The Leading Open Source Backup Solution

Bacula® Console and Operators Guide

Kern Sibbald

August 14, 2015
This manual documents Bacula version 7.2.0 (12 August 2015)

Contents

Chapter 1

Bacula Console

The **Bacula Console** (sometimes called the User Agent) is a program that allows the user or the System Administrator, to interact with the Bacula Director daemon while the daemon is running.
The current Bacula Console comes in two versions: a shell interface (TTY style), and a QT GUI interface (Bat). Both permit the administrator or authorized users to interact with Bacula. You can determine the status of a particular job, examine the contents of the Catalog as well as perform certain tape manipulations with the Console program.
Since the Console program interacts with the Director through the network, your Console and Director programs do not necessarily need to run on the same machine.
In fact, a certain minimal knowledge of the Console program is needed in order for Bacula to be able to write on more than one tape, because when Bacula requests a new tape, it waits until the user, via the Console program, indicates that the new tape is mounted.

1.1 Console Configuration

When the Console starts, it reads a standard Bacula configuration file named **bconsole.conf** or **bat.conf** in the case of the Bat QT Console version from the current directory unless you specify the **-c** command line option (see below). This file allows default configuration of the Console, and at the current time, the only Resource Record defined is the Director resource, which gives the Console the name and address of the Director. For more information on configuration of the Console program, please see the **Console Configuration** chapter (chapter 20 on page 213) of the Bacula Community Main Manual.

1.2 Running the Console Program

The console program can be run with the following options:

```
Usage: bconsole [-s] [-c config_file] [-d debug_level]
       -c <file>   set configuration file to file
       -dnn        set debug level to nn
       -n          no conio
       -s          no signals
       -u <nn>     set command execution timeout to <nn> seconds
       -t          test - read configuration and exit
       -?          print this message.
```

After launching the Console program (bconsole), it will prompt you for the next command with an asterisk (*). Generally, for all commands, you can simply enter the command name and the Console program will prompt you for the necessary arguments. Alternatively, in most cases, you may enter the command followed by arguments. The general format is:

```
 <command> <keyword1>[=<argument1>] <keyword2>[=<argument2>] ...
```

where **command** is one of the commands listed below; **keyword** is one of the keywords listed below (usually followed by an argument); and **argument** is the value. The command may be abbreviated to the shortest unique form. If two commands have the same starting letters, the one that will be selected is the one that appears first in the **help** listing. If you want the second command, simply spell out the full command. None of the keywords following the command may be abbreviated.
For example:

```
list files jobid=23
```

will list all files saved for JobId 23. Or:

```
show pools
```

will display all the Pool resource records.
The maximum command line length is limited to 511 characters, so if you are scripting the console, you may need to take some care to limit the line length.

1.3 Stopping the Console Program

Normally, you simply enter **quit** or **exit** and the Console program will terminate. However, it waits until the Director acknowledges the command. If the Director is already doing a lengthy command (e.g. prune), it may take some time. If you want to immediately terminate the Console program, enter the **.quit** command. There is currently no way to interrupt a Console command once issued (i.e. Ctrl-C does not work). However, if you are at a prompt that is asking you to select one of several possibilities and you would like to abort the command, you can enter a period (**.**), and in most cases, you will either be returned to the main command prompt or if appropriate the previous prompt (in the case of nested prompts). In a few places such as where it is asking for a Volume name, the period will be taken to be the Volume name. In that case, you will most likely be able to cancel at the next prompt.

1.4 Alphabetic List of Console Keywords

Unless otherwise specified, each of the following keywords takes an argument, which is specified after the keyword following an equal sign. For example:

```
jobid=536
```

Please note, this list is incomplete as it is currently in the process of being created and is not currently totally in alphabetic order ...

restart Permitted on the python command, and causes the Python interpreter to be restarted. Takes no argument.

all Permitted on the status and show commands to specify all components or resources respectively.

allfrompool Permitted on the update command to specify that all Volumes in the pool (specified on the command line) should be updated.

allfrompools Permitted on the update command to specify that all Volumes in all pools should be updated.

before Used in the restore command.

bootstrap Used in the restore command.

catalog Allowed in the use command to specify the catalog name to be used.

catalogs Used in the show command. Takes no arguments.

client — fd

clients Used in the show, list, and llist commands. Takes no arguments.

counters Used in the show command. Takes no arguments.

current Used in the restore command. Takes no argument.

days Used to define the number of days the "list nextvol" command should consider when looking for jobs to be run. The days keyword can also be used on the "status dir" command so that it will display jobs scheduled for the number of days you want.

devices Used in the show command. Takes no arguments.

dir — director

directors Used in the show command. Takes no arguments.

directory Used in the restore command. Its argument specifies the directory to be restored.

enabled This keyword can appear on the **update volume** as well as the **update slots** commands, and can allows one of the following arguments: yes, true, no, false, archived, 0, 1, 2. Where 0 corresponds to no or false, 1 corresponds to yes or true, and 2 corresponds to archived. Archived volumes will not be used, nor will the Media record in the catalog be pruned. Volumes that are not enabled, will not be used for backup or restore.

done Used in the restore command. Takes no argument.

file Used in the restore command.

files Used in the list and llist commands. Takes no arguments.

fileset

filesets Used in the show command. Takes no arguments.

help Used in the show command. Takes no arguments.

jobs Used in the show, list and llist commands. Takes no arguments.

jobmedia Used in the list and llist commands. Takes no arguments.

jobtotals Used in the list and llist commands. Takes no arguments.

jobid The JobId is the numeric jobid that is printed in the Job Report output. It is the index of the database record for the given job. While it is unique for all the existing Job records in the catalog database, the same JobId can be reused once a Job is removed from the catalog. Probably you will refer specific Jobs that ran using their numeric JobId.

job — jobname The Job or Jobname keyword refers to the name you specified in the Job resource, and hence it refers to any number of Jobs that ran. It is typically useful if you want to list all jobs of a particular name.

level

listing Permitted on the estimate command. Takes no argument.

limit

messages Used in the show command. Takes no arguments.

media Used in the list and llist commands. Takes no arguments.

nextvol — nextvolume Used in the list and llist commands. Takes no arguments.

on Takes no keyword.

off Takes no keyword.

pool

pools Used in the show, list, and llist commands. Takes no arguments.

select Used in the restore command. Takes no argument.

limit Used in the setbandwidth command. Takes integer in KB/s unit.

storages Used in the show command. Takes no arguments.

schedules Used in the show command. Takes no arguments.

sd — store — storage

ujobid The ujobid is a unique job identification that is printed in the Job Report output. At the current time, it consists of the Job name (from the Name directive for the job) appended with the date and time the job was run. This keyword is useful if you want to completely identify the Job instance run.

volume

volumes Used in the list and llist commands. Takes no arguments.

where Used in the restore command.

yes Used in the restore command. Takes no argument.

1.5 Alphabetic List of Console Commands

The following commands are currently implemented:

add [pool=<pool-name> storage=<storage> jobid=<JobId>] This command is used to add Volumes to an existing Pool. That is, it creates the Volume name in the catalog and inserts into the Pool in the catalog, but does not attempt to access the physical Volume. Once added, Bacula expects that Volume to exist and to be labeled. This command is not normally used since Bacula will automatically do the equivalent when Volumes are labeled. However, there may be times when you have removed a Volume from the catalog and want to later add it back.

Normally, the **label** command is used rather than this command because the **label** command labels the physical media (tape, disk, DVD, ...) and does the equivalent of the **add** command. The **add** command affects only the Catalog and not the physical media (data on Volumes). The physical media must exist and be labeled before use (usually with the **label** command). This command can, however, be useful if you wish to add a number of Volumes to the Pool that will be physically labeled at a later time. It can also be useful if you are importing a tape from another site. Please see the **label** command below for the list of legal characters in a Volume name.

autodisplay on/off This command accepts **on** or **off** as an argument, and turns auto-display of messages on or off respectively. The default for the console program is **off**, which means that you will be notified when there are console messages pending, but they will not automatically be displayed.

When autodisplay is turned off, you must explicitly retrieve the messages with the **messages** command. When autodisplay is turned on, the messages will be displayed on the console as they are received.

automount on/off This command accepts **on** or **off** as the argument, and turns auto-mounting of the Volume after a **label** command on or off respectively. The default is **on**. If **automount** is turned off, you must explicitly **mount** tape Volumes after a label command to use it.

cancel [jobid=<number> job=<job-name> ujobid=<unique-jobid>] This command is used to cancel a job and accepts **jobid=nnn** or **job=xxx** as an argument where nnn is replaced by the JobId and xxx is replaced by the job name. If you do not specify a keyword, the Console program will prompt you with the names of all the active jobs allowing you to choose one.

Once a Job is marked to be canceled, it may take a bit of time (generally within a minute but up to two hours) before the Job actually terminates, depending on what operations it is doing. Don't be surprised that you receive a Job not found message. That just means that one of the three daemons had already canceled the job. Messages numbered in the 1000's are from the Director, 2000's are from the File daemon and 3000's from the Storage daemon.

create [pool=<pool-name>] This command is not normally used as the Pool records are automatically created by the Director when it starts based on what it finds in the conf file. If needed, this command can be to create a Pool record in the database using the Pool resource record defined in the Director's configuration file. So in a sense, this command simply transfers the information from the Pool resource in the configuration file into the Catalog. Normally this command is done automatically for you when the Director starts providing the Pool is referenced within a Job resource. If you use this command on an existing Pool, it will automatically update the Catalog to have the same information as the Pool resource. After creating a Pool, you will most likely use the **label** command to label one or more volumes and add their names to the Media database.

When starting a Job, if Bacula determines that there is no Pool record in the database, but there is a Pool resource of the appropriate name, it will create it for you. If you want the Pool record to appear in the database immediately, simply use this command to force it to be created.

delete [**volume=<vol-name> pool=<pool-name> job jobid=<id>**] The delete command is used to delete a Volume, Pool or Job record from the Catalog as well as all associated catalog Volume records that were created. This command operates only on the Catalog database and has no effect on the actual data written to a Volume. This command can be dangerous and we strongly recommend that you do not use it unless you know what you are doing.

If the keyword **Volume** appears on the command line, the named Volume will be deleted from the catalog, if the keyword **Pool** appears on the command line, a Pool will be deleted, and if the keyword **Job** appears on the command line, a Job and all its associated records (File and JobMedia) will be deleted from the catalog. The full form of this command is:

```
delete pool=<pool-name>
```

or

```
delete volume=<volume-name> pool=<pool-name>  or

delete JobId=<job-id> JobId=<job-id2> ...  or

delete Job JobId=n,m,o-r,t ...
```

The first form deletes a Pool record from the catalog database. The second form deletes a Volume record from the specified pool in the catalog database. The third form deletes the specified Job record from the catalog database. The last form deletes JobId records for JobIds n, m, o, p, q, r, and t. Where each one of the n,m,... is, of course, a number. That is a "delete jobid" accepts lists and ranges of jobids.

disable job<job-name> This command permits you to disable a Job for automatic scheduling. The job may have been previously enabled with the Job resource **Enabled** directive or using the console **enable** command. The next time the Director is restarted or the conf file is reloaded, the Enable/Disable state will be set to the value in the Job resource (default enabled) as defined in the bacula-dir.conf file.

enable job<job-name> This command permits you to enable a Job for automatic scheduling. The job may have been previously disabled with the Job resource **Enabled** directive or using the console **disable** command. The next time the Director is restarted or the conf file is reloaded, the Enable/Disable state will be set to the value in the Job resource (default enabled) as defined in the bacula-dir.conf file.

estimate Using this command, you can get an idea how many files will be backed up, or if you are unsure about your Include statements in your FileSet, you can test them without doing an actual backup. The default is to assume a Full backup. However, you can override this by specifying a **level=Incremental** or **level=Differential** on the command line. A Job name must be specified or you will be prompted for one, and optionally a Client and FileSet may be specified on the command line. It then contacts the client which computes the number of files and bytes that would be backed up. Please note that this is an estimate calculated from the number of blocks in the file rather than by reading the actual bytes. As such, the estimated backup size will generally be larger than an actual backup.

The `estimate` command can use the accurate code to detect changes and give a better estimation. You can set the accurate behavior on command line using `accurate=yes/no` or use the Job setting as default value.

Optionally you may specify the keyword **listing** in which case, all the files to be backed up will be listed. Note, it could take quite some time to display them if the backup is large. The full form is:

```
estimate job=<job-name> listing client=<client-name> accurate=<yes/no>
       fileset=<fileset-name> level=<level-name>
```

Specification of the **job** is sufficient, but you can also override the client, fileset, accurate and/or level by specifying them on the estimate command line.

As an example, you might do:

```
@output /tmp/listing
estimate job=NightlySave listing level=Incremental
@output
```

which will do a full listing of all files to be backed up for the Job **NightlySave** during an Incremental save and put it in the file **/tmp/listing**. Note, the byte estimate provided by this command is based on the file size contained in the directory item. This can give wildly incorrect estimates of the actual storage used if there are sparse files on your systems. Sparse files are often found on 64 bit systems for certain system files. The size that is returned is the size Bacula will backup if the sparse option is not specified in the FileSet. There is currently no way to get an estimate of the real file size that would be found should the sparse option be enabled.

exit This command terminates the console program.

gui Invoke the non-interactive gui mode.

```
gui [on|off]
```

help This command displays the list of commands available.

label This command is used to label physical volumes. The full form of this command is:

```
label storage=<storage-name> volume=<volume-name>
      slot=<slot>
```

If you leave out any part, you will be prompted for it. The media type is automatically taken from the Storage resource definition that you supply. Once the necessary information is obtained, the Console program contacts the specified Storage daemon and requests that the Volume be labeled. If the Volume labeling is successful, the Console program will create a Volume record in the appropriate Pool.

The Volume name is restricted to letters, numbers, and the special characters hyphen (**-**), underscore (_), colon (**:**), and period (**.**). All other characters including a space are invalid. This restriction is to ensure good readability of Volume names to reduce operator errors.

Please note, when labeling a blank tape, Bacula will get **read I/O error** when it attempts to ensure that the tape is not already labeled. If you wish to avoid getting these messages, please write an EOF mark on your tape before attempting to label it:

```
mt rewind
mt weof
```

The label command can fail for a number of reasons:

1. The Volume name you specify is already in the Volume database.
2. The Storage daemon has a tape or other Volume already mounted on the device, in which case you must **unmount** the device, insert a blank tape, then do the **label** command.
3. The Volume in the device is already a Bacula labeled Volume. (Bacula will never relabel a Bacula labeled Volume unless it is recycled and you use the **relabel** command).
4. There is no Volume in the drive.

There are two ways to relabel a volume that already has a Bacula label. The brute force method is to write an end of file mark on the tape using the system **mt** program, something like the following:

```
mt -f /dev/st0 rewind
mt -f /dev/st0 weof
```

For a disk volume, you would manually delete the Volume.

Then you use the **label** command to add a new label. However, this could leave traces of the old volume in the catalog.

The preferable method to relabel a Volume is to first **purge** the volume, either automatically, or explicitly with the **purge** command, then use the **relabel** command described below.

If your autochanger has barcode labels, you can label all the Volumes in your autochanger one after another by using the **label barcodes** command. For each tape in the changer containing a barcode, Bacula will mount the tape and then label it with the same name as the barcode. An appropriate Media record will also be created in the catalog. Any barcode that begins with the same characters as specified on the "CleaningPrefix=xxx" directive in the Director's Pool resource, will be treated as a cleaning tape, and will not be labeled. However, an entry for the cleaning tape will be created in the catalog. For example with:

```
Pool {
  Name ...
  Cleaning Prefix = "CLN"
}
```

Any slot containing a barcode of CLNxxxx will be treated as a cleaning tape and will not be mounted. Note, the full form of the command is:

```
label storage=xxx pool=yyy slots=1-5,10 barcodes
```

list The list command lists the requested contents of the Catalog. The various fields of each record are listed on a single line. The various forms of the list command are:

```
list jobs

list jobid=<id>           (list jobid id)

list ujobid=<unique job name> (list job with unique name)

list job=<job-name>   (list all jobs with "job-name")

list jobname=<job-name>  (same as above)

    In the above, you can add "limit=nn" to limit the output to
    nn jobs.

list joblog jobid=<id> (list job output if recorded in the catalog)

list jobmedia

list jobmedia jobid=<id>

list jobmedia job=<job-name>

list files jobid=<id>

list files job=<job-name>

list pools

list clients

list jobtotals

list volumes

list volumes jobid=<id>

list volumes pool=<pool-name>

list volumes job=<job-name>

list volume=<volume-name>

list nextvolume job=<job-name>

list nextvol job=<job-name>

list nextvol job=<job-name> days=nnn
```

What most of the above commands do should be more or less obvious. In general if you do not specify all the command line arguments, the command will prompt you for what is needed.

The **list nextvol** command will print the Volume name to be used by the specified job. You should be aware that exactly what Volume will be used depends on a lot of factors including the time and what a prior job will do. It may fill a tape that is not full when you issue this command. As a consequence, this command will give you a good estimate of what Volume will be used but not a definitive answer. In addition, this command may have certain side effect because it runs through the same algorithm as a job, which means it may automatically purge or recycle a Volume. By default, the job specified must run within the next two days or no volume will be found. You can, however, use the **days=nnn**

specification to specify up to 50 days. For example, if on Friday, you want to see what Volume will be needed on Monday, for job MyJob, you would use **list nextvol job=MyJob days=3**.

If you wish to add specialized commands that list the contents of the catalog, you can do so by adding them to the **query.sql** file. However, this takes some knowledge of programming SQL. Please see the **query** command below for additional information. See below for listing the full contents of a catalog record with the **llist** command.

As an example, the command **list pools** might produce the following output:

```
+------+---------+---------+---------+----------+-------------+
| PoId | Name    | NumVols | MaxVols | PoolType | LabelFormat |
+------+---------+---------+---------+----------+-------------+
|    1 | Default |       0 |       0 | Backup   | *           |
|    2 | Recycle |       0 |       8 | Backup   | File        |
+------+---------+---------+---------+----------+-------------+
```

As mentioned above, the **list** command lists what is in the database. Some things are put into the database immediately when Bacula starts up, but in general, most things are put in only when they are first used, which is the case for a Client as with Job records, etc.

Bacula should create a client record in the database the first time you run a job for that client. Doing a **status** will not cause a database record to be created. The client database record will be created whether or not the job fails, but it must at least start. When the Client is actually contacted, additional info from the client will be added to the client record (a "uname -a" output).

If you want to see what Client resources you have available in your conf file, you use the Console command **show clients**.

llist The llist or "long list" command takes all the same arguments that the list command described above does. The difference is that the llist command list the full contents of each database record selected. It does so by listing the various fields of the record vertically, with one field per line. It is possible to produce a very large number of output lines with this command.

If instead of the **list pools** as in the example above, you enter **llist pools** you might get the following output:

```
          PoolId: 1
            Name: Default
         NumVols: 0
         MaxVols: 0
         UseOnce: 0
      UseCatalog: 1
 AcceptAnyVolume: 1
    VolRetention: 1,296,000
  VolUseDuration: 86,400
      MaxVolJobs: 0
     MaxVolBytes: 0
       AutoPrune: 0
         Recycle: 1
        PoolType: Backup
     LabelFormat: *

          PoolId: 2
            Name: Recycle
         NumVols: 0
         MaxVols: 8
         UseOnce: 0
      UseCatalog: 1
 AcceptAnyVolume: 1
    VolRetention: 3,600
  VolUseDuration: 3,600
      MaxVolJobs: 1
     MaxVolBytes: 0
       AutoPrune: 0
         Recycle: 1
        PoolType: Backup
     LabelFormat: File
```

messages This command causes any pending console messages to be immediately displayed.

memory Print current memory usage.

mount The mount command is used to get Bacula to read a volume on a physical device. It is a way to tell Bacula that you have mounted a tape and that Bacula should examine the tape. This command is normally used only after there was no Volume in a drive and Bacula requests you to mount a new Volume or when you have specifically unmounted a Volume with the **unmount** console command, which causes Bacula to close the drive. If you have an autoloader, the mount command will not cause Bacula to operate the autoloader unless you specify a **slot** and possibly a **drive**. The various forms of the mount command are:

mount storage=<storage-name> [slot=<num>] [drive=<num>]

mount [jobid=<id> — job=<job-name>]

If you have specified **Automatic Mount = yes** in the Storage daemon's Device resource, under most circumstances, Bacula will automatically access the Volume unless you have explicitly **unmount**ed it in the Console program.

prune The Prune command allows you to safely remove expired database records from Jobs, Volumes and Statistics. This command works only on the Catalog database and does not affect data written to Volumes. In all cases, the Prune command applies a retention period to the specified records. You can Prune expired File entries from Job records; you can Prune expired Job records from the database, and you can Prune both expired Job and File records from specified Volumes.

prune files—jobs—volume—stats client=<client-name> volume=<volume-name>

For a Volume to be pruned, the **VolStatus** must be Full, Used, or Append, otherwise the pruning will not take place.

purge The Purge command will delete associated Catalog database records from Jobs and Volumes without considering the retention period. **Purge** works only on the Catalog database and does not affect data written to Volumes. This command can be dangerous because you can delete catalog records associated with current backups of files, and we recommend that you do not use it unless you know what you are doing. The permitted forms of **purge** are:

purge files jobid=<jobid>—job=<job-name>—client=<client-name>

purge jobs client=<client-name> (of all jobs)

purge volume—volume=<vol-name> (of all jobs)

For the **purge** command to work on Volume Catalog database records the **VolStatus** must be Append, Full, Used, or Error.

The actual data written to the Volume will be unaffected by this command unless you are using the `ActionOnPurge=Truncate` option on those Media.

To ask Bacula to truncate your `Purged` volumes, you need to use the following command in interactive mode or in a RunScript:

```
*purge volume action=truncate storage=File allpools
# or by default, action=all
*purge volume action storage=File pool=Default
```

This is possible to specify the volume name, the media type, the pool, the storage, etc... (see `help purge`) Be sure that your storage device is idle when you decide to run this command.

python The python command takes a single argument **restart**:

python restart

This causes the Python interpreter in the Director to be reinitialized. This can be helpful for testing because once the Director starts and the Python interpreter is initialized, there is no other way to make it accept any changes to the startup script **DirStartUp.py**. For more details on Python scripting, please see the **Python Scripting** chapter (chapter 1 on page 1) of the Bacula Community Misc Manual.

query This command reads a predefined SQL query from the query file (the name and location of the query file is defined with the QueryFile resource record in the Director's configuration file). You are prompted to select a query from the file, and possibly enter one or more parameters, then the command is submitted to the Catalog database SQL engine.

The following queries are currently available (version 2.2.7):

```
Available queries:
1: List up to 20 places where a File is saved regardless of the directory
2: List where the most recent copies of a file are saved
3: List last 20 Full Backups for a Client
4: List all backups for a Client after a specified time
5: List all backups for a Client
6: List Volume Attributes for a selected Volume
7: List Volumes used by selected JobId
8: List Volumes to Restore All Files
9: List Pool Attributes for a selected Pool
10: List total files/bytes by Job
11: List total files/bytes by Volume
12: List Files for a selected JobId
13: List Jobs stored on a selected MediaId
14: List Jobs stored for a given Volume name
15: List Volumes Bacula thinks are in changer
16: List Volumes likely to need replacement from age or errors
Choose a query (1-16):
```

quit This command terminates the console program. The console program sends the **quit** request to the Director and waits for acknowledgment. If the Director is busy doing a previous command for you that has not terminated, it may take some time. You may quit immediately by issuing the **.quit** command (i.e. quit preceded by a period).

relabel This command is used to label physical volumes. The full form of this command is:

relabel storage=<storage-name> oldvolume=<old-volume-name> volume=<newvolume-name>

If you leave out any part, you will be prompted for it. In order for the Volume (old-volume-name) to be relabeled, it must be in the catalog, and the volume status must be marked **Purged** or **Recycle**. This happens automatically as a result of applying retention periods, or you may explicitly purge the volume using the **purge** command.

Once the volume is physically relabeled, the old data previously written on the Volume is lost and cannot be recovered.

release This command is used to cause the Storage daemon to rewind (release) the current tape in the drive, and to re-read the Volume label the next time the tape is used.

release storage=<storage-name>

After a release command, the device is still kept open by Bacula (unless Always Open is set to No in the Storage Daemon's configuration) so it cannot be used by another program. However, with some tape drives, the operator can remove the current tape and to insert a different one, and when the next Job starts, Bacula will know to re-read the tape label to find out what tape is mounted. If you want to be able to use the drive with another program (e.g. **mt**), you must use the **unmount** command to cause Bacula to completely release (close) the device.

reload The reload command causes the Director to re-read its configuration file and apply the new values. The new values will take effect immediately for all new jobs. However, if you change schedules, be aware that the scheduler pre-schedules jobs up to two hours in advance, so any changes that are to take place during the next two hours may be delayed. Jobs that have already been scheduled to run (i.e. surpassed their requested start time) will continue with the old values. New jobs will use the new values. Each time you issue a reload command while jobs are running, the prior config values will queued until all jobs that were running before issuing the reload terminate, at which time the old config values will be released from memory. The Directory permits keeping up to ten prior set of configurations before it will refuse a reload command. Once at least one old set of config values has been released it will again accept new reload commands.

While it is possible to reload the Director's configuration on the fly, even while jobs are executing, this is a complex operation and not without side effects. Accordingly, if you have to reload the Director's configuration while Bacula is running, it is advisable to restart the Director at the next convenient opportunity.

restore The restore command allows you to select one or more Jobs (JobIds) to be restored using various methods. Once the JobIds are selected, the File records for those Jobs are placed in an internal Bacula directory tree, and the restore enters a file selection mode that allows you to interactively walk up and down the file tree selecting individual files to be restored. This mode is somewhat similar to the standard Unix **restore** program's interactive file selection mode.

restore storage=<storage-name> client=<backup-client-name> where=<path> pool=<pool-name> fileset=<fileset-name> restoreclient=<restore-client-name> restorejob=<job-name> select current all done

Where **current**, if specified, tells the restore command to automatically select a restore to the most current backup. If not specified, you will be prompted. The **all** specification tells the restore command to restore all files. If it is not specified, you will be prompted for the files to restore. For details of the **restore** command, please see the **Restore** chapter (chapter 22 on page 221) of the Bacula Community Main Manual.

The client keyword initially specifies the client from which the backup was made and the client to which the restore will be make. However, if the restoreclient keyword is specified, then the restore is written to that client.

The restore job rarely needs to be specified, as bacula installations commonly only have a single restore job configured. However, for certain cases, such as a varying list of RunScript specifications, multiple restore jobs may be configured. The restorejob argument allows the selection of one of these jobs.

run This command allows you to schedule jobs to be run immediately. The full form of the command is:

run job=<job-name> client=<client-name> fileset=<FileSet-name> level=<level-keyword> storage=<storage-name> where=<directory-prefix> when=<universal-time-specification> spool-data=yes—no yes

Any information that is needed but not specified will be listed for selection, and before starting the job, you will be prompted to accept, reject, or modify the parameters of the job to be run, unless you have specified **yes**, in which case the job will be immediately sent to the scheduler.

On my system, when I enter a run command, I get the following prompt:

```
A job name must be specified.
The defined Job resources are:
     1: Matou
     2: Polymatou
     3: Rufus
     4: Minimatou
     5: Minou
     6: PmatouVerify
     7: MatouVerify
     8: RufusVerify
     9: Watchdog
Select Job resource (1-9):
```

If I then select number 5, I am prompted with:

```
Run Backup job
JobName:  Minou
FileSet:  Minou Full Set
Level:    Incremental
Client:   Minou
Storage:  DLTDrive
Pool:     Default
When:     2003-04-23 17:08:18
OK to run? (yes/mod/no):
```

If I now enter **yes**, the Job will be run. If I enter **mod**, I will be presented with the following prompt.

```
Parameters to modify:
     1: Level
     2: Storage
     3: Job
     4: FileSet
     5: Client
     6: When
     7: Pool
Select parameter to modify (1-7):
```

If you wish to start a job at a later time, you can do so by setting the When time. Use the **mod** option and select **When** (no. 6). Then enter the desired start time in YYYY-MM-DD HH:MM:SS format.

The spooldata argument of the run command cannot be modified through the menu and is only accessible by setting its value on the intial command line. If no spooldata flag is set, the job, storage or schedule flag is used.

setbandwidth This command is used to limit the bandwidth of a running job or a client.

setbandwidth limit=¡nb¿ [jobid=¡id¿ — client=¡cli¿]

setdebug This command is used to set the debug level in each daemon. The form of this command is:

setdebug level=nn [trace=0/1 client=<client-name> — dir — director — storage=<storage-name> — all]

If trace=1 is set, then tracing will be enabled, and the daemon will be placed in trace mode, which means that all debug output as set by the debug level will be directed to the file **bacula.trace** in the current directory of the daemon. Normally, tracing is needed only for Win32 clients where the debug output cannot be written to a terminal or redirected to a file. When tracing, each debug output message is appended to the trace file. You must explicitly delete the file when you are done.

setip Sets new client address – if authorized.

A console is authorized to use the **SetIP** command only if it has a Console resource definition in both the Director and the Console. In addition, if the console name, provided on the **Name =** directive, must be the same as a Client name, the user of that console is permitted to use the **SetIP** command to change the Address directive in the Director's client resource to the IP address of the Console. This permits portables or other machines using DHCP (non-fixed IP addresses) to "notify" the Director of their current IP address.

show The show command will list the Director's resource records as defined in the Director's configuration file (normally **bacula-dir.conf**). This command is used mainly for debugging purposes by developers. The following keywords are accepted on the show command line: catalogs, clients, counters, devices, directors, filesets, jobs, messages, pools, schedules, storages, all, help. Please don't confuse this command with the **list**, which displays the contents of the catalog.

sqlquery The sqlquery command puts the Console program into SQL query mode where each line you enter is concatenated to the previous line until a semicolon (;) is seen. The semicolon terminates the command, which is then passed directly to the SQL database engine. When the output from the SQL engine is displayed, the formation of a new SQL command begins. To terminate SQL query mode and return to the Console command prompt, you enter a period (.) in column 1.

Using this command, you can query the SQL catalog database directly. Note you should really know what you are doing otherwise you could damage the catalog database. See the **query** command below for simpler and safer way of entering SQL queries.

Depending on what database engine you are using (MySQL, PostgreSQL or SQLite), you will have somewhat different SQL commands available. For more detailed information, please refer to the MySQL, PostgreSQL or SQLite documentation.

status This command will display the status of all components. For the director, it will display the next jobs that are scheduled during the next 24 hours as well as the status of currently running jobs. For the Storage Daemon, you will have drive status or autochanger content. The File Daemon will give you information about current jobs like average speed or file accounting. The full form of this command is:

status [all — dir=<dir-name> — director [days=nnn] — client=<client-name> — [slots] storage=<storage-name>]

If you do a **status dir**, the console will list any currently running jobs, a summary of all jobs scheduled to be run in the next 24 hours, and a listing of the last ten terminated jobs with their statuses. The scheduled jobs summary will include the Volume name to be used. You should be aware of two things: 1. to obtain the volume name, the code goes through the same code that will be used when the job runs, but it does not do pruning nor recycling of Volumes; 2. The Volume listed is at best a guess. The Volume actually used may be different because of the time difference (more durations may expire when the job runs) and another job could completely fill the Volume requiring a new one.

In the Running Jobs listing, you may find the following types of information:

```
2507 Catalog MatouVerify.2004-03-13_05.05.02 is waiting execution
5349 Full    CatalogBackup.2004-03-13_01.10.00 is waiting for higher
             priority jobs to finish
5348 Differe Minou.2004-03-13_01.05.09 is waiting on max Storage jobs
5343 Full    Rufus.2004-03-13_01.05.04 is running
```

Looking at the above listing from bottom to top, obviously JobId 5343 (Rufus) is running. JobId 5348 (Minou) is waiting for JobId 5343 to finish because it is using the Storage resource, hence the "waiting on max Storage jobs". JobId 5349 has a lower priority than all the other jobs so it is waiting for higher priority jobs to finish, and finally, JobId 2507 (MatouVerify) is waiting because only one job can run at a time, hence it is simply "waiting execution"

If you do a **status dir**, it will by default list the first occurrence of all jobs that are scheduled today and tomorrow. If you wish to see the jobs that are scheduled in the next three days (e.g. on Friday you want to see the first occurrence of what tapes are scheduled to be used on Friday, the weekend, and Monday), you can add the **days=3** option. Note, a **days=0** shows the first occurrence of jobs scheduled today only. If you have multiple run statements, the first occurrence of each run statement for the job will be displayed for the period specified.

If your job seems to be blocked, you can get a general idea of the problem by doing a **status dir**, but you can most often get a much more specific indication of the problem by doing a **status storage=xxx**. For example, on an idle test system, when we do **status storage=File**, we get:

```
status storage=File
Connecting to Storage daemon File at 192.168.68.112:8103

rufus-sd Version: 1.39.6 (24 March 2006) i686-pc-linux-gnu redhat (Stentz)
Daemon started 26-Mar-06 11:06, 0 Jobs run since started.

Running Jobs:
No Jobs running.
====

Jobs waiting to reserve a drive:
====

Terminated Jobs:
 JobId  Level    Files          Bytes Status   Finished        Name
===================================================================
    59  Full        234      4,417,599 OK       15-Jan-06 11:54 kernsave
====

Device status:
Autochanger "DDS-4-changer" with devices:
   "DDS-4" (/dev/nst0)
Device "DDS-4" (/dev/nst0) is mounted with Volume="TestVolume002"
Pool="*unknown*"
    Slot 2 is loaded in drive 0.
    Total Bytes Read=0 Blocks Read=0 Bytes/block=0
    Positioned at File=0 Block=0

Device "DVD-Writer" (/dev/hdc) is not open.
Device "File" (/tmp) is not open.
====

In Use Volume status:
====
```

Now, what this tells us is that no jobs are running and that none of the devices are in use. Now, if we **unmount** the autochanger, which will not be used in this example, and then start a Job that uses the File device, the job will block. When we re-issue the status storage command, we get for the Device status:

```
status storage=File
...
Device status:
Autochanger "DDS-4-changer" with devices:
   "DDS-4" (/dev/nst0)
Device "DDS-4" (/dev/nst0) is not open.
    Device is BLOCKED. User unmounted.
    Drive 0 is not loaded.
```

```
Device "DVD-Writer" (/dev/hdc) is not open.
Device "File" (/tmp) is not open.
    Device is BLOCKED waiting for media.
====
...
```

Now, here it should be clear that if a job were running that wanted to use the Autochanger (with two devices), it would block because the user unmounted the device. The real problem for the Job I started using the "File" device is that the device is blocked waiting for media – that is Bacula needs you to label a Volume.

If you enter **status storage**, Bacula will prompt you with a list of the storage resources. When you select one, the Storage daemon will be requested to do a **status**. However, note that the Storage daemon will do a status of all the devices it has, and not just of the one you requested. In the current version of Bacula, when you enter the **status storage** command, it prompts you only with a subset of all the storage resources that the Director considers to be in different Storage daemons. In other words, it attempts to remove duplicate storage definitions. This can be a bit confusing at first, but can vastly simplify the promt listing if you have defined a large number of storage resources.

If you prefer to see the full list of all storage resources, simply add the keyword **select** to the command such as: **status select storage** and you will get a prompt that includes all storage resources even if they reference the same storage daemon.

time Prints the current time.

trace Turn on/off trace to file.

umount For old-time Unix guys. See the unmount command for full details.

unmount This command causes the indicated Bacula Storage daemon to unmount the specified device. The forms of the command are the same as the mount command:

```
unmount storage=<storage-name> [ drive=<num> ]

unmount [ jobid=<id> | job=<job-name> ]
```

Once you unmount a storage device, Bacula will no longer be able to use it until you issue a mount command for that device. If Bacula needs to access that device, it will block and issue mount requests periodically to the operator.

If the device you are unmounting is an autochanger, it will unload the drive you have specified on the command line. If no drive is specified, it will assume drive 1.

update This command will update the catalog for either a specific Pool record, a Volume record, or the Slots in an autochanger with barcode capability. In the case of updating a Pool record, the new information will be automatically taken from the corresponding Director's configuration resource record. It can be used to increase the maximum number of volumes permitted or to set a maximum number of volumes. The following main keywords may be specified:

```
media, volume, pool, slots, stats
```

In the case of updating a Volume, you will be prompted for which value you wish to change. The following Volume parameters may be changed:

```
Volume Status
Volume Retention Period
Volume Use Duration
Maximum Volume Jobs
Maximum Volume Files
Maximum Volume Bytes
Recycle Flag
Recycle Pool
Slot
InChanger Flag
Pool
Volume Files
Volume from Pool
All Volumes from Pool
All Volumes from all Pools
```

For slots **update slots**, Bacula will obtain a list of slots and their barcodes from the Storage daemon, and for each barcode found, it will automatically update the slot in the catalog Media record to correspond to the new value. This is very useful if you have moved cassettes in the magazine, or if you have removed the magazine and inserted a different one. As the slot of each Volume is updated, the InChanger flag for that Volume will also be set, and any other Volumes in the Pool that were last mounted on the same Storage device will have their InChanger flag turned off. This permits Bacula to know what magazine (tape holder) is currently in the autochanger.

If you do not have barcodes, you can accomplish the same thing in version 1.33 and later by using the **update slots scan** command. The **scan** keyword tells Bacula to physically mount each tape and to read its VolumeName.

For Pool **update pool**, Bacula will move the Volume record from its existing pool to the pool specified.

For **Volume from Pool**, **All Volumes from Pool** and **All Volumes from all Pools**, the following values are updated from the Pool record: Recycle, RecyclePool, VolRetention, VolUseDuration, MaxVolJobs, MaxVolFiles, and MaxVolBytes. (RecyclePool feature is available with bacula 2.1.4 or higher.)

The full form of the update command with all command line arguments is:

```
update volume=xxx pool=yyy slots volstatus=xxx VolRetention=ddd
  VolUse=ddd MaxVolJobs=nnn MaxVolBytes=nnn Recycle=yes|no
  slot=nnn enabled=n recyclepool=zzz
```

use This command allows you to specify which Catalog database to use. Normally, you will be using only one database so this will be done automatically. In the case that you are using more than one database, you can use this command to switch from one to another.

```
use [catalog=name-of-catalog]
```

var This command takes a string or quoted string and does variable expansion on it the same way variable expansion is done on the **LabelFormat** string. Thus, for the most part, you can test your LabelFormat strings. The difference between the **var** command and the actual LabelFormat process is that during the var command, no job is running so "dummy" values are used in place of Job specific variables. Generally, however, you will get a good idea of what is going to happen in the real case.

version The command prints the Director's version.

wait The wait command causes the Director to pause until there are no jobs running. This command is useful in a batch situation such as regression testing where you wish to start a job and wait until that job completes before continuing. This command now has the following options:

```
wait [jobid=nn] [jobuid=unique id] [job=job name]
```

If specified with a specific JobId, ... the wait command will wait for that particular job to terminate before continuing.

1.6 Special dot Commands

There is a list of commands that are prefixed with a period (.). These commands are intended to be used either by batch programs or graphical user interface front-ends. They are not normally used by interactive users. Once GUI development begins, this list will be considerably expanded. The following is the list of dot commands:

```
.backups job=xxx      list backups for specified job
.clients              list all client names
.defaults client=xxx fileset=yyy  list defaults for specified client
.die                  cause the Director to segment fault (for debugging)
.dir                  when in tree mode prints the equivalent to the dir command,
                        but with fields separated by commas rather than spaces.
.exit                 quit
.filesets             list all fileset names
.help                 help command output
.jobs                 list all job names
```

```
.levels              list all levels
.messages            get quick messages
.msgs                return any queued messages
.pools               list all pool names
.quit                quit
.status              get status output
.storage             return storage resource names
.types               list job types
```

1.7 Special At (@) Commands

Normally, all commands entered to the Console program are immediately forwarded to the Director, which may be on another machine, to be executed. However, there is a small list of **at** commands, all beginning with an at character (@), that will not be sent to the Director, but rather interpreted by the Console program directly. Note, these commands are implemented only in the tty console program and not in the Bat Console. These commands are:

@input <**filename**> Read and execute the commands contained in the file specified.

@output <**filename**> **w/a** Send all following output to the filename specified either overwriting the file (w) or appending to the file (a). To redirect the output to the terminal, simply enter **@output** without a filename specification. WARNING: be careful not to overwrite a valid file. A typical example during a regression test might be:

```
@output /dev/null
commands ...
@output
```

@tee <**filename**> **w/a** Send all subsequent output to both the specified file and the terminal. It is turned off by specifying **@tee** or **@output** without a filename.

@sleep <**seconds**> Sleep the specified number of seconds.

@time Print the current time and date.

@version Print the console's version.

@quit quit

@exit quit

@# anything Comment

@help Get the list of every special @ commands.

@separator <**char**> When using bconsole with readline, you can set the command separator to one of those characters to write commands who require multiple input on one line, or to put multiple commands on a single line.

```
!$%&'()*+,-/:;<>?[]^`{|}~
```

Note, if you use a semicolon (;) as a separator character, which is common, you will not be able to use the **sql** command, which requires each command to be terminated by a semicolon.

1.8 Running the Console from a Shell Script

You can automate many Console tasks by running the console program from a shell script. For example, if you have created a file containing the following commands:

```
./bconsole -c ./bconsole.conf <<END_OF_DATA
unmount storage=DDS-4
quit
END_OF_DATA
```

when that file is executed, it will unmount the current DDS-4 storage device. You might want to run this command during a Job by using the **RunBeforeJob** or **RunAfterJob** records.
It is also possible to run the Console program from file input where the file contains the commands as follows:

```
./bconsole -c ./bconsole.conf <filename
```

where the file named **filename** contains any set of console commands.
As a real example, the following script is part of the Bacula regression tests. It labels a volume (a disk volume), runs a backup, then does a restore of the files saved.

```
bin/bconsole -c bin/bconsole.conf <<END_OF_DATA
@output /dev/null
messages
@output /tmp/log1.out
label volume=TestVolume001
run job=Client1 yes
wait
messages
@#
@# now do a restore
@#
@output /tmp/log2.out
restore current all
yes
wait
messages
@output
quit
END_OF_DATA
```

The output from the backup is directed to /tmp/log1.out and the output from the restore is directed to /tmp/log2.out. To ensure that the backup and restore ran correctly, the output files are checked with:

```
grep "^ *Termination: *Backup OK" /tmp/log1.out
backupstat=$?
grep "^ *Termination: *Restore OK" /tmp/log2.out
restorestat=$?
```

1.9 Adding Volumes to a Pool

If you have used the **label** command to label a Volume, it will be automatically added to the Pool, and you will not need to add any media to the pool.
Alternatively, you may choose to add a number of Volumes to the pool without labeling them. At a later time when the Volume is requested by **Bacula** you will need to label it.
Before adding a volume, you must know the following information:

1. The name of the Pool (normally "Default")
2. The Media Type as specified in the Storage Resource in the Director's configuration file (e.g. "DLT8000")
3. The number and names of the Volumes you wish to create.

For example, to add media to a Pool, you would issue the following commands to the console program:

```
*add
Enter name of Pool to add Volumes to: Default
Enter the Media Type: DLT8000
Enter number of Media volumes to create. Max=1000: 10
Enter base volume name: Save
Enter the starting number: 1
10 Volumes created in pool Default
*
```

To see what you have added, enter:

```
*list media pool=Default
+-------+----------+---------+---------+-------+------------------+
| MedId | VolumeNa | MediaTyp| VolStat | Bytes | LastWritten      |
+-------+----------+---------+---------+-------+------------------+
|    11 | Save0001 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    12 | Save0002 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    13 | Save0003 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    14 | Save0004 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    15 | Save0005 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    16 | Save0006 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    17 | Save0007 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    18 | Save0008 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    19 | Save0009 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
|    20 | Save0010 | DLT8000 | Append  |     0 | 0000-00-00 00:00 |
+-------+----------+---------+---------+-------+------------------+
*
```

Notice that the console program automatically appended a number to the base Volume name that you specify (Save in this case). If you don't want it to append a number, you can simply answer 0 (zero) to the question "Enter number of Media volumes to create. Max=1000:", and in this case, it will create a single Volume with the exact name you specify.

Chapter 2

Bacula Copyright, Trademark, and Licenses

There are a number of different licenses that are used in Bacula. If you have a printed copy of this manual, the details of each of the licenses referred to in this chapter can be found in the online version of the manual at http://www.bacula.org .

2.1 CC-BY-SA

The Creative Commons Attribution-ShareAlike 4.0 International License (CC-BY-SA) is used for this manual, which is a free and open license. Though there are certain restrictions that come with this license you may in general freely reproduce it and even make changes to it. However, rather than distribute your own version of this manual, we would much prefer if you would send any corrections or changes to the Bacula project.
The most recent version of the manual can always be found online at http://www.bacula.org .

2.2 GPL

The vast bulk of the source code is released under the Affero GNU General Public License version 3..
Most of this code is copyrighted: Copyright ©2000-2015 Kern Sibbald.
Portions may be copyrighted by other people. These files are released under different licenses which are compatible with the Bacula AGPLv3 license.

2.3 LGPL

Some of the Bacula library source code is released under the GNU Lesser General Public License. This permits third parties to use these parts of our code in their proprietary programs to interface to Bacula.

2.4 Public Domain

Some of the Bacula code, or code that Bacula references, has been released to the public domain. E.g. md5.c, SQLite.

2.5 Trademark

Bacula® is a registered trademark of Kern Sibbald.

2.6 Fiduciary License Agreement

Developers who have contributed significant changes to the Bacula code should have signed a Fiduciary License Agreement (FLA), which guarantees them the right to use the code they have developed, and also

ensures that the Free Software Foundation Europe (and thus the Bacula project) has the rights to the code. This Fiduciary License Agreement is found on the Bacula web site at:
http://www.bacula.org/en/FLA-bacula.en.pdf
and if you are submitting code, you should fill it out then sent to:

Kern Sibbald
Cotes-de-Montmoiret 9
1012 Lausanne
Switzerland

When you send in such a complete document, please notify me: kern at sibbald dot com.

2.7 Disclaimer

NO WARRANTY
BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM ”AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

2.8 Authors

The following people below have contributed to making this document what it is today:
Alexandre Baron ¡balexfr at users dot sourceforge dot net¿ Arno Lehmann ¡arnol at users dot sourceforge dot net¿ Bastian Friedrich ¡bastian dot friedrich at collax dot com¿ Christopher S dot Hull ¡csh at raidersolutions dot com¿ Dan Langille Davide Franco ¡dfranco at dflc dot ch¿ Dirk H Bartley ¡dbartley at schupan dot com¿ Eric Bollengier ¡eric.bollengier at baculasystems dot com¿ Frank Sweetser James Harper bendigoit dot com dot au¿ Jeremy C dot Reed ¡jeremy-c-reed at users dot sourceforge dot net¿ Jose Herrera ¡herrerajs at yahoo dot com¿ Jo Simoens Juan Luis Francis ¡indpnday at users dot sourceforge dot net¿ Karl Cunningham ¡karlec at users dot sourceforge dot net¿ Kern Sibbald ¡kern at sibbald dot com¿ Landon Fuller ¡landonf at opendarwin dot org¿ Lucas Di Pentima Ludovic Strappazon Meno Abels Nicolas Boichat Peter Buschman Philippe Chauvat ¡philippe.chauvat at baculasystems dot com¿ Philipp Storz Richard Mortimer ¡richm at oldelvet dot org dot uk¿ Robert Nelson ¡robertn at the-nelsons dot org¿ Scott Barninger Sebastien Guilbaud Thomas Glatthor Thomas Mueller ¡thomas at chaschperli dot ch¿ Thorsten Engel ¡thorsten dot engel at matrix-computer dot com¿ Victor Hugo dos Santos ¡victorhugops at users dot sourceforge dot net

Creative Commons Attribution-ShareAlike 4.0 International

Attribution-ShareAlike 4.0 International

Creative Commons Corporation (Creative Commons) is not a law firm and does not provide legal service

Using Creative Commons Public Licenses

Creative Commons public licenses provide a standard set of terms and conditions that creators and ot

Considerations for licensors: Our public licenses are intended for use by those authorized to gi

Considerations for the public: By using one of our public licenses, a licensor grants the public

Creative Commons Attribution-ShareAlike 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and

Section 1 Definitions.

Adapted Material means material subject to Copyright and Similar Rights that is derived from or
Adapter's License means the license You apply to Your Copyright and Similar Rights in Your contr
BY-SA Compatible License means a license listed at creativecommons.org/compatiblelicenses, appro
Copyright and Similar Rights means copyright and/or similar rights closely related to copyright
Effective Technological Measures means those measures that, in the absence of proper authority,
Exceptions and Limitations means fair use, fair dealing, and/or any other exception or limitatio
License Elements means the license attributes listed in the name of a Creative Commons Public Li
Licensed Material means the artistic or literary work, database, or other material to which the
Licensed Rights means the rights granted to You subject to the terms and conditions of this Publ
Licensor means the individual(s) or entity(ies) granting rights under this Public License.
Share means to provide material to the public by any means or process that requires permission u
Sui Generis Database Rights means rights other than copyright resulting from Directive 96/9/EC o
You means the individual or entity exercising the Licensed Rights under this Public License. You

Section 2 Scope.

License grant.
Subject to the terms and conditions of this Public License, the Licensor hereby grants You a
reproduce and Share the Licensed Material, in whole or in part; and
produce, reproduce, and Share Adapted Material.
Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations app
Term. The term of this Public License is specified in Section 6(a).
Media and formats; technical modifications allowed. The Licensor authorizes You to exercise
Downstream recipients.
Offer from the Licensor Licensed Material. Every recipient of the Licensed Material aut
Additional offer from the Licensor Adapted Material. Every recipient of Adapted Materia
No downstream restrictions. You may not offer or impose any additional or different term
No endorsement. Nothing in this Public License constitutes or may be construed as permission

Other rights.
Moral rights, such as the right of integrity, are not licensed under this Public License, no
Patent and trademark rights are not licensed under this Public License.
To the extent possible, the Licensor waives any right to collect royalties from You for the

Section 3 License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

```
    Attribution.

        If You Share the Licensed Material (including in modified form), You must:
            retain the following if it is supplied by the Licensor with the Licensed Material:
                identification of the creator(s) of the Licensed Material and any others designated
                a copyright notice;
                a notice that refers to this Public License;
                a notice that refers to the disclaimer of warranties;
                a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
            indicate if You modified the Licensed Material and retain an indication of any previous
            indicate the Licensed Material is licensed under this Public License, and include the te
        You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medi
        If requested by the Licensor, You must remove any of the information required by Section 3(a
    ShareAlike.

    In addition to the conditions in Section 3(a), if You Share Adapted Material You produce, the fo
        The Adapters License You apply must be a Creative Commons license with the same License Elem
        You must include the text of, or the URI or hyperlink to, the Adapter's License You apply. Y
        You may not offer or impose any additional or different terms or conditions on, or apply any

Section 4  Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed

    for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, a
    if You include all or a substantial portion of the database contents in a database in which You
    You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under t

Section 5  Disclaimer of Warranties and Limitation of Liability.

    Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor off
    To the extent possible, in no event will the Licensor be liable to You on any legal theory (incl

    The disclaimer of warranties and limitation of liability provided above shall be interpreted in

Section 6  Term and Termination.

    This Public License applies for the term of the Copyright and Similar Rights licensed here. Howe

    Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:
        automatically as of the date the violation is cured, provided it is cured within 30 days of
        upon express reinstatement by the Licensor.
    For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to
    For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate ter
    Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7  Other Terms and Conditions.

    The Licensor shall not be bound by any additional or different terms or conditions communicated
    Any arrangements, understandings, or agreements regarding the Licensed Material not stated herei

Section 8  Interpretation.

    For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduc
    To the extent possible, if any provision of this Public License is deemed unenforceable, it shal
    No term or condition of this Public License will be waived and no failure to comply consented to
    Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver
```

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect

Creative Commons may be contacted at creativecommons.org.