



Bacula Utility Programs

It comes in the night and sucks the essence from your computers.

Kern Sibbald

November 19, 2012

This manual documents Bacula version 5.0.0 (26 January 2010)

Copyright ©1999-2010, Free Software Foundation Europe e.V.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Contents

1	Volume Utility Tools	5
1.1	Specifying the Configuration File	5
1.2	Specifying a Device Name For a Tape	5
1.3	Specifying a Device Name For a File	5
1.4	Specifying Volumes	5
1.5	bls	6
1.5.1	Listing Jobs	7
1.5.2	Listing Blocks	7
1.6	bextract	8
1.6.1	Extracting with Include or Exclude Lists	9
1.6.2	Extracting With a Bootstrap File	10
1.6.3	Extracting From Multiple Volumes	10
1.7	bscan	10
1.7.1	Using bscan to Compare a Volume to an existing Catalog	12
1.7.2	Using bscan to Recreate a Catalog from a Volume	12
1.7.3	Using bscan to Correct the Volume File Count	14
1.7.4	After bscan	14
1.8	bcopy	14
1.8.1	bcopy Command Options	14
1.9	btape	14
1.9.1	Using btape to Verify your Tape Drive	15
1.9.2	btape Commands	15
1.10	Other Programs	17
1.11	bsmtp	17
1.12	dbcheck	18
1.13	bregex	20

1.14	bwild	21
1.15	testfind	21
1.16	bimagemgr	23
1.16.1	bimagemgr installation	23
1.16.2	bimagemgr usage	24
2	Bacula RPM Packaging FAQ	27
2.1	Answers	27
2.2	Build Options	30
2.3	RPM Install Problems	31
3	GNU Free Documentation License	33

Chapter 1

Volume Utility Tools

This document describes the utility programs written to aid Bacula users and developers in dealing with Volumes external to Bacula.

1.1 Specifying the Configuration File

Starting with version 1.27, each of the following programs requires a valid Storage daemon configuration file (actually, the only part of the configuration file that these programs need is the **Device** resource definitions). This permits the programs to find the configuration parameters for your archive device (generally a tape drive). By default, they read **bacula-sd.conf** in the current directory, but you may specify a different configuration file using the **-c** option.

1.2 Specifying a Device Name For a Tape

Each of these programs require a **device-name** where the Volume can be found. In the case of a tape, this is the physical device name such as **/dev/nst0** or **/dev/rmt/0ubn** depending on your system. For the program to work, it must find the identical name in the Device resource of the configuration file. See below for specifying Volume names.

Please note that if you have Bacula running and you ant to use one of these programs, you will either need to stop the Storage daemon, or **unmount** any tape drive you want to use, otherwise the drive will **busy** because Bacula is using it.

1.3 Specifying a Device Name For a File

If you are attempting to read or write an archive file rather than a tape, the **device-name** should be the full path to the archive location including the filename. The filename (last part of the specification) will be stripped and used as the Volume name, and the path (first part before the filename) must have the same entry in the configuration file. So, the path is equivalent to the archive device name, and the filename is equivalent to the volume name.

1.4 Specifying Volumes

In general, you must specify the Volume name to each of the programs below (with the exception of **btape**). The best method to do so is to specify a **bootstrap** file on the command line with the **-b** option. As part

of the bootstrap file, you will then specify the Volume name or Volume names if more than one volume is needed. For example, suppose you want to read tapes **tape1** and **tape2**. First construct a **bootstrap** file named say, **list.bsr** which contains:

```
Volume=test1|test2
```

where each Volume is separated by a vertical bar. Then simply use:

```
./bls -b list.bsr /dev/nst0
```

In the case of Bacula Volumes that are on files, you may simply append volumes as follows:

```
./bls /tmp/test1\\|test2
```

where the backslash (\) was necessary as a shell escape to permit entering the vertical bar (—).

And finally, if you feel that specifying a Volume name is a bit complicated with a bootstrap file, you can use the **-V** option (on all programs except **bcopy**) to specify one or more Volume names separated by the vertical bar (—). For example,

```
./bls -V Vol001 /dev/nst0
```

You may also specify an asterisk (*) to indicate that the program should accept any volume. For example:

```
./bls -V* /dev/nst0
```

1.5 bls

bls can be used to do an **ls** type listing of a **Bacula** tape or file. It is called:

```
Usage: bls [options] <device-name>
      -b <file>      specify a bootstrap file
      -c <file>      specify a config file
      -d <level>     specify debug level
      -e <file>      exclude list
      -i <file>      include list
      -j             list jobs
      -k             list blocks
      (no j or k option) list saved files
      -L             dump label
      -p             proceed inspite of errors
      -v             be verbose
      -V             specify Volume names (separated by |)
      -?             print this message
```

For example, to list the contents of a tape:

```
./bls -V Volume-name /dev/nst0
```

Or to list the contents of a file:

```
./bls /tmp/Volume-name
or
./bls -V Volume-name /tmp
```

Note that, in the case of a file, the Volume name becomes the filename, so in the above example, you will replace the **xxx** with the name of the volume (file) you wrote.

Normally if no options are specified, **bls** will produce the equivalent output to the **ls -l** command for each file on the tape. Using other options listed above, it is possible to display only the Job records, only the tape blocks, etc. For example:

```
./bls /tmp/File002
bls: butil.c:148 Using device: /tmp
drwxrwxr-x  3 k k  4096 02-10-19 21:08 /home/kern/bacula/k/src/dird/
drwxrwxr-x  2 k k  4096 02-10-10 18:59 /home/kern/bacula/k/src/dird/CVS/
-rw-rw-r--  1 k k    54 02-07-06 18:02 /home/kern/bacula/k/src/dird/CVS/Root
-rw-rw-r--  1 k k    16 02-07-06 18:02 /home/kern/bacula/k/src/dird/CVS/Repository
-rw-rw-r--  1 k k  1783 02-10-10 18:59 /home/kern/bacula/k/src/dird/CVS/Entries
-rw-rw-r--  1 k k 97506 02-10-18 21:07 /home/kern/bacula/k/src/dird/Makefile
-rw-r--r--  1 k k  3513 02-10-18 21:02 /home/kern/bacula/k/src/dird/Makefile.in
-rw-rw-r--  1 k k  4669 02-07-06 18:02 /home/kern/bacula/k/src/dird/README-config
-rw-r--r--  1 k k  4391 02-09-14 16:51 /home/kern/bacula/k/src/dird/authenticate.c
-rw-r--r--  1 k k  3609 02-07-07 16:41 /home/kern/bacula/k/src/dird/autopruner.c
-rw-rw-r--  1 k k  4418 02-10-18 21:03 /home/kern/bacula/k/src/dird/bacula-dir.conf
...
-rw-rw-r--  1 k k    83 02-08-31 19:19 /home/kern/bacula/k/src/dird/.cvsignore
bls: Got EOF on device /tmp
84 files found.
```

1.5.1 Listing Jobs

If you are listing a Volume to determine what Jobs to restore, normally the **-j** option provides you with most of what you will need as long as you don't have multiple clients. For example,

```
./bls -j -V Test1 -c stored.conf DDS-4
bls: butil.c:258 Using device: "DDS-4" for reading.
11-Jul 11:54 bls: Ready to read from volume "Test1" on device "DDS-4" (/dev/nst0).
Volume Record: File:blk=0:1 SessId=4 SessTime=1121074625 JobId=0 DataLen=165
Begin Job Session Record: File:blk=0:2 SessId=4 SessTime=1121074625 JobId=1 Level=F Type=B
Begin Job Session Record: File:blk=0:3 SessId=5 SessTime=1121074625 JobId=5 Level=F Type=B
Begin Job Session Record: File:blk=0:6 SessId=3 SessTime=1121074625 JobId=2 Level=F Type=B
Begin Job Session Record: File:blk=0:13 SessId=2 SessTime=1121074625 JobId=4 Level=F Type=B
End Job Session Record: File:blk=0:99 SessId=3 SessTime=1121074625 JobId=2 Level=F Type=B
Files=168 Bytes=1,732,978 Errors=0 Status=T
End Job Session Record: File:blk=0:101 SessId=2 SessTime=1121074625 JobId=4 Level=F Type=B
Files=168 Bytes=1,732,978 Errors=0 Status=T
End Job Session Record: File:blk=0:108 SessId=5 SessTime=1121074625 JobId=5 Level=F Type=B
Files=168 Bytes=1,732,978 Errors=0 Status=T
End Job Session Record: File:blk=0:109 SessId=4 SessTime=1121074625 JobId=1 Level=F Type=B
Files=168 Bytes=1,732,978 Errors=0 Status=T
11-Jul 11:54 bls: End of Volume at file 1 on device "DDS-4" (/dev/nst0), Volume "Test1"
11-Jul 11:54 bls: End of all volumes.
```

shows a full save followed by two incremental saves.

Adding the **-v** option will display virtually all information that is available for each record:

1.5.2 Listing Blocks

Normally, except for debugging purposes, you will not need to list Bacula blocks (the "primitive" unit of Bacula data on the Volume). However, you can do so with:

```
./bls -k /tmp/File002
bls: butil.c:148 Using device: /tmp
Block: 1 size=64512
Block: 2 size=64512
```

```
...
Block: 65 size=64512
Block: 66 size=19195
bls: Got EOF on device /tmp
End of File on device
```

By adding the **-v** option, you can get more information, which can be useful in knowing what sessions were written to the volume:

```
./bls -k -v /tmp/File002
Volume Label:
Id                : Bacula 0.9 mortal
VerNo            : 10
VolName          : File002
PrevVolName      :
VolFile          : 0
LabelType        : VOL_LABEL
LabelSize        : 147
PoolName         : Default
MediaType        : File
PoolType         : Backup
HostName         :
Date label written: 2002-10-19 at 21:16
Block: 1 blen=64512 First rec FI=VOL_LABEL SessId=1 SessTim=1035062102 Strm=0 rlen=147
Block: 2 blen=64512 First rec FI=6 SessId=1 SessTim=1035062102 Strm=DATA rlen=4087
Block: 3 blen=64512 First rec FI=12 SessId=1 SessTim=1035062102 Strm=DATA rlen=5902
Block: 4 blen=64512 First rec FI=19 SessId=1 SessTim=1035062102 Strm=DATA rlen=28382
...
Block: 65 blen=64512 First rec FI=83 SessId=1 SessTim=1035062102 Strm=DATA rlen=1873
Block: 66 blen=19195 First rec FI=83 SessId=1 SessTim=1035062102 Strm=DATA rlen=2973
bls: Got EOF on device /tmp
End of File on device
```

Armed with the SessionId and the SessionTime, you can extract just about anything.

If you want to know even more, add a second **-v** to the command line to get a dump of every record in every block.

```
./bls -k -v -v /tmp/File002
bls: block.c:79 Dump block 80f8ad0: size=64512 BlkNum=1
      Hdrcksum=b1bdfd6d cksum=b1bdfd6d
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=VOL_LABEL Strm=0 len=147 p=80f8b40
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=SOS_LABEL Strm=-7 len=122 p=80f8be7
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=1 Strm=UATTR len=86 p=80f8c75
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=2 Strm=UATTR len=90 p=80f8cdf
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=3 Strm=UATTR len=92 p=80f8d4d
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=3 Strm=DATA len=54 p=80f8dbd
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=3 Strm=MD5 len=16 p=80f8e07
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=4 Strm=UATTR len=98 p=80f8e2b
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=4 Strm=DATA len=16 p=80f8ea1
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=4 Strm=MD5 len=16 p=80f8ec5
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=5 Strm=UATTR len=96 p=80f8ee9
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=5 Strm=DATA len=1783 p=80f8f5d
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=5 Strm=MD5 len=16 p=80f9668
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=6 Strm=UATTR len=95 p=80f968c
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=6 Strm=DATA len=32768 p=80f96ff
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=6 Strm=DATA len=32768 p=8101713
bls: block.c:79 Dump block 80f8ad0: size=64512 BlkNum=2
      Hdrcksum=9acc1e7f cksum=9acc1e7f
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=6 Strm=contDATA len=4087 p=80f8b40
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=6 Strm=DATA len=31970 p=80f9b4b
bls: block.c:92 Rec: Vid=1 VT=1035062102 FI=6 Strm=MD5 len=16 p=8101841
...
```

1.6 bextract

If you find yourself using **bextract**, you probably have done something wrong. For example, if you are trying to recover a file but are having problems, please see the Restoring When Things Go Wrong section of

the Restore chapter of this manual.

Normally, you will restore files by running a **Restore** Job from the **Console** program. However, **bextract** can be used to extract a single file or a list of files from a Bacula tape or file. In fact, **bextract** can be a useful tool to restore files to an empty system assuming you are able to boot, you have statically linked **bextract** and you have an appropriate **bootstrap** file.

Please note that some of the current limitations of bextract are:

1. It cannot restore access control lists (ACL) that have been backed up along with the file data.
2. It cannot restore encrypted files.
3. The command line length is relatively limited, which means that you cannot enter a huge number of volumes. If you need to enter more volumes than the command line supports, please use a bootstrap file (see below).

It is called:

```
Usage: bextract [-d debug_level] <device-name> <directory-to-store-files>
      -b <file>          specify a bootstrap file
      -dnn              set debug level to nn
      -e <file>          exclude list
      -i <file>          include list
      -p               proceed inspite of I/O errors
      -V               specify Volume names (separated by |)
      -?               print this message
```

where **device-name** is the Archive Device (raw device name or full filename) of the device to be read, and **directory-to-store-files** is a path prefix to prepend to all the files restored.

NOTE: On Windows systems, if you specify a prefix of say d:/tmp, any file that would have been restored to **c:/My Documents** will be restored to **d:/tmp/My Documents**. That is, the original drive specification will be stripped. If no prefix is specified, the file will be restored to the original drive.

1.6.1 Extracting with Include or Exclude Lists

Using the **-e** option, you can specify a file containing a list of files to be excluded. Wildcards can be used in the exclusion list. This option will normally be used in conjunction with the **-i** option (see below). Both the **-e** and the **-i** options may be specified at the same time as the **-b** option. The bootstrap filters will be applied first, then the include list, then the exclude list.

Likewise, and probably more importantly, with the **-i** option, you can specify a file that contains a list (one file per line) of files and directories to include to be restored. The list must contain the full filename with the path. If you specify a path name only, all files and subdirectories of that path will be restored. If you specify a line containing only the filename (e.g. **my-file.txt**) it probably will not be extracted because you have not specified the full path.

For example, if the file **include-list** contains:

```
/home/kern/bacula
/usr/local/bin
```

Then the command:

```
./bextract -i include-list -V Volume /dev/nst0 /tmp
```

will restore from the Bacula archive **/dev/nst0** all files and directories in the backup from **/home/kern/bacula** and from **/usr/local/bin**. The restored files will be placed in a file of the original name under the directory **/tmp** (i.e. **/tmp/home/kern/bacula/...** and **/tmp/usr/local/bin/...**).

1.6.2 Extracting With a Bootstrap File

The **-b** option is used to specify a **bootstrap** file containing the information needed to restore precisely the files you want. Specifying a **bootstrap** file is optional but recommended because it gives you the most control over which files will be restored. For more details on the **bootstrap** file, please see Restoring Files with the Bootstrap File chapter of this document. Note, you may also use a bootstrap file produced by the **restore** command. For example:

```
./bextract -b bootstrap-file /dev/nst0 /tmp
```

The bootstrap file allows detailed specification of what files you want restored (extracted). You may specify a bootstrap file and include and/or exclude files at the same time. The bootstrap conditions will first be applied, and then each file record seen will be compared to the include and exclude lists.

1.6.3 Extracting From Multiple Volumes

If you wish to extract files that span several Volumes, you can specify the Volume names in the bootstrap file or you may specify the Volume names on the command line by separating them with a vertical bar. See the section above under the **bls** program entitled **Listing Multiple Volumes** for more information. The same techniques apply equally well to the **bextract** program or read the Bootstrap chapter of this document.

1.7 bscan

If you find yourself using this program, you have probably done something wrong. For example, the best way to recover a lost or damaged Bacula database is to reload the database by using the bootstrap file that was written when you saved it (default bacula-dir.conf file).

The **bscan** program can be used to re-create a database (catalog) records from the backup information written to one or more Volumes. This is normally needed only if one or more Volumes have been pruned or purged from your catalog so that the records on the Volume are no longer in the catalog, or for Volumes that you have archived. Note, if you scan in Volumes that were previously purged, you will be able to do restores from those Volumes. However, unless you modify the Job and File retention times for the Jobs that were added by scanning, the next time you run any Job with the same name, the records will be pruned again. Since it takes a long time to scan Volumes this can be very frustrating.

With some care, **bscan** can also be used to synchronize your existing catalog with a Volume. Although we have never seen a case of bscan damaging a catalog, since bscan modifies your catalog, we recommend that you do a simple ASCII backup of your database before running **bscan** just to be sure. See Compacting Your Database for the details of making a copy of your database.

bscan can also be useful in a disaster recovery situation, after the loss of a hard disk, if you do not have a valid **bootstrap** file for reloading your system, or if a Volume has been recycled but not overwritten, you can use **bscan** to re-create your database, which can then be used to **restore** your system or a file to its previous state.

It is called:

```
Usage: bscan [options] <bacula-archive>
-b bootstrap    specify a bootstrap file
-c <file>       specify configuration file
-d <nn>         set debug level to nn
-m             update media info in database
-n <name>       specify the database name (default bacula)
-u <user>       specify database user name (default bacula)
-P <password>   specify database password (default none)
-h <host>       specify database host (default NULL)
```

-p	proceed inspite of I/O errors
-r	list records
-s	synchronize or store in database
-v	verbose
-V <Volumes>	specify Volume names (separated by)
-w <dir>	specify working directory (default from conf file)
-?	print this message

If you are using MySQL or PostgreSQL, there is no need to supply a working directory since in that case, bscan knows where the databases are. However, if you have provided security on your database, you may need to supply either the database name (**-b** option), the user name (**-u** option), and/or the password (**-p**) options.

NOTE: before **bscan** can work, it needs at least a bare bones valid database. If your database exists but some records are missing because they were pruned, then you are all set. If your database was lost or destroyed, then you must first ensure that you have the SQL program running (MySQL or PostgreSQL), then you must create the Bacula database (normally named bacula), and you must create the Bacula tables using the scripts in the **cats** directory. This is explained in the Installation chapter of the manual. Finally, before scanning into an empty database, you must start and stop the Director with the appropriate bacula-dir.conf file so that it can create the Client and Storage records which are not stored on the Volumes. Without these records, scanning is unable to connect the Job records to the proper client.

Forgetting for the moment the extra complications of a full rebuild of your catalog, let's suppose that you did a backup to Volumes "Vol001" and "Vol002", then sometime later all records of one or both those Volumes were pruned or purged from the database. By using **bscan** you can recreate the catalog entries for those Volumes and then use the **restore** command in the Console to restore whatever you want. A command something like:

```
bscan -c bacula-sd.conf -v -V Vol001\|Vol002 /dev/nst0
```

will give you an idea of what is going to happen without changing your catalog. Of course, you may need to change the path to the Storage daemon's conf file, the Volume name, and your tape (or disk) device name. This command must read the entire tape, so if it has a lot of data, it may take a long time, and thus you might want to immediately use the command listed below. Note, if you are writing to a disk file, replace the device name with the path to the directory that contains the Volumes. This must correspond to the Archive Device in the conf file.

Then to actually write or store the records in the catalog, add the **-s** option as follows:

```
bscan -s -m -c bacula-sd.conf -v -V Vol001\|Vol002 /dev/nst0
```

When writing to the database, if bscan finds existing records, it will generally either update them if something is wrong or leave them alone. Thus if the Volumes you are scanning are all or partially in the catalog already, no harm will be done to that existing data. Any missing data will simply be added.

If you have multiple tapes, you should scan them with:

```
bscan -s -m -c bacula-sd.conf -v -V Vol001\|Vol002\|Vol003 /dev/nst0
```

Since there is a limit on the command line length (511 bytes) accepted by **bscan**, if you have too many Volumes, you will need to manually create a bootstrap file. See the Bootstrap chapter of this manual for more details, in particular the section entitled Bootstrap for bscan.

You should, always try to specify the tapes in the order they are written. However, bscan can handle scanning tapes that are not sequential. Any incomplete records at the end of the tape will simply be ignored in that case. If you are simply repairing an existing catalog, this may be OK, but if you are creating a new catalog from scratch, it will leave your database in an incorrect state. If you do not specify all necessary Volumes on a single bscan command, bscan will not be able to correctly restore the records that span two volumes.

In other words, it is much better to specify two or three volumes on a single `bscan` command rather than run `bscan` two or three times, each with a single volume.

Note, the restoration process using `bscan` is not identical to the original creation of the catalog data. This is because certain data such as Client records and other non-essential data such as volume reads, volume mounts, etc is not stored on the Volume, and thus is not restored by `bscan`. The results of bscanning are, however, perfectly valid, and will permit restoration of any or all the files in the catalog using the normal Bacula console commands. If you are starting with an empty catalog and expecting `bscan` to reconstruct it, you may be a bit disappointed, but at a minimum, you must ensure that your `bacula-dir.conf` file is the same as what it previously was – that is, it must contain all the appropriate Client resources so that they will be recreated in your new database **before** running `bscan`. Normally when the Director starts, it will recreate any missing Client records in the catalog. Another problem you will have is that even if the Volumes (Media records) are recreated in the database, they will not have their autochanger status and slots properly set. As a result, you will need to repair that by using the **update slots** command. There may be other considerations as well. Rather than bscanning, you should always attempt to recover your previous catalog backup.

1.7.1 Using `bscan` to Compare a Volume to an existing Catalog

If you wish to compare the contents of a Volume to an existing catalog without changing the catalog, you can safely do so if and only if you do **not** specify either the **-m** or the **-s** options. However, at this time (Bacula version 1.26), the comparison routines are not as good or as thorough as they should be, so we don't particularly recommend this mode other than for testing.

1.7.2 Using `bscan` to Recreate a Catalog from a Volume

This is the mode for which `bscan` is most useful. You can either `bscan` into a freshly created catalog, or directly into your existing catalog (after having made an ASCII copy as described above). Normally, you should start with a freshly created catalog that contains no data.

Starting with a single Volume named **TestVolume1**, you run a command such as:

```
./bscan -V TestVolume1 -v -s -m -c bacula-sd.conf /dev/nst0
```

If there is more than one volume, simply append it to the first one separating it with a vertical bar. You may need to precede the vertical bar with a forward slash escape the shell – e.g. **TestVolume1\—TestVolume2**. The **-v** option was added for verbose output (this can be omitted if desired). The **-s** option that tells `bscan` to store information in the database. The physical device name `/dev/nst0` is specified after all the options.

For example, after having done a full backup of a directory, then two incrementals, I reinitialized the SQLite database as described above, and using the `bootstrap.bsr` file noted above, I entered the following command:

```
./bscan -b bootstrap.bsr -v -s -c bacula-sd.conf /dev/nst0
```

which produced the following output:

```
bscan: bscan.c:182 Using Database: bacula, User: bacula
bscan: bscan.c:673 Created Pool record for Pool: Default
bscan: bscan.c:271 Pool type "Backup" is OK.
bscan: bscan.c:632 Created Media record for Volume: TestVolume1
bscan: bscan.c:298 Media type "DDS-4" is OK.
bscan: bscan.c:307 VOL_LABEL: OK for Volume: TestVolume1
bscan: bscan.c:693 Created Client record for Client: Rufus
bscan: bscan.c:769 Created new JobId=1 record for original JobId=2
bscan: bscan.c:717 Created FileSet record "Kerns Files"
bscan: bscan.c:819 Updated Job termination record for new JobId=1
bscan: bscan.c:905 Created JobMedia record JobId 1, MediaId 1
```

```

bscan: Got EOF on device /dev/nst0
bscan: bscan.c:693 Created Client record for Client: Rufus
bscan: bscan.c:769 Created new JobId=2 record for original JobId=3
bscan: bscan.c:708 Fileset "Kerns Files" already exists.
bscan: bscan.c:819 Updated Job termination record for new JobId=2
bscan: bscan.c:905 Created JobMedia record JobId 2, MediaId 1
bscan: Got EOF on device /dev/nst0
bscan: bscan.c:693 Created Client record for Client: Rufus
bscan: bscan.c:769 Created new JobId=3 record for original JobId=4
bscan: bscan.c:708 Fileset "Kerns Files" already exists.
bscan: bscan.c:819 Updated Job termination record for new JobId=3
bscan: bscan.c:905 Created JobMedia record JobId 3, MediaId 1
bscan: Got EOF on device /dev/nst0
bscan: bscan.c:652 Updated Media record at end of Volume: TestVolume1
bscan: bscan.c:428 End of Volume. VolFiles=3 VolBlocks=57 VolBytes=10,027,437

```

The key points to note are that **bscan** prints a line when each major record is created. Due to the volume of output, it does not print a line for each file record unless you supply the **-v** option twice or more on the command line.

In the case of a Job record, the new JobId will not normally be the same as the original Jobid. For example, for the first JobId above, the new JobId is 1, but the original JobId is 2. This is nothing to be concerned about as it is the normal nature of databases. **bscan** will keep everything straight.

Although **bscan** claims that it created a Client record for Client: Rufus three times, it was actually only created the first time. This is normal.

You will also notice that it read an end of file after each Job (Got EOF on device ...). Finally the last line gives the total statistics for the bscan.

If you had added a second **-v** option to the command line, Bacula would have been even more verbose, dumping virtually all the details of each Job record it encountered.

Now if you start Bacula and enter a **list jobs** command to the console program, you will get:

JobId	Name	StartTime	Type	Lvl	JobFiles	JobBytes	JobStat
1	kernsave	2002-10-07 14:59	B	F	84	4180207	T
2	kernsave	2002-10-07 15:00	B	I	15	2170314	T
3	kernsave	2002-10-07 15:01	B	I	33	3662184	T

which corresponds virtually identically with what the database contained before it was re-initialized and restored with bscan. All the Jobs and Files found on the tape are restored including most of the Media record. The Volume (Media) records restored will be marked as **Full** so that they cannot be rewritten without operator intervention.

It should be noted that **bscan** cannot restore a database to the exact condition it was in previously because a lot of the less important information contained in the database is not saved to the tape. Nevertheless, the reconstruction is sufficiently complete, that you can run **restore** against it and get valid results.

An interesting aspect of restoring a catalog backup using **bscan** is that the backup was made while Bacula was running and writing to a tape. At the point the backup of the catalog is made, the tape Bacula is writing to will have say 10 files on it, but after the catalog backup is made, there will be 11 files on the tape Bacula is writing. This there is a difference between what is contained in the backed up catalog and what is actually on the tape. If after restoring a catalog, you attempt to write on the same tape that was used to backup the catalog, Bacula will detect the difference in the number of files registered in the catalog compared to what is on the tape, and will mark the tape in error.

There are two solutions to this problem. The first is possibly the simplest and is to mark the volume as Used before doing any backups. The second is to manually correct the number of files listed in the Media record of the catalog. This procedure is documented elsewhere in the manual and involves using the **update volume** command in **bconsole**.

1.7.3 Using bscan to Correct the Volume File Count

If the Storage daemon crashes during a backup Job, the catalog will not be properly updated for the Volume being used at the time of the crash. This means that the Storage daemon will have written say 20 files on the tape, but the catalog record for the Volume indicates only 19 files.

Bacula refuses to write on a tape that contains a different number of files from what is in the catalog. To correct this situation, you may run a **bscan** with the **-m** option (but **without** the **-s** option) to update only the final Media record for the Volumes read.

1.7.4 After bscan

If you use **bscan** to enter the contents of the Volume into an existing catalog, you should be aware that the records you entered may be immediately pruned during the next job, particularly if the Volume is very old or had been previously purged. To avoid this, after running **bscan**, you can manually set the volume status (VolStatus) to **Read-Only** by using the **update** command in the catalog. This will allow you to restore from the volume without having it immediately purged. When you have restored and backed up the data, you can reset the VolStatus to **Used** and the Volume will be purged from the catalog.

1.8 bcopy

The **bcopy** program can be used to copy one **Bacula** archive file to another. For example, you may copy a tape to a file, a file to a tape, a file to a file, or a tape to a tape. For tape to tape, you will need two tape drives. (a later version is planned that will buffer it to disk). In the process of making the copy, no record of the information written to the new Volume is stored in the catalog. This means that the new Volume, though it contains valid backup data, cannot be accessed directly from existing catalog entries. If you wish to be able to use the Volume with the Console restore command, for example, you must first bscan the new Volume into the catalog.

1.8.1 bcopy Command Options

```
Usage: bcopy [-d debug_level] <input-archive> <output-archive>
  -b bootstrap      specify a bootstrap file
  -c <file>         specify configuration file
  -dnn              set debug level to nn
  -i                specify input Volume names (separated by |)
  -o                specify output Volume names (separated by |)
  -p                proceed inspite of I/O errors
  -v                verbose
  -w dir            specify working directory (default /tmp)
  -?                print this message
```

By using a **bootstrap** file, you can copy parts of a Bacula archive file to another archive.

One of the objectives of this program is to be able to recover as much data as possible from a damaged tape. However, the current version does not yet have this feature.

As this is a new program, any feedback on its use would be appreciated. In addition, I only have a single tape drive, so I have never been able to test this program with two tape drives.

1.9 btape

This program permits a number of elementary tape operations via a tty command interface. It works only with tapes and not with other kinds of Bacula storage media (DVD, File, ...). The **test** command, described

below, can be very useful for testing older tape drive compatibility problems. Aside from initial testing of tape drive compatibility with **Bacula**, **btape** will be mostly used by developers writing new tape drivers.

btape can be dangerous to use with existing **Bacula** tapes because it will relabel a tape or write on the tape if so requested regardless that the tape may contain valuable data, so please be careful and use it only on blank tapes.

To work properly, **btape** needs to read the Storage daemon's configuration file. As a default, it will look for **bacula-sd.conf** in the current directory. If your configuration file is elsewhere, please use the **-c** option to specify where.

The physical device name must be specified on the command line, and this same device name must be present in the Storage daemon's configuration file read by **btape**

```
Usage: btape <options> <device_name>
      -b <file>    specify bootstrap file
      -c <file>    set configuration file to file
      -d <nn>      set debug level to nn
      -p           proceed inspite of I/O errors
      -s           turn off signals
      -v           be verbose
      -?           print this message.
```

1.9.1 Using btape to Verify your Tape Drive

An important reason for this program is to ensure that a Storage daemon configuration file is defined so that Bacula will correctly read and write tapes.

It is highly recommended that you run the **test** command before running your first Bacula job to ensure that the parameters you have defined for your storage device (tape drive) will permit **Bacula** to function properly. You only need to mount a blank tape, enter the command, and the output should be reasonably self explanatory. Please see the Tape Testing Chapter of this manual for the details.

1.9.2 btape Commands

The full list of commands are:

Command	Description
=====	=====
autochanger	test autochanger
bsf	backspace file
bsr	backspace record
cap	list device capabilities
clear	clear tape errors
eod	go to end of Bacula data for append
eom	go to the physical end of medium
fill	fill tape, write onto second volume
unfill	read filled tape
fsf	forward space a file
fsr	forward space a record
help	print this command
label	write a Bacula label to the tape
load	load a tape
quit	quit btape
rawfill	use write() to fill tape
readlabel	read and print the Bacula tape label
rectest	test record handling functions
rewind	rewind the tape
scan	read() tape block by block to EOT and report
scanblocks	Bacula read block by block to EOT and report
speed	report drive speed
status	print tape status
test	General test Bacula tape functions
weof	write an EOF on the tape

```

wr      write a single Bacula block
rr      read a single record
qfill   quick fill command

```

The most useful commands are:

- **test** – test writing records and EOF marks and reading them back.
- **fill** – completely fill a volume with records, then write a few records on a second volume, and finally, both volumes will be read back. This command writes blocks containing random data, so your drive will not be able to compress the data, and thus it is a good test of the real physical capacity of your tapes.
- **readlabel** – read and dump the label on a Bacula tape.
- **cap** – list the device capabilities as defined in the configuration file and as perceived by the Storage daemon.

The **readlabel** command can be used to display the details of a Bacula tape label. This can be useful if the physical tape label was lost or damaged.

In the event that you want to relabel a **Bacula**, you can simply use the **label** command which will write over any existing label. However, please note for labeling tapes, we recommend that you use the **label** command in the **Console** program since it will never overwrite a valid Bacula tape.

Testing your Tape Drive

To determine the best configuration of your tape drive, you can run the new **speed** command available in the **btape** program.

This command can have the following arguments:

file_size=n Specify the Maximum File Size for this test (between 1 and 5GB). This counter is in GB.

nb_file=n Specify the number of file to be written. The amount of data should be greater than your memory (*file_size * nb_file*).

skip_zero This flag permits to skip tests with constant data.

skip_random This flag permits to skip tests with random data.

skip_raw This flag permits to skip tests with raw access.

skip_block This flag permits to skip tests with Bacula block access.

```

*speed file_size=3 skip_raw
btape.c:1078 Test with zero data and bacula block structure.
btape.c:956 Begin writing 3 files of 3.221 GB with blocks of 129024 bytes.
+++++
btape.c:604 Wrote 1 EOF to "Drive-0" (/dev/nst0)
btape.c:406 Volume bytes=3.221 GB. Write rate = 44.128 MB/s
...
btape.c:383 Total Volume bytes=9.664 GB. Total Write rate = 43.531 MB/s

btape.c:1090 Test with random data, should give the minimum throughput.
btape.c:956 Begin writing 3 files of 3.221 GB with blocks of 129024 bytes.
+++++
btape.c:604 Wrote 1 EOF to "Drive-0" (/dev/nst0)
btape.c:406 Volume bytes=3.221 GB. Write rate = 7.271 MB/s

```



```

+++++
...
btape.c:383 Total Volume bytes=9.664 GB. Total Write rate = 7.365 MB/s

```

When using compression, the random test will give you the minimum throughput of your drive . The test using constant string will give you the maximum speed of your hardware chain. (cpu, memory, scsi card, cable, drive, tape).

You can change the block size in the Storage Daemon configuration file.

1.10 Other Programs

The following programs are general utility programs and in general do not need a configuration file nor a device name.

1.11 bsmtplib

bsmtplib is a simple mail transport program that permits more flexibility than the standard mail programs typically found on Unix systems. It can even be used on Windows machines.

It is called:

```

Usage: bsmtplib [-f from] [-h mailhost] [-s subject] [-c copy] [recipient ...]
  -c          set the Cc: field
  -dnn        set debug level to nn
  -f          set the From: field
  -h          use mailhost:port as the bsmtplib server
  -l          limit the lines accepted to nn
  -s          set the Subject: field
  -?          print this message.

```

If the **-f** option is not specified, **bsmtplib** will use your userid. If the option **-h** is not specified **bsmtplib** will use the value in the environment variable **bsmtplibSERVER** or if there is none **localhost**. By default port 25 is used.

If a line count limit is set with the **-l** option, **bsmtplib** will not send an email with a body text exceeding that number of lines. This is especially useful for large restore job reports where the list of files restored might produce very long mails your mail-server would refuse or crash. However, be aware that you will probably suppress the job report and any error messages unless you check the log file written by the Director (see the messages resource in this manual for details).

recipients is a space separated list of email recipients.

The body of the email message is read from standard input.

An example of the use of **bsmtplib** would be to put the following statement in the **Messages** resource of your **bacula-dir.conf** file. Note, these commands should appear on a single line each.

```

mailcommand = "/home/bacula/bin/bsmtplib -h mail.domain.com -f \"\\(Bacula\\) %r\"
               -s \"Bacula: %t %e of %c %l\" %r"
operatorcommand = "/home/bacula/bin/bsmtplib -h mail.domain.com -f \"\\(Bacula\\) %r\"
                  -s \"Bacula: Intervention needed for %j\" %r"

```

Where you replace **/home/bacula/bin** with the path to your **Bacula** binary directory, and you replace **mail.domain.com** with the fully qualified name of your bsmtplib (email) server, which normally listens on

port 25. For more details on the substitution characters (e.g. %r) used in the above line, please see the documentation of the MailCommand in the Messages Resource chapter of this manual.

It is **HIGHLY** recommended that you test one or two cases by hand to make sure that the **mailhost** that you specified is correct and that it will accept your email requests. Since **bsmtp** always uses a TCP connection rather than writing in the spool file, you may find that your **from** address is being rejected because it does not contain a valid domain, or because your message is caught in your spam filtering rules. Generally, you should specify a fully qualified domain name in the **from** field, and depending on whether your bsmtp gateway is Exim or Sendmail, you may need to modify the syntax of the from part of the message. Please test.

When running **bsmtp** by hand, you will need to terminate the message by entering a **ctl-d** in column 1 of the last line.

If you are getting incorrect dates (e.g. 1970) and you are running with a non-English language setting, you might try adding a **LANG="en_US"** immediately before the bsmtp call.

In general, **bsmtp** attempts to cleanup email addresses that you specify in the from, copy, mailhost, and recipient fields, by adding the necessary < and > characters around the address part. However, if you include a **display-name** (see RFC 5332), some SMTP servers such as Exchange may not accept the message if the **display-name** is also included in < and >. As mentioned above, you must test, and if you run into this situation, you may manually add the < and > to the Bacula **mailcommand** or **operatorcommand** and when **bsmtp** is formatting an address if it already contains a < or > character, it will leave the address unchanged.

1.12 dbcheck

dbcheck is a simple program that will search for logical inconsistencies in the Bacula tables in your database, and optionally fix them. It is a database maintenance routine, in the sense that it can detect and remove unused rows, but it is not a database repair routine. To repair a database, see the tools furnished by the database vendor. Normally dbcheck should never need to be run, but if Bacula has crashed or you have a lot of Clients, Pools, or Jobs that you have removed, it could be useful.

The **dbcheck** program can be found in the <**bacula-source**>/src/tools directory of the source distribution. Though it is built with the make process, it is not normally "installed".

It is called:

```
Usage: dbcheck [-c config ] [-B] [-C catalog name] [-d debug_level]
      <working-directory> <bacula-database> <user> <password> [<dbhost>] [<dbport>]
      -b                batch mode
      -C                catalog name in the director conf file
      -c                Director conf filename
      -B                print catalog configuration and exit
      -d <nn>          set debug level to <nn>
      -dt              print timestamp in debug output
      -f                fix inconsistencies
      -v                verbose
      -?                print this message
```

If the **-B** option is specified, dbcheck will print out catalog information in a simple text based format. This is useful to backup it in a secure way.

```
$ dbcheck -B
catalog=MyCatalog
db_type=SQLite
db_name=regress
db_driver=
db_user=regress
```

```
db_password=  
db_address=  
db_port=0  
db_socket=
```

If the **-c** option is given with the Director's conf file, there is no need to enter any of the command line arguments, in particular the working directory as **dbcheck** will read them from the file.

If the **-f** option is specified, **dbcheck** will repair (**fix**) the inconsistencies it finds. Otherwise, it will report only.

If the **-b** option is specified, **dbcheck** will run in batch mode, and it will proceed to examine and fix (if **-f** is set) all programmed inconsistency checks. If the **-b** option is not specified, **dbcheck** will enter interactive mode and prompt with the following:

```
Hello, this is the database check/correct program.  
Please select the function you want to perform.  
1) Toggle modify database flag  
2) Toggle verbose flag  
3) Repair bad Filename records  
4) Repair bad Path records  
5) Eliminate duplicate Filename records  
6) Eliminate duplicate Path records  
7) Eliminate orphaned Jobmedia records  
8) Eliminate orphaned File records  
9) Eliminate orphaned Path records  
10) Eliminate orphaned Filename records  
11) Eliminate orphaned FileSet records  
12) Eliminate orphaned Client records  
13) Eliminate orphaned Job records  
14) Eliminate all Admin records  
15) Eliminate all Restore records  
16) All (3-15)  
17) Quit  
Select function number:
```

By entering 1 or 2, you can toggle the modify database flag (**-f** option) and the verbose flag (**-v**). It can be helpful and reassuring to turn off the modify database flag, then select one or more of the consistency checks (items 3 through 9) to see what will be done, then toggle the modify flag on and re-run the check.

The inconsistencies examined are the following:

- Duplicate filename records. This can happen if you accidentally run two copies of Bacula at the same time, and they are both adding filenames simultaneously. It is a rare occurrence, but will create an inconsistent database. If this is the case, you will receive error messages during Jobs warning of duplicate database records. If you are not getting these error messages, there is no reason to run this check.
- Repair bad Filename records. This checks and corrects filenames that have a trailing slash. They should not.
- Repair bad Path records. This checks and corrects path names that do not have a trailing slash. They should.
- Duplicate path records. This can happen if you accidentally run two copies of Bacula at the same time, and they are both adding filenames simultaneously. It is a rare occurrence, but will create an inconsistent database. See the item above for why this occurs and how you know it is happening.
- Orphaned JobMedia records. This happens when a Job record is deleted (perhaps by a user issued SQL statement), but the corresponding JobMedia record (one for each Volume used in the Job) was not deleted. Normally, this should not happen, and even if it does, these records generally do not take much space in your database. However, by running this check, you can eliminate any such orphans.

- Orphaned File records. This happens when a Job record is deleted (perhaps by a user issued SQL statement), but the corresponding File record (one for each Volume used in the Job) was not deleted. Note, searching for these records can be **very** time consuming (i.e. it may take hours) for a large database. Normally this should not happen as Bacula takes care to prevent it. Just the same, this check can remove any orphaned File records. It is recommended that you run this once a year since orphaned File records can take a large amount of space in your database. You might want to ensure that you have indexes on JobId, FilenameId, and PathId for the File table in your catalog before running this command.
- Orphaned Path records. This condition happens any time a directory is deleted from your system and all associated Job records have been purged. During standard purging (or pruning) of Job records, Bacula does not check for orphaned Path records. As a consequence, over a period of time, old unused Path records will tend to accumulate and use space in your database. This check will eliminate them. It is recommended that you run this check at least once a year.
- Orphaned Filename records. This condition happens any time a file is deleted from your system and all associated Job records have been purged. This can happen quite frequently as there are quite a large number of files that are created and then deleted. In addition, if you do a system update or delete an entire directory, there can be a very large number of Filename records that remain in the catalog but are no longer used.
During standard purging (or pruning) of Job records, Bacula does not check for orphaned Filename records. As a consequence, over a period of time, old unused Filename records will accumulate and use space in your database. This check will eliminate them. It is strongly recommended that you run this check at least once a year, and for large database (more than 200 Megabytes), it is probably better to run this once every 6 months.
- Orphaned Client records. These records can remain in the database long after you have removed a client.
- Orphaned Job records. If no client is defined for a job or you do not run a job for a long time, you can accumulate old job records. This option allow you to remove jobs that are not attached to any client (and thus useless).
- All Admin records. This command will remove all Admin records, regardless of their age.
- All Restore records. This command will remove all Restore records, regardless of their age.

If you are using Mysql, dbcheck will ask you if you want to create temporary indexes to speed up orphaned Path and Filename elimination.

Mostly for PostgreSQL users, we provide a pure SQL script dbcheck replacement in `examples/database/dbcheck.sql` that works with global queries instead of many small queries like dbcheck. Execution instructions are at the top of the script and you will need to type `COMMIT` at the end to validate modifications.

If you are using bweb or brestore, don't eliminate orphaned Path, else you will have to rebuild `brestore_pathvisibility` and `brestore_pathhierarchy` indexes.

By the way, I personally run dbcheck only where I have messed up my database due to a bug in developing Bacula code, so normally you should never need to run dbcheck in spite of the recommendations given above, which are given so that users don't waste their time running dbcheck too often.

1.13 bregex

bregex is a simple program that will allow you to test regular expressions against a file of data. This can be useful because the regex libraries on most systems differ, and in addition, regex expressions can be complicated.

bregex is found in the `src/tools` directory and it is normally installed with your system binaries. To run it, use:

```
Usage: bregex [-d debug_level] -f <data-file>
        -f          specify file of data to be matched
        -l          suppress line numbers
        -n          print lines that do not match
        -?          print this message.
```

The <data-file> is a filename that contains lines of data to be matched (or not) against one or more patterns. When the program is run, it will prompt you for a regular expression pattern, then apply it one line at a time against the data in the file. Each line that matches will be printed preceded by its line number. You will then be prompted again for another pattern.

Enter an empty line for a pattern to terminate the program. You can print only lines that do not match by using the -n option, and you can suppress printing of line numbers with the -l option.

This program can be useful for testing regex expressions to be applied against a list of filenames.

1.14 bwild

bwild is a simple program that will allow you to test wild-card expressions against a file of data.

bwild is found in the src/tools directory and it is normally installed with your system binaries. To run it, use:

```
Usage: bwild [-d debug_level] -f <data-file>
        -f          specify file of data to be matched
        -l          suppress line numbers
        -n          print lines that do not match
        -?          print this message.
```

The <data-file> is a filename that contains lines of data to be matched (or not) against one or more patterns. When the program is run, it will prompt you for a wild-card pattern, then apply it one line at a time against the data in the file. Each line that matches will be printed preceded by its line number. You will then be prompted again for another pattern.

Enter an empty line for a pattern to terminate the program. You can print only lines that do not match by using the -n option, and you can suppress printing of line numbers with the -l option.

This program can be useful for testing wild expressions to be applied against a list of filenames.

1.15 testfind

testfind permits listing of files using the same search engine that is used for the **Include** resource in Job resources. Note, much of the functionality of this program (listing of files to be included) is present in the estimate command in the Console program.

The original use of testfind was to ensure that Bacula's file search engine was correct and to print some statistics on file name and path length. However, you may find it useful to see what bacula would do with a given **Include** resource. The **testfind** program can be found in the <bacula-source>/src/tools directory of the source distribution. Though it is built with the make process, it is not normally "installed".

It is called:

```
Usage: testfind [-d debug_level] [-] [pattern1 ...]
        -a          print extended attributes (Win32 debug)
        -dnn        set debug level to nn
```

```
-          read pattern(s) from stdin
-?         print this message.
Patterns are used for file inclusion -- normally directories.
Debug level>= 1 prints each file found.
Debug level>= 10 prints path/file for catalog.
Errors are always printed.
Files/paths truncated is a number with len> 255.
Truncation is only in the catalog.
```

Where a pattern is any filename specification that is valid within an **Include** resource definition. If none is specified, / (the root directory) is assumed. For example:

```
./testfind /bin
```

Would print the following:

```
Dir: /bin
Reg: /bin/bash
Lnk: /bin/bash2 -> bash
Lnk: /bin/sh -> bash
Reg: /bin/cpio
Reg: /bin/ed
Lnk: /bin/red -> ed
Reg: /bin/chgrp
...
Reg: /bin/ipcalc
Reg: /bin/usleep
Reg: /bin/aumix-minimal
Reg: /bin/mt
Lnka: /bin/gawk-3.1.0 -> /bin/gawk
Reg: /bin/pgawk
Total files      : 85
Max file length: 13
Max path length: 5
Files truncated: 0
Paths truncated: 0
```

Even though **testfind** uses the same search engine as **Bacula**, each directory to be listed, must be entered as a separate command line entry or entered one line at a time to standard input if the **-** option was specified.

Specifying a debug level of one (i.e. **-d1**) on the command line will cause **testfind** to print the raw filenames without showing the Bacula internal file type, or the link (if any). Debug levels of 10 or greater cause the filename and the path to be separated using the same algorithm that is used when putting filenames into the Catalog database.

1.16 bimagemgr

bimagemgr is a utility for those who backup to disk volumes in order to commit them to CDR disk, rather than tapes. It is a web based interface written in Perl and is used to monitor when a volume file needs to be burned to disk. It requires:

- A web server running on the bacula server
- A CD recorder installed and configured on the bacula server
- The cdrtools package installed on the bacula server.
- perl, perl-DBI module, and either DBD-MySQL DBD-SQLite or DBD-PostgreSQL modules

DVD burning is not supported by **bimagemgr** at this time, but both are planned for future releases.

1.16.1 bimagemgr installation

Installation from tarball: 1. Examine the Makefile and adjust it to your configuration if needed. 2. Edit config.pm to fit your configuration. 3. Do 'make install' as root. 4. Edit httpd.conf and change the Timeout value. The web server must not time out and close the connection before the burn process is finished. The exact value needed may vary depending upon your cd recorder speed and whether you are burning on the bacula server on on another machine across your network. In my case I set it to 1000 seconds. Restart httpd. 5. Make sure that cdrecord is setuid root.

Installation from rpm package: 1. Install the rpm package for your platform. 2. Edit /cgi-bin/config.pm to fit your configuration. 3. Edit httpd.conf and change the Timeout value. The web server must not time out and close the connection before the burn process is finished. The exact value needed may vary depending upon your cd recorder speed and whether you are burning on the bacula server on on another machine across your network. In my case I set it to 1000 seconds. Restart httpd. 4. Make sure that cdrecord is setuid root.

For bacula systems less than 1.36: 1. Edit the configuration section of config.pm to fit your configuration. 2. Run /etc/bacula/create_cdimage_table.pl from a console on your bacula server (as root) to add the CDImage table to your bacula database.

Accessing the Volume files: The Volume files by default have permissions 640 and can only be read by root. The recommended approach to this is as follows (and only works if bimagemgr and apache are running on the same host as bacula).

For bacula-1.34 or 1.36 installed from tarball - 1. Create a new user group bacula and add the user apache to the group for Red Hat or Mandrake systems. For SuSE systems add the user wwwrun to the bacula group. 2. Change ownership of all of your Volume files to root.bacula 3. Edit the /etc/bacula/bacula startup script and set SD_USER=root and SD_GROUP=bacula. Restart bacula.

Note: step 3 should also be done in /etc/init.d/bacula-sd but released versions of this file prior to 1.36 do not support it. In that case it would be necessary after a reboot of the server to execute '/etc/bacula/bacula restart'.

For bacula-1.38 installed from tarball - 1. Your configure statement should include: --with-dir-user=bacula --with-dir-group=bacula --with-sd-user=bacula --with-sd-group=disk --with-fd-user=root --with-fd-group=bacula 2. Add the user apache to the bacula group for Red Hat or Mandrake systems. For SuSE systems add the user wwwrun to the bacula group. 3. Check/change ownership of all of your Volume files to root.bacula

For bacula-1.36 or bacula-1.38 installed from rpm - 1. Add the user apache to the group bacula for Red Hat or Mandrake systems. For SuSE systems add the user wwwrun to the bacula group. 2. Check/change ownership of all of your Volume files to root.bacula

bimagemgr installed from rpm ; 1.38.9 will add the web server user to the bacula group in a post install script. Be sure to edit the configuration information in config.pm after installation of rpm package.

bimagemgr will now be able to read the Volume files but they are still not world readable.

If you are running bimagemgr on another host (not recommended) then you will need to change the permissions on all of your backup volume files to 644 in order to access them via nfs share or other means. This approach should only be taken if you are sure of the security of your environment as it exposes the backup Volume files to world read.

1.16.2 bimagemgr usage

Calling the program in your web browser, e.g. `http://localhost/cgi-bin/bimagemgr.pl` will produce a display as shown below in Figure 1. The program will query the bacula database and display all volume files with the date last written and the date last burned to disk. If a volume needs to be burned (last written is newer than last burn date) a "Burn" button will be displayed in the rightmost column.



Figure 1

Place a blank CDR disk in your recorder and click the "Burn" button. This will cause a pop up window as shown in Figure 2 to display the burn progress.



Figure 2

When the burn finishes the pop up window will display the results of `cdrecord` as shown in Figure 3. Close the pop up window and refresh the main window. The last burn date will be updated and the "Burn" button for that volume will disappear. Should you have a failed burn you can reset the last burn date of that volume by clicking its "Reset" link.



Figure 3

In the bottom row of the main display window are two more buttons labeled "Burn Catalog" and "Blank CDRW". "Burn Catalog" will place a copy of your bacula catalog on a disk. If you use CDRW disks rather than CDR then "Blank CDRW" allows you to erase the disk before re-burning it. Regularly committing your backup volume files and your catalog to disk with **bimagemgr** ensures that you can rebuild easily in the event of some disaster on the bacula server itself.

Chapter 2

Bacula RPM Packaging FAQ

1. How do I build Bacula for platform xxx?
2. How do I control which database support gets built?
3. What other defines are used?
4. I'm getting errors about not having permission when I try to build the packages. Do I need to be root?
5. I'm building my own rpms but on all platforms and compiles I get an unresolved dependency for something called /usr
6. I'm building my own rpms because you don't publish for my platform. Can I get my packages released to sourceforge
7. Is there an easier way than sorting out all these command line options?
8. I just upgraded from 1.36.x to 1.38.x and now my director daemon won't start. It appears to start but dies silently and
9. There are a lot of rpm packages. Which packages do I need for what?

2.1 Answers

1. **How do I build Bacula for platform xxx?** The bacula spec file contains defines to build for several platforms:

Red Hat 7.x (rh7), Red Hat 8.0 (rh8), Red Hat 9 (rh9),
Fedora Core (fc1, fc3, fc4, fc5, fc6, fc7, fc8),
Whitebox Enterprise Linux 3.0 (wb3),
Red Hat Enterprise Linux (rhel3, rhel4, rhel5),
Mandrake 10.x (mdk), Mandriva 2006.x (mdv),
CentOS (centos3, centos4, centos5)
Scientific Linux (sl3, sl4, sl5) and
SuSE (su9, su10, su102, su103, su110).

The package build is controlled by a mandatory define set at the beginning of the file. These defines basically just control the dependency information that gets coded into the finished rpm package as well as any special configure options required. The platform define may be edited in the spec file directly (by default all defines are set to 0 or "not set"). For example, to build the Red Hat 7.x package find the line in the spec file which reads

```
%define rh7 0
```

and edit it to read

```
%define rh7 1
```

Alternately you may pass the define on the command line when calling rpmbuild:

```
rpmbuild -ba --define "build_rh7 1" bacula.spec
rpmbuild --rebuild --define build_rh7 1" bacula-x.x.x-x.src.rpm
```

2. **How do I control which database support gets built?** Another mandatory build define controls which database support is compiled, one of build_sqlite, build_mysql or build_postgresql. To get the MySQL package and support either set the

```
%define mysql 0
OR
%define mysql4 0
OR
%define mysql5 0
```

to

```
%define mysql 1
OR
%define mysql4 1
OR
%define mysql5 1
```

in the spec file directly or pass it to rpmbuild on the command line:

```
rpmbuild -ba --define "build_rh7 1" --define "build_mysql 1" bacula.spec
rpmbuild -ba --define "build_rh7 1" --define "build_mysql4 1" bacula.spec
rpmbuild -ba --define "build_rh7 1" --define "build_mysql5 1" bacula.spec
```

3. **What other defines are used?**

Three other building defines of note are the depkgs.version, docs.version and _rescuever identifiers. These two defines are set with each release and must match the version of those sources that are being used to build the packages. You would not ordinarily need to edit these. See also the Build Options section below for other build time options that can be passed on the command line.

4. **I'm getting errors about not having permission when I try to build the packages. Do I need to be root?**

No, you do not need to be root and, in fact, it is better practice to build rpm packages as a non-root user. Bacula packages are designed to be built by a regular user but you must make a few changes on your system to do this. If you are building on your own system then the simplest method is to add write permissions for all to the build directory (/usr/src/redhat/, /usr/src/RPM or /usr/src/packages). To accomplish this, execute the following command as root:

```
chmod -R 777 /usr/src/redhat
chmod -R 777 /usr/src/RPM
chmod -R 777 /usr/src/packages
```

If you are working on a shared system where you can not use the method above then you need to recreate the appropriate above directory tree with all of its subdirectories inside your home directory. Then create a file named

.rpmmacros

in your home directory (or edit the file if it already exists) and add the following line:

```
%_topdir /home/myuser/redhat
%_tmppath /tmp
```

Another handy directive for the .rpmmacros file if you wish to suppress the creation of debug rpm packages is:

```
%debug_package %{nil}
```

5. **I'm building my own rpms but on all platforms and compiles I get an unresolved dependency for something called /usr/afsws/bin/pagsh.**

This is a shell from the OpenAFS (Andrew File System). If you are seeing this then you chose to include the docs/examples directory in your package. One of the example scripts in this directory is a pagsh script. Rpmbuild, when scanning for dependencies, looks at the shebang line of all packaged scripts in addition to checking shared libraries. To avoid this do not package the examples directory. If you are seeing this problem you are building a very old bacula package as the examples have been removed from the doc packaging.

6. **I'm building my own rpms because you don't publish for my platform. Can I get my packages released to sourceforge for other people to use?**

Yes, contributions from users are accepted and appreciated. Please examine the directory platforms/contrib-rpm in the source code for further information.

7. **Is there an easier way than sorting out all these command line options?**

Yes, there is a gui wizard shell script which you can use to rebuild the src rpm package. Look in the source archive for platforms/contrib-rpm/rpm_wizard.sh. This script will allow you to specify build options using GNOME dialog screens. It requires zenity.

8. **I just upgraded from 1.36.x to 1.38.x and now my director daemon won't start. It appears to start but dies silently and I get a "connection refused" error when starting the console. What is wrong?**

Beginning with 1.38 the rpm packages are configured to run the director and storage daemons as a non-root user. The file daemon runs as user root and group bacula, the storage daemon as user bacula and group disk, and the director as user bacula and group bacula. If you are upgrading you will need to change some file permissions for things to work. Execute the following commands as root:

```
chown bacula.bacula /var/bacula/*
chown root.bacula /var/bacula/bacula-fd.9102.state
chown bacula.disk /var/bacula/bacula-sd.9103.state
```

Further, if you are using File storage volumes rather than tapes those files will also need to have ownership set to user bacula and group bacula.

9. **There are a lot of rpm packages. Which packages do I need for what?**

For a bacula server you need to select the package based upon your preferred catalog database: one of bacula-mysql, bacula-postgresql or bacula-sqlite. If your system does not provide an mtz package you also need bacula-mtz to satisfy that dependency. For a client machine you need only install bacula-client. Optionally, for either server or client machines, you may install a graphical console bacula-gconsole and/or bacula-wxconsole. The Bacula Administration Tool is installed with the bacula-bat package. One last package, bacula-updatedb is required only when upgrading a server more than one database revision level.

10. **Support for RHEL3/4/5, CentOS 3/4/5, Scientific Linux 3/4/5 and x86_64**

The examples below show explicit build support for RHEL4 and CentOS 4. Build support for x86_64 has also been added.

Build with one of these 3 commands:

```
rpmbuild --rebuild \
--define "build_rhel4 1" \
--define "build_sqlite 1" \
bacula-1.38.3-1.src.rpm

rpmbuild --rebuild \
--define "build_rhel4 1" \
--define "build_postgresql 1" \
bacula-1.38.3-1.src.rpm

rpmbuild --rebuild \
--define "build_rhel4 1" \
```

```
--define "build_mysql4 1" \  
bacula-1.38.3-1.src.rpm
```

For CentOS substitute '--define "build_centos4 1"' in place of rhel4.
For Scientific Linux substitute '--define "build_sl4 1"' in place of rhel4.

For 64 bit support add '--define "build_x86_64 1"'

2.2 Build Options

The spec file currently supports building on the following platforms:

Red Hat builds

```
--define "build_rh7 1"  
--define "build_rh8 1"  
--define "build_rh9 1"
```

Fedora Core build

```
--define "build_fc1 1"  
--define "build_fc3 1"  
--define "build_fc4 1"  
--define "build_fc5 1"  
--define "build_fc6 1"  
--define "build_fc7 1"  
--define "build_fc8 1"  
--define "build_fc9 1"
```

Whitebox Enterprise build

```
--define "build_wb3 1"
```

Red Hat Enterprise builds

```
--define "build_rhel3 1"  
--define "build_rhel4 1"  
--define "build_rhel5 1"
```

CentOS build

```
--define "build_centos3 1"  
--define "build_centos4 1"  
--define "build_centos5 1"
```

Scientific Linux build

```
--define "build_sl3 1"  
--define "build_sl4 1"  
--define "build_sl5 1"
```

SuSE build

```
--define "build_su9 1"  
--define "build_su10 1"  
--define "build_su102 1"  
--define "build_su103 1"  
--define "build_su110 1"  
--define "build_su111 1"
```

Mandrake 10.x build

```
--define "build_mdk 1"
```

Mandriva build

```
--define "build_mdv 1"
```

MySQL support:

for mysql 3.23.x support define this

```
--define "build_mysql 1"
```

if using mysql 4.x define this,

currently: Mandrake 10.x, Mandriva 2006.0, SuSE 9.x & 10.0, FC4 & RHEL4

```
--define "build_mysql4 1"
```

if using mysql 5.x define this,

currently: SuSE 10.1 & FC5

```
--define "build_mysql5 1"
```

PostgreSQL support:

```
--define "build_postgresql 1"
```

```

Sqlite support:
--define "build_sqlite 1"

Build the client rpm only in place of one of the above database full builds:
--define "build_client_only 1"

X86-64 support:
--define "build_x86_64 1"

Supress build of bgnome-console:
--define "nobuild_gconsole 1"

Build the WXWindows console:
requires wxGTK >= 2.6
--define "build_wxconsole 1"

Build the Bacula Administration Tool:
requires QT >= 4.2
--define "build_bat 1"

Build python scripting support:
--define "build_python 1"

Modify the Packager tag for third party packages:
--define "contrib_packager Your Name <youremail@site.org>"

Install most files to /opt/bacula directory:
--define "single_dir_install 1"

Build the rescue files:
--define "build_rescue 1"

```

2.3 RPM Install Problems

In general the RPMs, once properly built should install correctly. However, when attempting to run the daemons, a number of problems can occur:

- Wrong /var/bacula Permissions

By default, the Director and Storage daemon do not run with root permission. If the /var/bacula is owned by root, then it is possible that the Director and the Storage daemon will not be able to access this directory, which is used as the Working Directory. To fix this, the easiest thing to do is:

```
chown bacula:bacula /var/bacula
```

Note: as of 1.38.8 /var/bacula is installed root:bacula with permissions 770.

- The Storage daemon cannot Access the Tape drive

This can happen in some older RPM releases where the Storage daemon ran under userid bacula, group bacula. There are two ways of fixing this: the best is to modify the /etc/init.d/bacula-sd file so that it starts the Storage daemon with group "disk". The second way to fix the problem is to change the permissions of your tape drive (usually /dev/nst0) so that Bacula can access it. You will probably need to change the permissions of the SCSI control device as well, which is usually /dev/sg0. The exact names depend on your configuration, please see the Tape Testing chapter for more information on devices.

Chapter 3

GNU Free Documentation License

Version 1.2, November 2002

Copyright ©2000,2001,2002 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject

(or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

General Index

- After bscan, 14
- Answers , 27
- Bacula® - RPM Packaging FAQ , 27
- Bcopy, 14
- Bcopy Command Options, 14
- Bextract, 8
- bimagemgr
 - Installation , 23
 - Usage , 24
- Bimagemgr , 23
- bimagemgr Installation , 23
- bimagemgr Usage , 24
- bls, 6
 - Listing Blocks, 7
 - Listing Jobs, 7
- bregex, 20
- Bscan
 - After, 14
- bscan, 10
- Bsmtp, 17
- Btape, 14
- Btape Commands, 15
- Build Options, 30
- bwild, 21
- Catalog
 - Using bscan to Compare a Volume to an existing, 12
- Commands
 - btape, 15
- Count
 - Using bscan to Correct the Volume File Count, 14
- Dbcheck, 18
- Drive
 - Using btape to Verify your Tape, 15
- Extracting From Multiple Volumes, 10
- Extracting With a Bootstrap File, 10
- Extracting with Include or Exclude Lists, 9
- FAQ
 - Bacula® - RPM Packaging , 27
- File
 - Extracting With a Bootstrap, 10
 - Specifying a Device Name For a, 5
- GNU Free Documentation License, 33
- License
 - GNU Free Documentation, 33
 - Listing Blocks with bls, 7
 - Listing Jobs with bls, 7
 - Lists
 - Extracting with Include or Exclude, 9
 - Options
 - bcopy Command, 14
 - Other Programs, 17
- program
 - bcopy, 14
 - bextract, 8
 - bls, 6
 - bregex, 20
 - bscan, 10
 - bsmtp, 17
 - btape, 14
 - bwild, 21
 - dbcheck, 18
 - testfind, 21
- Programs
 - Other, 17
- RPM Install Problems, 31
- Specifying a Device Name For a File, 5
- Specifying a Device Name For a Tape, 5
- Specifying the Configuration File, 5
- Specifying Volumes, 5
- Tape
 - Specifying a Device Name For a, 5
- Testfind, 21
- Tools
 - Volume Utility, 5
- Using bscan to Compare a Volume to an existing Catalog, 12
- Using bscan to Correct the Volume File Count, 14
- Using bscan to Recreate a Catalog from a Volume, 12
- Using btape to Verify your Tape Drive, 15
- Volume
 - Using bscan to Recreate a Catalog from a Volume, 12
- Volume Utility Tools, 5
- Volumes
 - Extracting From Multiple, 10
 - Specifying, 5