consequence, the option may be disabled or you may get an error message such as:

```
/usr/lib/gcc-lib/i586-suse-linux/3.3.1/.../ld:
cannot find -ltermcap
collect2: ld returned 1 exit status
```

while building the Bacula Console. In that case, you will need to set the **LDFLAGS** environment variable prior to building.

```
export LDFLAGS="-L/usr/lib/termcap"
```

The same library requirements apply if you wish to use the `readline` subroutines for command line editing, history and tab completion or if you are using a `MySQL` library that requires encryption. If you need encryption, you can either export the appropriate additional library options as shown above or, alternatively, you can include them directly on the `./configure` line as in:

```
LDFLAGS="-lssl -lcyrpto" \
        ./configure <your-options>
```

On some systems such as Mandriva, `readline` tends to gobble up prompts, which makes it totally useless. If this happens to you, use the disable option, or if you are using version 1.33 and above try using **--enable-conio** to use a built-in `readline` replacement. You will still need either the `termcap` or the `ncurses` library, but it is unlikely that the **conio** package will gobble up prompts.

`readline` is no longer supported after version 1.34. The code within Bacula remains, so it should be usable, and if users submit patches for it, we will be happy to apply them. However, due to the fact that each version of `readline` seems to be incompatible with previous versions, and that there are significant differences between systems, we can no longer afford to support it.

## 1.3 Quick Start

There are a number of options and important considerations given below that you can skip for the moment if you have not had any problems building Bacula with a simplified configuration as shown above.

If the `./configure` process is unable to find specific libraries (e.g. `libintl`, you should ensure that the appropriate package is installed on your system. Alternatively, if the package is installed in a non-standard location (as far as Bacula is concerned), then there is generally an option listed below (or listed with `./configure --help` that will permit you to specify the directory that should be searched. In other cases, there are options that will permit you to disable to feature (e.g. `--disable-nls`).

If you want to dive right into it, we recommend you skip to the next chapter, and run the example program. It will teach you a lot about Bacula and as an example can be installed into a single directory (for easy removal) and run as non-root. If you have any problems or when you want to do a real installation, come back to this chapter and read the details presented below.

## 1.4 Configure Options

The following command line options are available for `configure` to customize your installation.

--prefix=<patch> This option is meant to allow you to direct where the architecture independent files should be placed. However, we find this a somewhat vague concept, and so we have not implemented this option other than to use any explicit prefix that you may define. If you do not explicitly specify a prefix, Bacula's configure routine will not use the default value that `./configure --help` prints. As a consequence, we suggest that you avoid it. We have provided options that allow you to explicitly specify the directories for each of the major categories of installation files.

--sbindir=<binary-path> Defines where the Bacula binary (executable) files will be placed during a `make install` command.

--sysconfdir=<config-path> Defines where the Bacula configuration files should be placed during a `make install` command. Note, for security reasons, this directory should be unique to Bacula and not read/writable by any other user/group than `bacula` is running under.

--mandir=<path> Note, as of Bacula version 1.39.14, the meaning of any path specified on this option is change from prior versions. It now specifies the top level man directory. Previously the mandir specified the full path to where you wanted the man files installed. The man files will be installed in gzip'ed format under `mandir/man1` and `mandir/man8` as appropriate. For the install to succeed you must have `gzip` installed on your system.

By default, Bacula will install the Unix man pages in `/usr/share/man/man1` and `/usr/share/man/man8`. If you wish the man page to be installed in a different location, use this option to specify the path. Note, the main HTML and PDF Bacula documents are in a separate tar file that is not part of the source distribution.

--datadir=<path> If you translate Bacula or parts of Bacula into a different language you may specify the location of the po files using the **`--datadir`** option. You must manually install any po files as Bacula does not (yet) automatically do so.

--disable-ipv6

--enable-smartalloc This enables the inclusion of the Smartalloc orphaned buffer detection code. This option is highly recommended. Because we never build without this option, you may experience problems if it is not enabled. In this case, simply re-enable the option. We strongly recommend keeping this option enabled as it helps detect memory leaks. This configuration parameter is used while building Bacula

--enable-bat If you have Qt4 >= 4.3.4 installed on your computer including the libqt4 and libqt4-devel (libqt4-dev on Debian) libraries, and you want to use the BAT (Bacula Administration Tool) GUI Console interface to Bacula, you must specify this option. Doing so will build everything in the src/qt-console directory. The build with enable-bat will work only with a full Bacula build (i.e. it will not work with a client-only build).

Qt4 is available on OpenSUSE 10.2, CentOS 5, Fedora, and Debian. If it is not available on your system, you can download the depkgs-qt package from the Bacula Source Forge download area and build it. See the INSTALL file in that package for more details. In particular to use the Qt4 built by **depkgs-qt** you **must** source the file qt4-paths.

--enable-batch-insert This option enables batch inserts of the attribute records (default) in the catalog database, which is much faster (10 times or more) than without this option for large numbers of files. However, this option will automatically be disabled if your SQL libraries are not thread safe. If you find that batch mode is not enabled on your Bacula installation, then your database most likely does not support threads.

SQLite 2 is not thread safe. Batch insert cannot be enabled when using SQLite 2

On most systems, MySQL, PostgreSQL and SQLite 3 are thread safe.

To verify that your PostgreSQL is thread safe, you can try this (change the path to point to your particular installed libpq.a; these commands were issued on FreeBSD 6.2):

```
$ nm /usr/local/lib/libpq.a | grep PQputCopyData
00001b08 T PQputCopyData
$ nm /usr/local/lib/libpq.a | grep mutex
         U pthread_mutex_lock
         U pthread_mutex_unlock
         U pthread_mutex_init
         U pthread_mutex_lock
         U pthread_mutex_unlock
```

The above example shows a libpq that contains the required function PQputCopyData and is thread enabled (i.e. the pthread_mutex* entries). If you do not see PQputCopyData, your version of PostgreSQL is too old to allow batch insert. If you do not see the mutex entries, then thread support has not been enabled. Our tests indicate you usually need to change the configuration options and recompile/reinstall the PostgreSQL client software to get thread support.

Bacula always links to the thread safe MySQL libraries.

Running with Batch Insert turned on is recommended because it can significantly improve attribute insertion times. However, it does put a significantly larger part of the work on your SQL engine, so you may need to pay more attention to tuning it. In particular, Batch Insert can require large temporary table space, and consequently, the default location (often /tmp) may run out of space causing errors. For MySQL, the location is set in my.conf with "tmpdir". You may also want to increase the memory available to your SQL engine to further improve performance during Batch Inserts.

--enable-bwx-console If you have wxWidgets installed on your computer and you want to use the wxWidgets GUI Console interface to Bacula, you must specify this option. Doing so will build everything in the src/wx-console directory. This could also be useful to users who want a GUI Console and don't want to install QT, as wxWidgets can work with GTK+, Motif or even X11 libraries.

–enable-tray-monitor If you have GTK installed on your computer, you run a graphical environment or a window manager compatible with the FreeDesktop system tray standard (like KDE and GNOME) and you want to use a GUI to monitor Bacula daemons, you must specify this option. Doing so will build everything in the src/tray-monitor directory. Note, due to restrictions on what can be linked with GPLed code, we were forced to remove the egg code that dealt with the tray icons and replace it by calls to the GTK+ API, and unfortunately, the tray icon API necessary was not implemented until GTK version 2.10 or later.

–enable-static-tools This option causes the linker to link the Storage daemon utility tools (bls, bextract, and bscan) statically. This permits using them without having the

shared libraries loaded. If you have problems linking in the **src/stored** directory, make sure you have not enabled this option, or explicitly disable static linking by adding `--disable-static-tools`.

–enable-static-fd This option causes the make process to build a **static-bacula-fd** in addition to the standard File daemon. This static version will include statically linked libraries and is required for the Bare Metal recovery. This option is largely superseded by using `make static-bacula-fd` from with in the `src/filed` directory. Also, the `--enable-client-only` option described below is useful for just building a client so that all the other parts of the program are not compiled.

When linking a static binary, the linker needs the static versions of all the libraries that are used, so frequently users will experience linking errors when this option is used. The first thing to do is to make sure you have the static glibc library installed on your system. The second thing to do is the make sure you do not specify `--openssl` or other options on your `./configure` statement as these options require additional libraries. You may be able to enable those options, but you will need to load additional static libraries.

--enable-static-sd This option causes the make process to build a **static-bacula-sd** in addition to the standard Storage daemon. This static version will include statically linked libraries and could be useful during a Bare Metal recovery.

When linking a static binary, the linker needs the static versions of all the libraries that are used, so frequently users will experience linking errors when this option is used. The first thing to do is to make sure you have the static glibc library installed on your system. The second thing to do is the make sure you do not specify `--openssl` or other options on your `./configure` statement as these options require additional libraries. You may be able to enable those options, but you will need to load additional static libraries.

--enable-static-dir This option causes the make process to build a **static-bacula-dir** in addition to the standard Director. This static version will include statically linked libraries and could be useful during a Bare Metal recovery.

When linking a static binary, the linker needs the static versions of all the libraries that are used, so frequently users will experience linking errors when this option is used. The first thing to do is to make sure you have the static glibc library installed on your system. The second thing to do is the make sure you do not specify `--openssl` or other options on your `./configure` statement as these options require additional libraries. You may be able to enable those options, but you will need to load additional static libraries.

--enable-static-cons This option causes the make process to build a **static-console** in addition to the standard console. This static version will include statically linked libraries and could be useful during a Bare Metal recovery.

When linking a static binary, the linker needs the static versions of all the libraries that are used, so frequently users will experience linking errors when this option is used. The first thing to do is to make sure you have the static glibc library installed on your system. The second thing to do is the make sure you do not specify `--openssl` or other options on your `./configure` statement as these options require additional libraries. You may be able to enable those options, but you will need to load additional static libraries.

--enable-client-only This option causes the make process to build only the File daemon and the libraries that it needs. None of the other daemons, storage tools, nor the console will be built. Likewise a `make install` will then only install the File daemon. To cause all daemons to be built, you will need to do a configuration without this option. This option greatly facilitates building a Client on a client only machine.

When linking a static binary, the linker needs the static versions of all the libraries that are used, so frequently users will experience linking errors when this option is used. The first thing to do is to make sure you have the static glibc library installed on your system. The second thing to do is the make sure you do not specify `--openssl` or other options on your `./configure` statement as these options require additional libraries. You may be able to enable those options, but you will need to load additional static libraries.

–enable-build-dird This option causes the make process to build the Director and the Director's tools. By default, this option is on, but you may turn it off by using **--disable-build-dird** to prevent the Director from being built.

--enable-build-stored This option causes the make process to build the Storage daemon. By default, this option is on, but you may turn it off by using **--disable-build-stored** to prevent the Storage daemon from being built.

--enable-largefile This option (default) causes Bacula to be built with 64 bit file address support if it is available on your system. This permits Bacula to read and write files greater than 2 GBytes in size. You may disable this feature and revert to 32 bit file addresses by using **--disable-largefile**.

--disable-nls By default, Bacula uses the GNU Native Language Support (NLS) libraries. On some machines, these libraries may not be present or may not function correctly (especially on non-Linux implementations). In such cases, you may specify **--disable-nls** to disable use of those libraries. In such a case, Bacula will revert to using English.

--disable-ipv6 By default, Bacula enables IPv6 protocol. On some systems, the files for IPv6 may exist, but the functionality could be turned off in the kernel. In that case, in order to correctly build Bacula, you will explicitly need to use this option so that Bacula does not attempt to reference OS function calls that do not exist.

--with-sqlite3=<sqlite3-path> This enables use of the SQLite version 3.x database. The **sqlite3-path** is not normally specified as Bacula looks for the necessary components in a standard location (**depkgs/sqlite3**). See Installing and Configuring SQLite chapter of this manual for more details. SQLite 3 is not supported on Solaris.

--with-mysql=<mysql-path> This enables building of the Catalog services for Bacula. It assumes that MySQL is running on your system, and expects it to be installed in the **mysql-path** that you specify. Normally, if MySQL is installed in a standard system location, you can simply use **--with-mysql** with no path specification. If you do use this option, please proceed to installing MySQL in the **Installing and Configuring MySQL** chapter (chapter 50 page 555) before proceeding with the configuration.

See the note below under the --with-postgresql item.

--with-postgresql=<path> This provides an explicit path to the PostgreSQL libraries if Bacula cannot find it by default. Normally to build with PostgreSQL, you would simply use **--with-postgresql**.

Note, for Bacula to be configured properly, you must specify one of the four database options supported. That is: --with-sqlite, --with-sqlite3, --with-mysql, or --with-postgresql, otherwise the ./configure will fail.

--with-openssl=<path> This configuration option is necessary if you want to enable TLS (ssl), which encrypts the communications within Bacula or if you want to use File Daemon PKI data encryption. Normally, the **path** specification is not necessary since the configuration searches for the OpenSSL libraries in standard system locations. However, you must ensure that all the libraries are loaded including libssl-dev or the equivalent on your system. Enabling OpenSSL in Bacula permits secure communications between the daemons and/or data encryption in the File daemon. For more information on using TLS, please see the Bacula TLS – Communications Encryption chapter of this manual. For more information on using PKI data encryption, please see the Bacula PKI – Data Encryption chapter of this manual.

If you get errors linking, you need to load the development libraries, or you need to disable SSL by setting without-openssl.

--with-libintl-prefix=<DIR> This option may be used to tell Bacula to search DIR/include and DIR/lib for the libintl headers and libraries needed for NLS.

--enable-conio Tells Bacula to enable building the small, light weight readline replacement routine. It is generally much easier to configure than readline, although, like readline, it needs either the termcap or ncurses library.

--with-readline=<readline-path> Tells Bacula where **readline** is installed. Normally, Bacula will find readline if it is in a standard library. If it is not found and no --with-readline is specified, readline will be disabled. This option affects the Bacula build. readline provides the Console program with a command line history and editing capability and is no longer supported, so you are on your own if you have problems.

--enable-readline Tells Bacula to enable readline support. It is normally disabled due to the large number of configuration problems and the fact that the package seems to change in incompatible ways from version to version.

--with-tcp-wrappers=<path> This specifies that you want TCP wrappers (man hosts_access(5)) compiled in. The path is optional since Bacula will normally find the libraries in the standard locations. This option affects the Bacula build. In specifying your restrictions in the /etc/hosts.allow or /etc/hosts.deny files, do not use the **twist** option (hosts_options(5)) or the Bacula process will be terminated. Note, when setting up your /etc/hosts.allow or /etc/hosts.deny, you must identify the Bacula daemon in question with the name you give it in your conf file rather than the name of the executable.

For more information on configuring and testing TCP wrappers, please see the Configuring and Testing TCP Wrappers section in the Security Chapter.

On SuSE, the libwrappers libraries needed to link Bacula are contained in the tcpd-devel package. On Red Hat, the package is named tcp_wrappers.

--with-archivedir=<path> The directory used for disk-based backups. Default value is /tmp. This parameter sets the default values in the bacula-dir.conf and bacula-sd.conf configuration files. For example, it sets the Where directive for the default restore job and the Archive Device directive for the FileStorage device.

This option is designed primarily for use in regression testing. Most users can safely ignore this option.

--with-working-dir=<working-directory-path> This option is mandatory and specifies a directory into which Bacula may safely place files that will remain between Bacula executions. For example, if the internal database is used, Bacula will keep those files in this directory. This option is only used to modify the daemon configuration files. You may also accomplish the same thing by directly editing them later. The working directory is not automatically created by the install process, so you must ensure that it exists before using Bacula for the first time.

--with-baseport=<port=number> In order to run, Bacula needs three TCP/IP ports (one for the Bacula Console, one for the Storage daemon, and one for the File daemon). The **--with-baseport** option will automatically assign three ports beginning at the base port address specified. You may also change the port number in the resulting configuration files. However, you need to take care that the numbers correspond correctly in each of the three daemon configuration files. The default base port is 9101, which assigns ports 9101 through 9103. These ports (9101, 9102, and 9103) have been officially assigned to Bacula by IANA. This option is only used to modify the daemon configuration files. You may also accomplish the same thing by directly editing them later.

--with-dump-email=<email-address> This option specifies the email address where any core dumps should be set. This option is normally only used by developers.

--with-pid-dir=<PATH> This specifies where Bacula should place the process id file during execution. The default is: /var/run. This directory is not created by the install process, so you must ensure that it exists before using Bacula the first time.

--with-subsys-dir=<PATH> This specifies where Bacula should place the subsystem lock file during execution. The default is /var/run/subsys. Please make sure that you do not specify the same directory for this directory and for the sbindir directory. This directory is used only within the autostart scripts. The subsys directory is not created by the Bacula install, so you must be sure to create it before using Bacula.

--with-dir-password=<Password> This option allows you to specify the password used to access the Director (normally from the Console program). If it is not specified, configure will automatically create a random password.

--with-fd-password=<Password> This option allows you to specify the password used to access the File daemon (normally called from the Director). If it is not specified, configure will automatically create a random password.

--with-sd-password=<Password> This option allows you to specify the password used to access the Storage daemon (normally called from the Director). If it is not specified, configure will automatically create a random password.

--with-dir-user=<User> This option allows you to specify the Userid used to run the Director. The Director must be started as root, but doesn't need to run as root, and after doing preliminary initializations, it can "drop" to the UserId specified on this option. If you specify this option, you must create the User prior to running `make install`, because the working directory owner will be set to **User**.

--with-dir-group=<Group> This option allows you to specify the GroupId used to run the Director. The Director must be started as root, but doesn't need to run as root, and after doing preliminary initializations, it can "drop" to the GroupId specified on this option. If you specify this option, you must create the Group prior to running `make install`, because the working directory group will be set to **Group**.

--with-sd-user=<User> This option allows you to specify the Userid used to run the Storage daemon. The Storage daemon must be started as root, but doesn't need to run as root, and after doing preliminary initializations, it can "drop" to the UserId specified on this option. If you use this option, you will need to take care that the Storage daemon has access to all the devices (tape drives, ...) that it needs.

--with-sd-group=<Group> This option allows you to specify the GroupId used to run the Storage daemon. The Storage daemon must be started as root, but doesn't need to run as root, and after doing preliminary initializations, it can "drop" to the GroupId specified on this option.

--with-fd-user=<User> This option allows you to specify the Userid used to run the File daemon. The File daemon must be started as root, and in most cases, it needs to run as root, so this option is used only in very special cases, after doing preliminary initializations, it can "drop" to the UserId specified on this option.

--with-fd-group=<Group> This option allows you to specify the GroupId used to run the File daemon. The File daemon must be started as root, and in most cases, it must be run as root, however, after doing preliminary initializations, it can "drop" to the GroupId specified on this option.

--with-mon-dir-password=<Password> This option allows you to specify the password used to access the Directory from the monitor. If it is not specified, configure will automatically create a random password.

--with-mon-fd-password=<Password> This option allows you to specify the password used to access the File daemon from the Monitor. If it is not specified, configure will automatically create a random password.

--with-mon-sd-password=<Password> This option allows you to specify the password used to access the Storage daemon from the Monitor. If it is not specified, configure will automatically create a random password.

--with-db-name=<database-name> This option allows you to specify the database name to be used in the conf files. The default is bacula.

--with-db-user=<database-user> This option allows you to specify the database user name to be used in the conf files. The default is bacula.

Note, many other options are presented when you do a `./configure --help`, but they are not implemented.

## 1.5 Recommended Options for Most Systems

For most systems, we recommend starting with the following options:

```
./configure \
  --enable-smartalloc \
  --sbindir=/opt/bacula/bin \
  --sysconfdir=/opt/bacula/etc \
  --with-pid-dir=/opt/bacula/working \
  --with-subsys-dir=/opt/bacula/working \
  --with-working-dir=/opt/bacula/working
```

If you want to install Bacula in an installation directory rather than run it out of the build directory (as developers will do most of the time), you should also include the `--sbindir` and `--sysconfdir` options with appropriate paths. Neither are necessary if you do not use "make install" as is the case for most development work. The install process will create the sbindir and sysconfdir if they do not exist, but it will not automatically create the pid-dir, subsys-dir, or working-dir, so you must ensure that they exist before running Bacula for the first time.

## 1.6 Red Hat

Using `SQLite` :

```
CFLAGS="-g -Wall" \btool{./configure} \
  --sbindir=/opt/bacula/bin \
  --sysconfdir=/opt/bacula/etc \
  --enable-smartalloc \
  --with-sqlite=$HOME/bacula/depkgs/sqlite \
  --with-working-dir=/opt/bacula/working \
  --with-pid-dir=/opt/bacula/working \
  --with-subsys-dir=/opt/bacula/working \
  --enable-bat \
  --enable-\texttt{readline}
```

or

```
CFLAGS="-g -Wall" \btool{./configure} \
  --sbindir=/opt/bacula/bin \
  --sysconfdir=/opt/bacula/etc \
  --enable-smartalloc \
  --with-mysql \
  --with-working-dir=/opt/bacula/working
  --with-pid-dir=/opt/bacula/working \
  --with-subsys-dir=/opt/bacula/working
  --enable-\texttt{readline}
```

or finally, a completely traditional Red Hat Linux install, which we do not recommend, because it make it harder to backup Bacula for disaster recovery purposes:

```
CFLAGS="-g -Wall" \btool{./configure} \
  --sbindir=/usr/sbin \
  --sysconfdir=/etc/bacula \
  --with-scriptdir=/etc/bacula \
  --enable-smartalloc \
  --enable-bat \
  --with-mysql \
  --with-working-dir=/var/bacula \
  --with-pid-dir=/var/run \
  --enable-\texttt{readline}
```

Note, Bacula assumes that /var/bacula, /var/run, and /var/lock/subsys exist so it will not automatically create them during the install process.

## 1.7 Solaris

To build Bacula from source, you will need the following installed on your system (they are not by default): libiconv, gcc 3.3.2, stdc++, libgcc (for stdc++ and gcc_s libraries), make 3.8 or later.

You will probably also need to: Add /usr/local/bin to PATH and Add /usr/ccs/bin to PATH for ar.

It is possible to build Bacula on Solaris with the Solaris compiler, but we recommend using GNU C++ if possible.

A typical configuration command might look like:

```
#!/bin/sh
CFLAGS="-g" \btool{./configure} \
  --sbindir=/opt/bacula/bin \
  --sysconfdir=/opt/bacula/etc \
  --with-mysql \
  --enable-smartalloc \
  --with-pid-dir=/opt/bacula/working \
  --with-subsys-dir=/opt/bacula/working \
  --with-working-dir=/opt/bacula/working
```

Note, you may need to install the following packages to build Bacula from source:

```
SUNWbinutils,
SUNWarc,
SUNWhea,
SUNWGcc,
SUNWGnutls
SUNWGnutls-devel
SUNWGmake
SUNWgccruntime
SUNWlibgcrypt
SUNWzlib
SUNWzlibs
SUNW\texttt{readline}
SUNWbinutilsS
SUNWGmakeS
SUNWlibm

export
PATH=/usr/bin::/usr/ccs/bin:/etc:/usr/openwin/bin:/usr/local/bin:/usr/sfw/bin:/opt/sfw/bin:/usr/ucb:/usr/sbin
```

If you have installed special software not normally in the Solaris libraries, such as OpenSSL, or the packages shown above, then you may need to add `/usr/sfw/lib` to the library search path. Probably the simplest way to do so is to run:

```
setenv LDFLAGS "-L/usr/sfw/lib -R/usr/sfw/lib"
```

Prior to running the `./configure` command.

Alternatively, you can set the LD_LIBARY_PATH and/or the LD_RUN_PATH environment variables appropriately.

It is also possible to use the `crle` program to set the library search path. However, this should be used with caution.

## 1.8 FreeBSD

Please see: The FreeBSD Diary for a detailed description on how to make Bacula work on your system. In addition, users of FreeBSD prior to 4.9-STABLE dated Mon Dec 29 15:18:01 2003 UTC who plan to use tape devices, please see the **Tape Testing** section (section 3.3.6 page 36) of Bacula Community Edition Problems Resolution Guide for **important** information on how to configure your tape drive for compatibility with Bacula.

If you are using Bacula with MySQL, you should take care to compile MySQL with FreeBSD native threads rather than LinuxThreads, since Bacula is normally built with FreeBSD native threads rather than LinuxTreads. Mixing the two will probably not work.

## 1.9 Win32

To install the binary Win32 version of the File daemon please see the Win32 Installation Chapter in this document.

## 1.10 One File Configure Script

The following script could be used if you want to put everything in a single directory (except for the working directory):

```
#!/bin/sh
CFLAGS="-g -Wall" \
  \btool{./configure} \
    --sbindir=$HOME/bacula/bin \
    --sysconfdir=$HOME/bacula/bin \
    --mandir=$HOME/bacula/bin \
    --enable-smartalloc \
    --enable-bat \
    --with-pid-dir=$HOME/bacula/bin/working \
    --with-subsys-dir=$HOME/bacula/bin/working \
    --with-mysql \
    --with-working-dir=$HOME/bacula/bin/working \
    --with-dump-email=$USER@your-site.com \
    --with-job-email=$USER@your-site.com \
    --with-smtp-host=mail.your-site.com
exit 0
```

You may also want to put the following entries in your /etc/services file as it will make viewing the connections made by Bacula easier to recognize (i.e. netstat -a):

```
bacula-dir      9101/tcp
bacula-fd       9102/tcp
bacula-sd       9103/tcp
```

## 1.11 Installing Bacula

Before setting up your configuration files, you will want to install Bacula in its final location. Simply enter:

```
make install
```

If you have previously installed Bacula, the old binaries will be overwritten, but the old configuration files will remain unchanged, and the "new" configuration files will be appended with a **.new**. Generally if you have previously installed and run Bacula you will want to discard or ignore the configuration files with the appended **.new**.

## 1.12 Building a File Daemon or Client

If you run the Director and the Storage daemon on one machine and you wish to back up another machine, you must have a copy of the File daemon for that machine. If the machine and the Operating System are identical, you can simply copy the Bacula File daemon binary file `bacula-fd` as well as its configuration file `bacula-fd.conf` then modify the name and password in the conf file to be unique. Be sure to make corresponding additions to the Director's configuration file (`bacula-dir.conf`).

If the architecture or the OS level are different, you will need to build a File daemon on the Client machine. To do so, you can use the same `./configure` command as you did for your main program, starting either from a fresh copy of the source tree, or using `make distclean` before the `./configure`.

Since the File daemon does not access the Catalog database, you can remove the **--with-mysql** or **--with-sqlite** options, then add **--enable-client-only**. This will compile only the necessary libraries and the client programs and thus avoids the necessity of installing one or another of those database programs to build the File daemon. With the above option, you simply enter `make` and just the client will be built.

## 1.13 Auto Starting the Daemons

If you wish the daemons to be automatically started and stopped when your system is booted (a good idea), one more step is necessary. First, the ./configure process must recognize your system – that is it must be a supported platform and not **unknown**, then you must install the platform dependent files by doing:

```
(become root)
make install-autostart
```

Please note, that the auto-start feature is implemented only on systems that we officially support (currently, FreeBSD, Red Hat/Fedora Linux, and Solaris), and has only been fully tested on Fedora Linux.

The `make install-autostart` will cause the appropriate startup scripts to be installed with the necessary symbolic links. On Red Hat/Fedora Linux systems, these scripts reside in `/etc/rc.d/init.d/bacula-dir`, `/etc/rc.d/init.d/bacula-fd`, and `/etc/rc.d/init.d/bacula-sd`. However the exact location depends on what operating system you are using.

If you only wish to install the File daemon, you may do so with:

```
make install-autostart-fd
```

## 1.14 Other Make Notes

To simply build a new executable in any directory, enter:

```
make
```

To clean out all the objects and binaries (including the files named 1, 2, or 3, which are development temporary files), enter:

```
make clean
```

To really clean out everything for distribution, enter:

```
make distclean
```

note, this cleans out the Makefiles and is normally done from the top level directory to prepare for distribution of the source. To recover from this state, you must redo the `./configure` in the top level directory, since all the Makefiles will be deleted.

To add a new file in a subdirectory, edit the Makefile.in in that directory, then simply do a `make`. In most cases, the make will rebuild the Makefile from the new Makefile.in. In some case, you may need to issue the `make` a second time. In extreme cases, cd to the top level directory and enter: `make Makefiles`.

To add dependencies:

```
make depend
```

The `make depend` appends the header file dependencies for each of the object files to `Makefile` and `Makefile.in`. This command should be done in each directory where you change the dependencies. Normally, it only needs to be run when you add or delete source or header files. `make depend` is normally automatically invoked during the configuration process.

To install:

```
make install
```

This not normally done if you are developing Bacula, but is used if you are going to run it to back up your system.

After doing a `make install` the following files will be installed on your system (more or less). The exact files and location (directory) for each file depends on your `./configure` command (e.g. if you are using `SQLite` instead of `MySQL`, some of the files will be different).

NOTE: it is quite probable that this list is out of date. But it is a starting point.

```
bacula
bacula-dir
bacula-dir.conf
bacula-fd
bacula-fd.conf
bacula-sd
bacula-sd.conf
bacula-tray-monitor
tray-monitor.conf
bextract
bls
bscan
btape
btraceback
btraceback.gdb
bconsole
bconsole.conf
```

```
create_mysql_database
dbcheck
delete_catalog_backup
drop_bacula_tables
drop_mysql_tables
make_bacula_tables
make_catalog_backup
make_mysql_tables
mtx-changer
query.sql
bsmtp
startmysql
stopmysql
bwx-console
bwx-console.conf
9 man pages
```

## 1.15 Installing Tray Monitor

The Tray Monitor is already installed if you used the **--enable-tray-monitor** configure option and ran `make install`.

As you don't run your graphical environment as `root` (if you do, you should change that bad habit), don't forget to allow your user to read `tray-monitor.conf`, and to execute `bacula-tray-monitor` (this is not a security issue).

Then log into your graphical environment (KDE, GNOME or something else), run `bacula-tray-monitor` as your user, and see if a cassette icon appears somewhere on the screen, usually on the task bar. If it doesn't, follow the instructions below related to your environment or window manager.

### 1.15.1 GNOME

System tray, or notification area if you use the GNOME terminology, has been supported in GNOME since version 2.2. To activate it, right-click on one of your panels, open the menu **Add to this Panel**, then **Utility** and finally click on **Notification Area**.

### 1.15.2 KDE

System tray has been supported in KDE since version 3.1. To activate it, right-click on one of your panels, open the menu **Add**, then **Applet** and finally click on **System Tray**.

### 1.15.3 Other Window Managers

Read the documentation to know if the Freedesktop system tray standard is supported by your window manager, and if applicable, how to activate it.

## 1.16 Modifying the Bacula Configuration Files

See the chapter Configuring Bacula in this manual for instructions on how to set Bacula configuration files.

# Chapter 2

# Variable Expansion

Variable expansion is somewhat similar to Unix shell variable expansion. Currently (version 1.31), it is used only in format labels, but in the future, it will most likely be used in more places.

## 2.1 General Functionality

This is basically a string expansion capability that permits referencing variables, indexing arrays, conditional replacement of variables, case conversion, substring selection, regular expression matching and replacement, character class replacement, padding strings, repeated expansion in a user controlled loop, support of arithmetic expressions in the loop start, step and end conditions, and recursive expansion.

When using variable expansion characters in a Volume Label Format record, the format should always be enclosed in double quotes (**"**).

For example, **${HOME}** will be replaced by your home directory as defined in the environment. If you have defined the variable **xxx** to be **Test**, then the reference **${xxx:p/7/Y/r}** will right pad the contents of **xxx** to a length of seven characters filling with the character **Y** giving **YYYTest**.

## 2.2 Bacula Variables

Within Bacula, there are three main classes of variables with some minor variations within the classes. The classes are:

Counters Counters are defined by the **Counter** resources in the Director's conf file. The counter can either be a temporary counter that lasts for the duration of Bacula's execution, or it can be a variable that is stored in the catalog, and thus retains its value from one Bacula execution to another. Counter variables may be incremented by postfixing a plus sign (+ after the variable name).

Internal Variables Internal variables are read-only, and may be related to the current job (i.e. Job name), or maybe special variables such as the date and time. The following variables are available:

Year – the full year

Month – the current month 1-12

Day – the day of the month 1-31

Hour – the hour 0-24

Minute – the current minute 0-59

Second – the current second 0-59

WeekDay – the current day of the week 0-6 with 0 being Sunday

Job – the job name

general – the Director's name

Level – the Job Level

Type – the Job type

JobId – the JobId

JobName – the unique job name composed of Job and date

Storage – the Storage daemon's name

Client – the Client's name

NumVols – the current number of Volumes in the Pool

Pool – the Pool name

Catalog – the Catalog name

MediaType – the Media Type

Environment Variables Environment variables are read-only, and must be defined in the environment prior to executing Bacula. Environment variables may be either scalar or an array, where the elements of the array are referenced by subscripting the variable name (e.g. **${Months[3]}**). Environment variable arrays are defined by separating the elements with a vertical bar (|), thus **set Months="Jan|Feb|Mar|Apr|..."** defines an environment variable named **Month** that will be treated as an array, and the reference **${Months[3]}** will yield **Mar**. The elements of the array can have differing lengths.

## 2.3 Full Syntax

Since the syntax is quite extensive, below, you will find the pseudo BNF. The special characters have the following meaning:

```
::=     definition
( )     grouping if the parens are not quoted
|       separates alternatives
'/'     literal / (or any other character)
CAPS    a character or character sequence
*       preceding item can be repeated zero or more times
?       preceding item can appear zero or one time
+       preceding item must appear one or more times
```

And the pseudo BNF describing the syntax is:

```
input       ::= ( TEXT
                | variable
                | INDEX_OPEN input INDEX_CLOSE (loop_limits)?
                )*
variable    ::= DELIM_INIT (name|expression)
name        ::= (NAME_CHARS)+
expression  ::= DELIM_OPEN
                (name|variable)+
                (INDEX_OPEN num_exp INDEX_CLOSE)?
                (':' command)*
                DELIM_CLOSE
command     ::= '-' (TEXT_EXP|variable)+
              | '+' (TEXT_EXP|variable)+
              | 'o' NUMBER ('-'|',') (NUMBER)?
```

```
              | '#'
              | '*' (TEXT_EXP|variable)+
              | 's' '/' (TEXT_PATTERN)+
                    '/' (variable|TEXT_SUBST)*
                    '/' ('m'|'g'|'i'|'t')*
              | 'y' '/' (variable|TEXT_SUBST)+
                    '/' (variable|TEXT_SUBST)*
                    '/'
              | 'p' '/' NUMBER
                    '/' (variable|TEXT_SUBST)*
                    '/' ('r'|'l'|'c')
              | '%' (name|variable)+
                    ('(' (TEXT_ARGS)? ')')?
              | 'l'
              | 'u'
num_exp     ::= operand
              | operand ('+'|'-'|'*'|'/'|'%') num_exp
operand     ::= ('+'|'-')? NUMBER
              | INDEX_MARK
              | '(' num_exp ')'
              | variable
loop_limits ::= DELIM_OPEN
                (num_exp)? ',' (num_exp)? (',' (num_exp)?)?
                DELIM_CLOSE
NUMBER      ::= ('0'|...|'9')+
TEXT_PATTERN::= (^('/'))+
TEXT_SUBST  ::= (^(DELIM_INIT|'/'))+
TEXT_ARGS   ::= (^(DELIM_INIT|')'))+
TEXT_EXP    ::= (^(DELIM_INIT|DELIM_CLOSE|':'|'+'))+
TEXT        ::= (^(DELIM_INIT|INDEX_OPEN|INDEX_CLOSE))+
DELIM_INIT  ::= '$'
DELIM_OPEN  ::= '{'
DELIM_CLOSE ::= '}'
INDEX_OPEN  ::= '['
INDEX_CLOSE ::= ']'
INDEX_MARK  ::= '#'
NAME_CHARS  ::= 'a'|...|'z'|'A'|...|'Z'|'0'|...|'9'
```

## 2.4 Semantics

The items listed in **command** above, which always follow a colon (**:**) have the following meanings:

```
-    perform substitution if variable is empty
+    perform substitution if variable is not empty
o    cut out substring of the variable value
#    length of the variable value
*    substitute empty string if the variable value is not empty,
     otherwise substitute the trailing parameter
s    regular expression search and replace. The trailing
     options are: m = multiline, i = case insensitive,
                  g = global,    t = plain text (no regexp)
y    transpose characters from class A to class B
p    pad variable to l = left, r = right or c = center,
     with second value.
%    special function call (none implemented)
l    lower case the variable value
u    upper case the variable value
```

The **loop_limits** are start, step, and end values.

A counter variable name followed immediately by a plus (+) will cause the counter to be incremented by one.

## 2.5 Examples

To create an ISO date:

```
DLT-${Year}-${Month:p/2/0/r}-${Day:p/2/0/r}
```

on 20 June 2003 would give **DLT-2003-06-20**

If you set the environment variable **mon** to

```
January|February|March|April|May|...
File-${mon[${Month}]}/${Day}/${Year}
```

on the first of March would give **File-March/1/2003**

# Chapter 3

# Using Stunnel to Encrypt Communications

Prior to version 1.37, Bacula did not have built-in communications encryption. Please see the TLS chapter if you are using Bacula 1.37 or greater.

Without too much effort, it is possible to encrypt the communications between any of the daemons. This chapter will show you how to use `stunnel` to encrypt communications to your client programs. We assume the Director and the Storage daemon are running on one machine that will be called **server** and the Client or File daemon is running on a different machine called **client**. Although the details may be slightly different, the same principles apply whether you are encrypting between Unix, Linux, or Win32 machines. This example was developed between two Linux machines running stunnel version 4.04-4 on a Red Hat Enterprise 3.0 system.

## 3.1 Communications Ports Used

First, you must know that with the standard Bacula configuration, the Director will contact the File daemon on port 9102. The File daemon then contacts the Storage daemon using the address and port parameters supplied by the Director. The standard port used will be 9103. This is the typical server/client view of the world, the File daemon is a server to the Director (i.e. listens for the Director to contact it), and the Storage daemon is a server to the File daemon.

## 3.2 Encryption

The encryption is accomplished between the Director and the File daemon by using an stunnel on the Director's machine (server) to encrypt the data and to contact an stunnel on the File daemon's machine (client), which decrypts the data and passes it to the client.

Between the File daemon and the Storage daemon, we use an stunnel on the File daemon's machine to encrypt the data and another stunnel on the Storage daemon's machine to decrypt the data.

As a consequence, there are actually four copies of stunnel running, two on the server and two on the client. This may sound a bit complicated, but it really isn't. To accomplish this, we will need to construct four separate conf files for stunnel, and we will need to make some minor modifications to the Director's conf file. None of the other conf files need to be changed.

## 3.3 A Picture

Since pictures usually help a lot, here is an overview of what we will be doing. Don't worry about all the details of the port numbers and such for the moment.

```
File daemon (client):
                stunnel-fd1.conf
                  |===========|
Port 29102  >----| Stunnel 1 |-----> Port 9102
                  |===========|
                stunnel-fd2.conf
                  |===========|
Port 9103   >----| Stunnel 2 |-----> server:29103
                  |===========|
Director (server):
                stunnel-dir.conf
                  |===========|
Port 29102  >----| Stunnel 3 |-----> client:29102
                  |===========|
                stunnel-sd.conf
                  |===========|
Port 29103  >----| Stunnel 4 |-----> 9103
                  |===========|
```

## 3.4 Certificates

In order for stunnel to function as a server, which it does in our diagram for Stunnel 1 and Stunnel 4, you must have a certificate and the key. It is possible to keep the two in separate files, but normally, you keep them in one single `.pem` file. You may create this certificate yourself in which case, it will be self-signed, or you may have it signed by a Certificate Authority (CA).

If you want your clients to verify that the server is in fact valid (Stunnel 2 and Stunnel 3), you will need to have the server certificates signed by a CA , and you will need to have the CA's public certificate (contains the CA's public key).

Having a CA signed certificate is **highly** recommended if you are using your client across the Internet, otherwise you are exposed to the man in the middle attack and hence loss of your data.

See below for how to create a self-signed certificate.

## 3.5 Securing the Data Channel

To simplify things a bit, let's for the moment consider only the data channel. That is the connection between the File daemon and the Storage daemon, which takes place on port 9103. In fact, in a minimalist solution, this is the only connection that needs to be encrypted, because it is the one that transports your data. The connection between the Director and the File daemon is simply a control channel used to start the job and get the job status.

Normally the File daemon will contact the Storage daemon on port 9103 (supplied by the Director), so we need an stunnel that listens on port 9103 on the File daemon's machine, encrypts the data and sends it to the Storage daemon. This is depicted by Stunnel 2 above. Note that this stunnel is listening on port 9103 and sending to server:29103. We use port 29103 on the server because if we would send the data to port 9103, it would go directly to the Storage daemon, which doesn't understand encrypted data. On the server machine, we run Stunnel 4, which listens on port 29103, decrypts the data and sends it to the Storage daemon, which is listening on port 9103.

## 3.6 Data Channel Configuration

The Storage resource of the bacula-dir.conf normally looks something like the following:

```
Storage {
  Name = File
  Address = server
  SDPort = 9103
  Password = storage_password
  Device = File
  Media Type = File
}
```

Notice that this is running on the server machine, and it points the File daemon back to server:9103, which is where our Storage daemon is listening. We modify this to be:

```
Storage {
  Name = File
  Address = localhost
  SDPort = 9103
  Password = storage_password
  Device = File
  Media Type = File
}
```

This causes the File daemon to send the data to the stunnel running on localhost (the client machine). We could have used client as the address as well.

## 3.7 Stunnel Configuration for the Data Channel

In the diagram above, we see above Stunnel 2 that we use stunnel-fd2.conf on the client. A pretty much minimal config file would look like the following:

```
client = yes
[29103]
accept = localhost:9103
connect = server:29103
```

The above config file does encrypt the data but it does not require a certificate, so it is subject to the man in the middle attack. The file I actually used, stunnel-fd2.conf, looked like this:

```
#
# Stunnel conf for Bacula client -> SD
#
pid = /home/kern/bacula/bin/working/stunnel.pid
#
# A cert is not mandatory here. If verify=2, a
#  cert signed by a CA must be specified, and
#  either CAfile or CApath must point to the CA's
#  cert
#
cert = /home/kern/stunnel/stunnel.pem
CAfile = /home/kern/ssl/cacert.pem
verify = 2
client = yes
# debug = 7
# foreground = yes
[29103]
accept = localhost:9103
connect = server:29103
```

You will notice that I specified a pid file location because I ran stunnel under my own userid so I could not use the default, which requires root permission. I also specified a certificate that I have as well as verify level 2 so that the certificate is required and verified, and I must supply the location of the CA certificate so that the stunnel certificate can be verified. Finally, you will see that there are two lines commented out, which when enabled, produce a lot of nice debug info in the command window.

If you do not have a signed certificate (stunnel.pem), you need to delete the cert, CAfile, and verify lines.

Note that the stunnel.pem, is actually a private key and a certificate in a single file. These two can be kept and specified individually, but keeping them in one file is more convenient.

The config file, stunnel-sd.conf, needed for Stunnel 4 on the server machine is:

```
#
# Bacula stunnel conf for Storage daemon
#
pid = /home/kern/bacula/bin/working/stunnel.pid
#
# A cert is mandatory here, it may be self signed
#  If it is self signed, the client may not use
#  verify
#
cert   = /home/kern/stunnel/stunnel.pem
client = no
# debug = 7
# foreground = yes
[29103]
accept = 29103
connect = 9103
```

## 3.8 Starting and Testing the Data Encryption

It will most likely be the simplest to implement the Data Channel encryption in the following order:

- Setup and run Bacula backing up some data on your client machine without encryption.
- Stop Bacula.
- Modify the Storage resource in the Director's conf file.
- Start Bacula
- Start stunnel on the server with:

  ```
  stunnel stunnel-sd.conf
  ```

- Start stunnel on the client with:

  ```
      stunnel stunnel-fd2.conf
  ```

- Run a job.
- If it doesn't work, turn debug on in both stunnel conf files, restart the stunnels, rerun the job, repeat until it works.

## 3.9 Encrypting the Control Channel

The Job control channel is between the Director and the File daemon, and as mentioned above, it is not really necessary to encrypt, but it is good practice to encrypt it as well. The two stunnels

that are used in this case will be Stunnel 1 and Stunnel 3 in the diagram above. Stunnel 3 on the server might normally listen on port 9102, but if you have a local File daemon, this will not work, so we make it listen on port 29102. It then sends the data to client:29102. Again we use port 29102 so that the stunnel on the client machine can decrypt the data before passing it on to port 9102 where the File daemon is listening.

## 3.10 Control Channel Configuration

We need to modify the standard Client resource, which would normally look something like:

```
Client {
  Name = client-fd
  Address = client
  FDPort = 9102
  Catalog = BackupDB
  Password = "xxx"
}
```

to be:

```
Client {
  Name = client-fd
  Address = localhost
  FDPort = 29102
  Catalog = BackupDB
  Password = "xxx"
}
```

This will cause the Director to send the control information to localhost:29102 instead of directly to the client.

## 3.11 Stunnel Configuration for the Control Channel

The stunnel config file, `stunnel-dir.conf`, for the Director's machine would look like the following:

```
#
# Bacula stunnel conf for the Directory to contact a client
#
pid = /home/kern/bacula/bin/working/stunnel.pid
#
# A cert is not mandatory here. If verify=2, a
#  cert signed by a CA must be specified, and
#  either CAfile or CApath must point to the CA's
#  cert
#
cert   = /home/kern/stunnel/stunnel.pem
CAfile = /home/kern/ssl/cacert.pem
verify = 2
client = yes
# debug = 7
# foreground = yes
[29102]
accept = localhost:29102
connect = client:29102
```

and the config file, `stunnel-fd1.conf`, needed to run stunnel on the Client would be:

```
#
# Bacula stunnel conf for the Directory to contact a client
#
pid = /home/kern/bacula/bin/working/stunnel.pid
#
# A cert is not mandatory here. If verify=2, a
#  cert signed by a CA must be specified, and
#  either CAfile or CApath must point to the CA's
#  cert
#
cert   = /home/kern/stunnel/stunnel.pem
CAfile = /home/kern/ssl/cacert.pem
verify = 2
client = yes
# debug = 7
# foreground = yes
[29102]
accept = localhost:29102
connect = client:29102
```

## 3.12 Starting and Testing the Control Channel

It will most likely be the simplest to implement the Control Channel encryption in the following order:

- Stop Bacula.
- Modify the Client resource in the Director's conf file.
- Start Bacula
- Start `stunnel` on the server with:

  ```
  stunnel stunnel-dir.conf
  ```

- Start `stunnel` on the client with:

  ```
  stunnel stunnel-fd1.conf
  ```

- Run a job.
- If it doesn't work, turn debug on in both stunnel conf files, restart the stunnels, rerun the job, repeat until it works.

## 3.13 Using stunnel to Encrypt to a Second Client

On the client machine, you can just duplicate the setup that you have on the first client file for file and it should work fine.

In the `bacula-dir.conf` file, you will want to create a second client pretty much identical to how you did for the first one, but the port number must be unique. We previously used:

```
Client {
  Name = client-fd
  Address = localhost
  FDPort = 29102
  Catalog = BackupDB
  Password = "xxx"
}
```

so for the second client, we will, of course, have a different name, and we will also need a different port. Remember that we used port 29103 for the Storage daemon, so for the second client, we can use port 29104, and the Client resource would look like:

```
Client {
  Name = client2-fd
  Address = localhost
  FDPort = 29104
  Catalog = BackupDB
  Password = "yyy"
}
```

Now, fortunately, we do not need a third stunnel to on the Director's machine, we can just add the new port to the config file, `stunnel-dir.conf`, to make:

```
#
# Bacula stunnel conf for the Directory to contact a client
#
pid = /home/kern/bacula/bin/working/stunnel.pid
#
# A cert is not mandatory here. If verify=2, a
#  cert signed by a CA must be specified, and
#  either CAfile or CApath must point to the CA's
#  cert
#
cert   = /home/kern/stunnel/stunnel.pem
CAfile = /home/kern/ssl/cacert.pem
verify = 2
client = yes
# debug = 7
# foreground = yes
[29102]
accept = localhost:29102
connect = client:29102
[29104]
accept = localhost:29102
connect = client2:29102
```

There are no changes necessary to the Storage daemon or the other stunnel so that this new client can talk to our Storage daemon.

## 3.14 Creating a Self-signed Certificate

You may create a self-signed certificate for use with stunnel that will permit you to make it function, but will not allow certificate validation. The `.pem` file containing both the certificate and the key can be made with the following, which I put in a file named `makepem`:

```
#!/bin/sh
#
# Simple shell script to make a .pem file that can be used
# with stunnel and Bacula
#
OPENSSL=openssl
   umask 77
   PEM1="/bin/mktemp openssl.XXXXXX"
   PEM2="/bin/mktemp openssl.XXXXXX"
   ${OPENSSL} req -newkey rsa:1024 -keyout $PEM1 -nodes \
       -x509 -days 365 -out $PEM2
   cat $PEM1 > stunnel.pem
   echo ""   >>stunnel.pem
   cat $PEM2 >>stunnel.pem
   rm $PEM1 $PEM2
```

The above script will ask you a number of questions. You may simply answer each of them by entering a return, or if you wish you may enter your own data.

## 3.15 Getting a CA Signed Certificate

The process of getting a certificate that is signed by a CA is quite a bit more complicated. You can purchase one from quite a number of PKI vendors, but that is not at all necessary for use with Bacula.

To get a CA signed certificate, you will either need to find a friend that has setup his own CA or to become a CA yourself, and thus you can sign all your own certificates. The book OpenSSL by John Viega, Matt Mesier & Pravir Chandra[1] from O'Reilly explains how to do it, or you can read the documentation provided in the Open-source PKI Book project at Source Forge.

Note, the link may change.

## 3.16 Using ssh to Secure the Communications

Please see the script `ssh-tunnel.sh` in the `examples` directory. It was contributed by Stephan Holl.

[1] http://ospkibook.sourceforge.net/docs/OSPKI-2.4.7/OSPKI-html/ospki-book.htm

# Appendices

# Index

## Symbols



## A

## B



## C



## D



## E



## F



## G



## I



## K

## L



## M



## N



## O



## P



## Q



## R



## S



## T



## U

# Appendix A

# Acronyms

**BAT** Bacula Administration Tool
**CA** Certificate Authority
**GNU** Gnu is not Unix
**GPL** Gnu General Public License
**GTK** Graphic ToolKit
**GUI** Graphical User Interface
**NLS** Native Language Support
**PKI** Public-Key Infrastructure
**PHP** PHP Hypertext Processor
**SQL** Structured Query Language
**SVN** Subversion
**TLS** Transport Layer Security
**VSS** Volume Snapshot Service