MSSQL Plugins

Bacula Systems Documentation

Contents

Contents

The MSSQL VDI (Virtual Device Interface) and MSSQL VSS (Volume Shadow Copy Service) Plugins for Bacula Enterprise are both designed to facilitate backups of Microsoft SQL Server databases, but they operate differently and offer distinct features.

They seamlessly integrate with Bacula Enterprise's scheduling, cataloging, and management functionalities, enabling automated and uniform backup operations. Each plugin guarantees that SQL Server databases are preserved in a consistent state, thereby mitigating the risk of data corruption and ensuring dependable restores. Furthermore, both plugins are tailored for optimal performance with Microsoft SQL Server, utilizing various technologies to meet backup and recovery objectives.

- *MSSQL VDI Plugin* offers:
- Direct interaction with SQL Server via VDI.
- Full, differential, and log backups.
- High-performance backup operations.
- Granular control over backup types and restoration processes.

Thus, MSSQL VDI Plugin is best suited for environments that are heavily focused on SQL Server, offering fine-grained control over backup types, including differential and log backups.

- *MSSQL VSS Plugin* offers:
- Volume Shadow Copy Service used for backup.
- System-wide consistency ensured across multiple services.
- Full database and log backups supported via snapshots.
- Backup management simplified in mixed application environments.

Thus, MSSQL VSS Plugin is ideal for mixed environments where multiple applications might be sharing storage, as it coordinates the backup across different services beyond just SQL Server, ensuring overall system consistency.

1 MSSQL VDI plugin

1.1 Overview

This user's guide presents how to use the MSSQL Plugin feature with **Bacula Enterprise**.

1.2 Features Summary

Bacula Systems provides a plugin for Microsoft SQL Server for **Bacula Enterprise** named `mssql-fd.dll`. The MSSQL Plugin provides the following main features:

- Full and Differential support
- Incremental (Log) level support
- Database level backup
- Ability to include/exclude databases from the backup job
- Support for "Copy Only" backups
- Restore MSSQL backup files to disk
- Send the backup stream directly to the Storage Daemon without requiring much local disk space.
- *Point in time recovery restore*

The MSSQL VDI plugin is compatible with Copy/Migration jobs. Please read the CopyMigrationJobsReplication for more information.

1.3 Scope

This document will present solutions for **Bacula Enterprise** 8.4 and later, which are not applicable to prior versions. The MSSQL Plugin has been tested and is supported from Windows Server 2003 R2 up to Windows Server 2019 and from Microsoft SQL Server 2005 up to 2019.

1.4 Installation

You must ensure that the `mssql-driver.dll` is in the Bacula program directory and the `mssql-fd.dll` plugin is in the plugins directory on the FD doing the backup (done by default with the installer), and that the `Plugin Directory` directive line is present and enabled in the FD's configuration file `bacula-fd.conf`. An example configuration file and status output of a client with the MSSQL plugin available is shown below.

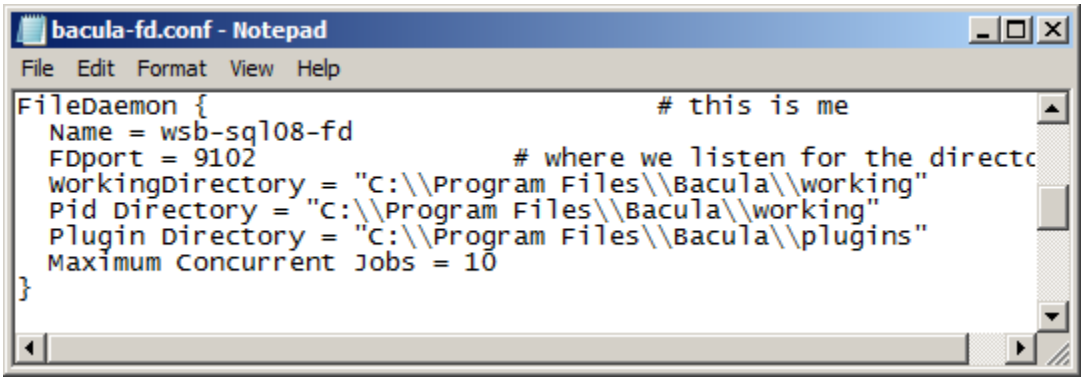

Fig. 1: File Daemon Configuration Excerpt with "Plugin Directory" Line

```
*status client
Automatically selected Client: win2008-fd
Connecting to Client win2008-fd at win2008-64-r2:9102

win-fd Version: 8.4.3 (30 November 2015)  VSS Linux Cross-compile Win64
Daemon started 11-Dec-15 05:13. Jobs: run=1 running=0.
Microsoft Windows Server 2008 R2 Standard Edition (build 7600), 64-bit
 Heap: heap=0 smbytes=348,013 max_bytes=481,620 bufs=134 max_bufs=162
 Sizes: boffset_t=8 size_t=8 debug=0 trace=1 mode=0,2010 bwlimit=0kB/s
 Plugin: alldrives-fd.dll mssql-fd.dll
```

If the SQL Server database is running under an account that is not `NT AUTHORIZED/SYSTEM`, it will be mandatory to configure the SQL Server instance to allow the Bacula File Daemon service account to connect and perform backup operations. By default, the Bacula File Daemon service runs under the `NT AUTHORIZED/SYSTEM` account.

The permission `sysadmin` can be granted with the following SQL command:


```
ALTER SERVER ROLE [sysadmin] ADD MEMBER [NT AUTHORITY\SYSTEM]
```

1.5 Configuration

Note: In case of a clustered MSSQL environment, or AlwaysOn Availability Groups, the Bacula Enterprise MSSQL VDI plugin will not automatically detect the configuration. Precautions need to be taken on both the MSSQL Server and Bacula Plugin side to make sure databases are adequately protected in such an environment.

Please contact Bacula Systems (support@baculasystems.com) for help on configuring the MSSQL VDI plugin if in doubt.

To activate the MSSQL plugin you have to put the following into the Include section of the File Set which will be used to back up the SQL Server data:

```
Plugin = "mssql"
```

This will back up all SQL server databases (`tempdb` is excluded by default).

The plugin directive must be specified exactly as shown above.

If everything is set up correctly as above then the backup will include the SQL server data. The SQL server data files backed up will appear in a **bconsole** or **bat** restore as follows:

```
/@mssql/MSSQLSERVER/master/data.bak
/@mssql/MSSQLSERVER/production/data.bak
...
etc
```

It is possible to select different instances or databases to be backed up with the following parameters:

- `instance`
- `database`
- `include`
- `exclude`

Full, Differential and Logs (Incremental) backups are supported.

Following the creation of a new database, you should run a Full backup of the SQL server data. A new database will not be backed up at a different level, and a Differential backup will automatically be upgraded to a Full backup on this specific new database.

The MSSQL Plugin does not use VSS snapshots to perform the backup so, unless some disk folder is present in the fileset, **Enable VSS** can be set to "no".

A complete example of the Job setup for MSSQL Server data is shown below:

```
FileSet {
  Name = MSSQL
  Enable VSS = no      # VSS is not required
  Include {
    Options {
      Signature = SHA1
    }
```

(continues on next page)

(continued from previous page)

```
    Plugin = "mssql: database=production"
  }
}

Job {
  Name = MSSQL08
  File Set = MSSQL
  Client = wsb-sql08-fd
  Job Defs = DefaultJob
  Level = Differential
}
```

```
FileSet {
  Name = MSSQL09
  Enable VSS = no      # VSS is not required
  Include {
    Options {
      Signature = SHA1
    }
    Plugin = "mssql: abort_on_error exclude=test1 exclude=test2 copyonly"
  }
}

Job {
  Name = MSSQL09
  File Set = MSSQL09
  Client = wsb-sql08-fd
  Job Defs = DefaultJob
  Level = Full
}
```

```
FileSet {
  Name = MSSQL10
  Enable VSS = no      # VSS is not required
  Include {
    Options {
      Signature = SHA1
    }
    Plugin = "mssql: include=test2 include=prod1 include=r7*"
  }
}
Job {
  Name = MSSQL10
  File Set = MSSQL10
  Client = wsb-sql08-fd
  Job Defs = DefaultJob
  Level = Full
}
```

```
FileSet {
  Name = SQLExpress
```

(continues on next page)

(continued from previous page)

```
  Enable VSS = no      # VSS is not required
  Include {
    Options {
      Signature = SHA1
    }
    Plugin = "mssql: instance=SQLExpress hostname=."
  }
}

Job {
  Name = SQLExpress
  File Set = SQLExpress
  Client = wsb-sql08-fd
  Job Defs = DefaultJob
  Level = Differential
}
```

Attention: New in Bacula Enterprise 12.8.0

MS SQL Server Multi Instances Backup Job}footnote{Available in Bacula Enterprise 12.8.0}

```
FileSet {
  Name = SQLAllInstances
  Enable VSS = no      # VSS is not required
  Include {
    Options {
      Signature = SHA1
    }
    Plugin = "mssql: all_instances"
  }
}

Job {
  Name = SQLAllInstances
  File Set = SQLAllInstances
  Client = wsb-sql08-fd
  Job Defs = DefaultJob
  Level = Incremental
}
```

Attention: New in Bacula Enterprise 12.8.0

MS SQL Server 2 Instances Backup Job

```
FileSet {
  Name = SQL2Instances
  Enable VSS = no      # VSS is not required
  Include {
    Options {
      Signature = SHA1
    }
```



```
    Plugin = "mssql: instance=MSSQLSERVER instance=PRODUCTION"
  }
}

Job {
  Name = SQL2Instances
  File Set = SQL2Instances
  Client = wsb-sql08-fd
  Job Defs = DefaultJob
  Level = Incremental
}
```

1.6 Plugin Options

- **database=<glob>** Specifies the name of the databases to backup. This parameter is optional. By default, all databases (except `tempdb`) will be backed up.
- **include=<glob>** Specifies the names of databases to backup. It is possible to specify the `include` parameter multiple times on the plugin command line.

  If a database that was explicitly specified is not found, a warning or error message will be printed to the Job report. The **abort_on_error** option is adhered to.
- **exclude=<glob>** Specifies the names of databases to exclude from the backup. It is possible to specify the `exclude` parameter multiple times on the plugin command line.
- **user=<str>** The username used to connect to the MSSQL instance. If the user is part of a domain, please also specify the domain parameter below. If no domain is specified the user parameter will refer to a local account. This parameter is optional, and if not set, the `bacula-fd` service account will be used to connect to the MSSQL instance.
- **domain=<str>** The domain of the user used to connect to the MSSQL instance. This parameter is optional. If not set and no user is set, the `bacula-fd` service account will be used to connect to the MSSQL instance. If not set and a user is set, the user is assumed to be a local account.
- **password=<str>** The password used to connect to the MSSQL instance. This parameter is optional, and if set, the password might be printed in various places such as `status client` output, job log or debug messages.
- **passfile=<file>** The file where the password can be found. This parameter is optional. The plugin will use the first line (limited to 127 characters) as the connection password.
- **authtype=[windows | server]** The authentication type used to connect to the MSSQL instance. This parameter is optional. By default the `Windows` authentication type is used.
- **instance=<str>** The instance name used to connect to the MSSQL instance. This parameter is optional. By default, the instance name is “MSSQLSERVER”.

  > **Attention: New in Bacula Enterprise 12.8.0**
  >
  > Multiple `instance` parameters are allowed.

-

  > **Attention: New in Bacula Enterprise 12.8.0**

all_instances This parameter is optional. If set, the MSSQL plugin will list and backup all instances defined on the SQL Server. The same authentication settings should be available on each instance.

- **hostname=<str>** This parameter is optional. If set, the `hostname` string will be used in the `Server` connection string of the ODBC driver. If not set, the plugin will configure automatically the ODBC driver to use `(localdb)` or `(local)`. This advanced option might be used in conjunction with the `instance` option, often, it is necessary to specify "`.`" as `hostname` parameter. Ex: `instance=MYINSTANCE hostname=.`
- **abort_on_error** By default, if the plugin is not able to reach the MSSQL instance, or to find an explicitly named database, an error will be generated and the job will continue. Some users might prefer to abort the Job, which will happen if this option is set.
- **copyonly[=incremental|all]** A copy-only backup is a MSSQL backup that is independent of the sequence of conventional MSSQL backups. The next Differential or Transaction Log (Incremental) backup will not use it. If the `incremental` option is set, only Transaction Log (Incremental) backups will use the option. The last Incremental job will contain all the information necessary to perform a point in time recovery.
- **fullbackup** Force the level of the MSSQL backup. If specified in the plugin command line, a job can run with the incremental level for standard files, and the databases backed up with the plugin will always be backed up like with a Full backup. (Available since 8.4.11)
- **skipreadonly** Skip read only databases when the backup level is incremental (BACKUP LOG) and the last modification/creation time for tables and views is prior to the last backup. By default, read only databases are upgraded to Full. (Available since 8.6.20)
- **dblayout** Store each database layout as a RestoreObject in the Bacula Catalog with a Full backup. The layout can be displayed in `bconsole` with `list restoreobjects` command. (Available since 8.6.20)
- **lock_timeout=<int>** Use the `SET LOCK_TIMEOUT` before issuing queries. (Available since 8.6.20)
- **buffercount=<int>** Specifies the total number of I/O buffers to be used for the backup operation. You can specify any positive integer; however, large numbers of buffers might cause "out of memory" errors because of inadequate virtual address space in the Sqlservr.exe process. This parameter is optional and the default value is 10.
- **blocksize=<bytes>** Specifies the physical block size, in bytes. The supported sizes are 512, 1024, 2048, 4096, 8192, 16384, 32768, and 65536 (64 KB) bytes. This parameter is optional and the default is 65536.
- **maxtransfersize=<bytes>** Specifies the largest unit of transfer in bytes to be used between the MSSQL Server and the backup media. The possible values are multiples of 65536 bytes (64 KB) ranging up to 4194304 bytes (4 MB). This parameter is optional and the default is 65536.
- **connection_string=<str>** Specifies the ODBC connection string. This parameter is optional.
- **checksum=<No/Yes>**

Available since Bacula Enterprise 8.11.6

Use the `WITH CHECKSUM` option in the backup query. This parameter is optional and the default is no.

- **driver=<str>**

Available since Bacula Enterprise 10.2.3

The driver name used in the ODBC connection string. Default is SQL Server. If the `connection_string` option is set, it will overwrite this option.

- **target_backup_recovery_models=<str>**

Available since Bacula Enterprise 10.2.4

Only the databases with the specified recovery model are backuped. The other databases are skipped with warning. The parameter should be on of the following:

- `SIMPLE`
- `FULL`
- `BULK_LOGGED`

- **simple_recovery_models_incremental_action=<str>**

Available since Bacula Enterprise 10.2.4

Specifies the behavior when incremental backup is requested over a simple recovery model database from the following parameters:

- `make_full` : The incremental backup is automatically upgraded to full.
- `ignore_with_error` : The backup is skipped with an error (the error translated to a warning at the job level).
- `ignore` : The backup is skipped silently.

1.7 Backup

1.8 Full Backup

The Full backup saves the database files and MSSQL to provide complete protection against media failure. If one or more data files are damaged, media recovery can restore all committed transactions. In-process transactions are rolled back. The master and the mbdb databases are always backed up in this mode.

1.9 Differential Backup

A differential backup is based on the most recent, previous full database backup. A differential backup captures only the data that has changed since that full backup. When using the Differential backup feature, the backup chain is very critical. If for some reason, the Full backup used as referenced by MSSQL is not available, the Differential data will not be usable. The plugin uses different techniques to avoid this problem, so if a problem is detected, the Differential database backup might be automatically upgraded to a Full backup.

1.10 Transaction Log Backup

The "Transaction Log Backup" MSSQL feature is implemented as the "Incremental" level with Bacula. The database must be configured with the full recovery model or bulk-logged recovery model. If the database uses the simple recovery model, the MSSQL file will be truncated after each checkpoint. The full restore will be possible, but not the restore to a point in time. For more information, see https://msdn.microsoft.com/en-us/library/ms189275.aspx.

1.11 Point in Time Restore (PITR)

Point In Time Restore (PITR) requires the database to be configured with the full recovery model. If the database uses the simple recovery model, the MSSQL file will be truncated after each checkpoint. For more information, see https://msdn.microsoft.com/en-us/library/ms189275.aspx.

1.12 MSSQL Database Configuration

The master database must be backed up. If master is damaged in some way, for example because of media failure, an instance of MSSQL may not be able to start. In this event, it is necessary to rebuild master, and then restore the database from a backup. Only full database backups of master can be created. See https://technet.microsoft.com/en-us/library/aa213839%28v=sql.80%29.aspx for more information.

1.13 Restore

You can use all the regular ways to start a restore. However, you must make sure that if restoring differential data, the previous full backup is also restored. This happens automatically if you start the restore, in `bconsole`, using the restore options 5 or 12. In the file tree generated, you should mark either complete databases or databases instances.

1.14 Restore Options

It is possible to restore the data with different scenarios using the following options:

- The common restore "Where" parameter (**where=<path>**)
- The common restore "Replace" parameter (**replace=<never|always>**)
- The Plugin restore option accessible in the "Plugin Options" menu at the restore prompt (Item 13).

```
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /tmp/regress/working/127.0.0.1-dir.restore.9.bsr
Where:           c:/tmp
Replace:         Never
FileSet:         Full Set
Backup Client:   win2008-fd
Restore Client:  win2008-fd
Storage:         File
When:            2016-02-22 12:02:56
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (yes/mod/no): mod                        <--------------------
Parameters to modify:
     1: Level
     2: Storage
     3: Job
     4: FileSet
     5: Restore Client
     6: When
     7: Priority
     8: Bootstrap
```

(continues on next page)

(continued from previous page)

```
     9: Where
    10: File Relocation
    11: Replace
    12: JobId
    13: Plugin Options
Select parameter to modify (1-13): 13              <-----------------
Automatically selected : mssql: database=db29187
Plugin Restore Options
instance:            *None*
database:            *None*
username:            *None*
password:            *None*
domain:              *None*
hostname:            *None*
authtype:            *None*
recovery:            *None*              (yes)
stop_before_mark:    *None*
stop_at_mark:        *None*
stop_at:             *None*
restricted_user:     *None*              (no)
verify:              *None*              (no)
connection_string:   *None*
checksum:            *None*              (no)
driver:              *None*              (SQL Server)

Use above plugin configuration? (yes/mod/no): mod  <------------------
You have the following choices:
You have the following choices:
     1: instance (Instance used to restore)
     2: database (New database name)
     3: username (Username used for restore)
     4: password (Password used for restore)
     5: domain (Domain name of user (default to local))
     6: hostname (Server ODBC parameter (default to (local/localdb)))
     7: authtype (Authentication type (server or windows))
     8: recovery (Start Recovery)
     9: stop_before_mark (Stop the recovery before a mark (STOPBEFOREMARK). {lsn:lsn_
↪number | mark_name})
    10: stop_at_mark (Stop the recovery at a mark (STOPATMARK). {lsn:lsn_number | mark_
↪name})
    11: stop_at (Stop at (STOPAT). {datetime})
    12: restricted_user (Restrict access to the restored database)
    13: verify (Verify backup integrity)
    14: connection_string (ODBC Connection string)
    15: checksum (Restore using the WITH CHECKSUM option)
    16: driver (ODBC Driver Name)

Select parameter to modify (1-15):
```

The Plugin restore options are:

- **instance=<str>** The instance name used to connect to the MSSQL instance. This parameter is optional, and if not set, the restore will use the value set during at the backup time. By default, the instance name is

"MSSQLSERVER".

- **database=<name>** Specifies the name of the databases to restore. This parameter is optional. By default, the plugin will use the `where` option to determine the name of the new database. If both `where` and `database` are set to a valid database name, `database` will be used. A valid database name can contain the following characters: `A-Za-z0-9#_`
- **username=<str>** The username used to connect to the MSSQL instance. This parameter is optional, and if not set, the restore will use the value set during at the backup time.
- **password=<str>** The password used to connect to the MSSQL instance. This parameter is optional and if not set, the restore will use the value set during at the backup time.
- **domain=<filestr>** The domain used to connect to the MSSQL instance. This parameter is optional and if not set, the restore will use the value set during at the backup time.
- **hostname=<str>** The MSSQL server host name.
- **authtype=[windows | server]** The authentification type. Windows is default.
- **recovery=<Yes/no>** Specifies if the database will use the RECOVERY or the NORECOVERY option during the restore. By default, the restored database will be recovered.
- **stop_before_mark=<markname>** Use the WITH STOPBEFOREMARK = '<point>' clause to specify that the log record that is immediately before the mark is the recovery point. The point can be a LSN number or a mark_name.
- **stop_at_mark=<markname>** Use the WITH STOPATMARK = '<point>' clause to specify that the marked transaction is the recovery point. STOPATMARK rolls forward to the mark and includes the marked transaction in the roll forward. The point can be a LSN number or a mark_name.
- **stop_at=<datetime>** Use the WITH STOPAT = '<datetime>' clause to specify that the date time is the recovery point.
- **restricted_user=<No/yes>** Use the WITH RESTRICT_USER clause to restrict access to the restored database. The default is no.
- **connection_string=<str>** Specifies the ODBC connection string. This parameter is optional.
- **checksum=<No/yes>**

  Available since Bacula Enterprise 8.11.6

  Use the WITH CHECKSUM clause to the restored database. The default is no.

- **driver=<str>**

  Available since Bacula Enterprise 10.2.3

  The driver name used in the ODBC connection string. Default is SQL Server. If the `connection_string` option is set, it will overwrite this option.

On BWeb Management Suite, the Plugin Options are available in the restore tab.

1.15 Point In Time Restore

This topic is relevant only for SQL Server databases that use the full or bulk-logged recovery models. Under the bulk-logged recovery model, if a log backup contains bulk-logged changes, point-in-time recovery is not possible to a point within that backup. The database must be recovered to the end of the MSSQL backup.

More information can be found on https://msdn.microsoft.com/en-us/library/ms179451.aspx

It is possible to do Point In Time Restore of a MSSQL database directly from the MSSQL Plugin. It is also possible to restore files locally and do the operation from the Microsoft SQL Server Mangement Console to have more options.

LSN Information

LSNs are used internally during a RESTORE sequence to track the point in time to which data has been restored. When a backup is restored, the data is restored to the LSN corresponding to the point in time at which the backup was taken. More information can be found on https://msdn.microsoft.com/en-us/library/ms190925.aspx.

The LSN of a log record at which a given backup and restore event occurred is viewable using one or more of the following:

- Bacula Backup job output
- Log file names
- msdb.backupset table
- msdb.backupfile table

During a backup job with MSSQL Plugin, the following information about LSN numbers will be displayed in the Job output:

```
win-fd JobId 3: LSN for "db29187": First: 42000146037, Last: 44000172001
```

The **First LSN** number corresponds to the last LSN of the last transaction logs backup. It can be the very first Full backup, or the last transactional backup (Incremental). The **Last LSN** number corresponds to the last transaction recorded in the log.

With a MSSQL backup (Incremental), the file name associated with this database in the Incremental job will be:

```
/@mssql/MSSQLSERVER/db29187/log-42000162001.trn
```

The number in the name, here **42000162001** corresponds to the last LSN of the previous job (Full or Incremental).

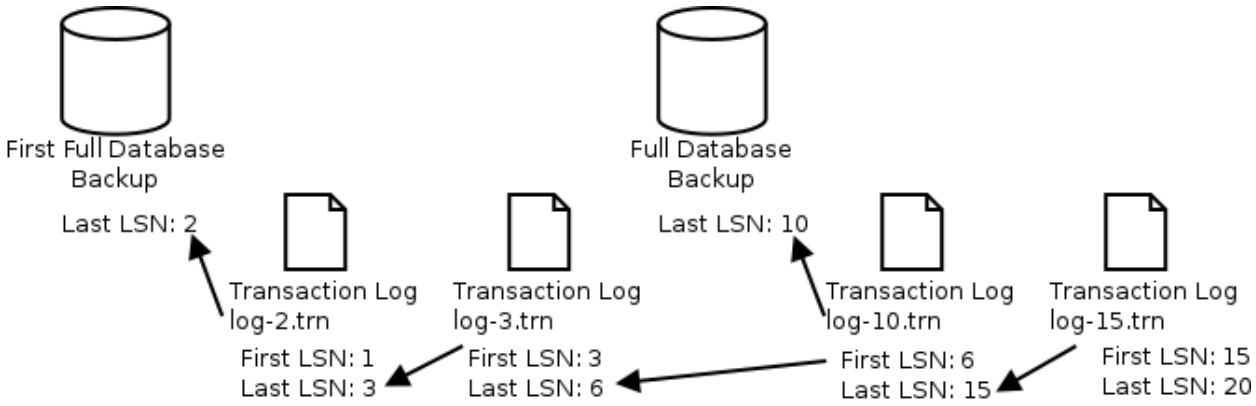

Fig. 2: First, Last and Filename LSNs

In the example shown above, if the administrator needs to restore the database at the state that corresponds to LSN 14, it can be done with the following actions:

- Use restore menu option 5
- Browse the database directory "/@mssql/db29187"
- Select last Full backup file "data.bak" (LSN: 10)
- Select incremental backup "log-10.trn"
- Specify the stop_at_mark option to "LSN:14"
- Run the restore job

or if the last full backup is not available but the previous full backup is.

- Use restore menu option 3, select the relevant jobids
- Browse the database directory "/@mssql/db29187"
- Select Full backup file "data.bak" (LSN: 2)
- Select incremental backups "log-2.trn", "log-3.trn", "log-10.trn"
- Specify the stop_at_mark option to "LSN:14"
- Run the restore job

1.16 Restore Scenarios Overview

Table 1: Restore Scenarios

Description	where	regexwhere	database	Example
Restore files **to disk**	Path			`where=c:/tmp`
Restore original database to MSSQL				`where=/`
Restore with a new name to MSSQL	Name			`where=newdb`
Restore with a new name to MSSQL			Name	`database=newdb`
Restore with a new name and file relocation to MSSQL	Path		Name	`where=c:/tmp` `database=newdb`

Restore with a new name and individual file relocation to MSSQL		Regex	Name	`regexwhere=!CLUSTER!`
				`MSSQLSERVER!`
				`database=newdb`

Restore Using Advanced Relocation

In some cases, a database can use different disks to store the data.

```
c:\data\db1.MDF
e:\logs\db1_log.LDF
```

Available since Bacula Enterprise 8.6.17

Using the Bacula `RegexWhere` function, it is possible to manipulate each file and generate new filenames. The destination directories must exist prior to the restore.

In the following example, the database will be renamed and the path will be adapted to the new server.


```
db1                 ->  db1-old                    !db1!db1-old!i
c:\data\db1.MDF     ->  f:\data\db1-old.MDF        !c:!f:!i
e:\logs\db1_log.LDF ->  g:\data\db1-old_log.MDF    !e:!g:!i
```

The resulting `RegexWhere` will be something like:

```
 *restore regexwhere="!db1!db1-old!i,!c:!f:!i,!e:!g:!i"
...
  database:            db1-old
```

Note that the **database** name must be specified in the Plugin Options menu, else, files will be stored on disk and the SQL restore commands will have to be executed manually.

To convert a path, the windows path separator must be escaped correctly in the console:

```
 *restore regexwhere="!c:\\\\data\\\\db1!c:\\data2\\db1!i"
or
 *restore regexwhere="!c:.data.db1!c:/data2/db1!i"
or
 *restore regexwhere="!c:\\\\data\\\\db1!c:/data2/db1!i"
```

Restore With Same Name

To restore a database with the same name, the `where` parameter should be empty or "/" and the `replace`= flag should be set to `always` or the original database should be dropped first.

```
* restore where=/ replace=always
...
Using Catalog "MyCatalog"
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /opt/bacula/working/127.0.0.1-dir.restore.1.bsr
Where:           /
Replace:         Always
FileSet:         Full Set
Backup Client:   win2008-fd
Restore Client:  win2008-fd
Storage:         File
When:            2015-12-14 09:53:36
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (yes/mod/no):
```


Restore Database With a New Name

To restore a database with a new name, it might be required to relocate database files on disk. It depends if the original database is still present.

If the original database is no longer available, the `where` parameter or the "Plugin Options" `database` can contain the new database name, and the plugin will automatically handle the database creation with the new name.

If the original database is still required, the `where` parameter is used to relocate files on disk, and the new database name should be be set with the "Plugin Options" menu with the `database` option. The `layout.dat` must be selected in the restore tree.

```
* restore where=c:/tmp replace=always
...
Using Catalog "MyCatalog"
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /opt/bacula/working/127.0.0.1-dir.restore.1.bsr
Where:           c:/tmp
Replace:         Always
FileSet:         Full Set
Backup Client:   win2008-fd
Restore Client:  win2008-fd
Storage:         File
When:            2015-12-14 09:53:36
Catalog:         MyCatalog
Priority:        10
Plugin Options:  *None*
OK to run? (yes/mod/no): mod                        <-----------------
Parameters to modify:
     1: Level
     2: Storage
     3: Job
     4: FileSet
     5: Restore Client
     6: When
     7: Priority
     8: Bootstrap
     9: Where
    10: File Relocation
    11: Replace
    12: JobId
    13: Plugin Options
Select parameter to modify (1-13): 13               <-----------------
Automatically selected : mssql: database=db29187
Plugin Restore Options
instance:            *None*
database:            *None*
username:            *None*
password:            *None*
domain:              *None*
recovery:            *None*              (yes)
stop_before_mark:    *None*
stop_at_mark:        *None*
```

(continues on next page)

(continued from previous page)

```
stop_at:             *None*
Use above plugin configuration? (yes/mod/no): mod  <------------------
You have the following choices:
   1: instance (Instance used to restore)
   2: database (New database name)
   3: username (Username used for restore)
   4: password (Password used for restore)
   5: domain (Domain name of user (default to local))
   6: recovery (Start Recovery)
   7: stop_before_mark (Stop the recovery before a mark (STOPBEFOREMARK).
   8: stop_at_mark (Stop the recovery at a mark (STOPATMARK).
   9: stop_at (Stop at (STOPAT). {datetime})
Select parameter to modify (1-9): 2                <------------------
Please enter a value for database: newdb           <------------------
Use above plugin configuration? (yes/mod/no): yes  <------------------

Using Catalog "MyCatalog"
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /opt/bacula/working/127.0.0.1-dir.restore.1.bsr
Where:           c:/tmp
Replace:         Always
FileSet:         Full Set
Backup Client:   win2008-fd
Restore Client:  win2008-fd
Storage:         File
When:            2015-12-14 09:53:36
Catalog:         MyCatalog
Priority:        10
Plugin Options:  User Specified
OK to run? (yes/mod/no): yes                       <-----------------
```

Restore to Local Disk

When specifying `where=c:/path/`, files will be restored to the local filesystem and the MSSQL administrator can use a TSQL or the Microsoft SQL Server Mangement Console to restore the database. SQL commands needed to restore the database are printed in the Job output as showed in the next example.

```
* restore where=c:/tmp

First you select one or more JobIds that contain files
to be restored. You will be presented several methods
of specifying the JobIds. Then you will be allowed to
select which files from those JobIds are to be restored.

To select the JobIds, you have the following choices:
     1: List last 20 Jobs run
     2: List Jobs where a given File is saved
     3: Enter list of comma separated JobIds to select
     4: Enter SQL list command
     5: Select the most recent backup for a client
```

(continues on next page)

(continued from previous page)

```
     6: Select backup for a client before a specified time
     7: Enter a list of files to restore
     8: Enter a list of files to restore before a specified time
     9: Find the JobIds of the most recent backup for a client
    10: Find the JobIds for a backup for a client before a specified time
    11: Enter a list of directories to restore for found JobIds
    12: Select full restore to a specified Job date
    13: Cancel
Select item:  (1-13): 5
Automatically selected Client: win2008-fd
+-------+-------+----------+----------+---------------------+---------------+
| jobid | level | jobfiles | jobbytes | starttime           | volumename    |
+-------+-------+----------+----------+---------------------+---------------+
|     1 | F     |        3 |   65,771 | 2015-12-14 09:52:31 | TestVolume001 |
|     2 | I     |        2 |   65,771 | 2015-12-14 09:52:42 | TestVolume001 |
|     3 | I     |        2 |   65,771 | 2015-12-14 09:52:52 | TestVolume001 |
+-------+-------+----------+----------+---------------------+---------------+
You have selected the following JobIds: 1,2,3

Building directory tree for JobId(s) 1,2,3 ...
6 files inserted into the tree.

You are now entering file selection mode where you add (mark) and
remove (unmark) files to be restored. No files are initially added, unless
you used the "all" keyword on the command line.
Enter "done" to leave this mode.

cwd is: /
$ cd @mssql
cwd is: /@mssql/
$ cd MSSQLSERVER
cwd is: /@mssql/MSSQLSERVER/
$ m db1684
6 files marked.
$ done
Bootstrap records written to /opt/bacula/working/127.0.0.1-dir.restore.1.bsr

The Job will require the following (*=>InChanger):
   Volume(s)                 Storage(s)                SD Device(s)
===========================================================================

    TestVolume001             File                      FileStorage

Volumes marked with "*" are in the Autochanger.


2 files selected to be restored.

Using Catalog "MyCatalog"
Run Restore job
JobName:         RestoreFiles
Bootstrap:       /opt/bacula/working/127.0.0.1-dir.restore.1.bsr
```

(continues on next page)

(continued from previous page)

```
Where:            /tmp
Replace:          Always
FileSet:          Full Set
Backup Client:    win2008-fd
Restore Client:   win2008-fd
Storage:          File
When:             2015-12-14 09:53:36
Catalog:          MyCatalog
Priority:         10
Plugin Options:   *None*
OK to run? (yes/mod/no): yes
Job queued. JobId=6
wait
You have messages.
* messages

$ done

17:18 dir JobId 6: Start Restore Job RestoreFiles.2015-12-14_17.18.18_14
17:18 dir JobId 6: Using Device "FileStorage" to read.
17:18 sd JobId 6: Ready to read from volume "TestVolume001" on file device "FileStorage"␣
↪(/tmp/regress/tmp).
17:18 sd JobId 6: Forward spacing Volume "TestVolume001" to file:block 0:224.
17:18 fd JobId 6: RESTORE DATABASE [db1684] FROM DISK='c:/tmp/@mssql/MSSQLSERVER/db1684/
↪data.bak'
                    WITH BLOCKSIZE=65536, BUFFERCOUNT=10, MAXTRANSFERSIZE=65536,␣
↪NORECOVERY , REPLACE
17:18 fd JobId 6: RESTORE DATABASE [db1684]
                     FROM DISK='c:/tmp/@mssql/MSSQLSERVER/db1684/log-34000000014400001.
↪bak'
                     WITH BLOCKSIZE=65536, BUFFERCOUNT=10, MAXTRANSFERSIZE=65536,␣
↪NORECOVERY
17:18 fd JobId 6: RESTORE DATABASE [db1684]
                     FROM DISK='c:/tmp/@mssql/MSSQLSERVER/db1684/log-34000000018400001.
↪bak'
                     WITH BLOCKSIZE=65536, BUFFERCOUNT=10, MAXTRANSFERSIZE=65536,␣
↪NORECOVERY
17:18 fd JobId 6: RESTORE DATABASE [db1684]
                     FROM DISK='c:/tmp/@mssql/MSSQLSERVER/db1684/log-34000000029100001.
↪bak'
                     WITH BLOCKSIZE=65536, BUFFERCOUNT=10, MAXTRANSFERSIZE=65536,␣
↪NORECOVERY
17:18 sd JobId 6: End of Volume at file 0 on device "FileStorage" (/tmp/regress/tmp),␣
↪Volume "TestVolume001"
17:18 fd JobId 6: RESTORE DATABASE [db1684]
                     FROM DISK='c:/tmp/@mssql/MSSQLSERVER/db1684/log-36000000017200001.
↪bak'
                     WITH BLOCKSIZE=65536, BUFFERCOUNT=10, MAXTRANSFERSIZE=65536,␣
↪NORECOVERY
17:18 sd JobId 6: Elapsed time=00:00:01, Transfer rate=9.372 M Bytes/second
17:18 fd JobId 6: RESTORE DATABASE [db1684]
17:18 dir JobId 6: Bacula dir 8.4.8 (22Feb16):
```

(continues on next page)

(continued from previous page)

```
  Build OS:               x86_64-unknown-linux-gnu archlinux
  JobId:                  6
  Job:                    RestoreFiles.2015-12-11_17.18.18_14
  Restore Client:         win2008-fd
  Start time:             14-Dec-2015 17:18:20
  End time:               14-Dec-2015 17:18:22
  Files Expected:         6
  Files Restored:         6
  Bytes Restored:         9,371,785
  Rate:                   4685.9 KB/s
  FD Errors:              0
  FD termination status:  OK
  SD termination status:  OK
  Termination:            Restore OK
```

1.17 Restore the "master" Database

Instructions on how to restore the "master" database are detailed in this article: https://technet.microsoft.com/en-us/library/aa213839%28v=sql.80%29.aspx

1.18 Database in *restoring* State

At the end of a restore, if the plugin option `recovery` was set to `no`, the restored database will be in the "restoring" state. To end the restore process, the recovery process must be run. It can be done with the following SQL command:

```
RESTORE [yourdatabase] WITH RECOVERY;
```

1.19 Always On Availability Groups

Availability groups can be created in MSSQL to provide high data availability over WSFC cluster nodes, as detailed in this article: https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/overview-of-always-on-availability-groups-sql-server?view=sql-server-2017

It can be interesting depending on the system scale to perform backups on secondary replicas to free up time and resource costs from the primary replica. The Bacula MSSQL Plugin is compatible with this type of architecture, with some restrictions.

1.20 Backup

When avalability group(s) are defined, automated backup preference should also be specified to indicate the type of backup strategy (prefer secondary, primary only, etc.). The MSSQL plugin enforces this preference, only allowing backup of the prefered replica.

Note that depending on replica, not all backup type are allowed. Secondary will support copy-only full database backups and log backups while differential is not supported. This needs to be taken in account when creating Bacula's backup Jobs. Refer to this article for more details: https://docs.microsoft.com/en-us/sql/database-engine/availability-groups/windows/active-secondaries-backup-on-secondary-replicas-always-on-availability-groups?view=sql-server-2017

1.21 Restore

Database restores can not be performed when they are part of one or many availability groups. This is detected by the plugin and the database restore will be skipped when this is the case. To workaround this, you need to remove the database from the availability group(s), then restore the database and eventually put it back in its original availability group(s). More details can be found here: https://social.technet.microsoft.com/wiki/contents/articles/28148.restoring-a-backup-database-to-an-availability-group-sql-server.aspx

1.22 Limitations

The current version of the plugin has the following limitations:

- The plugin can have only one Plugin command line in the FileSet. To backup multiple databases in the same Job, it is possible to use the **include** parameter multiple times on the plugin command string.
- To restore a database with a new name and a new location, the `layout.dat` file must be selected in the restore process. The relocation function will not work correctly if data files were added after the last Full backup job.
- It is not possible to restore multiple databases with a new name in a single restore job. The restore should be performed in different restore jobs.
- Database snapshots are automatically excluded from the backup.
- The `restart` command has limitations with plugins, as it initiates the Job from scratch rather than continuing it. Bacula determines whether a Job is restarted or continued, but using the `restart` command will result in a new Job.

These limitations will be addressed in a future version.

2 MSSQL VSS plugin

Attention: The MSSQL VSS plugin is deprecated

Support for the MSSQL VSS plugin will stop in September 2022. All instances of Microsoft SQL Server should use the *MSSQL VDI plugin* for future backups.

Please do not run any differential backup with the MSSQL VSS plugin.

In order to access the MSSQL VDI plugin, please contact our Support Team. The mssqlvss-to-mssqlvdi will guide you through the migration process.

2.1 Overview

This white paper presents how to use the Microsoft Windows VSS plugin's SQl Server support with **Bacula Enterprise** version 6.0 or newer. These solutions are not applicable to prior versions. This document is intended to be used by **Bacula Enterprise** administrators.

2.2 Bacula Windows VSS Plugin

Bacula Systems provides a single plugin for Bacula Enterprise named `vss-fd.dll` that permits you to backup a number of different components on Windows machines. One of those components is Microsoft SQL Server (MSSQL), which is the subject of this white paper.

Backing up and restoring MSSQL databases is supported with Full and Differential level backups. It is not possible to do Incremental backups because Microsoft does not support that backup level for the SQL Server product.

To activate the MSSQL component you have to put the following into the Include section of the File Set which will be used to back up the SQL Server data:

```
Plugin = "vss:/@MSSQL/"
```

This will back up all SQL server data except for those databases owned by Sharepoint, if that VSS plugin component is also specified.

The plugin directive must be specified exactly as shown above. A Job may have one or more of the vss plugin components specified.

You must ensure that the `vss-fd.dll` plugin is in the plugins directory on the FD doing the backup (done by default with the installer), and that the `Plugin Directory` directive line is present and enabled in the FD's configuration file `bacula-fd.conf`. An example configuration file is shown in figure *File Daemon Configuration Excerpt with "Plugin Directory" Line*. The status output of a client with the VSS plugin available is shown in figure *Status of a File Daemon with VSS Plugin Available*.

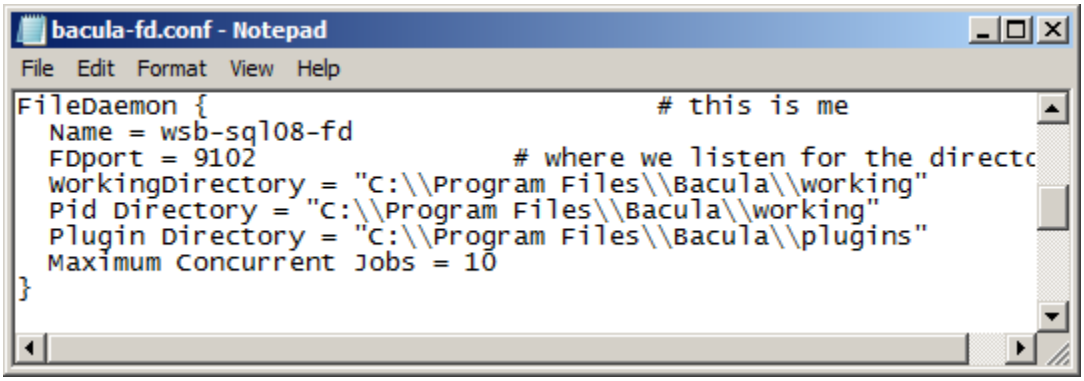

Fig. 3: File Daemon Configuration Excerpt with "Plugin Directory" Line


```
*status client=wsb-sql08-fd
Connecting to Client wsb-sql08-fd at wsb-sql08:9102

wsb-sql08-fd Version: 6.0.1 (02 Apr 2012)  VSS Linux Cross-compile Win64
Daemon started 20-Apr-12 13:14. Jobs: run=15 running=0.
Microsoft Windows Server 2008 R2 Standard Edition Service Pack 1 (build ↙
→7601), 64-bit
 Heap: heap=0 smbytes=1,061,455 ...
 Sizes: boffset_t=8 size_t=8 debug=0 ...
 Plugin: alldrives-fd.dll delta-fd.dll vss-fd.dll
```

Fig. 4: Status of a File Daemon with VSS Plugin Available

BMR and VSS

The VSS plugin will not work correctly during a Bare Metal Recovery because Microsoft does not include the VSS infrastructure in WinPE, the environment used during Bare Metal Recovery. As a consequence, any backup you do with the VSS plugin cannot be used for a Bare Metal Recovery. To have a good backup for BMR purposes, you must run the Bacula FD **without** using the `Plugin =` directive in your FileSet (i.e. the plugin must not be used).

2.3 Backup

If everything is set up correctly as above then the backup will include the SQL server data. The SQL server data files backed up will appear in a bconsole or bat restore like:

```
/@MSSQL/
...
etc
```

Only a backup of the complete SQL server data is supported, i. e. all database server instances, all databases and all tables are being backed up. It is not currently possible to back up only parts of the SQL server data like certain server instances, databases, or tables.

Both Full and Differential backups are supported. Microsoft does not support Incremental backups for MSSQL. If you run one, you will get errors.

Following the creation of a new database, you should run a Full backup of the SQL server data. A new database will not be backed up until a Full backup has been done, and Differential backups will fail on this specific new database.

As Windows does not update the time and date of modification of all SQL server files, you **must** use the **Accurate** Job option if you want correct Differential backups. If you do not use this option, the plugin will print a warning message, and restores will probably fail.

A complete example of the Job setup for MS SQL Server data looks as shown in figure *MS SQL Server Backup Job*. As for all VSS-enabled components, it is the administrator's responsibility to make sure that the required VSS snapshots are created by explicitly mentioning at least one file or directory for each drive where data that is handled by the plugin is stored. In the example, we use the file `c:/backmeup` to ensure this.

Note the inclusion of `c:/backmeup`, which is required to ensure that **Bacula** creates the required VSS snapshot of the drive the backed up data is stored on. If SQL Server data is also stored on other drives, you need to create similar `File =`-lines for these drives, too[1].

[1] Starting with version 12.5, specifying the volumes is not mandatory anymore


```
File Set {
  Name = MSSQL
  Include {
    Options {
      Signature = SHA1
    }
    File = C:/backmeup
    Plugin = "vss:/@MSSQL/"
  }
}

Job {
  Name = MSSQL08
  Accurate = Yes
  File Set = MSSQL
  Client = wsb-sql08-fd
  Job Defs = DefaultJob
  Level = Differential
}
```

Fig. 5: MS SQL Server Backup Job

```
File Set {
  Name = MSSQL-TestDB
  Include {
    Options {
      Signature = SHA1
    }
    File = C:/backmeup
    # backup only TestDB on the server
    Plugin = "vss:/@MSSQL/ cinclude=*/TestDB cexclude=*"
  }
}
```

In this example, only the database TestDB will be included in the backup. Use of multiple `cinclude` parameters is possible in the Plugin command line.

2.4 Restore

To restore you can use all regular ways to start a restore. However, you must make sure that, if restoring differential data, the previous full backup is also restored. This happens automatically if you start the restore, in `bconsole`, using the restore options 5 or 12. In the file tree generated, you should either mark complete database instances or databases. Note that it is important **not** to include master databases in a restore as those have to be handled specially. We show an example of that below.

Additionally, a database must be restored to the MSSQL server from which is has been backed up; restoring to another machine is not possible.[2]

The **master** database cannot be restored while the MSSQL server is running. Thus you must take care not to select the master database for a regular restore. If you do, the restore will fail. Accordingly, we provide instructions how to restore the **master** database of a MSSQL server instance, see chapter *Restoring master Databases*.

[2] The MS SQL plugin is not intended to be used to migrate data. To do that, you should use specialized tools. In the simplest case, a complete SQl data dump may be created, moved to the new machine, and fed to the database locally.

All other databases can be restored while the system is running. The Windows VSS writer will automatically unmount each database before it is restored, then apply all the outstanding log files if any exist, and finally remount the database.

It is important to understand that, using relocation of the restored files, data can be restored to a different and even a new database. This may be useful, but it may also be unintended. We recommend to make sure that any relocation is either turned off, or the effects are well understood. See figure *Relocated Database* for an example of how things may look when relocating to a default location.

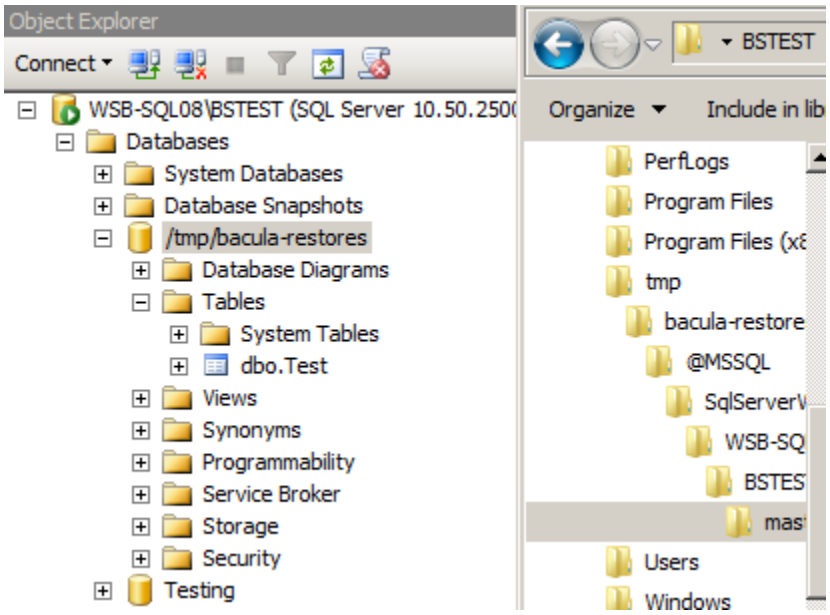

Fig. 6: Relocated Database

This result would probably not be what you expect – the result is a new database with a name that most likely does not match any reasonable naming scheme.

If you restore SQL Server data, it is important to understand that, during the restore process, **all** existing database log files in the database directory will be applied. For this reason, it can be important to remove old log files. This can only be done when the database is offline, so in this situation, the restore sequence may be like this:

- Take database offline. This can be done through the `Microsoft SQL Server Management Studio` by right-clicking on the database in the Explorer pane, clicking "Tasks", then "Take Offline". This can be seen in figure *Off-Line a Database in SQL Server Management Studio*.
- Delete, move or rename the database log files. An example is shown in figure *Renaming a Log File in Explorer*.
- Run the actual restore.
- Bring the database online again, similar to the procedure in step 1.

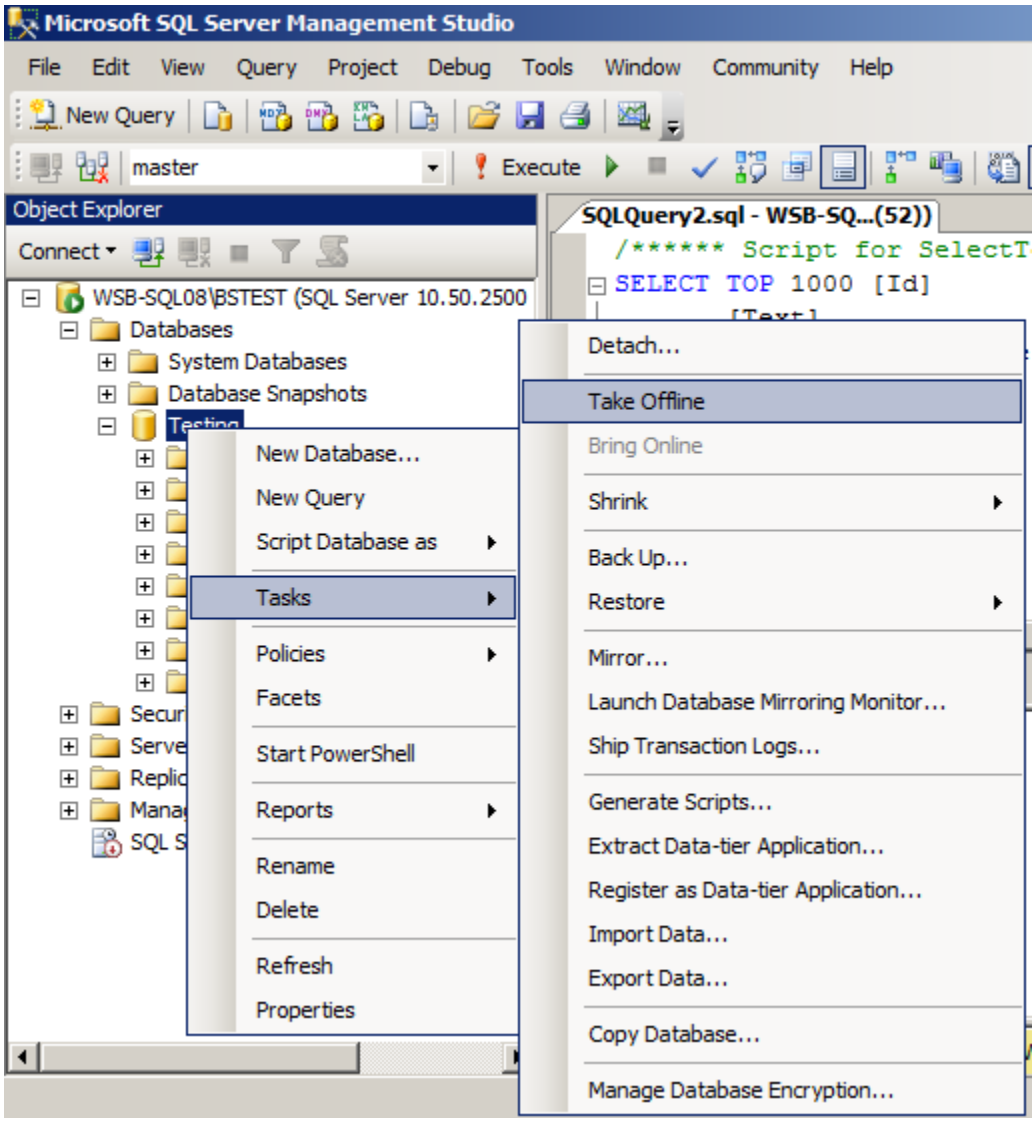

Fig. 7: Off-Line a Database in SQL Server Management Studio

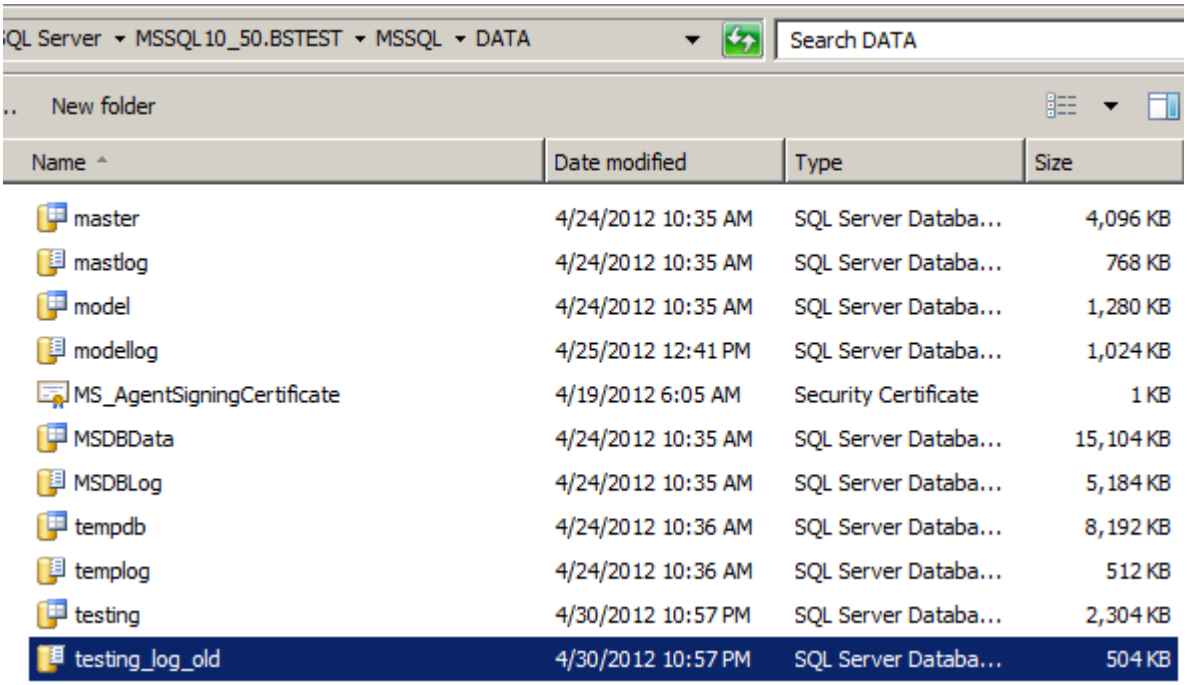

Fig. 8: Renaming a Log File in Explorer

Example

We assume that a correct backup of MSSQL data exists and you start the restore with option 5 in `bconsole`'s **restore** command, mark the complete tree of data backed up by the MSSQL component of the VSS plugin, then finally do `lsmark @MSSQL` to show all the files selected to be restored. Then the output you see should be similar to that presented in figure *Output for lsmark Command with SQL Server 2005 Data Marked*.

Note, the MSSQL files are generally under @MSSQL/MSDEWriter as in figure *Output for lsmark Command with SQL Server 2005 Data Marked* if you are running MSSQL 2005 on Windows Server 2003. If you are running MSSQL 2008 on Windows Server 2008, they will be under @MSSQL/SqlServerWriter.

Following that part of the data tree is the name of the MSSQL server host, in this case RUFUS-WIN2003, possibly the database server instance, and then the various databases.

In figure *Output for lsmark Command with SQL Server 2005 Data Marked* you can see that the name of the database master immediately follows RUFUS-WIN2003. The other databases, model, and msdb are at the same indentation level as master. So, for the restore to work, in the above example, you will need to unmark all the items that are associated with database master and below. If you have only selected database msdb for restoration, you would have output that looks like the one shown in figure *Marked Files to Restore Excluding master Database*.


```
$ mark @M*
14 files marked.
$ lsmark
*@MSSQL/
  *MSDEWriter/
    *RUFUS-WIN2003/
      *master/
        *:component_info_4f771ba1
        *c:/
          *program files/
            *microsoft sql server/
              *mssql.1/
                *mssql/
                  *data/
                    *master.mdf
                    *mastlog.ldf
      *model/
        *:component_info_4f771ba1
        *c:/
          *program files/
            *microsoft sql server/
              *mssql.1/
                *mssql/
                  *data/
                    *model.mdf
                    *modellog.ldf
      *msdb/
        *:component_info_4f771ba1
        *c:/
          *program files/
            *microsoft sql server/
              *mssql.1/
                *mssql/
                  *data/
                    *msdbdata.mdf
                    *msdblog.ldf
```

Fig. 9: Output for lsmark Command with SQL Server 2005 Data Marked

```
$ lsmark
*@MSSQL/
  *MSDEWriter/
    *RUFUS-WIN2003/
      *msdb/
        *:component_info_4f771ba1
        *c:/
          *program files/
            *microsoft sql server/
              *mssql.1/
                *mssql/
                  *data/
                    *msdbdata.mdf
                    *msdblog.ldf
```

Fig. 10: Marked Files to Restore Excluding master Database

Restoring master Databases

To restore the SQL Server **master** database, the only reliable way is to turn off plugins, restore the database files to some location, and copy the restored files to their original location.

To turn off the VSS plugin, use `notepad` to edit the **file deamon** (fd)'s configuration file, `bacula-fd.conf`, put a hash sign "#" in front of the `plugin directory` line, and re-start the **file deamon** (fd). A typical command sequence (in a command window with elevated privileges!) would look something like this:

```
net stop bacula-fd
notepad "C:\Program Files\Bacula\bacula-fd.conf"
net start bacula-fd
```

```
$ ls
master.mdf
mastlog.ldf
$ pwd
cwd is: /@MSSQL/SqlServerWriter/WSB-SQL08/BSTEST/master/c:/program files/microsoft
 sql server/mssql10_50.bstest/mssql/data/
$ mark *
2 files marked.
```

This example shows SQL Server 2010 paths.

When restoring, you should navigate directly to the directory containing the data files, which will pe represented in the virtual directory tree similarly to what we show in figure **fig:masterrestore**. In there, mark the data files (there should be two: `master.mdf` and `mastlog.ldf`), and continue the restore process. Using file relocation features to put the files into a new location is strongly recommended.

After the restore of the data files, you have to shut down the SQL server instance, move the restored data files to their correct location, and start the server instance again.

Afterwards, make sure to turn on plugins for the again.

Restoring the **master** database while MSSQL server is running is not possible.

2.5 Plugin Notes

Windows VSS Plugin Items to Note

- One file from each drive needed by the plugins must be explicitly listed in File Set used. This is to ensure that the main **Bacula** code does a snapshot of all the required drives. At a later time, we will find a way to accomplish this automatically.
- When doing a backup that is to be used for Bare Metal Recovery, do **not** use the VSS plugin. The reason is that during a Bare Metal Recovery, VSS is not available nor are the writers from the various components that are needed to do the restore. You might do a full backup to be used with a Bare Metal Recovery once a month or once a week, and all other days, do a backup using the VSS plugin, but under a different Job name. Then to restore your system, use the last Full non-VSS backup during the bare metal restoration of your system, and after rebooting do a restore with the VSS plugin to get everything fully up to date.

General Plugin Items to Note

- The 'estimate' command does not handle plugins. When estimating a job that uses plugins, an error message regarding the plugin will be displayed. However, backup jobs will use the plugin.
- The File Set Include Option `CheckFileChanges = Yes` does not work with plugin-generated data. Thus, you must not use that Option in the Include section of the FileSet where you specify using the MSSQL plugin.
- The `restart` command has limitations with plugins, as it initiates the Job from scratch rather than continuing it. Bacula determines whether a Job is restarted or continued, but using the `restart` command will result in a new Job.

2.6 Problem Resolution

Most problems that can happen are reported in the job report **Bacula** sends. In the following table, we have collected information about common problems and the suggested resolution.

Job Report Message	Cause	Resolution
Unable to do a Differential backup of MSSQL master. Database excluded.		The master table is only backed up at full level. This is desing limitation of MS SQL Server.[3]
Warning: VSS Writer "SqlServerWriter" has invalid sttate. ERR=The writer vetoed the shadow copy creation creation process during the backup preparation state.	Various	Check for other messages in the Job Report
VSS Writer (PrepareForBackup): "SqlServerWriter", State: 0x7 (VSS_WS_FAILED_AT_PREPARE_BACKUP)		
Warning: VSS Writer "SqlServerWriter" has invalid sttate. ERR=The writer vetoed the shadow copy creation creation process during the backup preparation state.	Various	Check for other messages in the Job Report
SQL writer error: Backup type 2 not supported.	Incremental	Do not run incremental use Full and Differential levels only
Error: Unstable writer state=13: Restore skipped for file: /@MSSQL/SqlServerWriter/… /… /master/:component_info_… Writer="SqlServerWriter" … Sqllib error: OLEDB Error encountered calling ICommandText::Execute. hr = 0x80040e14.		
SQLSTATE: 42000, Native Error: 3013 Error state: 1, Severity: 16 … Error message: RESTORE master WITH SNAPSHOT is not supported. To restore master from a snapshot backup, stop the service and copy the data and log file.		The master table can not be restored with SQL Server running normally. Follow instructions given above
Error: Unstable writer state=0: Restore skipped for file: /@MSSQL/SqlServerWriter/… /… /master/:component_info_… Writer="SqlServerWriter" Sqllib error: OLEDB Error encountered calling IDBInitialize::Initialize. hr = 0x80004005 SQLSTATE: 42000, Native Error: 18461 Error state: 1, Severity: 14 Source: Microsoft SQL Server Native Client 10.0 Error message: Login failed for user 'NT AUTHORITYSYSTEM'. Reason: Server is in single user mode. Only one administrator can connect at this time.		
DBPROP_INIT_DATASOURCE: WSB-SQL08BSTEST DBPROP_INIT_CATALOG: master DBPROP_AUTH_INTEGRATED: SSPI		Follow instructions above to restore master table

[3] An explanation can be found at a Microsoft Web site

