Attention, version 13.2.2 is out

This information is available for reference only as we have released already a newer stable release.

Please use Version 13.2.2 for new installations, as it contains everything that 12.4 contains and much more.

Bandwidth limitation

You can add a bandwidth limitation in the client resource definition:

Maximum Bandwidth Per Job  = 1024 k/s

This will limit the network bandwidth used for this client per job by 1024 kilo bits per second.

Console commands

import

Automatic Tapechangers offer special slots for importing new tape cartridges or exporting written tape cartridges. This can happen without having to set the device offline. With the new console commands import and export, the importing and exporting of tapes is now much easier.

To import new tapes into the autochanger, you only have to load the new tapes into the import/export slots and call import from the cmdline.

The import command will automatically transfer the new tapes into free slots of the autochanger. The slots are filled in order of the slot numbers. To import all tapes, there have to be enough free slots to load all tapes.

Example with a Library with 36 Slots and 3 Import/Export Slots:

*import storage=TandbergT40
Connecting to Storage daemon TandbergT40 at bareos:9103 ...
3306 Issuing autochanger "slots" command.
Device "Drive-1" has 39 slots.
Connecting to Storage daemon TandbergT40 at bareos:9103 ...
3306 Issuing autochanger "listall" command.
Connecting to Storage daemon TandbergT40 at bareos:9103 ...
3306 Issuing autochanger transfer command.
export

The export command does exactly the opposite of the import command. You can specify which slots should be transferred to import/export slots. The most useful application of the export command is the possibility to automatically transfer the volumes of a certain backup into the import/export slots for external storage.

To be able to do this, the export command also accepts a list of volume names to be exported.

Example:

```
export volume=A00020L4|A00007L4|A00005L4
```

This is exactly the format of the volume list in Variable `%V` (Captive v) after the Backup.

So to automatically export the Volumes used by a certain backup job, you can use the following RunScript in that job:

```
RunScript {
    Console = "export storage=TandbergT40 volume=%V"
    RunsWhen = After
    RunsOnClient = no
}
```

**e-mail notification via Messages resource regarding export tapes**

Variable `%V` substitution in the Messages resource is implemented in Bareos 13.2. However, also in earlier release it does already work inside job resources. So in version prior to Bareos 13.2 following workaround can be used:

```
RunAfterJob = "'/bin/bash -c \"'/bin/echo Remove Tape %V | \" /usr/sbin/bsmtp -h localhost -f root@localhost -s 'Remove Tape %V' root@localhost \""'
```
**move**

The new move command allows to move volumes between slots without having to leave the bconsole.

To move a volume from slot 32 to slots 33, use:

```
*move storage=TandbergT40 srcslots=32 dstslots=33
```

Connecting to Storage daemon TandbergT40 at bareos:9103 ...

3306 Issuing autochanger "slots" command.

Device "Drive-1" has 39 slots.

Connecting to Storage daemon TandbergT40 at bareos:9103 ...

3306 Issuing autochanger "listall" command.

Connecting to Storage daemon TandbergT40 at bareos:9103 ...

3306 Issuing autochanger transfer command.

3308 Successfully transfered volume from slot 32 to 33.

**rerun command**

In Bareos, the job configuration is often altered by job overrides. These overrides alter the configuration of the job just for one job run. If because of any reason, a job with overrides fails, it is not easy to restart a new job that is exactly configured as the job that failed. The whole job configuration is automatically set to the defaults and it is hard to configure everything like it was.

By using the rerun command, it is now much easier to rerun a jobs exactly as it was configured. You only have to specify the JobId of the failed job:

```
rerun jobid=
```

**Scheduler Enhancements**

**last keyword**

Until now, the scheduler was able to schedule in a certain week of the month, e.g. 1st, 2nd... 5th. Unfortunately, it is not possible to schedule a job on "the last friday of the month", because the last friday of the month can either be in the 4th or 5th week of the month.

Now the keyword "last" is available. If given, the Scheduler will trigger the job if we are in the last week of the month.
Example:

```yaml
Schedule {
    Name = "Last Friday"
    Run = Level=Full last fri at 21:00
}
```

**modulo scheduler**

The modulo scheduler makes it easy to specify schedules like odd or even days/weeks, or more generally every n days or weeks. It is called modulo scheduler because it uses the modulo to determine if the schedule must be run or not. Some examples:

```yaml
Schedule {
    Name = "Odd Days"
    Run = 1/2 at 23:10
}
```

```yaml
Schedule {
    Name = "Even Days"
    Run = 2/2 at 23:10
}
```

```yaml
Schedule {
    Name = "Odd Weeks"
    Run = w01/w02 at 23:10
}
```

```yaml
Schedule {
    Name = "Even Weeks"
    Run = w02/w02 at 23:10
}
```

Without the modulo scheduler specifying a schedule for even weeks would look like this:

```yaml
Schedule {
    Name = "Even Weeks"
    Run = w02,w04,w06,w08,w10,w12,w14,w16,w18,w20,w22,w24,w26,w28,w30,w32,w34,w36,w38,w40,w42,w44,w46,w48,w50,w52 at 23:10
}
```
Fileset Shadowing Detection

A fileset shadow occurs, if you define a directory in your fileset and a subdirectory. E.g. / and /usr, which makes sense in the case that /usr is a separate filesystem. On the other hand it will lead to doubled backedup data, if /usr is not on a separate filesystem. With Bareos you can detect such shadows. To activate this feature put `shadowing = localremove` in your fileset options, which will exclude detected fileset shadows from your backup. With `localwarn`, only a warning will be issued, if shadows are detected.

Example for a fileset resource with fileset shadow warning enabled:

```plaintext
FileSet {
    Name = "Test Set"
    Include {
        Options {
            signature = MD5
            shadowing = localwarn
        }
        File = /
        File = /usr
    }
}
```

Configuration Parser Enhancements

**default values for strings**

The configuration parser was previously not able to have default settings for configuration strings. Now, also configuration strings can have default values. Using this possibility, we can omit a lot of redundant information leading to shorter and more comprehensible configuration files.

**default value for catalog**

While theoretically having the possibility to use multiple catalogs, virtually every installation only uses one catalog.

Before, the only used catalog had to be configured in multiple places, especially in every single client resource.

Now, the

```plaintext
Catalog =
```
can be left out of the client definition. If so, the first defined catalog is automatically chosen.

**default value for Cleaning Prefix**

Cleaning Tapes cannot be used to write on them. By setting the *Cleaning Prefix* directive in the *Pool* resource, you could tell Bacula that Volumes starting with the defined prefix are cleaning tapes and have to be ignored during labeling.

Es the Cleaning Prefix is **CLN** in most cases, the default value now also is set to **CLN**, as if the following line was configured:

```
Cleaning Prefix = "CLN"
```

Now, cleaning tapes should be recognized in 99% of the cases automagically without having to configure anything. By setting the directive to another value, the default can of course be overridden.

**LTO Hardware Encryption**

LTO4 and newer LTO generation drives as well as other modern tape drives support hardware encryption. There are several ways of using encryption with these drives. The following three types of key management are available for doing encryption. The transmission of the keys to the volumes is accomplished by:

- A backup application that supports Application Managed Encryption (AME)
- A tape library that supports Library Managed Encryption (LME)
- Using a Key Management Appliance (KMA).

We added support for Application Managed Encryption (AME) scheme where on labeling a crypto key is generated for a volume and when the volume is mounted the crypto key is loaded and when unloaded the key is cleared from the memory of the Tape Drive using the SCSI SPOUT command set.

There is a comprehensive [README.scsirypto](README.scsirypto) about this subject.

**Quota**
With the bareos quota code, it is possible to limit the amount that a certain client is able to backup.

For all calculations, the amount of data stored by a specific client is regarded.

The quota support adds the following directives and the needed parameter type to the Client resource:

- Soft Quota (amount of data)
- Soft Quota Grace Period (time interval)
- Strict Quotas (yes/no)
- Hard Quota (amount of data)
- Quota Include Failed Jobs (yes/no)

**Soft Quota and Soft Quota Grace Time Period**

When the amount of data backed up by the client outruns the value specified by the soft quota directive, the next start of a backup job will start the soft quota grace time period. This is written to the job log:

Error: Softquota Exceeded, Grace Period starts now.

In the Job Overview, the value of Grace Expiry Date: will then change from Soft Quota was never exceeded to the date when the grace time expires, e.g. 11-Dec-2012 04:09:05

During that period, it is possible to do backups even if the total amount of stored data is over the limit specified by soft quota

If in this state, the Job Log will write:

Error: Softquota Exceeded, will be enforced after Grace Period expires.

After the grace time expires, in the next Backup Job of the Client, the value for Burst Quota will be set to the value that the client has stored at this point in time. Also, the Job will be terminated. The following Information in the Job Log shows what happened:

Warning: Softquota Exceeded and Grace Period expired.
Setting Burst Quota to 122880000 Bytes.

At this point, it is not possible to do any backup of the client. To be able to do more backups, the amount of stored data for this client has to fall under the burst quota value.

**Quota Include Failed Jobs**
The directive **Quota Include Failed Jobs** determines, if failed jobs are considered in the calculation of the space used by the client for Hard and Soft Quotas or not.

The default value is **yes**.

**Strict Quotas**

The directive **Strict Quotas** determines, if after the **Grace Time Period** is over, the **Burst Limit** is enforced (\textit{Strict Quotas} = \textit{No}) or the **Soft Limit** is enforced (\textit{Strict Quotas} = \textit{Yes}).

The Job Log shows either

\textit{Softquota Exceeded, enforcing Burst Quota Limit}.

or

\textit{Softquota Exceeded, enforcing Strict Quota Limit}.

The default value is **No**.

**Hard Quota**

The amount of data determined by the **Hard Quota** directive sets the hard limit of backup space that cannot be exceeded.

If the **Hard Quota** is exceeded, the running job is terminated:

\textit{Fatal error: append.c:218 Quota Exceeded. Job Terminated}.

**Example for Quota Configuration in Client resource**

```
# Quota
Soft Quota = 50 mb
Soft Quota Grace Period = 15 second
Strict Quotas = Yes
Hard Quota = 150 mb
Quota Include Failed Jobs = yes
```
NDMP

No filed plugin but a proper implementation with support in the director to act as a NDMP DMA (Data Management Application) and for NDMP tape agent support in the storage daemon for saving data using the NDMP protocol.

Please read more in the README.NDMP.

Windows Drive Discovery

Until now, available Windows drives could not be automatically discovered. This lead to the problem, that for every single Windows Client all available drives had to be configured in the fileset.

Also, if a new drive was added to the client, it was not automatically backed up.

With the new Windows Drive Discovery code, the available drives can automatically be discovered by the File Daemon on start of the backup job.

Therefore, the FileSet has to contain the Entry

File = /

The given '/' will be expanded to all available local drives.

If the "drive type" directive is configured, only drives of the specified type will be selected.

If VSS is used (default=yes), only drives of type "fixed" will be snapshotted via VSS. The VSS Snapshot of drives of other than "fixed" type is not possible and would lead to an error.

The following example shows a FileSet that automatically will backup all local fixed drives and exclude usually unwanted data like the pagefile or the recyclers.

FileSet {
    Name = "Windows All Drives"
    Enable VSS = yes
    Include {
        Options {
            Signature = MD5
            Drive Type = fixed # only backup fixed drives (e.g no CD-ROM)
            IgnoreCase = yes
            WildFile = "[A-Z]:/pagefile.sys"
        }
    }
}
WildDir = "[A-Z]:/RECYCLER"
WildDir = "[A-Z]://RECYCLE.BIN"
WildDir = "[A-Z]:/System Volume Information"
Exclude = yes

Windows Installer

The windows installer was significantly enhanced. The interactive inputs masks have been enhanced to be more understandable. Also, all inputs that are given during interactive install can now directly be configured on the commandline, so that an automatic silent install is possible.

Commandline Switches

/? shows the list of available parameters.

/S sets the installer to silent. The Installation is done without user interaction. This switch is also available for the uninstaller.

By setting the Installation Parameters via commandline and using the silent installer, you can install the bareos client without having to do any configuration after the installation:

winbareos-12.4.0-64-bit-r11.1.exe /S /CLIENTNAME=windows64-fd /CLIENTPASSWORD="verysecretpassword" /DIRECTORNAME=bareos-dir

This will install the bareos windows client without user interaction.

New console commands in 12.4.4

status scheduler

We have the new command status scheduler available in bareos 12.4.4. Before, it was not possible to check when a certain schedule would trigger. The preview in the status director is not powerful enough.
With **status scheduler**, it is easy to see when a certain scheduler will trigger jobs.

Called without parameters, **status scheduler** shows a preview for all schedules for the next 14 days.

**status scheduler** first shows a list of the known schedules and the jobs that will be triggered by these jobs:

```
*status scheduler
Scheduler Jobs:

Schedule               Jobs Triggered
===========================================================================
WeeklyCycle             BackupClient1
WeeklyCycleAfterBackup  BackupCatalog

====
```

Next, a table with Date (including weekday), schedule name and applied overrides is displayed:

```
Scheduler Preview for 14 days:

<table>
<thead>
<tr>
<th>Date</th>
<th>Schedule</th>
<th>Overrides</th>
</tr>
</thead>
<tbody>
<tr>
<td>Di 04-Jun-2013</td>
<td>WeeklyCycle</td>
<td>Level=Incremental</td>
</tr>
<tr>
<td>Di 04-Jun-2013</td>
<td>WeeklyCycleAfterBackup</td>
<td>Level=Full</td>
</tr>
<tr>
<td>Mi 05-Jun-2013</td>
<td>WeeklyCycle</td>
<td>Level=Incremental</td>
</tr>
<tr>
<td>Mi 05-Jun-2013</td>
<td>WeeklyCycleAfterBackup</td>
<td>Level=Full</td>
</tr>
<tr>
<td>Do 06-Jun-2013</td>
<td>WeeklyCycle</td>
<td>Level=Incremental</td>
</tr>
<tr>
<td>Do 06-Jun-2013</td>
<td>WeeklyCycleAfterBackup</td>
<td>Level=Full</td>
</tr>
<tr>
<td>Fr 07-Jun-2013</td>
<td>WeeklyCycle</td>
<td>Level=Incremental</td>
</tr>
<tr>
<td>Fr 07-Jun-2013</td>
<td>WeeklyCycleAfterBackup</td>
<td>Level=Full</td>
</tr>
<tr>
<td>Sa 08-Jun-2013</td>
<td>WeeklyCycle</td>
<td>Level=Differential</td>
</tr>
<tr>
<td>Mo 10-Jun-2013</td>
<td>WeeklyCycle</td>
<td>Level=Incremental</td>
</tr>
<tr>
<td>Mo 10-Jun-2013</td>
<td>WeeklyCycleAfterBackup</td>
<td>Level=Full</td>
</tr>
</tbody>
</table>
```
status scheduler accepts the following parameters:

- **client=**clientname shows only the schedules that affect the given client.
- **job=**jobname shows only the schedules that affect the given job.
- **schedule=**schedulename shows only the given schedule.
- **days=**number of days shows only the number of days in the scheduler preview. Positive numbers show the future, negative numbers show the past. **days=** can be combined with the other selection criteria.

status subscriptions

To make it easier for users that have a bareos subscription to keep the overview over the subscriptions that are used or available, subscriptions can now be automatically checked.

To enable this functionality, just add the configuration directive Subscriptions to the director configuration in the director resource:

```
Director {
    ...
    Subscriptions = 4
}
```

Using the console command **status subscriptions**, the status of the subscriptions can be checked any time interactively:

```
Ok: available subscriptions: 1 (3/4) (used/total)
```
Also, the number of subscriptions is checked after every job. If the number of clients is bigger than the configured limit, a Job warning is created a message like this:

```
JobId 7: Warning: Subscriptions exceeded: (used/total) (5/4)
```

Important: **Nothing else than the warning is issued, no enforcement on backup, restore or any other operation will happen.**

Setting the value for Subscriptions to 0 disables this functionality:

```plaintext
Director {
    ...
    Subscriptions = 0
}
```

Not configuring the directive at all also disables it, as the default value for the Subscriptions directive is zero.

**time**

The **time** command shows the current date and time and was available in bareos and bacula since ever but did not show the weekday.

As usually backup schedules refer to weekdays, we added the weekday to the output of the **time** command.

**rerun command**

In Bareos, the job configuration is often altered by job overrides. These overrides alter the configuration of the job just for one job run. If because of any reason, a job with overrides fails, it is not easy to restart a new job that is exactly configured as the job that failed. The whole job configuration is automatically set to the defaults and it is hard to configure everything like it was.

By using the rerun command, it is now much easier to rerun a jobs exactly as it was configured. You only have to specify the JobId of the failed job:

Before 12.4.4, only the parameter jobid was available to select a single jobid:

```
rerun jobid=
```

With version 12.4.4, we now also have the options that allow to automatically select multiple jobids, as it is
not uncommon that multiple jobids fail cased by the same error.

- **days=number of days** or **hours=number of hours**. This will automatically select all failed jobids in the last *number of days* or *number of hours* respectively for rerunning.
- **since_jobid=jobid**. This will automatically select all jobs failed after and including the given jobid for rerunning.