Bacula Community API

Using Bacula Community Version

This document is intended to provide insight into the considerations and processes required to design and implement a GUI Interface upon Bacula.

Bacula Systems White Paper

Version 1.3, May 14, 2017

Copyright (C) 2008-2017, Bacula Systems S.A.
All rights reserved.

www.baculasystems.com

Contents

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

Bacula Programming API

1.1 General

This White Paper will outline the best practices to design custom GUI and Panel for Bacula Community version 9.0.0 or later. The current Bacula interface can be described as a *human* interface, doing so, interfacing a program with Bacula requires a bit of work.

1.2 Assumptions

The following are the assumptions that we have made:

- You have a basic understanding of Bacula, Volumes, recycling, and backup strategies.
- You are using Bacula 9.0.0 or later.
- You are using PostgreSQL as your Bacula catalog database. This is highly recommended. If you are using another database such as MySQL, you may need to adjust some of the SQL in this document.
- You have a basic understanding of SQL.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

1.3 Implementing a GUI Interface

To implement a GUI interface with Bacula, you must be able to send commands to the Director. Although this is possible to do directly, in this manual, we will use `bconsole` to send commands to the Director. In addition, it is possible to directly query the catalog to get certain information. Even if the `bconsole` provides a command `sqlquery` that sends queries to the Catalog, this command is not really designed to handle multiple queries at the same time or large volume of data. Using a direct read-only SQL link is sometimes the best way to query the catalog.

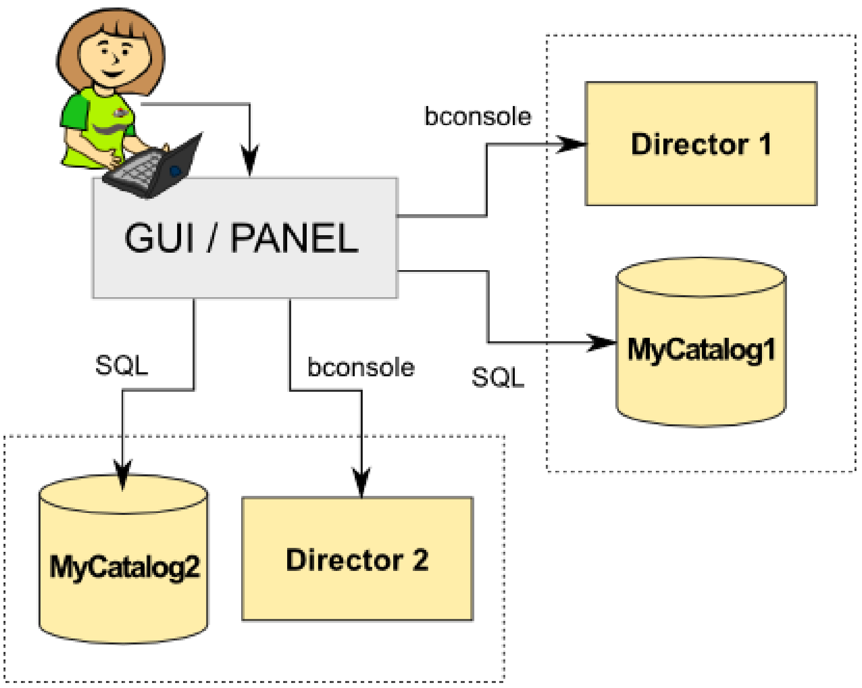


Figure 1.1: GUI Interaction with Bacula

If you need to edit or display resources such as Client, Jobs, FileSet or Schedule, Bacula provides a set of tools that extract the configuration in a friendly JSON output.

```
# bdirjson -r client -n 127.0.0.1-fd
{
    "Name": "127.0.0.1-fd",
    "Address": "127.0.0.1",
    "Password": "xxx",
    "Catalog": "MyCatalog",
    "FileRetention": 2592000,
    "JobRetention": 15552000,
    "AutoPrune": true,
    "MaximumConcurrentJobs": 10
}
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

1.3.1 Simple Console Interface

When writing a PHP, Perl, Python GUI interface, you might want to interact with Bacula using bconsole directly. One major drawback of this technique is that some of the lower level information available from a direct connection with the Director will not be available through a bconsole connection.

```
# printf "gui on\n.jobs\n" | bconsole
Connecting to Director 127.0.0.1:9101
1000 OK: 127.0.0.1-dir Version: 6.2.1 (18 January 2013)
Enter a period to cancel a command.
gui on
.jobs
NightlySave
MonsterSave
MonsterFileSet
VerifyVolume
#
```

In this example, the first 5 lines are not really interesting for your application, and need to be discarded to get the essential information. The 1000 OK indicates a successful connection with the Director.
Using a library such as Expect allows send multiple queries to bconsole. To know when a command has completed, bconsole will send a * at the beginning of the next line of output, or you can use @# bconsole comment command to print a specific marker. For example:

```
.bvfs_get_jobids client=127.0.0.1-fd
@echo ENDOFCOMMAND
```

will display the list of all JobIds and the word `ENDOFCOMMAND`.

1.3.2 Native Console Interface

Minimal Code in Console Program

Nearly all the database catalog interface code is in the Director (with the exception of dbcheck and bscan). This is because at some point we would like to add user level security and access. If we had spread code everywhere such as in a GUI this will be more difficult. The other advantage is that any code you add to the Director is automatically available to both the tty console program and the WX program. The major disadvantage is it increases the size of the code – however, compared to Networker the Bacula Director is really tiny.

GUI Interface is Difficult

Interfacing to an interactive program such as Bacula with the current API can be difficult because the interfacing program must interpret all the prompts that may come. This can be next to impossible. There are are a number of ways that Bacula is designed to facilitate this:

- The Bacula network protocol is packet based, and thus pieces of information sent can be ASCII or binary.
- The packet interface permits special “signals” to be passed rather than data.
- The packet interface permits knowing where the end of a list is.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

- The Director has a number of commands that are non-interactive. They all begin with a period, and provide things such as the list of all Jobs, list of all Clients, list of all Pools, list of all Storage, ... Thus the GUI interface can get to virtually all information that the Director has in a deterministic way. See src/dird/ua_dotcmds.c for more details on this.
- Most console commands allow all the arguments to be specified on the command line: e.g. **run job=NightlyBackup level=Full**

One of the first things to do is to establish a conversation with the Director. Below is a discussion how you can do this directly with your own program rather than interfacing through bconsole.

Although you can write your own code, it is probably easier to use the Bacula subroutines. The following code is used by the Console program to begin a conversation.

```
static BSOCK *UA_sock = NULL;
static JCR *jcr;
...
  read-your-config-getting-address-and-pasword;
  UA_sock = bnet_connect(NULL, 5, 15, "Director daemon", dir->address,
                          NULL, dir->DIRport, 0);
   if (UA_sock == NULL) {
      terminate_console(0);
      return 1;
   }
   jcr.dir_bsock = UA_sock;
   if (!authenticate_director(\&jcr, dir)) {
      fprintf(stderr, "ERR=%s", UA_sock->msg);
      terminate_console(0);
      return 1;
   }
   read_and_process_input(stdin, UA_sock);
   if (UA_sock) {
      bnet_sig(UA_sock, BNET_TERMINATE); /* send EOF */
      bnet_close(UA_sock);
   }
   exit 0;
```

Then the read_and_process_input routine looks like the following:

```
   get-input-to-send-to-the-Director;
   bnet_fsend(UA_sock, "%s", input);
   stat = bnet_recv(UA_sock);
   process-output-from-the-Director;
```

For a GUI program things will be a bit more complicated. Basically in the inner loop, you will need to check and see if any output is available on the UA_sock. For an example, please take a look at the WX GUI interface code in: `src/wx-console`

1.4 Bacula Commands

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>] ...
```

This document will list some useful command for a GUI front end, for the complete list, you can refer to the Console manual.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

1.4.1 Job Management

run This command allows you to run job immediately. The full form of the command is:

```
run job=<job-name> client=<client-name>
  fileset=<FileSet-name>  level=<level-keyword>
  storage=<storage-name> when=<universal-time-specification>
  spooldata=yes|no accurate=yes|no yes
```

Any information that is needed but not specified will be listed as a prompt for selection, and before starting the job, you will be prompted to accept, reject, or modify the parameters of the job to be run. To skip this menu you need to specify **yes** in the command line and the job will be immediately sent to the scheduler. All resources needed to run the job can be listed using dot commands such as `.clients`, `.filesets`, ...

cancel 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.

```
cancel [jobid=<number>|job=<job-name>|ujobid=<unique-jobid>]
```

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.

stop The **stop** command is very similar to the **cancel** command with the main difference that the Job that is stopped is marked as Incomplete so that it can be restarted later by the **restart** command where it left off (see below). The **stop** command with no arguments, will like the cancel command, prompt you with the list of running jobs allowing you to select one, which might look like the following:

```
stop [jobid=<number>|job=<job-name>|ujobid=<unique-jobid>]
```

restart The **restart** command allows console users to restart a canceled, failed, or incomplete Job. For canceled and failed Jobs, the Job will restart from the beginning. For incomplete Jobs the Job will restart at the point that it was stopped either by a stop command or by some recoverable failure.

.clients The **.clients** command will display all Client resources.

```
.clients
127.0.0.1-fd
127.0.0.2-fd
```

.filesets The **.fileset** command will display all FileSet resources.

```
.filesets
Full Set
Catalog
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

.jobs The **.jobs** command will display all Job resources. You can select only a certain type of jobs using the **type=** parameter.

```
.jobs type=R
RestoreJob

.jobs type=B
BackupJob
BackupCatalog

.jobs
BackupJob
BackupCatalog
RestoreJob
```

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

```
setbandwidth limit=<nb> [ jobid=<id> | client=<cli> ]
```

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 yes
```

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.

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.

status This command will display the status of all components. For the director, it will display the jobs that are scheduled during the next 24 hours as well as the status of currently running jobs. For the Storage Daemon, you will also get 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 client**, the console will connect to the specified client and display information about current jobs. This output is essential to know what the job is currently doing.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

```
status client=<client-name>

127.0.0.1-fd Version: 6.3.5 (01 Feb 2013)  x86_64-linux-gnu archlinux
Daemon started 13-Feb-13 17:01. Jobs: run=0 running=1.
 Heap: heap=135,168 smbytes=25,258 max_bytes=25,275 bufs=74 max_bufs↙
→=74
 Sizes: boffset_t=8 size_t=8 debug=0 trace=0 mode=1,2010 bwlimit=0kB/s
 Plugin: bpipe-fd.so vsphere-fd.so

Running Jobs:
JobId 1 Job NightlySave.2013-02-13_17.04.11_03 is running.
    Full Backup Job started: 13-Feb-13 17:04
    Files=1,526 Bytes=43,434,400 AveBytes/sec=1,447,813 LastBytes/sec↙
→=2,895,626 Errors=0
    Bwlimit=3,000,000
    Files: Examined=1,526 Backed up=1,526
    Processing file: /tmp/regress/build/src/cscope.out
    SDReadSeqNo=5 fd=5
JobId 5 Job RestoreFiles.2012-03-09_09.54.38_09 is running.
      Restore Job started: 09-Mar-12 00:54
    Files=7 Bytes=0 Bytes/sec=0 Errors=7
    Bwlimit=0
    Files: Restored=8 Expected=32 Completed=25%
    Processing file: c:/test2/
Director connected at: 13-Feb-13 17:04
```

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. When running a large set of Jobs, the `status dir` might take a long time and is not suitable for GUI.

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.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

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 I do **`status storage=File`**, I 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
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 indicates 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 reissue 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 "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 qw started using the "File" device is that the device is blocked waiting for media – that is Bacula needs you to label a Volume.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

1.4.2 Volume Management

The following section will present `bconsole` commands that are used to handle volumes.

add This command is used to add Volumes to an existing Pool.

```
add [pool=<pool-name>] [storage=<storage>]
```

This command 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.

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

```
label storage=<storage-name> volume=<volume-name> pool=<pool-name>
      [barcodes] slot=<slot> drive=<drive-number>
```

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
```

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

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

and then label it with the same name as the barcode. An appropriate Media record will also be created in the catalog.

Note, the full form of the command is:

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

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
   Scratch Pool
   Slot
   InChanger Flag
   Pool
   Action On Purge
   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.

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.

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
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

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.

delete 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.

```
delete volume=<vol-name> | pool=<pool-name> | jobid=<id> | job=<job-↙
→name>
```

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 **JobId** appears on the command line, a Job and all its associated records (File and JobMedia) will be deleted from the catalog.

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:

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

```
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.

.pool This command will list all Pool resources defined.

```
.pool
Default
Full
Inc
```

show The show command will list the Director's resource records as defined in the Director's configuration file. The following keywords are accepted on the show command line: catalogs, clients, counters, devices, directors, filesets, jobs, messages, pools, schedules, storages, all.

.storage This command will list all Storage resources defined.

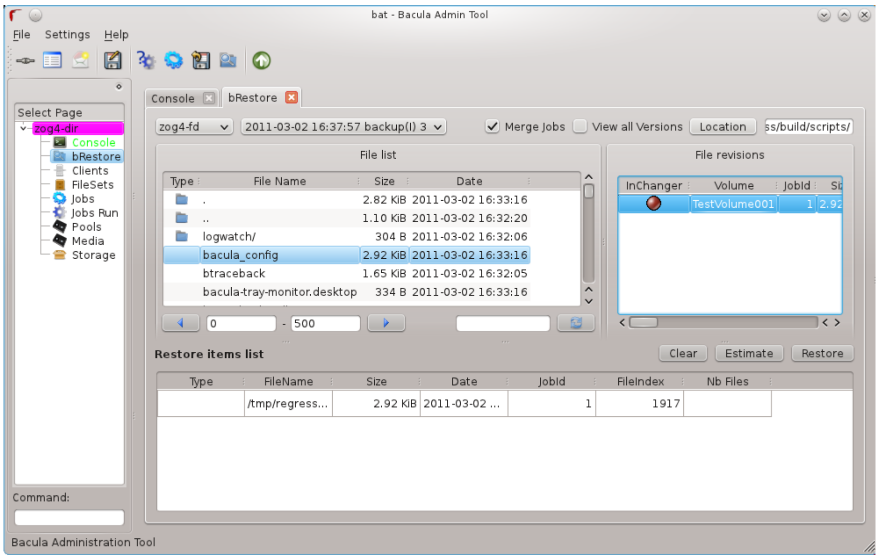
```
.storage
LTO3
File
```

.msgs This command will list all Messages resources defined.

1.4.3 Restore GUI Using Bvfs API

To help developers of restore GUI interfaces, we have added new *dot commands* that permit to browse the catalog in a very simple way. We strongly advise you to use BVFS instead of writing your own set of queries. When adding a new feature in Bacula, such as Delta, the BVFS is updated to support the new feature, when using your own set of queries, you will have to adapt them to support new features.
Bat has now a bRestore panel that uses Bvfs to display files and directories.
The Bvfs module works correctly with BaseJobs, Delta, Copy and Migration jobs.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners



Type	File Name	Size	Date
	.	2.82 KiB	2011-03-02 16:33:16
	..	1.10 KiB	2011-03-02 16:32:20
	logwatch/	304 B	2011-03-02 16:32:06
	bacula_config	2.92 KiB	2011-03-02 16:33:16
	btraceback	1.65 KiB	2011-03-02 16:32:05
	bacula-tray-monitor.desktop	334 B	2011-03-02 16:33:16

0 - 500

InChanger	Volume	JobId	Si
●	TestVolume001	1	2.92

Type	FileName	Size	Date	JobId	FileIndex	Nb Files
	/tmp/regress...	2.92 KiB	2011-03-02 ...	1	1917	

Figure 1.2: Bat Brestore Panel

General notes

- All fields are separated by a tab
- You can specify `limit=` and `offset=` to list smoothly records in very big directories
- All operations (except cache creation) are designed to run instantly
- At this time, Bvfs works faster on PostgreSQL than MySQL catalog.
- The cache creation is dependent of the number of directories. As Bvfs shares information across jobs, the first creation can be slow
- Due to potential encoding problem, it's advised to always use pathid in queries.
- BVFS can use Console ACLs to restrict results with Bacula version 9.0.0 before 9.0.0 the custom Bacula interface must ensure that a user is allowed to browse or restore files with custom procedures.

Get dependent jobs from a given JobId

Bvfs allows to query the catalog against any combination of jobs. You can combine all Jobs and all FileSet for a Client in a single session.
To get all JobId needed to restore a particular job, you can use the `.bvfs_get_jobids` command.

```
.bvfs_get_jobids ujobid=name|jobid=num [client=name] [only|all]
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

```
.bvfs_get_jobids jobid=100
10,20,50,100
.bvfs_get_jobids jobid=100 all
10,20,30,50,100
.bvfs_get_jobids ujobid=BackupJob.2013-02-11_15.41.01_03 only
10
.bvfs_get_jobids client=localhost-fd
2,3,4,10,20,30,50,100
```

In this example, a normal restore will need to use JobIds 10,20,50,100 to compute a complete restore of the system.

- **all**: the Director will use all defined FileSet for this client.
- **only**: the Director will convert a ujobid to a jobid.
- **client**: the Director will display all Jobs for this client. (requires Bacula Version 9.0.0 or later)

With the `all` option, the Director will use all defined FileSet for this client.

Generating Bvfs cache

The `.bvfs_update` command computes the directory cache for jobs specified in argument, or for all jobs if unspecified.

```
.bvfs_update [jobid=numlist]
```

Example:

```
.bvfs_update jobid=1,2,3
```

You can run the cache update process in a RunScript after the catalog backup.
This cache is used by the `.bvfs_lsdirs` function, other functions such as `.bvfs_lsfiles` are not using the cache.

Get all versions of a specific file

Bvfs allows to find all versions of a specific file for a given Client with the `.bvfs_version` command. To avoid problems with encoding, this function uses only PathId and FilenameId. The jobid argument is mandatory but unused up to Bacula version 9.0.0.

```
.bvfs_versions client=filedaemon pathid=num filenameid=num
PathId FilenameId FileId JobId LStat Md5 VolName Inchanger
PathId FilenameId FileId JobId LStat Md5 VolName Inchanger
...
```

Example:

```
.bvfs_versions client=localhost-fd pathid=1 fnid=47
1  47  52  12  gD HRid IGk D Po Po A P BAA I A   /uPWagKZlnMti7LChyA  Vol1  ↙
→1
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

Get volume information for a particular file

Bvfs allows to get volume information for a particular file. (requires Bacula version 9.0.0)

```
.bvfs_get_volumes fileid=1
VolName
VolName2
...
```

If the file spans over multiple volumes, the function will return one volume per line.

List directories

Bvfs allows to list directories in a specific path.

```
.bvfs_lsdirs pathid=num path=/apath jobid=numlist limit=num offset=num
PathId  FilenameId  FileId  JobId  LStat  Path
PathId  FilenameId  FileId  JobId  LStat  Path
PathId  FilenameId  FileId  JobId  LStat  Path
...
```

You need to pathid or path. Using path="" will list "/" on Unix and all drives on Windows. If FilenameId is 0, the record listed is a directory.

```
.bvfs_lsdirs pathid=4 jobid=1,11,12
4       0       0       0       A A A A A A A A A A A A A A     .
5       0       0       0       A A A A A A A A A A A A A A     ..
3       0       0       0       A A A A A A A A A A A A A A     regress/
```

In this example, to list directories present in regress/, you can use

```
.bvfs_lsdirs pathid=3 jobid=1,11,12
3       0       0       0       A A A A A A A A A A A A A A     .
4       0       0       0       A A A A A A A A A A A A A A     ..
2       0       0       0       A A A A A A A A A A A A A A     tmp/
```

Depending on the FileSet configuration, root directory might not be part of the backup. In this case, a dummy LStat will be displayed (having all fields to 0).

List files

Bvfs allows to list files in a specific path.

```
.bvfs_lsfiles pathid=num path=/apath jobid=numlist limit=num offset=num
PathId  FilenameId  FileId  JobId  LStat  Path
PathId  FilenameId  FileId  JobId  LStat  Path
PathId  FilenameId  FileId  JobId  LStat  Path
...
```

You need to pathid or path. Using path="" will list "/" on Unix and all drives on Windows. If FilenameId is 0, the record listed is a directory.

```
.bvfs_lsfiles path=/tmp/regress/build/po/ jobid=1,11,12
26   34   2263   1   b   IST  IGk  B  Po  Bk  A  Blr  BA  Mw  BRFg  BRFg  BRG  A  A  C   nl.po
```

In this example, to list files present in regress/, you can use

```
.bvfs_lsfiles pathid=1 jobid=1,11,12
1    47    52    12    gD  HRid  IGk  BAA  I  BMqcPH  BMqcPE  BMqe+t  A      titi
1    49    53    12    gD  HRid  IGk  BAA  I  BMqe/K  BMqcPE  BMqe+t  B      toto
1    48    54    12    gD  HRie  IGk  BAA  I  BMqcPH  BMqcPE  BMqe+3  A      tutu
1    45    55    12    gD  HRid  IGk  BAA  I  BMqe/K  BMqcPE  BMqe+t  B      fich1.txt
1    46    56    12    gD  HRie  IGk  BAA  I  BMqe/K  BMqcPE  BMqe+3  D      fich2.txt
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

Decode LStat Field

The LStat field contains basic `stat()` information about the file or directory. You can decode this Base64 field quite easily in Perl, PHP or even PL/SQL. BVFS provides also a basic tool to decode the value.

```
.bvfs_decode_lstat lstat="b ISB EHt L Po Bk A Mg BAA A BRGhs0 BRGgam BRGgam ↙
→A  A C"
st_nlink=11
st_mode=16877
st_uid=1000
st_gid=100
st_size=800
st_blocks=0
st_ino=33921
st_ctime=1360660134
st_mtime=1360660134
st_mtime=1360665396
st_dev=27
LinkFI=0
```

Restore set of files

Bvfs allows to create a SQL table that contains files that you want to restore. This table can be provided to a restore command with the file option.

```
.bvfs_restore fileid=numlist dirid=numlist hardlink=numlist path=b2num
OK
restore file=?b2num ...
```

To include a directory (with `dirid`), Bvfs needs to run a query to select all files. This query could be time consuming.
`hardlink` list is always composed of a series of two numbers (jobid, fileindex). This information can be found in the LinkFI field of the LStat packet.
The `path` argument represents the name of the table that Bvfs will store results. The format of this table is `b2[0-9]+`. (Should start by b2 and followed by digits).
Example:

```
.bvfs_restore fileid=1,2,3,4 hardlink=10,15,10,20 jobid=10 path=b20001
OK
```

Cleanup after Restore

To drop the table used by the restore command, you can use the `.bvfs_cleanup` command.

```
.bvfs_cleanup path=b20001
```

Clearing the BVFS Cache

If needed, to clear the BVFS cache, you can use the `.bvfs_clear_cache` command.

```
.bvfs_clear_cache yes
OK
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

1.5 Object Detailed Information

When using the `llist` command, you are able to get detailed information about Bacula's objects such as volume or jobs.

```
*llist jobid=8
           jobid: 8
             job: NightlySave.2013-02-11_15.41.30_10
            name: NightlySave
     purgedfiles: 0
            type: B
           level: I
        clientid: 1
            name: 127.0.0.1-fd
       jobstatus: T
       schedtime: 2013-02-11 15:41:30
       starttime: 2013-02-11 15:41:32
         endtime: 2013-02-11 15:41:33
     realendtime: 2013-02-11 15:41:33
...
```

```
*llist volume=TestVolume001
          mediaid: 1
       volumename: TestVolume001
             slot: 0
           poolid: 1
        mediatype: File
     firstwritten: 2013-02-11 15:41:03
      lastwritten: 2013-02-11 15:41:32
        labeldate: 2013-02-11 15:41:03
...
```

This is also possible to get information on objects through SQL queries.

1.6 Resource List

To list resources defined in the Director configuration file, you can use very simple *dot* commands. Resources will be displayed as they are defined in the configuration file.

- **.clients** will display all clients, one by line
- **.pools** will display all pools, one by line
- **.jobs** will display all jobs, one by line
- **.filesets** will display all jobs, one by line
- **.catalogs** will display all catalogs, one by line
- **.msgs** will display all messages, one by line

```
*.client
127.0.0.1-fd
127.0.0.2-fd
127.0.0.3-fd
```

For example, to fill the run job screen (Fig 1.3 on the following page), you will need to use the following commands:

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

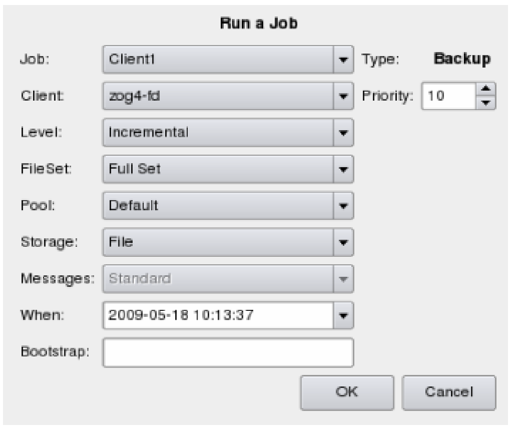


Figure 1.3: Run a Job

```
.clients
.jobs
.filesets
.pools
.storages
.msgs
```

When using a "script" based on bconsole, you might want to use the following bconsole script to easily extract all information.

```
gui on
.clients
@echo ENDOFCMD
.jobs
@echo ENDOFCMD
.filesets
@echo ENDOFCMD
.pools
@echo ENDOFCMD
.storage
@echo ENDOFCMD
.msgs
@echo ENDOFCMD
```

It will display the following output:

```
[eric@zog8 /tmp/regress] cat script | ./bin/bconsole
Connecting to Director 127.0.0.1:8101
1000 OK: 127.0.0.1-dir Version: 6.3.6 (14 Feb 2013)
Enter a period to cancel a command.
gui on
.clients
127.0.0.1-fd
127.0.0.1-fd2
@#ENDOFCMD
.jobs
NightlySave
NightlySave1
NightlySave2
RestoreFiles
@#ENDOFCMD
.filesets
Full Set
Verify Set
@#ENDOFCMD
.pools
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

```
Default
Full
Test
Inc
Diff
Scratch
@#ENDOFCMD
.storage
LTO1
LTO3
LTO1-ANSI_6
File
@#ENDOFCMD
.msgs
Standard
Daemon
@#ENDOFCMD
```

1.7 JSON Bacula Resource Description

Bacula 9.0.0 introduced new set of tools that handle all components configuration and produce computer friendly interface using JSON format.

- **bdirjson** will display Director resources
- **bsdjson** will display Storage Daemon resources
- **bfdjson** will display File Daemon resources
- **bbconsjson** will display Console resources

All JSON tools are using the following options to configure the JSON output:

- **-c** Path to the configuration file
- **-r** Select a specific resource type
- **-n** Select a specific resource name (use with -r)
- **-l** Get only directives matching a pattern (use with -r)
- **-D** Get only data

```
# bdirjson -r client
[
{
  "Client": {
    "Name": "127.0.0.1-fd",
    "Address": "127.0.0.1",
    "Password": "xxx",
    "Catalog": "MyCatalog",
  }
},
{
  "Client": {
    "Name": "127.0.0.2-fd",
    "Address": "127.0.0.2",
    "Password": "xxx",
    "Catalog": "MyCatalog",
  }
}
]
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

When combining with **-D** option, all resources will be placed in an array.

```
# bdirjson -r client -D
[
{
    "Name": "127.0.0.1-fd",
    "Address": "127.0.0.1",
    "Password": "xxx",
    "Catalog": "MyCatalog",
},
{
    "Name": "127.0.0.2-fd",
    "Address": "127.0.0.2",
    "Password": "xxx",
    "Catalog": "MyCatalog",
}
]
```

When combining with **-l** option, you can display a set of Directives matching the **-l** argument.

```
# bdirjson -r client -D -l 'Name|Address'
[
{
    "Name": "127.0.0.1-fd",
    "Address": "127.0.0.1",

},
{
    "Name": "127.0.0.2-fd",
    "Address": "127.0.0.1",
}
]
```

1.8 Generate Bacula Configuration

One of the major strengths of Bacula is the ability to generate configuration in with very simple scripts.

1.9 Enable Bacula Statistics Management

If you (or probably your boss) want to have statistics on your backups to provide some *Service Level Agreement* indicators, you could use a few SQL queries on the Job table to report how many:

- jobs have run
- jobs have been successful
- files have been backed up
- ...

However, these statistics are accurate only if your job retention is greater than your statistics period. Ie, if jobs are purged from the catalog, you won't be able to use them.
Now, you can use the **update stats [days=num]** console command to fill the `JobHisto` table with new Job records. If you want to be sure to take in account only **good jobs**, ie if one of your important job has failed but you have fixed the problem

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

and restarted it on time, you probably want to delete the first *bad* job record and keep only the successful one. For that simply let your staff do the job, and update JobHisto table after two or three days depending on your organization using the **[days=num]** option.

1.10 Autochanger Operations

The **status slots storage=<storage-name>** command displays autochanger content.

```
status slots storage=LTO1 drive=0
Automatically selected Storage: LTO1
Connecting to Storage daemon LTO1 at 127.0.0.1:8103 ...
3306 Issuing autochanger "slots" command.
Device "LTO" has 80 slots.
Connecting to Storage daemon LTO1 at 127.0.0.1:8103 ...
3306 Issuing autochanger "list" command.
 Slot |  Volume Name   |  Status   |  Media Type        |   Pool     |
------+---------------+----------+-------------------+------------|
    1 |         00001 |   Append |  DiskChangerMedia |    Default |
    2 |         00002 |   Append |  DiskChangerMedia |    Default |
    3*|         00003 |   Append |  DiskChangerMedia |    Scratch |
    4 |               |          |                   |            |
```

If an asterisk (*) appears after the slot number, you must run an **update slots** command to synchronize autochanger content with your catalog.

1.11 Bacula Catalog

The Bacula Catalog is a very important piece of the Bacula infrastructure, and the Catalog size is closely related to the number of files that you are protecting. Depending on how you access and query the Catalog, you might see big performance differences. This guide will propose you a set of queries that can be used on your GUI.

1.11.1 Catalog Schema

Order of Magnitude

This is quite important to have some knowledge about the order of magnitude in the Bacula Catalog. Getting information from the Job table is rather easy, getting information on File or Path requires good SQL queries.
A quite large Catalog can contain:

- thousands of Clients
- thousand of Media (Volumes)
- hundreds of thousands of Jobs
- hundreds of thousands of JobMedia
- tens of millions of Paths
- hundreds of millions of Filenames
- billions Files

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

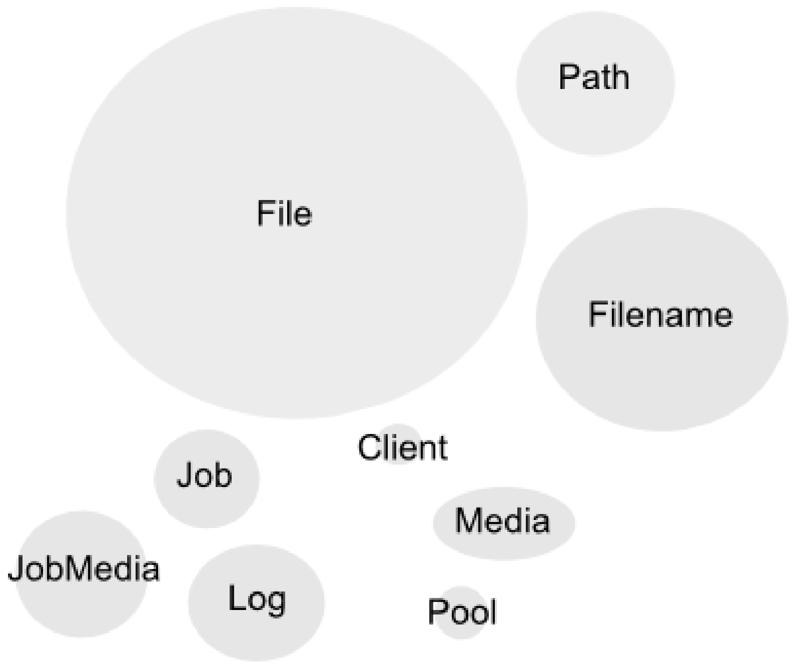


Figure 1.4: Main Tables Overview

If you need a specific index on some Bacula tables, such as on Job.StartTime or Job.ClientId for example, it can be added without too much risk for the overall Bacula performance. Adding a new index on Path or File without checking with Bacula experts will probably lead to serious performance degradation.

Jobs

The Job table is the central table of the Catalog, this table is usually joined with Client, Pool and FileSet. It's common to have between 10,000 and 200,000 records in this table. You will find information about the start date, the status, the size or the number of files for all jobs.
The Job table is maintaining only jobs that can still be restored, if you want to maintain a copy of the Job table for statistic or billing purpose, you can use the "`update stats`" command and query the JobHisto table.
As the Job table can be quite large, this is a good idea to always query this table using the Job name, or the Client or using a filter on the start date.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

(Illustrative material only)

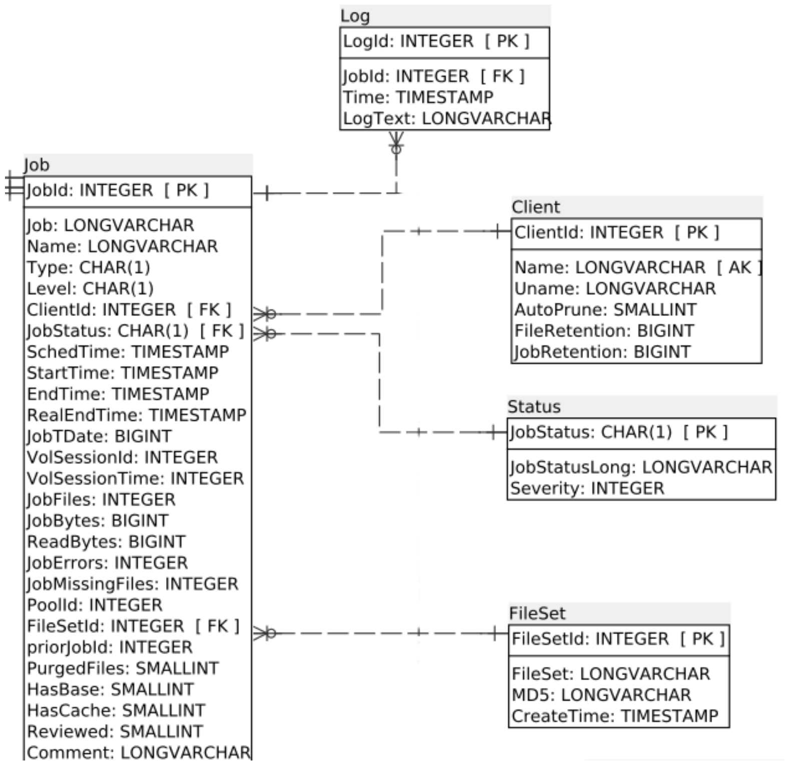


Figure 1.5: Job Table

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

(Illustrative material only)

Status	Info
C	Created, not yet running
R	Running
B	Blocked
T	Completed successfully
W	Completed with warnings
E	Terminated with errors
e	Non-fatal error
f	Fatal error
A	Canceled by user
F	Waiting for Client
S	Waiting for Storage daemon
m	Waiting for new media
M	Waiting for media mount
s	Waiting for storage resource
j	Waiting for job resource
c	Waiting for client resource
d	Waiting on maximum jobs
t	Waiting on start time
p	Waiting on higher priority jobs
a	SD despooling attributes
i	Doing batch insert file records
I	Incomplete Job

Figure 1.6: Status Code in the Job Table

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

Type	Info
B	Backup Job
M	A previous backup job that was migrated
V	Verify Job
R	Restore Job
D	Admin job
A	Archive Job
C	Copy of a Job
c	Copy Job
g	Migration Job

Figure 1.7: Job Type

Status	Info
Append	Ready to append data
Full	Volume Full, got end of media
Used	Volume marked administratively full
Purged	Retention time passed
Recycled	Volume is selected for reuse
Error	Marked on Error by Bacula, need to be checked

Figure 1.8: Volume Status

Clients and Groups

Media

The Media table contains a reference to all media (Volumes) that are accessible by Bacula. Each volume is linked with a Pool. As the Media table can contain a large number of items, this is a good idea to browse Media by selecting a Pool first.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

(Illustrative material only)

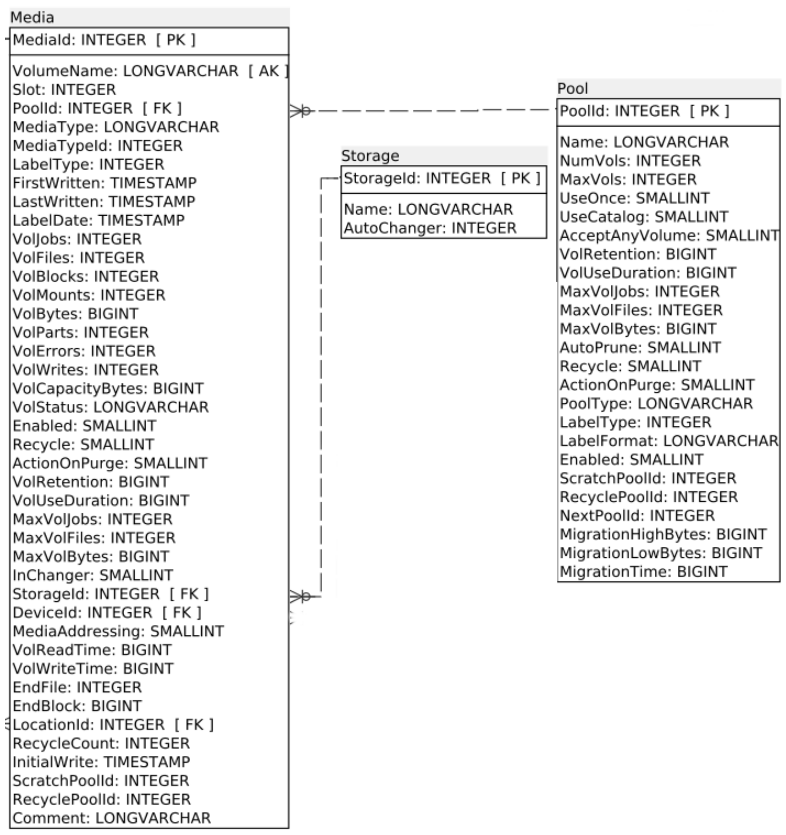


Figure 1.9: Job Media Table

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

Files

The File table contains a reference to all files that are on Bacula volumes (except if you don't store file information or if the job is pruned). This table can be very large (billions of records are not uncommon).
Path name and File name are stored in separate tables and can be also quite large. You always need to use the JobId when querying this table, this is even better to have the Path, and this is very efficient when you have the FileId or the Job, Path and Filename. Bacula is using Path and Filename encoded in UTF8 and the catalog can use a case sensitive index. Searching into the File table using an insensitive operator such as `ILIKE` gives very poor performances.
When using this table in your application, this is a good idea to get a Client, then to get a list of Jobs. Sometime it's a bit difficult to get information from the final user, but searching into File without the Job name or the Path is very inefficient, queries can run during minutes or even hours.

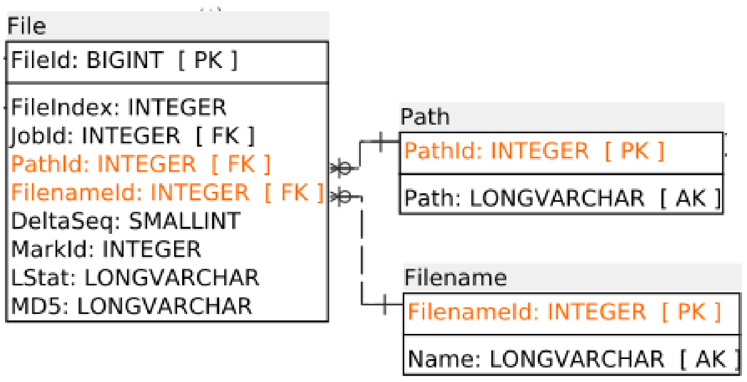


Figure 1.10: File Table

We advise you to use the BVFS layer to access File, Filename and Path.

1.11.2 Common Queries

We advise you to never run a SQL query that would modify Bacula tables (such as `UPDATE` or `DELETE` commands), instead use Bacula bconsole commands. This will ensure that if several tables are involved, they will be correctly handled by the Director. If you do it directly, you may end up with database inconsistencies that can cause Bacula to fail.

MySQL and PostgreSQL Comparison

When you start to write your Bacula GUI, you need to decide if you want to support both MySQL and PostgreSQL or just one of them. You need to know that some parts of Bacula such as BVFS run faster on PostgreSQL.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

Operation	MySQL	PostgreSQL
Match a string	REGEXP	~ *
Unix timestamp from date	UNIX_TIMESTAMP(val)	date_part('epoch', val)
Convert secs to time	val	interval '1 second' * val
Concat two strings	CONCAT(s1,s2,s3)	s1 || s2 || s3
...		

If you want to support MySQL and PostgreSQL in your GUI, you will have to test all SQL queries on both engines to make sure that your syntax is valid.
Depending on MySQL settings you might have to respect the case (upper/lower) in SQL queries, so it's better to always use the mixed case format (ie `SELECT Job.Name` and not `SELECT job.name`).

List Running Jobs

To list all jobs that are currently running on a Director, you can use the `.status dir running` command or use the following SQL statement.

```
SELECT Job.JobId AS jobid,
       Job.Name  AS jobname,
       Job.Level     AS level,
       Job.StartTime AS starttime,
       Job.JobFiles  AS jobfiles,
       Job.JobBytes  AS jobbytes,
       Job.JobStatus AS jobstatus,
       Job.JobTDate AS  jobtdate,
       Client.Name AS clientname
FROM Job JOIN Client USING (ClientId)
WHERE
  JobStatus IN ('C','R','B','e','D','F','S','m',
                'M','s','j','c','d','t','p');
```

Display Client List

In the Catalog, you will find some useful information in the Uname field about Clients such as:

- Bacula version
- OS version
- CPU architecture type

Extracting this information is not really complex, but the format may vary from one OS to an other.

```
SELECT Name   AS name,
       Uname  AS uname,
       AutoPrune AS autoprune,
       FileRetention AS fileretention,
       JobRetention  AS jobretention
FROM Client;
```

Display Job Information

For a particular JobId (101 here), the following SQL will give you a lot of useful information.

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

```
SELECT DISTINCT Job.JobId       AS jobid,
                Client.Name     AS client,
                Job.Name        AS jobname,
                FileSet.FileSet AS fileset,
                Level           AS level,
                Pool.Name       AS poolname,
                StartTime       AS starttime,
                JobFiles        AS jobfiles,
                JobBytes        AS jobbytes,
                JobStatus       AS jobstatus,
                JobErrors       AS joberrors,
                Type            AS jobtype,
                ReadBytes       AS readbytes,
                Job.Comment     AS comment,
                JobTDate        AS jobtdate
 FROM Client JOIN USING (ClientId) JOIN
      Job LEFT JOIN FileSet ON (Job.FileSetId = FileSet.FileSetId)
          LEFT JOIN Pool    ON (Job.PoolId    = Pool.PoolId)
 WHERE Job.JobId = 101;
```

You can compute the duration of the job with the following kind of code.

```
 $now = time();
 $duration = $now - $result['jobtdate'];
```

To get the list of jobs for a particular Client or a group of Clients, you can use the following code in the `WHERE` clause.

```
  AND Client.Name IN ('127.0.0.1-fd', 'localhost-fd')
```

If you want to limit your search to a particular date, you can use the `JobTDate` or the `StartTime` fields. When using the `StartTime` field, you need to adapt your code to run on PostgreSQL or MySQL, using `JobTDate` implies to compute the time limit in your program which is probably simpler.

```
  $lastweek = time() - 7 * 3600;

  SELECT ... FROM Job ... WHERE JobTDate > $lastweek
```

Display Pools

To get the space that is used by Purged and Recycled volumes, you can use something like:

```
SELECT Pool.Name AS name,
       sum(VolBytes) AS size
FROM   Media JOIN Pool ON (Media.PoolId = Pool.PoolId)
WHERE  Media.VolStatus IN ('Recycled', 'Purged')
GROUP BY Pool.Name;
```

Display Media

As Bacula is handling media automatically, you need to really think if you want to give this level of information to your end-users.

```
  SELECT InChanger     AS online,
       Media.Enabled   AS enabled,
       VolBytes        AS nb_bytes,
       VolumeName      AS volumename,
       VolStatus       AS volstatus,
       VolMounts       AS nb_mounts,
       Media.VolUseDuration    AS voluseduration,
       Media.MaxVolJobs   AS maxvoljobs,
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

```
       Media.MaxVolFiles AS maxvolfiles,
       Media.MaxVolBytes AS maxvolbytes,
       VolErrors         AS nb_errors,
       Pool.Name         AS poolname,
       Location.Location AS location,
       Media.Recycle     AS recycle,
       Media.VolRetention AS volretention,
       Media.LastWritten  AS lastwritten,
       Media.VolReadTime/1000000  AS volreadtime,
       Media.VolWriteTime/1000000 AS volwritetime,
       Media.RecycleCount AS recyclecount,
       Media.Comment      AS comment,
 FROM Pool,
      Media LEFT JOIN Location ON (Media.LocationId = Location.LocationId)
 WHERE Pool.PoolId = Media.PoolId
 AND VolumeName IN ('Volume001', 'Volume002')
```

1.12 .api version 2

In Bacula version 9.0.0, we introduced a new .api version to help external tools to parse various Bacula bconsole output.
The `api_opts` option can use the following arguments:

C Clear current options

tn Use a specific time format (1 ISO format, 2 Unix Timestamp, 3 Default Bacula time format)

sn Use a specific separator between items (new line by default).

Sn Use a specific separator between objects (new line by default).

o Convert all keywords to lowercase and convert all non *isalpha* characters to _

```
  .api 2 api_opts=t1s43S35
  .status dir running
==================================
jobid=10
job=AJob
...
```

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

Revision History

Version	Date	Owner	Changes
1.0	12 February 2013	Eric	Initial creation
1.1	20 February 2013	Kern	Updates – proof read
1.2	11 April 2013	Eric	Updates
1.3	13 May 2017	Kern	Add Eric's API 2 documentation

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners

For More Information

For more information on Bacula Enterprise Edition, or any part of the broad Bacula Systems services portfolio, visit www.baculasystems.com.

Headquarters
Bacula Systems SA
Rue Galilée 5
CH-1400 Yverdon-les-Bains
Switzerland
Phone: +41 21 641 6080
Fax: +41 21 641 6081

USA
Western Region
100 100th Ave SE. #3
Bellevue, WA 98004

Eastern Region & Canada
269 Carmita Ave
Rutherford, NJ 07070

Toll Free: +1 800 256 0192

Rev : 2 V. 1.3
Author(s): KES

Copyright © May 2017 Bacula Systems SA www.baculasystems.com/contactus
All trademarks are the property of their respective owners