Bacula Enterprise Upgrade and Removal

Bacula Systems Documentation

Contents

Contents

The following chapter aims at explaining how to upgrade and remove Bacula Enterprise.

1 Bacula Enterprise Upgrade

The following chapter aims at explaining how to upgrade Bacula Enterprise.

Request the newest Bacula Enterprise version from your Customer Portal.

1.1 Bacula Enterprise Upgrade on Linux

The following chapter aims at explaining how to upgrade Bacula Enterprise on Linux.

A major Bacula upgrade involves changing the first or second digit, such as going from 16.0.10 to 16.2.3 or from 16.0.10 to 18.0.3.

On the other hand, a minor Bacula upgrade only involves changing the last digit, for example, from 16.0.10 to 16.0.11.

If you upgrade only the Client or File Daemon (FD), follow *Bacula Enterprise Upgrade Procedure* for any version

If you upgrade the Director and the Storage Daemon:

- Major upgrade case

  Follow the document: *Migration to New Major Version of Bacula*.

- Minor upgrade case

  Follow the document: *Bacula Enterprise Upgrade Procedure*.

Important:

- The upgrade of the Director and the Storage Daemon should happen at the same time as both must run the same version.
- The FD running on the same system than the Director and the Storage Daemon will also be upgraded at the same time.
- The plugins running on the same system than the Director and the Storage Daemon or the Client must also be upgraded at the same time.

Bacula Enterprise Upgrade Procedure

The following article aims at proving you with necessary information about BE upgrade, and instructing you on how to perform it safely.

Important: Always have the Director and Storage Daemon(s) at the same version level. You can upgrade the File Daemons of your clients afterwards.

On the Director and any Storage Daemon machine(s), the File Daemon version must also match the DIR or SD version on that machine. For other machines, the FD version may be less than or equal to the DIR and SD, but not greater.

Note: It is recommended to have a testing environment and to upgrade it first, prior to upgrading a production environment.

Steps

1. Configuration files backup.

   Backup your main configuration file folder (the content of `/opt/bacula/etc`) as well as any other files (script, services) you have created or modified. If you are using Bacula Enterprise plugin(s), backup the `/opt/bacula/working` directory too.

   The upgrade will overwrite your service files, startup files, and some scripts, but it will not overwrite any `.conf` files.

2. Catalog backup if you upgrade the Director.

   You can back it up to temporary media following these instructions:

   ```
   sudo -u bacula /opt/bacula/scripts/make_catalog_backup.pl MyCatalog
   ```

   ```
   cp /opt/bacula/working/bacula.sql /tmp
   ```

   Change `/tmp` to any mount point you prefer.

3. Repository update.

   - BIM usage

     If you used BIM to install the Bacula Enterprise components, simply run BIM again and it will propose to install the new Bacula version. Do run BIM again for any plugin installed on that system.

   - Package manager usage

     Modify your package manager configuration file(s) in order to reflect the version you want to upgrade to. Do so for all plugins and products installed as well.

4. Upgrade.

   - Patch or minor

     If you upgrade to a new version of Bacula Enterprise and only the last digit changes, for instance from 12.2.1 to 12.2.2, the process is straightforward. Just use your package manager to upgrade different Bacula components and plugins. Install new binaries and libraries. You should schedule a maintenance window for this, and restart the Bacula daemons that have received package upgrades, so that the latest versions are up and running.

- Major

  If you upgrade your Bacula environment to a new feature release (change of minor version digit) or a new major release, the procedure may involve a schema update of the Bacula Catalog database: *Migration to a new major version of Bacula*.

Note: If you run into trouble or have any error messages while upgrading your Bacula Enterprise infrastructure, open a ticket from your Customer Portal.

See also:

Go back to *Migration to New Major Version of Bacula*.

Go back to the *Bacula Enterprise Upgrade on Linux* chapter.

Go back to the *Bacula Enterprise Upgrade* chapter.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

Migration to New Major Version of Bacula

Overview

This document demonstrates how to upgrade Bacula Enterprise from previous versions to version **16.0**. If you are upgrading from one Bacula version to another you should first carefully read the ReleaseNotes of all major versions between your current version and the version to which you are upgrading. For many upgrades, especially for minor patch upgrades (e.g. between **16.0**.0 and **16.0**.1), there will be no database upgrade so the process is to simply install the new software version. However, in many of the major version upgrades there are also changes made to the Bacula Catalog database which require the additional steps of running one or more database upgrade scripts.

Contact Bacula Systems Support if you have any questions about the upgrade procedure.

Scope

This document is specifically written for Bacula Systems Enterprise version **16.0**. The upgrade procedures are very similar from one version to another, however this paper is directly applicable only to upgrading to Bacula Enterprise **16.0**.x and the outlined procedures must be modified to be applicable to other versions.

Upgrading Bacula

File Daemon

As always, in any new release we attempt to support the prior File daemon version and usually even older File daemons can be supported. This avoids the need to do a simultaneous upgrade of many machines. For precisely which older versions of the File daemon are supported, please see the ReleaseNotes for the new version.

Linux/Unix File Daemon Deployment

On Linux/Unix, you can upgrade the File daemon by simply copying the new version (rpm, deb or tar) to the server, then install it using the package manager native to that system.

Tools such as CFEngine or Puppet may also be used for automatic distribution and upgrade. It is even possible to create a Bacula Restore job that will put the Bacula binaries on the remote server, then execute a ClientRunAfterJob runscript at the end of the job to deploy the new version.

Microsoft Windows File Daemon Deployment

On Windows, it is possible to use Samba tools to deploy your new version remotely from the Director. First, use the `net` command to stop the `bacula-fd` service, then copy new files on the server using the `smbclient` or `smbtar` tool and the restart the bacula-fd service.

```
$ net -U administrator%passwd -S your-server rpc service stop bacula-fd
$ smbclient //your-server/c$ '' -U administrator%passwd -N -Tx bacula-win32.tar
$ net -U administrator%passwd -S your-server rpc service start bacula-fd
```

Director and Storage Daemons

You must always upgrade both the Director and the Storage Daemon(s) at the same time, and you must also upgrade any File daemon that is running on the same machine as a Director or a Storage daemon. Note: In general if only the patch number changes (the third digit of the Bacula version), you will not need to upgrade Storage daemons that are on separate machines. However, for major version changes, for example from 6.4.x to 16.0.x, you must also upgrade all of your Storage daemons on all machines.

BWeb Management Suite Catalog Upgrade

When upgrading BWeb Management Suite from 6.x, 8.0, 8.2, 8.4, 8.6, you must update the BWeb SQL tables with the script `upgrade-6.0-8.8_postgresql.sql` or `upgrade-6.0-8.8_mysql.sql`.

For MySQL users, when upgrading BWeb Management Suite from 8.8, 10.0 you must update the BWeb SQL tables with the script `upgrade-8.8-10.2_mysql.sql`.

BCloud Service Upgrade

The major release 12.2 of BCloud Service uses a new catalog format. We provide a SQL script that converts 12.0.x and earlier catalog versions (2) to the new 12.2 format (3).

```
% psql -U bacula bacula < /opt/bacula/bcloud/www/protected/conf/upgrade-10.0_12.2_
↪postgresql.sql
```

After applying the SQL script, please go to Admin => Directors page on the BCloud Service interface and please fill for each defined Director the Address and Port fields.

Upgrade Process Overview

This release of Bacula uses a new catalog format.

We provide a set of scripts that convert earlier Catalog versions to the new **16.0**.0 format (database version **1026**). You can utilize the scripts provided in order to perform the manual Catalog upgrade process as described in the Section *2*. We strongly recommend that you save a copy of your existing database before upgrading.

Bacula Systems Support can assist if you have questions prior to or during the upgrade process.

In the case where you did not perform the manual Catalog upgrade prior to the installation of the Bacula upgrade binaries, the upgrade process will be carried out automatically. Please note that as a first step of the automatic Catalog upgrade process, a dump of the Catalog database will be placed in the /opt/bacula/working directory. Please make sure that this directory has enough free space available to hold this Catalog dump. Once the automatic upgrade process is complete, the database backup file will not be deleted automatically.

Generally, we recommend the automatic Catalog upgrade approach. However, in scenarios where the size of the Catalog database is relatively large and/or that there is not enough free space on the partition that holds the /opt/bacula/working directory we recommend performing the manual Catalog upgrade process or to stop the Catalog service prior to the upgrade.

The **1026** Catalog format (for Bacula version **16.0**.x) is a quick and simple schema upgrade.

[upgradeoverview] General Bacula upgrade process:

1. Stop any current version of Bacula from running.
2. To skip the automatic Catalog upgrade, stop the Catalog service.
3. Install the **16.0**.x version of Bacula.
4. If you have not performed the manual Catalog upgrade, the install will upgrade your database automatically.
5. If you have stopped the Catalog service, restart it and run the update_bacula_tables script.
6. Start the new Bacula daemons. If everything works correctly, no error messages should be printed.

Catalog Upgrade

Catalog Backup

We strongly advise you to backup your Bacula catalog database before making any changes to it. To do so, you can use the `make_catalog_backup.pl` script provided with Bacula (see your catalog backup job for the exact command). Keeping the SQL result file on your local disk rather than in a Bacula volume or on tape can speed up recovery if something goes wrong.

Bacula Enterprise Binaries

If you are using the Bacula Enterprise binaries, you will need the update scripts that are discussed in the following sections. These scripts are contained in a special updatedb package. For rpms you should install:

```
bacula-enterprise-updatedb-x.y.z.rpm
```

where the x and y vary depending on what rpm and version you are installing. Normally this rpm will install the scripts in the /opt/bacula/scripts/updatedb directory. If this is not the case please take note that some of the paths shown in subsequent sections will need to be adjusted accordingly.

For systems that use the .deb packages, these upgrade scripts will already be installed in the /opt/bacula/scripts directory.

Scripts may be extracted from packages using a tool such a `mc`, or the Bacula Systems support team can provide the latest version of the migration scripts on demand. If the database server is stopped during the installation, automatic backup and update procedures are skipped.

Upgrade Procedure

Upgrading the catalog is normally done after Bacula is installed by:

```
cd /opt/bacula/scripts       (or where the scripts can be found)
./update_bacula_tables
```

If there are several database upgrades between your version and the version to which you are upgrading (see table below), you will need to run the database upgrade script for each version. Version 12, 13, 14, 1014, 1015, 1017, 1018, 1019, 1020, 1025, **1026** catalogs are converted using the `update_bacula_tables` script. For your convenience, you can find all the old upgrade scripts in the upgradedb package for RPMs and in /opt/bacula/scripts. If you don't find these scripts, please contact the Bacula Systems support team. You will need to edit the scripts to correspond to your system configuration. The final upgrade script can be applied as noted above.

Catalog Versions

Bacula version	Catalog version
1.38.x	9
2.0.x	10
2.2.x	10
2.4.x	10
2.6.x	11*
3.0.x	11*
5.0.x	12
4.0.x	13
5.2.x	14
6.0.x	1014
6.4.x	1015
6.6.x	1016*
8.0.x	1016*
8.2.x	1017
8.6.x	1018
8.8.x	1019
8.10.x	1019
10.0.x	1020
10.2.x	1020
12.0.x	1021
12.2.x	1021
12.4.x	1022
12.6.x	1023
12.8.x	1024
14.0.x	1025
16.0.x	1026

The time and storage space required for the database upgrades with "" in the table above depends on the number of files that your catalog has stored and the capacity of your hardware. For versions listed without the "*", upgrades should proceed very quickly.*

How to Verify the Current Version Number

```
$ bconsole
*sql
Entering SQL query mode.
Terminate each query with a semicolon.
Terminate query mode with a blank line.
Enter SQL query: SELECT VersionId FROM Version;
+-----------+
| VersionId |
+-----------+
|      1026 |
+-----------+
```

Important Note on 2.4.x to 3.0.x Upgrade (Catalog Versions 10 to 11)

In this upgrade, we changed the size of the FileId field from 32 bits (integer) to 64 bits (bigint). If you have already done this prior to this upgrade, you need to remove the `ALTER TABLE File` command in the script `update_postgresql_tables_10_to_11` or `update_mysql_tables_10_to_11` (see below), otherwise the script may fail. If you have questions about this, or the script fails, please contact Bacula Systems support team.

On PostgreSQL:

```
$ psql -U bacula bacula
bacula=> \d File
...
fileid     | integer        (can be integer or bigint)
```

Or on MySQL:

```
$ mysql -u bacula bacula
mysql> describe File;
...
| FileId     | bigint(20) unsigned
```

Important Note on 6.0, 6.2, 6.4 to 6.6, 8.x,10.x,12.0 Upgrade (Catalog Version 1016 and 1015)

For a PostgreSQL database, the upgrade from 1015 to 1016 will create a large temporary file in the local directory (where the script is executed). Make sure that the filesystem is big enough to create the file (up to 20% of the database size).

For a MySQL database, or when using the Advanced Upgrade Procedure for PostgreSQL, the upgrade from 1015 to 1016 will create a copy of the File table in the database, please make sure that the database filesystem is big enough to create the table copy (up to 80% of the database size).

For more information, see Section **myupgrade**, and Section **pgupgrade**.

When installing RPMs or Debs, the package installer will try to dump the old catalog and upgrade the catalog automatically. This means that you must have the space available for a dump, and during a major upgrade such as moving from 6.6.x to **12.8**.0, the database upgrade procedure will require additional disk space equal to the size of your database. Thus in the worst case, unless you off-load the dump, you will need additional disk of two times the size of your existing database.

In the case of extremely large databases where you do not have sufficient disk space, it is possible to perform the database upgrade prior to doing the Bacula package upgrade. This will prevent the package upgrade from detecting an old database version and the database dump will be skipped, thereby removing the requirement of additional disk space of two times the size of your existing database.

MySQL Upgrade Process

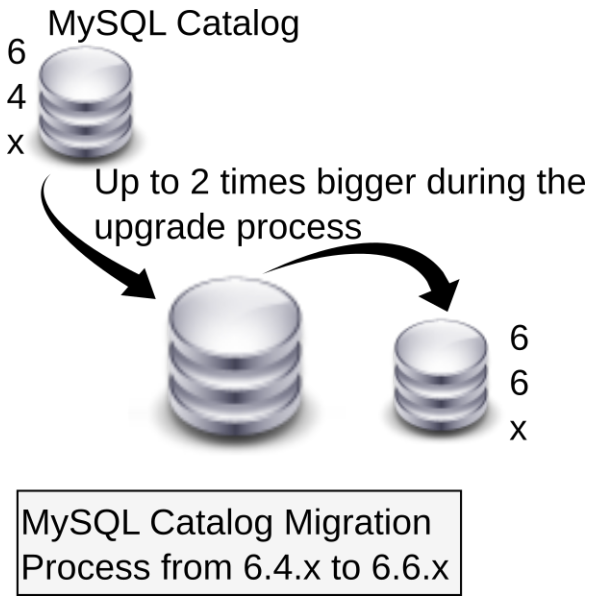


Fig. 1: MySQL Upgrade Procedure

1. Stop any current version of Bacula from running.
2. Save a copy of your existing database.
3. Ensure that there is enough free space on the catalog filesystem to hold a copy of the File table. The upgrade process will probably require 50% of the catalog size of free disk space before starting the upgrade process.
4. If the database is large, it is possible to upgrade the catalog prior the package installation (see the next section).
5. Install the **12.8**.x version of Bacula.
6. If the install didn't upgrade your database automatically, run the upgrade script by reading section **runupgrade** section of this document.
7. Start the new Bacula daemons. If everything worked correctly, no error messages should be printed.

The MySQL `tmpdir` directory (usually /tmp) must have adequate space for the upgrade. This may be as large as many gigabytes for catalogs with many files. You must also check the `innodb_data_file_path` setting. For example, the following might cause the upgrade to fail if more than 512MB is required:

```
innodb_data_file_path = ibdata1:10M:autoextend:max:512M
```

This should be changed to something like:

```
ibdata1:10M:autoextend
```

to allow unlimited space.

PostgreSQL Upgrade Process

The following parameters can be set to avoid extra work for the database. It will require a restart of the PostgreSQL (9.x) database.

```
archive_mode = off
wal_level = minimal
```

If you are using PostgreSQL replication, we advise you to stop it during the migration process, and re-synchronize your PostgreSQL slave instance after the migration. Don't forget to change your settings back to their original values once the upgrade is finished.

When using PostgreSQL, it is possible to choose between two migration processes depending on the size of the catalog.

Default PostgreSQL Upgrade Procedure

The default method of upgrading the database requires the Director to be shut down during the upgrade. If your database is large and it is not possible to have a long downtime, please read the next section (**advancedpg**) about the advanced upgrade procedure, then contact the Bacula Systems support team for additional help.

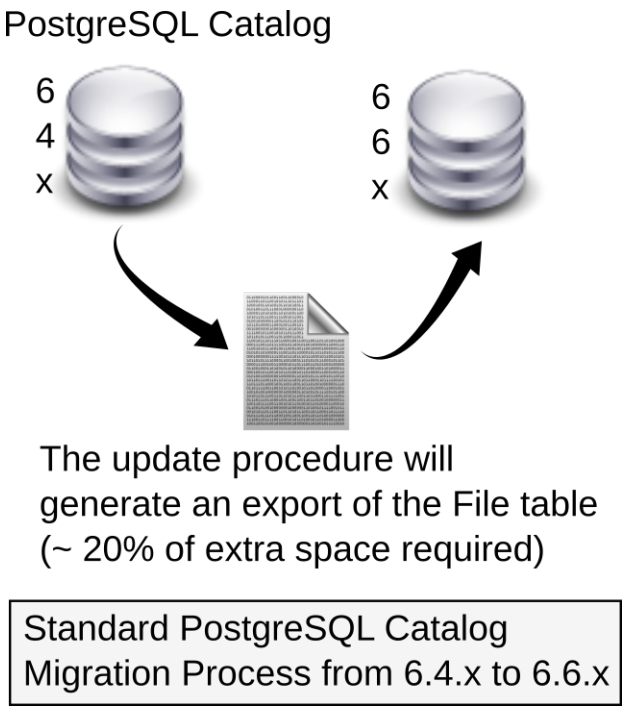


Fig. 2: Default PostgreSQL Upgrade Procedure

1. Install the `pigz` or the `lzop` compression program. This step is not mandatory, but using a parallel compression program (like `pigz` or `pbzip2`) will speed up the process. By default, the upgrade script will detect if `pigz`, `lzop` or `pbzip2` are installed. If neither are available, the upgrade script will use `gzip`.
2. Ensure you have enough free space on the catalog filesystem to hold a compressed copy of the File table (around 20% of the database size).
3. Stop any current version of Bacula from running.
4. Make sure you have a copy of your existing database.

5. Install the **12.8**.x version of Bacula.
6. If the install didn't upgrade your database automatically, run the upgrade script described in the **runupgrade** section of this document.
7. Start the new Bacula daemons. If everything worked correctly, no error messages should be printed.

In testing, a server with 16GB of RAM, RAID5 storage, and a 5GB File table with 17 million records produced 1.3GB of compressed dump and took 25 minutes to migrate the catalog format from 1015 to 1016.

The Bacula Systems installation packages (rpm, deb, ...) will try to dump the Catalog before the actual upgrade, so if you don't want to run the dump step, you can manually upgrade the catalog *before* installing the packages then skip the post-install step, which normally does the catalog upgrade when installing packages. (Possible with RPMs using the `-nopost` option).

Advanced Upgrade Procedure from 1015 to 1016

The advanced upgrade procedure permits running the biggest part of the catalog upgrade 1015 to 1016 while Bacula backup and restore jobs are running. Please, contact the Bacula Systems support team to evaluate your needs and to prepare the migration with you.

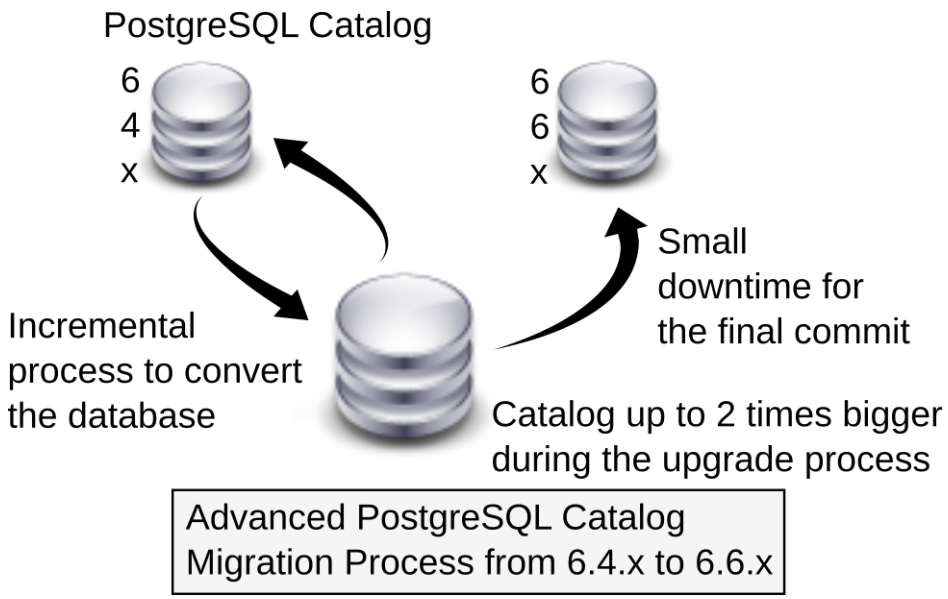


Fig. 3: Advanced PostgreSQL Upgrade Procedure

1. If upgrading from 6.2.x, 6.0.x or a previous version, first upgrade to 6.4.x.
2. Install the perl DBI module and the perl DBD::Pg (perl-DBI and perl-DBD-Pg on Redhat or libdbi-perl and libdbd-pg-perl on Debian/Ubuntu).
3. Ensure you have enough free space on the catalog filesystem to hold a copy of the File table (you can use `\d+` psql command to display table size). The upgrade process will probably require between 50% and 100% of the catalog size of free space where the database is stored before to start.
4. Save a copy of your existing database.
5. Follow the section **advpg2** to run the migration script
6. Install the **12.8**.0 version of Bacula.
7. If not done automatically, upgrade the catalog form from 1016 to 1020

8. Start the new Bacula daemons. If everything worked correctly, no error messages should be printed.

Notes on Directives and Commands

Deprecated Directives

The "Allow Higher Duplicates" directive is now deprecated. It did not work as documented and was confusing.

The "Dedup Index Directory" File Daemon directive is no longer used to store information. Instead, the new "Client Rehydration" directive enables the "restore cache" feature. Please see the Global Endpoint Deduplication user's guide for more information.

Changed Default Directives

Since version 6.0.0, the default for "Allow Duplicate Jobs" has been changed from no to yes.

Since version 8.6.4, the default for "Heartbeat Interval" has been changed from 0 (disabled) to 300 seconds.

Since version 8.8.0, restricted Console users must modify their existing Console resources to specify Directory-ACL=*all* and UserIdAcl=*all* in order to continue to use the restore command. If these two directives are not set, the restore command will return an error.

Since version 12.0, all network communication are encrypted by default. To disable encryption, use `TLS Enabled=no`.

Changed Command Option

The default for the "setbandwidth limit" option has been changed from kb/s to b/s. The option now accepts a speed suffix such as "kb/s", "mb/s".

Suse Enterprise Linux Changes

On Suse Linux, the Director and the Storage Daemons are now started with the unix account "bacula". To upgrade a Suse system, the ownership of the Bacula volumes and the configuration files must be changed to "bacula:bacula" and the PostgreSQL/MySQL configuration may also have to be adapted.

```
chown -R bacula:bacula /opt/bacula/archives
chown -R bacula:bacula /opt/bacula/working

# If needed
chown bacula:bacula /opt/bacula/etc/bacula-dir.conf
chown bacula:bacula /opt/bacula/etc/bacula-sd.conf
```


Package Changes

Since version 16.0, The `bacula-enterprise-client` RPM package does not provide the `bconsole` program anymore. The program is available separatly in the `bacula-enteprise-console` RPM package.

Since version 10.2.0, Cloud packages are distributed with specific packages for each driver (S3, Google, Oracle, and Azure). To upgrade from a previous version, it is recommended to remove the previous `bacula-enterprise-cloud-storage` package and install the specific cloud type packages required.

Checklist

Whether you are migrating to Bacula Enterprise from a different product or a prior, possibly community Bacula version, we recommend that you take your time before implementing any production Bacula backup system.

If you haven't already done so, we recommend that you read Disk Backup Design article.

If you follow the instructions in this chapter, you will have covered most of the major problems that can occur. It goes without saying that if you ever find that we have left out an important point, please inform us, so that we can document it to the benefit of everyone.

Critical Items

The following assumes that you have installed Bacula, you more or less understand it, you have at least worked through the tutorial or have equivalent experience, and that you have set up a basic production configuration. If you haven't done the above, please do so and then come back here. The following is a sort of checklist. In most cases, you will find additional details elsewhere in the manual. The order is more or less the order you would use in setting up a production system (if you already are in production, please use the checklist anyway).

- If you have a Bacula Systems subscription, we encourage you to contact us prior to upgrading so that we may schedule your upgrade when our support personnel are available. Most of us are in the CET timezone, and upgrading is best not done on a Friday. By scheduling it, you can be sure we will be available in case you have any unusual problems.
- If you are upgrading, follow the instructions at the beginning of this white paper (**upgradeoverview**).
- If you use tapes for backup, walk through all of the steps in the "Testing Your Tape Drive With Bacula" chapter of the Problem Resolution Guide. It may take you a bit of time, but it will eliminate any surprises.
- Do at least one restore of files. If you backup multiple OS types (Linux, Solaris, HP, MacOS, FreeBSD, Win32, ...), restore files from each system type.
- Write a bootstrap file to a separate system for each backup job. The "Write Bootstrap" directive is described in the manual, and more details are available in the "The Bootstrap File" chapter of the Bacula Main Reference Guide. Also, the default bacula-dir.conf comes with a "Write Bootstrap" directive defined. This allows you to recover the state of your system as of the last backup.
- Backup your catalog. An example of this is found in the default bacula-dir.conf file. The backup script is installed by default and should handle any database, though you may want to make your own local modifications.
- Write a bootstrap file for the catalog. An example of this is found in the default bacula-dir.conf file. This will allow you to quickly restore your catalog in the event it is wiped out – otherwise it may take many hours of work to recover.
- Make a copy of the bacula-dir.conf, bacula-sd.conf, and bacula-fd.conf files that you are using on your server. Put them it in a safe place (on another machine) as these files can be difficult to reconstruct if they are lost or damaged.

- Decide whether or not you need a bare metal recover, then either do it yourself or see about getting the Bacula Enterprise Bare Metal Restore programs for Linux and Windows.
- Bacula assumes all filenames are in UTF-8 format. This is important when saving the filenames to the catalog. For Win32 machines, Bacula will automatically convert from Unicode to UTF-8, but on Unix, Linux, *BSD, and Apple OS X machines, you must explicitly ensure that your locale is set properly. Typically this means that the bf LANG environment variable must end in **.UTF-8**. An full example is **en_US.UTF-8**. The exact syntax may vary a bit from OS to OS, and exactly how you define it may also vary.

  On most modern Win32 machines, you can edit the conf files with **notepad** and choose output encoding UTF-8.

Examples

You should use the following commands only after installing the new version of Bacula, unless you want to explicitly skip the automatic backup and upgrade procedure due to a complex setup, for example.

On Redhat with PostgreSQL catalog backend, you must run the update scripts as the `postgres` Unix user.

```
# su - postgres
$ cd /opt/bacula/scripts
$ ./update_bacula_tables
$ ./grant_bacula_privileges
...
```

On Debian/Ubuntu, scripts can run as `bacula` Unix user. Further it is necessary to specify the database credentials through the shell variables such as **db_user** and sometimes editing the script to set the variable **db_password**.

```
# su - bacula
$ cd /opt/bacula/scripts
$ ./update_bacula_tables
$ db_user=bacula ./grant_bacula_privileges
...
```

The `update_bacula_tables` script handles multiple catalog versions in a single execution beginning with catalog version 12. For example a single execution of the script will permit upgrading from Bacula Enterprise 6.0.x to **16.0**.

Example : Upgrade from 10.x to 16.0.x

From the version table shown above, we can see that 10.0 and 10.2 uses catalog version 1020 and version **16.0** is **1026**. The following procedure will upgrade your MySQL or PostgreSQL catalog:

```
$ cd /opt/bacula/scripts    # or possibly bacula/updatedb
$ ./update_bacula_tables
$ ./grant_bacula_privileges
```


Example : Upgrade from 8.10.x to 16.0.x

From the version table shown above, we can see that 8.10 uses catalog version 1019 and version **16.0** is **1026**. The following procedure will upgrade your MySQL or PostgreSQL catalog:

```
$ cd /opt/bacula/scripts    # or possibly bacula/updatedb
$ ./update_bacula_tables
$ ./grant_bacula_privileges
```

Example : Upgrade from 6.4.x to 16.0.x

From the version table shown above, we can see that 6.4.x uses catalog version 1015, and version **16.0**.x is **1026**. The following procedure will upgrade your Postgresql catalog by applying 4 separate upgrade scripts:

```
$ cd /opt/bacula/scripts    # or possibly bacula/updatedb
$ ./update_bacula_tables
$ ./grant_bacula_privileges
```

The time and the space needed to run the upgrade from version 1015 to 1016 depends on your catalog size. As Bacula must be shutdown during this upgrade, you may want to perform it during non-working hours.

Typical output from the **./update_bacula_tables** script will look like the following:

```
./update_bacula_tables
Altering postgresql tables

This script will update a Bacula PostgreSQL database from version
 12-14,1014-1024-1025 to 1026 which is needed to convert from
 Bacula Enterprise version 4.0.x to 16.0.x
Dumping File table to /opt/bacula/scripts/file1016.data.
The process may fail if the current user
 doesn't have write permission on the current directory,
 or if the system doesn't have enough space to store a
 compressed export of the File table
BEGIN
DROP TABLE
DROP TABLE
CREATE TABLE
COMMIT
Loading the File table from /opt/bacula/scripts/file.3341.data...
date
Creation of indexes and PK on the File table...
SET
BEGIN
CREATE INDEX
CREATE INDEX
psql:-:6: NOTICE:  ALTER TABLE / ADD PRIMARY KEY will create implicit index "file_pkey"
for table "file"
ALTER TABLE
CREATE SEQUENCE
ALTER SEQUENCE
 setval
```

(continues on next page)

(continued from previous page)

```
-----------
230909040
(1 row)
ALTER TABLE
ANALYZE
ALTER TABLE
ALTER TABLE
psql:-:29: NOTICE:  CREATE TABLE will create implicit sequence "snapshot_snapshotid_seq"
for serial column "snapshot.snapshotid"
psql:-:29: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "snapshot_pkey"
for table "snapshot"
CREATE TABLE
CREATE INDEX
UPDATE 1
COMMIT
BEGIN
CREATE TABLE
CREATE INDEX
UPDATE 1
COMMIT
Upgrade of the File table succeeded.
SET
```

Example : Upgrade from 6.0.x to 16.0.x

From the version table shown above, we can see that 6.0.x uses catalog version 1014, and version **16.0**.x is **1026**. The following procedure will upgrade your Postgresql catalog by applying three separate upgrade scripts:

```
$ cd /opt/bacula/scripts    # or possibly bacula/updatedb
$ ./update_bacula_tables
$ ./grant_bacula_privileges
```

The time and the space needed to run the upgrade from version 1015 to 1016 depends on your catalog size. As Bacula must be shutdown during this upgrade, you may want to run it during non-working hours.

Example : Upgrade from 2.6.x to 16.0.x

In the above table, we can see that 2.6.x uses catalog version 11, and version **16.0**.x uses **1026**. The following procedure will upgrade your Postgresql catalog in two steps.

```
$ cd /opt/bacula/scripts
$ ./update_postgresql_tables_11_to_12
$ ./update_postgresql_tables
$ ./grant_bacula_privileges
```

During the upgrade to database version 12, you may see errors about `file_fp_idx` and `file_jpfid_idx` indexes, but they can be ignored.

You may change the script parameters (or edit it) to fit your security or installation. (All extra command line parameters are passed to the database interpreter). In this example, the database server is located on host “192.168.0.1”


```
$ cd /opt/bacula/scripts    # or possibly bacula/updatedb
$ ./update_postgresql_tables_11_to_12 -h 192.168.0.1
$ cd /opt/bacula/scripts    # or possibly bacula/src/cats
$ ./update_postgresql_tables -h 192.168.0.1
$ ./grant_bacula_privileges -h 192.168.0.1
```

Example : Upgrade from 2.6.x to 16.0.x with MySQL

For a MySQL database, the procedure is very similar the PostgreSQL upgrade described above:

```
$ cd /opt/bacula/scripts    # or possibly bacula/updatedb
$ ./update_mysql_tables_11_to_12
$ cd /opt/bacula/scripts    # or possibly bacula/src/cats
$ ./update_mysql_tables -u bacula
$ ./grant_bacula_privileges
```

Example : Upgrade from 2.0.x, 2.2.x or 2.4.x to 16.0.x

From the version table shown above, we can see that 2.4.x uses catalog version 10, and version **16.0**.x is **1026**. The following procedure will upgrade your Postgresql catalog by applying four separate upgrade scripts:

```
$ cd /opt/bacula/scripts    # or possibly bacula/updatedb
$ ./update_postgresql_tables_10_to_11
$ ./update_postgresql_tables_11_to_12
$ cd /opt/bacula/scripts    # or possibly bacula/src/cats
$ ./update_postgresql_tables
$ ./grant_bacula_privileges
```

The time needed to run the upgrade from version 10 to 11 and 1015 to **1026** depends on your catalog size. As Bacula must be shutdown during this upgrade, you may want to perform it during non-working hours.

Example: Advanced PostgreSQL Upgrade from 6.4.x to 16.0.x

The advanced migration script is designed to convert a 6.4.x schema to **16.0**.x. You may need to convert your catalog from a previous version first up to 1015, please see Section **stop1015**.

As the upgrade from 6.4.x (1015) to 6.6.x (1016) can take quite long time, we advise you to use run the command from a session started with "`screen`". In case of a network disconnection, the migration script will continue to run, and it is possible to re-attach the `screen` session using `screen -r` option.

```
$ screen
$ su - postgres
$ cd /opt/bacula/scripts
$ ./update_large_catalog_1015_1016 -D bacula
$ ./update_postgresql_tables
$ ./grant_bacula_privileges
```

The `update_large_catalog_1015_1016` script is designed to be interactive and it is possible to stop the process at almost anytime using Ctrl-C (of course, if the break is done in a middle of a query, the script will have to do the work again at the next run).


```
$ su - postgres
$ /opt/bacula/scripts/update_large_catalog_1015_1016 -D bacula
16:23:14 INFO: Connexion to the catalog OK
16:23:14 INFO: Found catalog version 1015
16:23:14 INFO: Your catalog is already pretty big.
       Make sure to stay in touch with Bacula Systems Support team
       during the catalog migration

       The best way to proceed with your Catalog is to create a second
       File table while your Bacula is running. This method requires
       storing at least 2 times the size of your database during the
       migration.

       PLEASE MAKE SURE YOUR CATALOG FILESYSTEM IS BIG ENOUGH!!

Filesystem              Size  Used Avail Use% Mounted on
/dev/md0                 70G   31G  34.7G 43% /
/dev/md1                978G  887G   42G  96% /home
/dev/sdb3               493G   14G  454G   3% /boot

  -- Press enter key to continue or press Ctrl-C to cancel ---
```

If your system doesn't have enough space to hold two copies of the File table, you will need to add more temporary space (for example using tablespace). If it is not possible to add extra space, the default migration procedure will be a better option. However, it will require stopping the Bacula Director during the migration (see Section *Example : Upgrade from 6.4.x to 16.0.x*).

```
16:33:16 INFO: Creating the temporary table 'nb_file_temp'
16:33:16 INFO: Creating the temporary File1016 table
16:33:16 INFO: Will update the temp File table with records
16:33:16 INFO: 2427 records have been inserted in the new table
16:33:16 INFO: Creating indexes if not already created
16:33:16 INFO: Now it's time to STOP bacula and start the final part of
               the upgrade

         Everything was committed to the database, so you can stop here
         and restart the process later.

  -- Press enter key to continue or press Ctrl-C to cancel ---
```

At this point, most of the work is done, the new table and all indexes are created. It is possible to stop here and re-run the script later, the script will ask if you want to continue the migration or restart from scratch.

For large databases, the first part of the migration can take up to 20 hours, it is advised to stop here and run the script again to reduce the final downtime.

Once the Bacula Director is stopped, press enter in the terminal and let the script finish the offline part of the migration.

```
16:39:34 INFO: Do a last update from file to temporary file table
16:39:34 INFO: 10 records have been inserted in the new table
16:39:34 INFO: Copying GRANT information
16:39:34 INFO: Dropping original File table
16:39:34 INFO: Restoring GRANT information for File sequence
16:39:34 INFO: Deleting orphan records
```

(continues on next page)

(continued from previous page)

```
16:39:34 INFO: 0 records deleted
16:39:34 INFO: You can now upgrade bacula itself with your package manager
```

Once done, install the new 8.2.x version of bacula using your package manager.

Example: Advanced Upgrade from 6.0.x to 16.0.x with PostgreSQL

To use the advanced migration script starting from a Catalog format prior to 1015 (6.4.x), it is possible to use the special `-stop1015` option of the `update_postgresql_tables` script. The `-stop1015` option will convert the catalog from an old revision up to the 1015 schema.

```
$ cd /opt/bacula/scripts
$ ./update_postgresql_tables --stop1015   # other options...
$ ./grant_bacula_privileges
```

Once done, follow instructions described in Section **advpg2**.

Upgrading a Major PostgreSQL Version

Standard Procedure

Note, this section applies to upgrading PostgreSQL rather than the Bacula table format.

Please note that the Bacula Systems QA team tests Bacula Enterprise with the default PostgeSQL version that comes with the RHEL, CentOS, Debian, SLES, and Ubuntu platforms. If you wish to run a more recent version of PostgreSQL or install the Catalog on another distribution, please contact the Bacula Systems Support for advice.

The standard procedure to upgrade PostgreSQL is described in https://www.postgresql.org/docs/10/static/upgrading.html It consists of making a SQL dump of your database, stopping it, installing the new version and reloading the SQL dump. This procedure can take a lot of time during which the database is unavailable. You can speed up the loading phase by changing the `postgresql.conf` file:

```
fsync = no
checkpoint_segments = 1G
checkpoint_timeout = 3000
checkpoint_completion_target = 0.9
autovacuum = off
```

These settings are not suitable for production. Please do not forget reset your production settings when the upgrade is finished.

At the end of the procedure, run the "ANALYZE" command.

Advanced Procedure

PostgreSQL has the capability to restore a database using multiple connections at the same time. To use this feature, the dump should be done using `pg_dump` and the custom format.

First, you need to dump all roles:

```
postgres$ pg_dumpall --globals > /tmp/roles.sql
```

Then, you can to dump all databases:

```
postgres$ psql -AtU postgres -c "SELECT datname FROM pg_database WHERE NOT datistemplate
↪"| \
while read f;
   do pg_dump -Upostgres --format=c --file=/tmp/$f.sqlc $f;
done;
```

In this example, dumps will be generated in /tmp. Feel free to change this location. Now you can install and start the new PostgreSQL version, and restore roles:

```
postgres$ psql -f /tmp/roles.sql
```

Finally, you can restore each database with the `pg_restore` command, ex:

```
postgres$ pg_restore -j 4 -C -d postgres /tmp/bacula.sqlc
```

Using the pg_upgrade Tool

This method is very interesting because the migration is very fast compared to other methods.

`pg_upgrade` (or `pg_migrator` supports upgrades from/to 8.3.X and 8.4.X, including snapshot and alpha releases. For upgrading to 9.0.X or later, use the `pg_upgrade` tool that is part of /contrib. To use `pg_upgrade`, you need to install it from the postgresql contrib directory.

```
$ cd <postgresql source>/contrib/pg_upgrade
$ make install
$ cd ../pg_upgrade_support
$ make install
```

To continue, you need to define some variables about your installation.

```
postgres$ export NEWBINDIR=/opt/pg9.0/bin
postgres$ export NEWDATADIR=/opt/pg9.0/data
postgres$ export OLDBINDIR=/opt/pg8.4/bin
postgres$ export OLDDATADIR=/opt/pg8.4/data
```

Your catalog should be stopped to run the upgrade. The `pg_upgrade` tool will guide you if something is misconfigured.

```
postgres$ /opt/pg9.0/bin/pg_upgrade
Performing Consistency Checks
-----------------------------
Checking old data directory (/var/lib/postgres/data)        ok
Checking new data directory (/tmp/pg9/DATA)                 ok
Checking for /contrib/isn with bigint-passing mismatch      ok
```

(continues on next page)

(continued from previous page)

```
... skip ...
Setting next oid for new cluster                            ok
Creating script to delete old cluster                       ok
Checking for large objects                                  ok

Upgrade complete
```

Important Note After Upgrading

In all cases, you need to run the ANALYZE command after the upgrade. Be sure to adapt the new postgresql.conf file to your previous tuning.

See also:

Go to *Bacula Enterprise Upgrade Procedure*.

Go back to the *Bacula Enterprise Upgrade on Linux* chapter.

Go back to the *Bacula Enterprise Upgrade* chapter.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

See also:

Go to *Bacula Enterprise Upgrade on Windows*.

Go back to the *Bacula Enterprise Upgrade* chapter.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

1.2 Bacula Enterprise Upgrade on Windows

If you wish to upgrade the FD running on your Windows system, use the same method as the one used to install it:

- Download the new Bacula executable from your download area and follow the steps described in Install FD with Windows Installer.
- Follow the installation instruction with BIM in Windows: Install File Daemon (Client).

See also:

Go to *Bacula Enterprise Upgrade on Linux*.

Go back to the *Bacula Enterprise Upgrade* chapter.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

2 Bacula Enterprise Removal

The following chapter aims at explaining how to remove Bacula Enterprise. It is organized by operating systems.

Important: Following the below procedures will remove all the Bacula Enterprise packages, the Catalog database and backup volumes. Please make sure you understand all the backups will be deleted.

Choose the operating system from which you wish to remove Bacula:

- Bacula Enterprise Removal on Linux (Director, Storage Daemon and Client)
- Bacula Enterprise Removal on Windows (Client only)

2.1 Linux: Complete Removal

The following article aims at explaining how to remove Bacula Enterprise from your Linux system.

Following this guide all Bacula components will be uninstalled, including the Catalog database and backup volumes.

Stopping Bacula Services

Before uninstalling Bacula Enterprise packages stop all Bacula Daemons first.

```
systemctl stop bacula-dir
systemctl stop bacula-sd
systemctl stop bacula-fd
```

Remove Catalog Database

The Catalog (Bacula database) service is responsible for maintaining the bacula database for all the backed up files. Without the Catalog, you are unable to restore files using a restore job. By default, the Catalog is installed on the Bacula Director, but it can also be hosted on a dedicated SQL Server.

Bacula Enterprise provides a script to drop the Catalog from the SQL Server.

If you are using PostgreSQL, change to a **postgres** user before running the script.

```
su - postgres
```

Execute the script to drop the Catalog.

```
/opt/bacula/scripts/drop_bacula_database
```


Remove Disk-Based Volumes

Volume is an archive unit where Bacula stores backed up data. Once volumes are removed, all backed up data will be lost. Disk-based Bacula volumes are located in a location pointed to by Storage Daemon Devices through the "Archive Device" directive.

You can quickly find all the locations Bacula volumes are stored in a bsys_report of your Storage Daemon(s).

```
zgrep -i "archive*device" /opt/bacula/working/bsys_report.*.gz | sort -u
```

Important: Do make sure there is no data you may need inside the **ArchiveDevice** directories before removing it.

Remove all the content of directories **ArchiveDevice** points to.

```
rm -rf @@archive-device@@
```

@@**archive-device**@@ refers to the directories Storage Daemon Devices point to as found by the command above.

Remove Bacula Packages

You will uninstall Bacula Enterprise packages using your OS package manager. We provide instructions for **deb** and **rpm** based Operating Systems.

Linux: Bacula Enterprise Removal on Debian/Ubuntu

Use the Package Manager to remove all Bacula Enterprise packages

```
apt-get remove bacula-enterprise*
```

See also:

Go to:

- PackageManagerBERHELCentOS
- *Linux: Client (FD) Removal*

Go back to Linux: Complete Removal.

Go back to Bacula Enterprise Removal.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

Linux: Bacula Enterprise Removal on on RHEL/CentOS

Use the Package Manager to remove all Bacula Enterprise packages

```
yum remove bacula-enterprise*
```

See also:

Go to

- BEDebianUbuntuUninstall
- *Linux: Client (FD) Removal*

Go back to Linux: Complete Removal.

Go back to Bacula Enterprise Removal.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

Linux: Client (FD) Removal

The following article aims at explaining how to remove Bacula Enterprise Client from your Linux system.

It is important to avoid uninstalling only the Client on a system that is running the Director and the Storage Daemon. The Client serves as a crucial dependency for both components.

Following this guide, the Bacula Client components will be uninstalled.

Stopping Bacula Client Service

Before uninstalling Bacula Enterprise Client package, stop Bacula Client Daemon first.

```
systemctl stop bacula-fd
```

Remove Bacula Packages

You will uninstall Bacula Enterprise Client package using your OS package manager. We provide instructions for **deb** and **rpm** based Operating Systems.

```
apt-get remove bacula-enterprise-fd
```

```
yum remove bacula-enterprise-fd
```

Remove Bacula Installation Directories

Important: Do make sure there is no data you may need inside the **/opt/bacula** directories before removing it.

Remove all the content of Bacula Enterprise Client.

```
rm -rf /opt/bacula
```

See also:

Go to:

- BEDebianUbuntuUninstall
- PackageManagerBERHELCentOS

Go back to Bacula Enterprise Removal.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

Remove Bacula Installation Directories

Important: Do make sure there is no data you may need inside the **/opt/bacula** and **/opt/bweb** directories before removing it.

Remove all the content of Bacula Enterprise and Bweb directories.

```
rm -rf /opt/bacula
rm -rf /opt/bweb
```

See also:

Go to BEUninstallWindows.

Go back to Bacula Enterprise Removal.

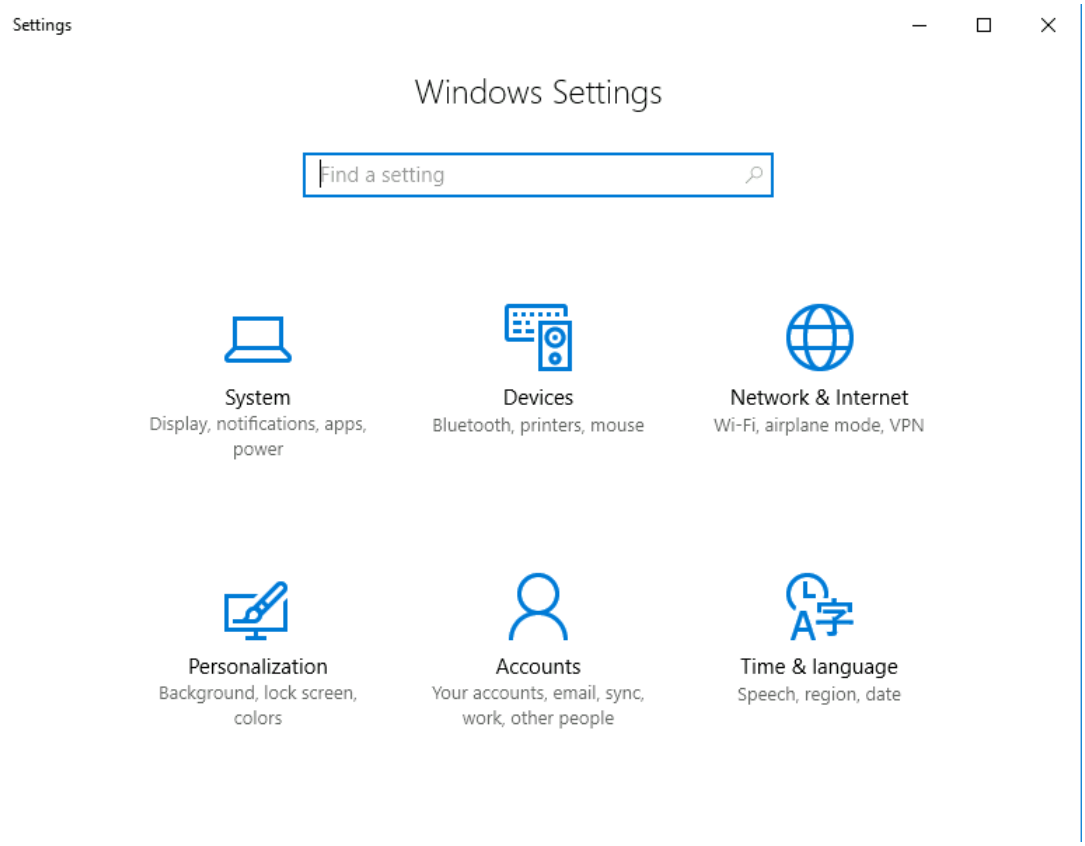
Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

2.2 Windows: Remove Bacula Enterprise Packages

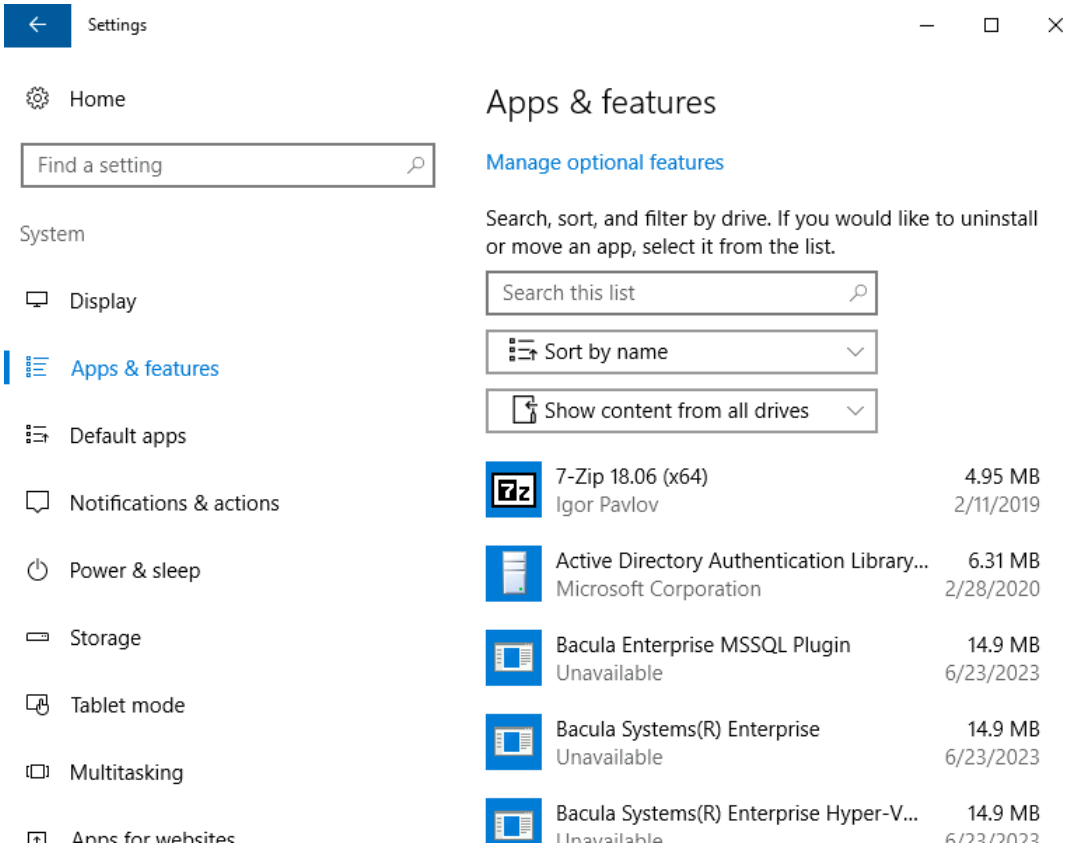
To uninstall Bacula Enterprise packages on Windows we will use Windows Settings application.

Steps

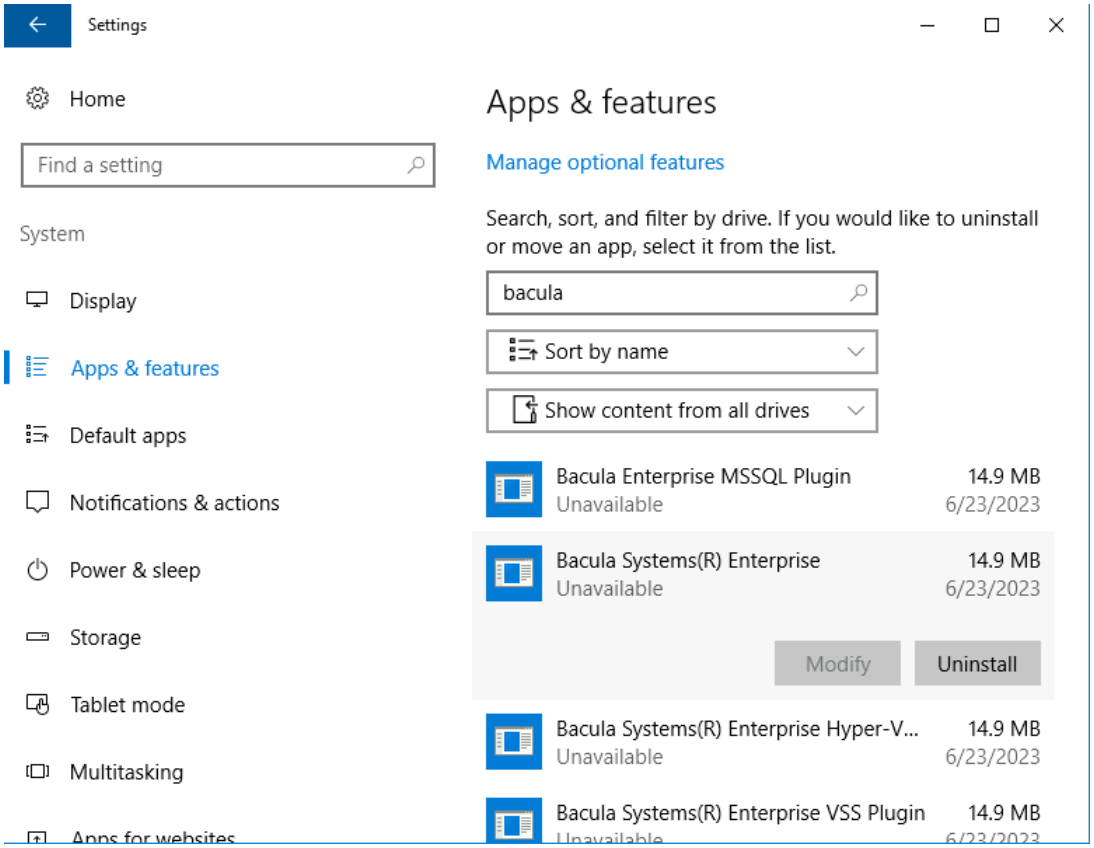
1. Open Windows Settings Application from Start Menu.



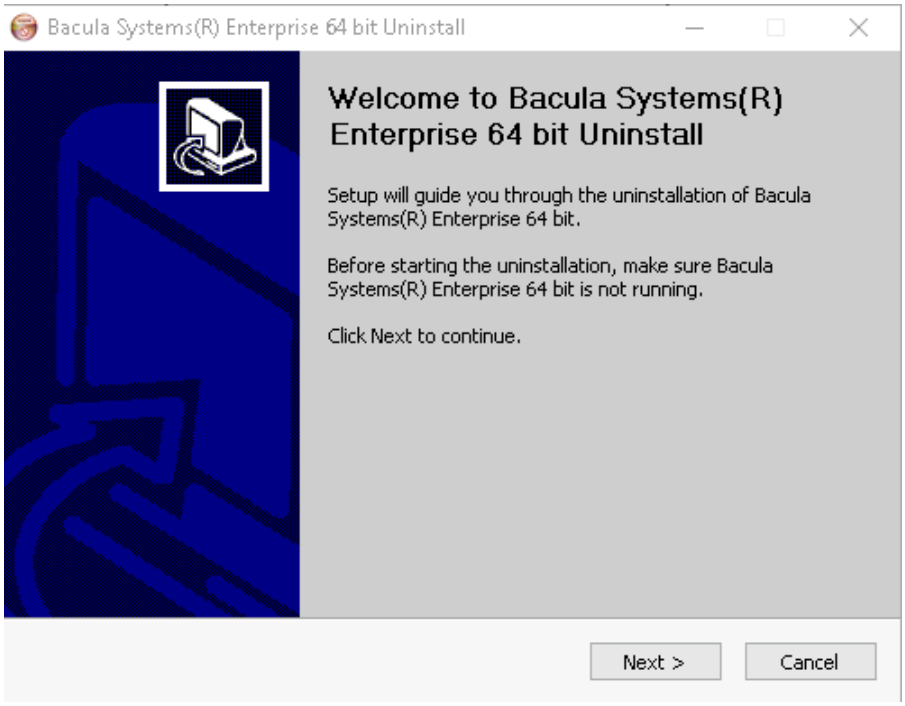
2. Navigate to “Apps & features”.



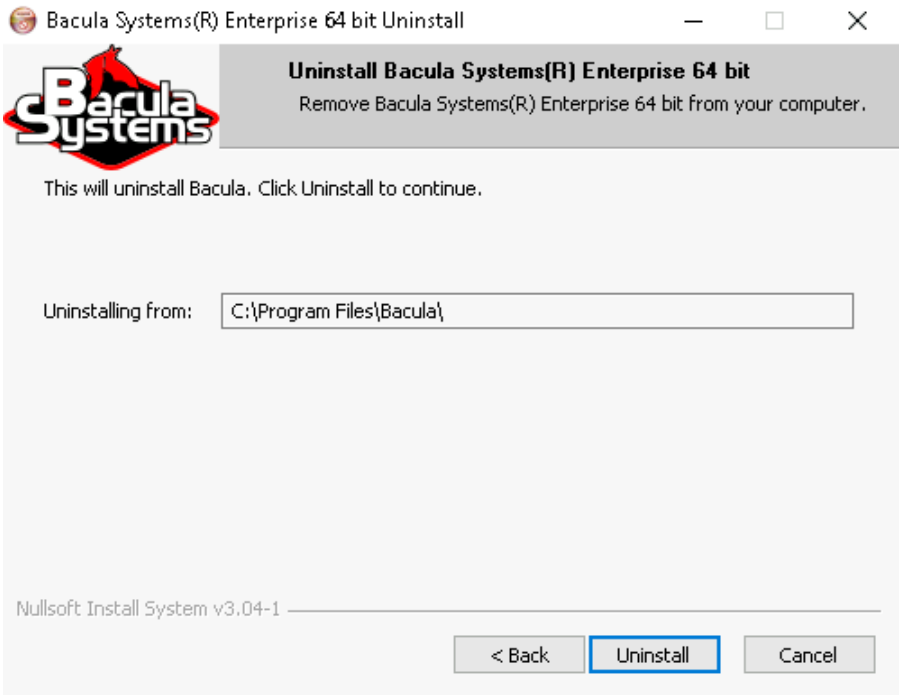
3. Click on “Bacula Systems(R) Enterprise” item and click on “Uninstall” button. Confirm you want to uninstall packages.



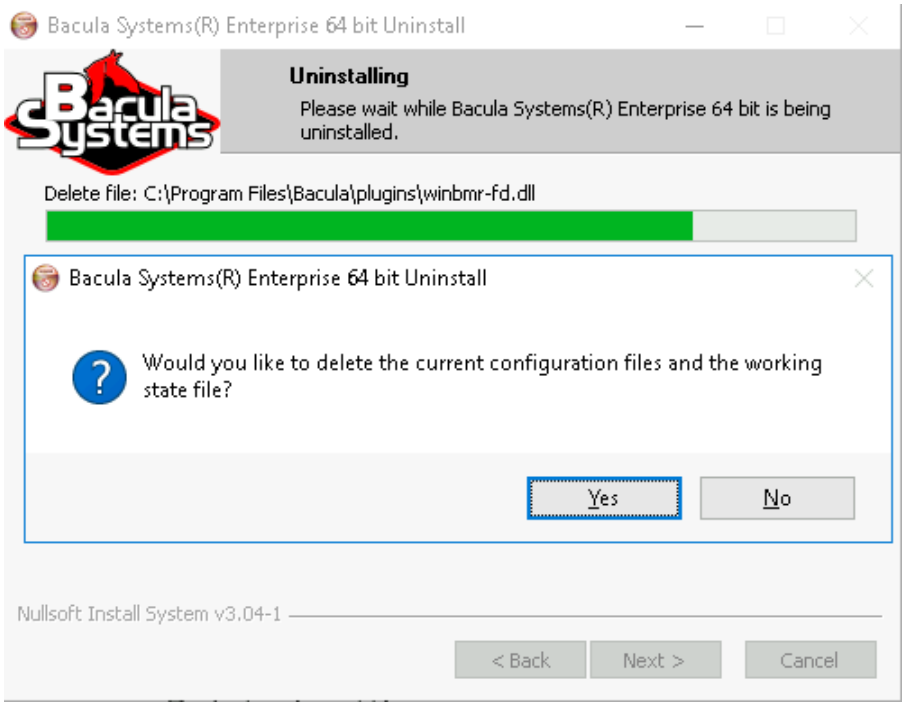
4. The Bacula packages uninstall wizard will start.



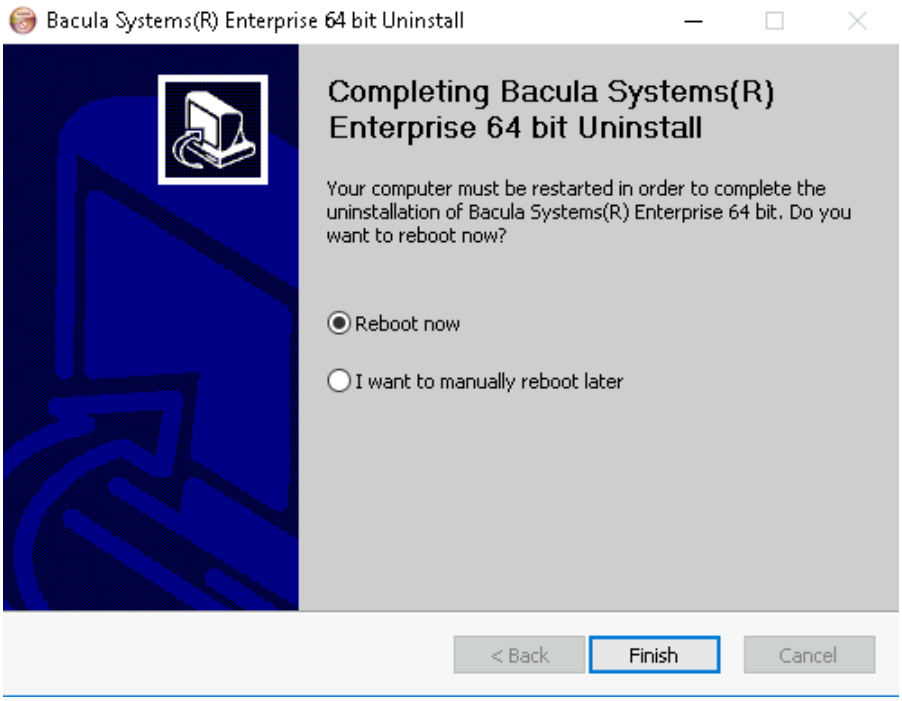
5. Click on the “Uninstall” button.



6. When prompted confirm to remove the configuration and working files.



7. Reboot the computer to finalize the deinstallation.



See also:

Go to BEUninstallLinux.

Go back to Bacula Enterprise Removal.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.

Go back to the *Bacula Enterprise Upgrade and Removal* chapter.