VEEAM
IT JUST WORKS!™

SQL Server Backup and Restore in a Veeam environment

Tibor Karaszi
Consultant, SQL Server expert

AVAILABILITY™
for the Modern Data Center

The purpose of this paper is to describe SQL Server backup in general and also the options you have for backing up your Microsoft SQL Server databases in conjunction with Veeam®. The paper is both targeted to the SQL Server DBA, as well as the backup operator who may have more experience with Veeam and less with SQL Server. This is not intended to be a reference paper covering all the options in the graphical user interfaces (GUIs) or the SQL commands. For that, please refer separately to the Veeam and SQL Server documentation that accompanies each product.

When you use Veeam with SQL Server, you have two options regarding SQL Server backup:

The first is to let SQL Server produce its backups, typically to files, as if you were not using Veeam. Then, allow Veeam to pick up the backup files with the snapshot of the virtual machine. We call this the DBA-centric way of thinking. You need space where you initially store the backup files – often on that same machine. I typically keep three days back in time locally - if I am not using differential daily backups (more information on differential daily backups will follow). You will also need space for the backup files on your backup server – where Veeam stores the snapshot of the virtual machines.

The second option, what we call the backup operator centric way of thinking, is to not perform backups in SQL Server. Veeam performs backups by producing a snapshot of the machine, usually once a day. This includes your SQL Server databases – at that point in time. As you will see, Veeam can also complement this with SQL Server transaction log backups, based on this snapshot. This makes for a very storage-effective solution – you do not store the database backups separately, instead they are a part of the snapshot.

Regardless of the method you choose to follow, there is some important groundwork that must be discussed before covering backup specifics.

In an attempt to keep this discussion at a suitable technical level, I have made simplifications at various places throughout the document. Therefore, if your experience varies slightly, please keep in mind that a number of generalizations have been made to describe the process succinctly and produce a document that is helpful to the largest potential audience of readers.

Transaction logging and management of the transaction log

SQL Server supports transaction. Every modification is logged in the transaction log before the modification is performed on the actual data page. The transaction log lives in the ldf file(s) of the database. Please reference the first paper in this series for a more detailed discussion about storage architecture and transaction logging.

Ultimately, it is the DBA's responsibility to make sure the transaction log doesn't fill up the disk, as log records are generated for our modifications.

Virtual Log Files (VLFs)

The transaction log file (or files) is internally divided into Virtual Log Files(VLFs). This is performed automatically by SQL Server, and a DBA typically does not have to be aware of VLFs.

There are some disadvantages of having "too many" VLFs, such as the case when the ldf file has grown frequently. Things such as startup and restore of the database can be slower with many VLFs. Search the Internet for terms such as "VLF" and "shrink" and you will find details on how to determine if you have many VLFs and how to properly manage them.

So, think of the ldf file internally as a series of VLFs. A VLF can be in use or it can be free for SQL Server to use (slightly simplified, but enough for our purposes). Also, imagine SQL Server having a series of log records with a head and a tail. When the head reaches the end of the current VLF, SQL Server has to find a VLF that it can use. If all VLFs in the ldf file are in use, then the ldf file has to grow – or if it cannot grow, then the modification will return an error message and fail.

What you need to do is make VLFs reusable. We sometimes refer to this as "truncate the log," or as I prefer to say "empty the log." However, technically, we make SQL Server mark as many VLFs as possible as OK to use – as free, reusable, or "OK to overwrite".

The recovery model setting

A database option called the *recovery model*, is all about management of the transaction log. The available modes are full, simple and bulk logged. Most installations and databases are either in simple or full recovery model. The default value – what you get when you create a database – is inherited from the model database, and by default is in full recovery.

Simple recovery

This recovery model is designed to be used when you do not perform backup of the transaction log of the database. In simple recovery, it is not your responsibility to "empty the log" (or "truncate the log"), as SQL Server will do that for you. However, you can still end up with large ldf files due to long-running transactions and problems with the log reader when using transactional replication. Since SQL Server will truncate the log for you, you cannot perform backup of the transaction log – the BACKUP LOG command will return an error if you try.

The log is typically truncated when a checkpoint occurs (reference the first paper), which is done automatically now and then. You can even stress this using the CHECKPOINT command.

Full recovery

In full recovery, it is your responsibility to truncate the log. This happens when you perform backup of the transaction log, i.e., the BACKUP LOG command will truncate the log after producing the backup. It is worth mentioning that other backup types (full, differential, snapshot, etc.) do not empty the log – only log backup will do this. If you are in full recovery and do not take a log backup, then the log file will continue to grow until it reaches maximum size or the disk is full. There is a setting in Veeam that will make Veeam empty the transaction log after producing its snapshot backups, and essentially manage the transaction log for you, even if you are in full recovery. I will discuss this in more detail later. However, it is important to note that you will not want to use this setting in Veeam if you produce your own log backups (outside of Veeam).

Bulk logged recovery

Bulk logged recovery is not commonly used, but for the right situation it can be valuable. In order to explain this properly, we need to first explain minimally logged operations.

There are some operations that can be logged in a minimal fashion to the transaction log. One such operation is mass-*loading data* into a table, such as importing them from a file. This is usually referred to as bulk loading data. Imagine you need to import one million rows of data from a file into a table. If fully logged, this operation will log at least one million log records – or two million, or three million etc., as each index is also maintained and reflected in the transaction log. In full recovery model, all operations are fully logged, as there are no minimally logged operations. However, in simple or bulk logged recovery, these operations do not log actual modifications of your data but only the fact that it allocates storage (basically "now this extent is used by this table," and so on).

In bulk logged recovery, these operations can be performed as minimally logged operations and you can also produce a log backup after those operations. Such a log backup will not only include log records from the ldf file, but also the data (extents) modified by the minimally logged operations. However, you can only produce such a log backup if the data files are available (having the data files available is not a requirement for a "normal" log backup). Also, you cannot restore this type of log backup to any point in time using the STOPAT option for the RESTORE LOG command.

The other operations that can be minimally logged, beside bulk loading of data, are SELECT INTO and create, rebuild, and drop of indexes.

In the end, deciding which recovery model to use isn't particularly difficult, if we leave bulk logged aside. If you are to produce log backups, then use full. If not, then use simple. Or, if you only want Veeam snapshot backups, you can let Veeam truncate the log for you, if the database happens to be in full recovery after the snapshot has been produced.

SQL Server backup types

Whether you choose to produce your own SQL Server backups or use Veeam's ability to back up your SQL Server, it is important to better understand the types of backups in SQL Server. Sure, you can always point'n'click in the Veeam GUI and let it produce your backups – but then you wouldn't be reading this paper in the first place! You want to better understand the technology, so you can use it correctly and handle unexpected situations. It is important you understand the various backup types since this will allow you to make an informed decision about how to perform your backups and which types of backups you want to use.

Full backup

A full backup includes everything in the database. SQL Server will copy all of the data in the database's data files (all extents) to the backup destination, which is typically a file. Changes that are made to the data while the backup is running are reflected in the transaction log, and when all data (all extents) have been copied, SQL Server will then also copy the log records that were produced while the backup was running. When you restore from a full backup, SQL Server will copy all pages from the backup file into the data file(s), and all log records from the backup files into the ldf file(s). And finally perform the same type of recovery as when you start SQL Server (see the first paper in this series). For example, you start a full backup at 02:00, and the backup finishes at 02:45. When you restore from that backup, the database will look like it did at 02:45 – not 02:00.

A full backup is performed using the BACKUP DATABASE command.

Differential backup

A Differential backup is very much like a full backup, except that SQL Server will only backup the extents that have been modified since the last full backup. It also uses the log records produced while copying the extents, the exact same way as for a full backup. For example, let's say you have a full backup F1 and then differential backups D1, D2 and D3. When you restore, you would restore F1 and D3, assuming you want to restore to the most recent time as possible (a full backup and then the last differential backup since). Note that a differential backup is based on the most recent full backup. Say you have F1, D1, D2, D3, F2, D4, D5 and D6. If you want to restore D6, you would restore F2 and then D6. You cannot base D6 on the F1 backup.

The BACKUP DATABASE is also used for differential backups, adding the option DIFFERENTIAL to the WITH clause.

Differential backups can be a huge space saver considering how much backup data is produced in the end. Here is an example from one of our customers. The figures used in the example have been slightly rounded. Initially, we did daily full backups. One such backup produced 100 GB of (compressed) backup data for the SQL Servers. This was stored on backup servers for four weeks, equaling 2.8 TB. We changed it to weekly full backup and daily differential backups. About 1 GB of data was modified each day, therefore, we produced 121 GB per week (100 + 1 + 2 + 3 + 4 + 5 + 6), meaning 484 GB for four weeks. So, the amount of SQL Server backup data we produced and stored on the backup servers decreased from 2.8 TB to 0.48 TB.

We had to adjust based upon the amount of time we stored the backup files on local machines. Three days could mean that we cannot perform a restore from what exists only on that machine, which is something I always recommend if you let SQL Server produce backup files. Imagine that we only have differential backup files on the machine. So, we changed it from three days to 13 days and in the end, the amount of data stored in the local backup files reduced some but not significantly.

Transaction log backup

Transaction log backup is defined as backing up the changes made since the last transaction log backup. This option is similar to *incremental backup*. Technically, SQL Server reads the log records in the ldf file and copies them to the backup file. Log backups have several advantages. First, you can produce a log backup even if the database files are damaged or even lost (using the NO_TRUNCATE option for the BACKUP LOG command). In many cases this means you can achieve zero data loss in the event of an accident. Another advantage is the possibility to perform log backups very frequently, perhaps every hour, every 10 minutes, or 5 minutes.

The command to produce a log backup is BACKUP LOG.

It is important to note that when you restore log backups, you need to restore them in sequence and cannot skip a log backup.

The restore sequence for SQL Server is pretty straight forward:

1. Restore from a full backup
2. If you have differential backups, restore from the most recent differential backup produced after that full backup.
3. If you have log backups, restore all subsequent log backups with an option to stop at a certain point in time when you restore the last log backup.

Snapshot backup

Snapshot backups are completely different. From a high abstraction viewpoint, your backup software tells SQL Server to stop using I/O for a certain time period, and while SQL Server isn't performing I/O, the backup software can produce a snapshot copy of the data in the database files. SQL Server is informed that this snapshot is being produced using the *SQL Server VSS Writer* service in the operating system. In other words, SQL Server does not produce any backup data, it is just halting modifications activity (not doing any I/O) while the snapshot is being performed (while being "frozen"). You can see that snapshots are produced by looking in the SQL Server errorlog file, where you will see messages such as "Freezing I/O for database …", for each database; and later "Resuming I/O for database …".

An interesting and important fact is that SQL Server will consider such a snapshot a full backup – even though SQL Server did not produce any backup data itself. This is important from several viewpoints, as we will explain later. Another important fact is that this is a fully supported backup type. There is nothing strange about snapshot backups assuming they are produced the right way (utilizing the SQL Server VSS Writer service).

More details about how snapshot backups work in SQL Server can be found in the following article: https://technet.microsoft.com/en-us/library/cc966520.aspx .

The COPY_ONLY option

Sometimes you produce a backup to simply get a copy of a database to restore on a test-server, for instance, especially if you want to avoid impacting the chain of your scheduled backups. For these purposes, we have an option to the backup command named COPY_ONLY. This is relevant for two backup types:

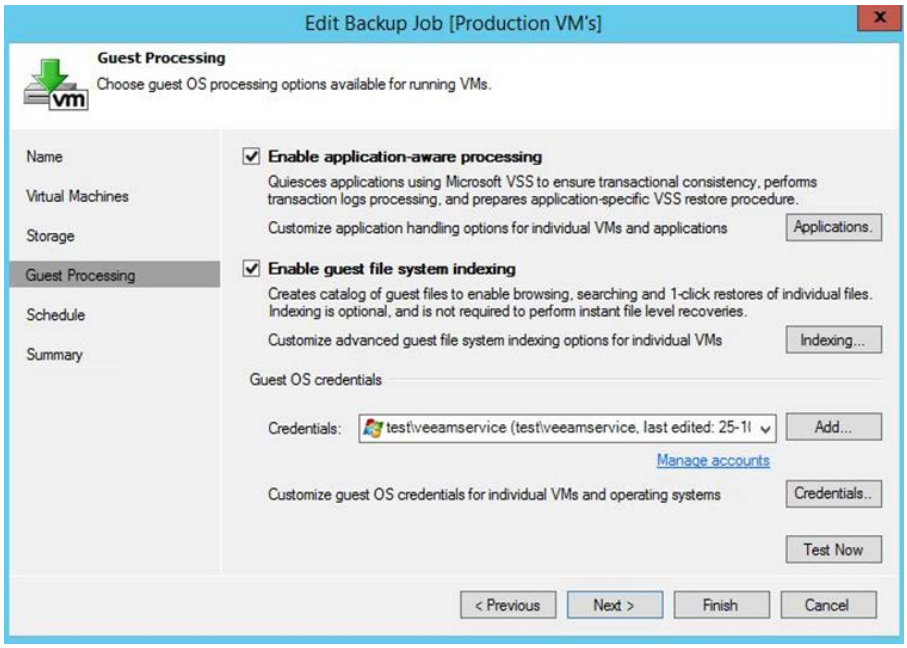
1. COPY_ONLY used with full backups. This means that this full backup will not impact your differential backups. For example, you have scheduled weekly full backups (Sunday, for instance), and daily differential backups (all days except Sunday). Now, if you perform a full backup just for the purpose of getting a copy of your database, say on Tuesday afternoon, then the differential backups for the rest of the week will be based on this Tuesday "out-of-bands" backup you performed. Imagine if the administrator who performed this full Tuesday backup after restore deleted that backup file. The following differential backups for that week will be based on the Tuesday full backup – but this no longer exists. This is a disaster! So what we do is specify the COPY_ONLY option for this Tuesday "out-of-band" backup and this way it will not impact the following differential backups.
2. COPY_ONLY used with transaction log backups. This is a far less common situation. When specifying COPY_ONLY when performing a log backup, then that log backup will not impact the subsequent log backups. Basically it will not truncate the log.

More advanced backup options

There are other backup options which we will not explain in this document – being a document about Veeam and SQL Server backups. These other options are well described in the SQL Server documentation. They include backup at the file or filegroup level.

Scheduling SQL Server to perform its own backups

Scheduling SQL Server to perform its own backups is probably what most experienced SQL Server DBAs will initially be most comfortable with. Let me first say that there are several advantages to using Veeam to back up your SQL Server, so I suggest you also read the following section before deciding what strategy to choose. Having said that, if you want to produce your own backups to files, then there are some things you must consider when using Veeam. Basically, you want to avoid Veeam interfering with your SQL Server backups.

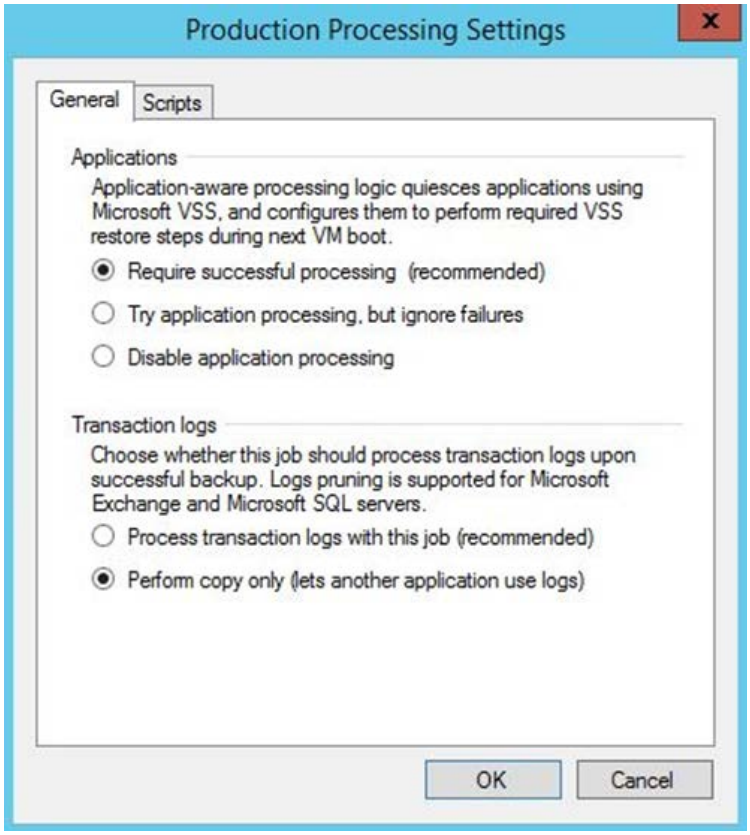


In the Veeam backup job, we strongly recommend you specify "**Enable application-aware processing**." This will make Veeam do the backup using the SQL Server VSS Writer service. This means that the machine snapshot will be a valid backup of also the SQL Server databases. So even if you are also producing your own SQL Server full backups, you have a **second level of safety**, using the Veeam machine snapshot backup for your databases. This also means that the Veeam snapshot backup is seen by SQL Server as a full database backup.

In order to play nice with your own SQL Server backups, you want to select "Applications" and make sure your Veeam backup is configured in a suitable manner.

The "Processing Setting" configuration dialog, the "General" tab

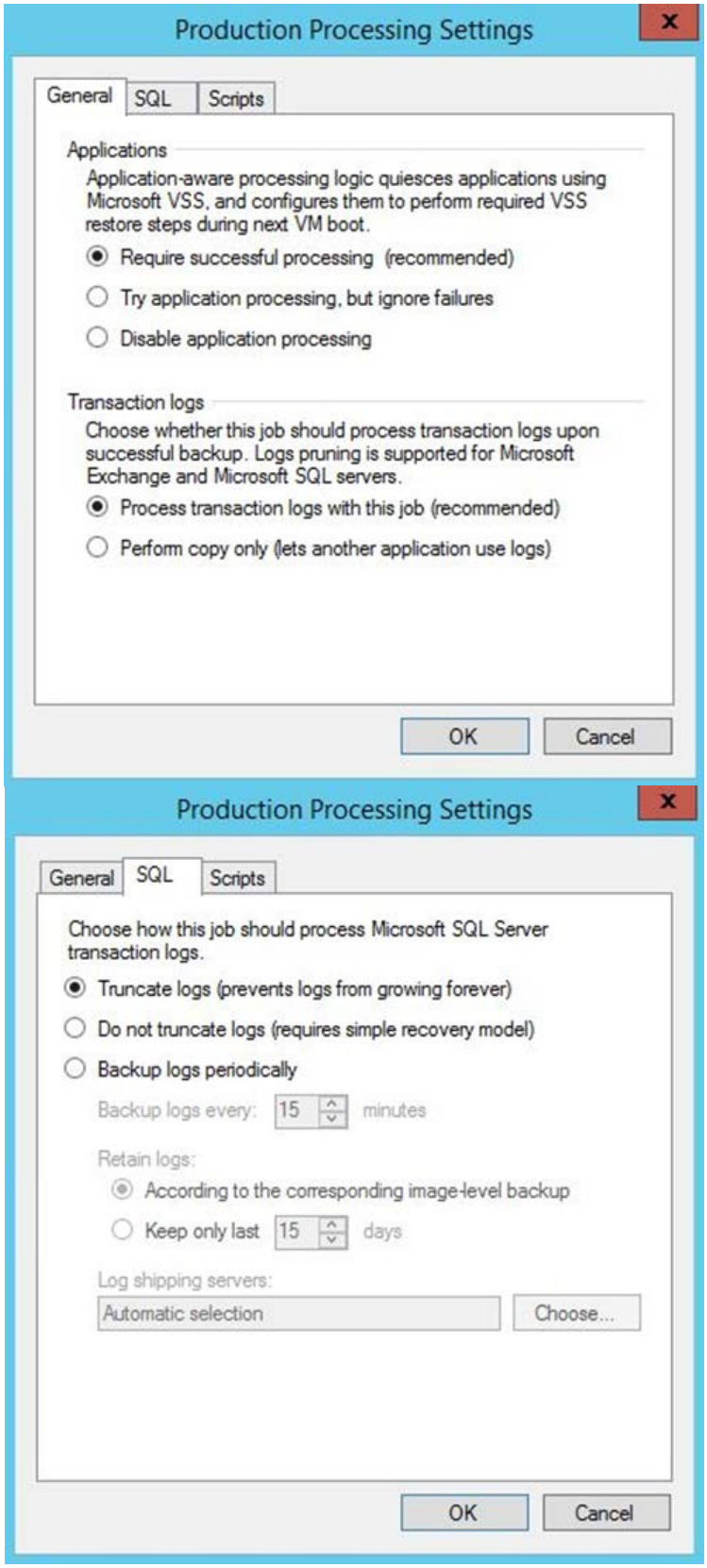
Regarding the "Application" setting, which isn't specific to only SQL Server, try to imagine the SQL Server VSS Writer service isn't available, for any reason. I strongly recommend you use the topmost option – to fail the backup in these situations. This way the backup operator can be alerted of the error and manage the situation.



As for the "Transaction logs" option, you want to select "Perform copy only" if you perform your own SQL Server backups. This way the Veeam snapshot backup will be seen by SQL Server as a COPY_ONLY backup and will not interfere with any differential backups that you produce in SQL Server. Even if you don't produce differential backups today, you still want to select this since you or someone within your organization might want to start using differential SQL Server backups in the future.

The "SQL tab"

If you select "Process transaction logs with this job" and do not specify "Perform copy only," then it is very important you select the "SQL" tab, which is not available if you select the copy only option..



If you select "Truncate logs" inside the SQL tab then Veeam will perform a log backup to the file name "nul" after the snapshot was produced. It will do this for all databases that are in full or bulk logged recovery model. This will render your own log backups performed after this useless. So, to avoid this, choose "Do not truncate logs." However, if you want to perform your own SQL Server backups then simply select "Perform copy only" on the General tab and Veeam will not interfere with your backup strategy. Simply put, the "Perform copy only" option is to help avoid Veeam interfering with your backup strategy.

How do you produce your backups?

Most SQL Server DBAs use some tool to generate the SQL Server backup commands. These tools typically also have the ability to do things besides performing backups, such as defragmenting indexes and checking that your databases are free of corruption, etc.

- SQL Server comes with Maintenance Plans which provide the ability to, among other things, produce SQL Server backups. These are typically produced to disk and the maintenance plan components will name the backup files so you will have the database name, date, and time in the backup file name, as well as a clean-up process to remove old backup files.
- There are other maintenance tools (scripts) available, which have advantages compared to the maintenance plans that come with the product. Perhaps the most commonly used is Ola Hallengren's Maintenance Solutions (https://ola.hallengren.com/). One advantage of Ola's tools is the smart index defragmentation handling which is designed to check the fragmentation level and only perform defragmentation for the indexes where we have fragmentation in the first place. This will save time including reducing time when the data isn't available, and also save space in the transaction log files and subsequent transaction log backups.

In the end, the above solutions will execute a job by the SQL Server Agent service. If you decide to let Veeam perform your SQL Server backup, then you will most likely still use some type of maintenance solution for all tasks except backup.

Should we use compression for the SQL Server backup files?

SQL Server has a compression option in the backup command. You may ask yourself whether or not it is appropriate to use this since Veeam will perform deduplication in the end.

Imagine a database where only a few pages have been modified between two backup occasions (two backup files). Without compression, these backup files will mostly be identical, where only small parts of the files will differ (for our example, remember that we only modified a few pages). Theoretically, deduplication would pick up on this and store the matching data only once. Compare this to the case where you let SQL server compress the backup data. Compression will likely "scramble" the bit-pattern so that the backup files will have little in common.

This might lead you to the conclusion that you shouldn't compress SQL Server backups. However, the way that deduplication works, the data being served by the hypervisor doesn't provide the data at a file-by-file level to the deduplication parts in Veeam. The end result is that deduplication might not do as much deduplication as is theoretically possible, and compression is likely to save on storage in the end. As always you should take in consideration the CPU cost for SQL Server to compress the backup data.

The bottom-line is that we do not recommend that you treat compression differently just because you happen to be in an environment where your SQL Server backup files will be picked up by Veeam.

Restore

We all know that it is important to practice restore and that a production failure is not the ideal time to practice a restore!

SQL Server has a GUI to perform restore built into the SQL Server Management Studio tool. The restore GUI will use backup history, which is stored in a few tables in the msdb database, to construct your RESTORE commands – and then execute these RESTORE commands if you wish (or you can use the script button to script them to a query window). It is, of course, important that it gets the restore commands right and this is where it gets a bit complicated.

The basic design principal for the restore GUI is that it uses backup history to figure out what RESTORE command to execute, based on what date and time you specify that you want to restore the database to. Unfortunately there are some "gotchas" to watch out for in this case.

First, Microsoft did a major change to the restore GUI between SQL Server 2008 R2 and SQL server 2012 and there have been minor changes with other versions as well. Obviously, we cannot point out every behavior change in every version, so consider the points below to be cautious about and verify whether they apply to you, if you want to use the restore GUI in the first place.

Perhaps the most obvious aspect is that the restore GUI only knows about the backups takes from the point in time of the machine. This might sound strange, so let me explain this better with an example. Say that you performed your Veeam snapshot on Wednesday at 04:00, you performed your SQL Server full backups Tuesday at 19:00 and transaction log backups every hour. Now, a problem occurred Wednesday at 10:43 and you want to restore the database to the point in time it had at 10:00 (your most recent log backup). This means you want to restore the Tuesday 19:00 full backup and all transaction log backups since, up to the one taken Wednesday at 10:00. Also, let's say the virtual machine also broke so you start by restoring the virtual machine from your snapshot taken Wednesday at 04:00. Your SQL Server backup history will now be from Wednesday 04:00 and there is no information in the restored backup history about the backups takes since 04:00. This means that the restore GUI in SQL Server can only help you to restore to 04:00, and the rest is up to you. This of course is assuming you have the log backup files copied somewhere else than your virtual machine, which is VERY important.

Another aspect is that the restore GUI might try to include COPY_ONLY backups and even snapshot backups when generating its RESTORE commands. Consider snapshot backups, for instance. Once again, let's take the above example but do not restore the virtual machine from a Veeam snapshot backup. Now, we want to use our SQL Server backups and bring the database to the point it had at 10:00. To do this, you specify the time 10:00 in the restore GUI and it will generate the first RESTORE command from the 04:00 snapshot backup, pointing to a filename for a file which doesn't exist (remember that this was a snapshot backup produced by Veeam). The restore GUI isn't smart enough to realize that this is a snapshot backup and skip/ignore it. I am not saying this happens in all versions of SQL Server Management Studio, but I have seen it happen in SQL Server 2012 and 2014. This may eventually be fixed at some point in time, assuming Microsoft considers it a bug in the first place. But it exemplifies how important it is that you **practice your restore routines**. And practicing restore cannot be easier than if you are using Veeam. With Veeam, simply perform a SureBackup restore and test/practice in that environment. Make sure you document your findings while doing this, as such a document is very valuable in a production situation emergency.

All of my clients have decided to not use the restore GUI for production restore. They have trained themselves to use the RESTORE commands from a query window, possibly aided by the ability to use the GUI, script the commands from the GUI, and then make adjustments in that script, if necessary, before executing the SQL commands.

There is nothing wrong with doing your own backups when using SQL Server with Veeam. However, there are several advantages of letting Veeam perform your SQL Server backups, as I will identify in the next section.

We have several components in play in our modern environments, such as virtualization software, backup solutions, and snapshot solutions at various levels, etc. It is more important than ever that we have actually practiced how to perform a restore. A production situation is not the right time to be surprised.

Using Veeam for your SQL Server backups

Using Veeam for your SQL Server backups may feel a bit unusual for the seasoned SQL Server DBA, so let us first lay out some facts before we dive into the details.

- There is nothing unsupported about the SQL Server backups produced by Veeam. Veeam uses supported and documented methods of producing SQL Server backups. More details will follow.
- There are potentially huge savings by letting Veeam handle our SQL Server backups. Consider the example we had above: SQL Server produced 2.8 TB per four weeks, to be stored on the backup server, when we didn't use differential backups and 0.48 TB when we used differential backups. What if we let Veeam do our SQL Server backups? It would take 0 TB since we already have the backup in our machine snapshots, which is a significant savings on our backup servers. We also save on disk space for each SQL Server virtual machine, since we do not produce and store SQL Server databases or differential backup files on the local machines.

- The restore process is all performed by the Veeam toolset. Veeam knows where all the backups are located and you have the same tools to perform both the machine restore as well as the SQL Server restore and in many cases it will be in the same place, your snapshot.
- Your transaction log backups will be copied away to a separate location immediately after the log backups have been performed. This is an interesting aspect, since if you keep the log backups only on the SQL Server machine and the machine breaks altogether, then your log backups will be lost as well.

The remainder of this document will focus on how Veeam works with SQL Server and what the SQL Server related configuration option in Veeam means.

Veeam supports two types of SQL Server backups:

- **Snapshot backup**. As explained above, this is a snapshot of the whole machine, from the point in time when the snapshot was produced. The SQL Server databases will be in a consistent state, thanks to the snapshot that was produced with assistance of the SQL Server VSS Writer service.
- **Transaction log backups**. You can complement your snapshot backups with transaction log backups. As we will see, Veeam will produce the log backups to files, using an ordinary BACKUP LOG command and then copy the log backup files to the Veeam backup repository. Transaction log backups will only be performed for the databases which are in full or bulk logged recovery model. So make sure you have the desired recovery model for your database.

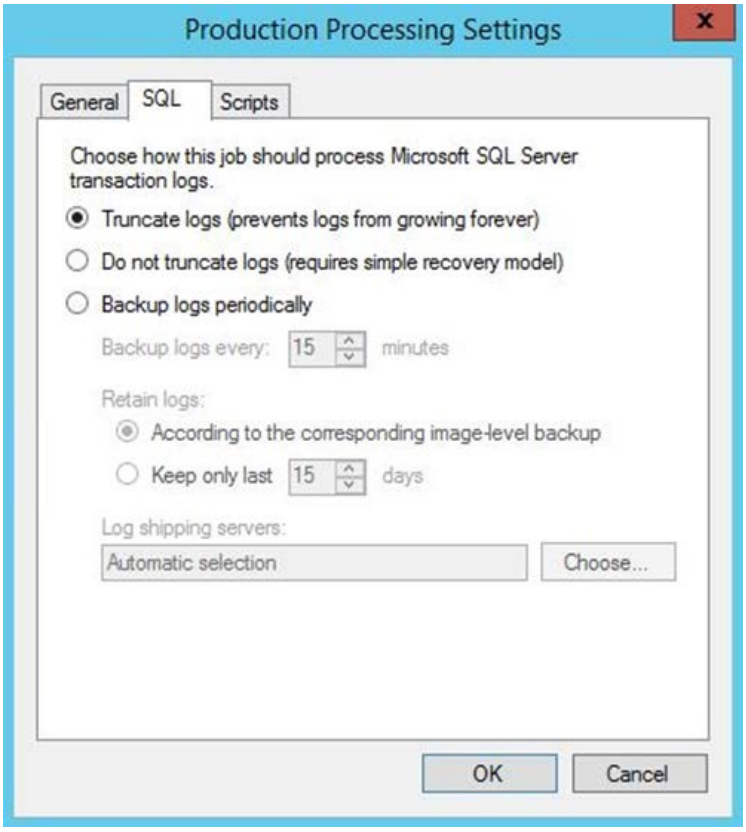
The full database backup – the snapshot

As we already explained above, Veeam produces a snapshot of your virtual machine using the VSS functionality in Windows. This, in conjunction with the SQL Server VSS Writer Service produces a consistent view of your databases. Veeam uses the VDI API to communicate with SQL Server that a backup is performed and SQL Server will consider this a full database backup. You might wonder whether or not it is important that SQL Server considers this a full backup (as long as it is supported and we can perform a restore from it)? Well, there is one aspect that might not be immediately obvious and that is the ability to perform log backups after a snapshot backup and restore such log backups.

Once again, let me share a link to an article that describes how snapshot backups work with SQL Server and VDI (Virtual Device Interface, the API used): https://technet.microsoft.com/en-us/library/cc966520.aspx. This article is technical, however, it will help answer deeper technical questions that you may have.

If you know you will not be doing any log backups, then you have two options to manage your transaction logs, i.e., make sure the ldf files don't grow indefinitely:

- You can make sure all your databases are in simple recovery model
- You can let Veeam truncate the transaction logs after the snapshot has been produced. This is implemented by Veeam executing the BACKUP LOG command to the file name 'nul'.



These two options are not mutually exclusive.

Using Veeam to produce transaction log backups

An interesting aspect with snapshot backups in SQL Server is that we can also complement them with log backups – just as we can use log backup to complement full and differential backups.

In order to do that, select "Backup logs periodically" in the "Processing Settings" dialog, the SQL tab – as you can see available above. It is that easy!

The implementation of log backups in Veeam

If you are curious how log backups work, then read on. If not, then skip to the next section.

Veeam installs a couple components on your SQL Server machines:

VeeamAgent which executes the BACKUP LOG command with the frequency you have specified. This component is started from the Backup Server. The log backups are performed to files, so there is nothing strange going on here. The following is an example of a backup command executed by the VeeamAgent:

```
DECLARE @database_name_var NVARCHAR(255)

SET @database_name_var = N'Tandlaege'

BACKUP LOG @database_name_var TO DISK = N'C:\ProgramData\Veeam\
Backup\SqlLogBackup\{78d18633-05ae-4613-b903-b2ea8854ad34}.bak'
```

As you can see, it sets the database name into a variable and uses that variable in the BACKUP LOG command.

VeeamLogShipper is the task that grabs the backup file produced by VeeamAgent and copies it to the Veeam backup repository (where the Veeam backup data is stored). If this server isn't directly reachable from the SQL Server machine, then one or more intermediate machines are attempted – log shipping servers. You can configure which log shipping server Veeam will attempt to use, but in the vast majority of cases, you want to let Veeam determing this for you. The VeeamLogShipper component is implemented as a Windows service.

If you look at the backup files in the Veeam backup repository, you will find files with the extension VLB. These are several log backup files compressed into a single file.

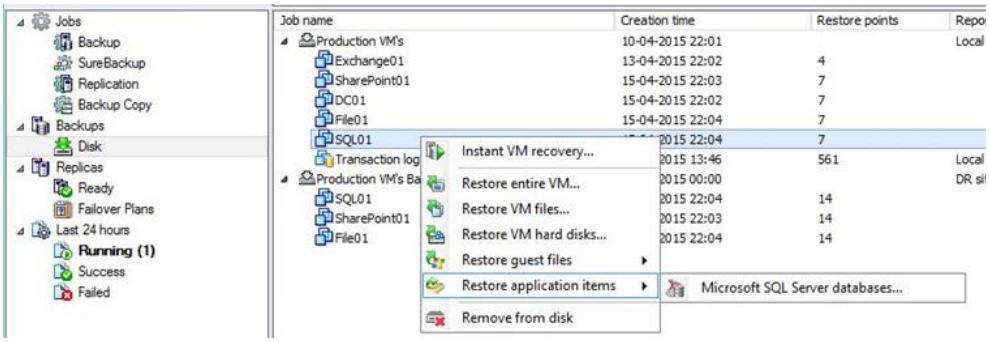
Restore databases using Veeam

The easiest and simplest restore situation, from a SQL Server point of view, is to simply restore the whole machine snapshot, to the Restore Point. There is nothing "SQL Server specific" about this type of restore. Of course, your databases will be based from when the snapshot was produced.

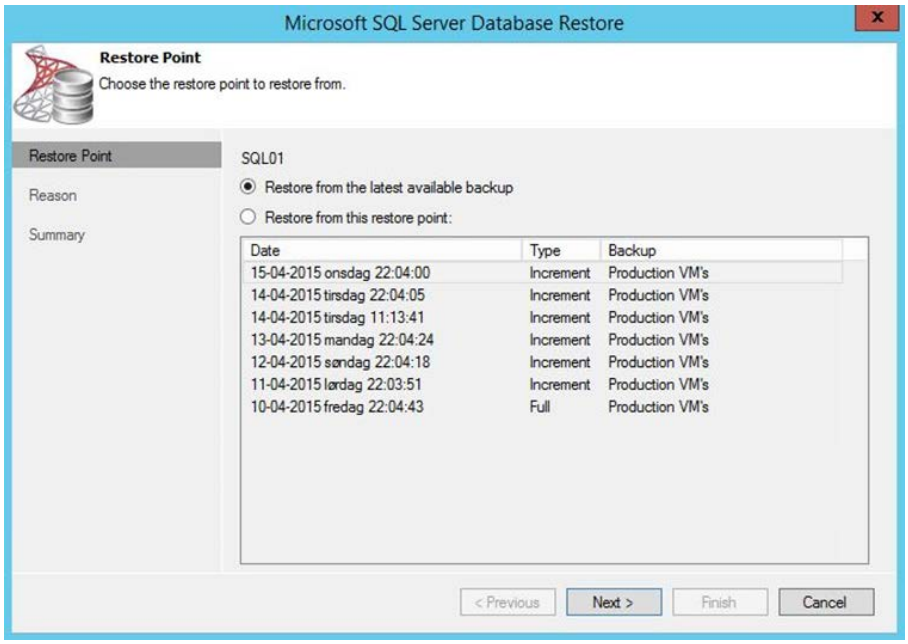
The other option is to restore from a Veeam restore point and then restore transaction log backups to the point in time you desire. Restoring to anything but your restore points will only be available for databases in full or bulk logged recovery model – and only if you let Veeam produce log backups. You don't have to be aware of what backups you have, restore commands, etc. You use any of the Veeam tools and specify your restore point, and optionally point in time – and Veeam will figure out what restore operations to perform.

Restore using the Veeam Backup & Replication tool

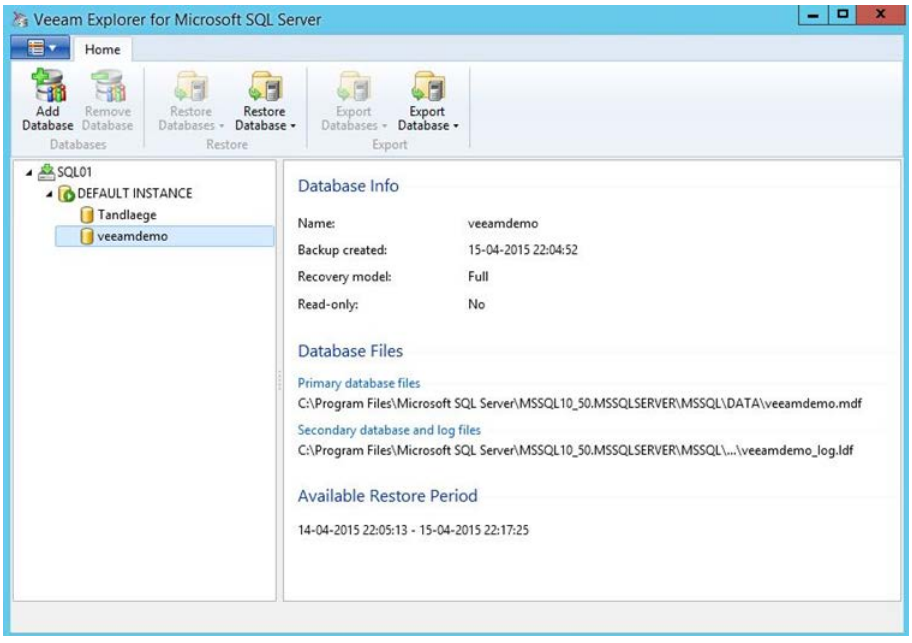
This is the tool that Veeam backup operators will be most familiar with. This tool is used to configure your Veeam environment and you can also use this tool to restore, as well as a number of other things. If you want to restore to anything else than a restore point, then you will use the "Restore Application Items," "Microsoft SQL Server Databases" option when you right-click your machine with SQL Server installed:



You then select the restore point you want to base your restore from:

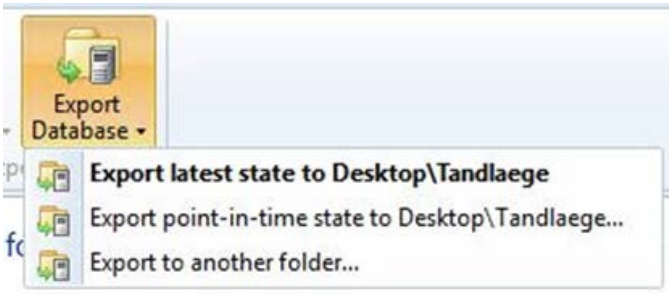


This will bring up a new tool titled "Veeam Explorer for Microsoft SQL Server":



When you select a database, you will see what time interval you can restore within. This is based on the Restore point you selected earlier. In technical terms, you have the snapshot from that restore point and then the subsequent log backups taken since.

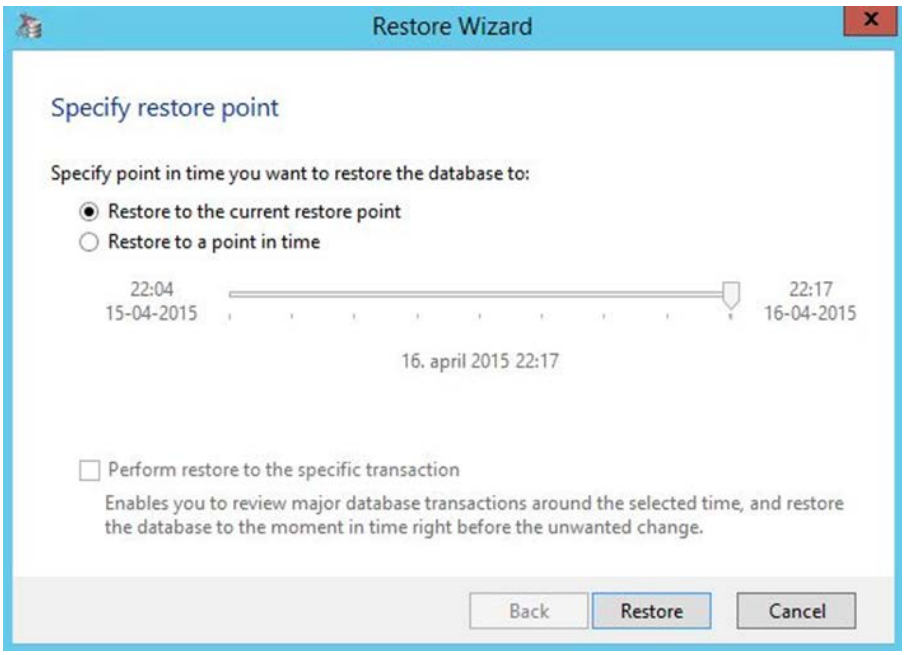
In this tool, you can select either Restore Database, or Export Database:



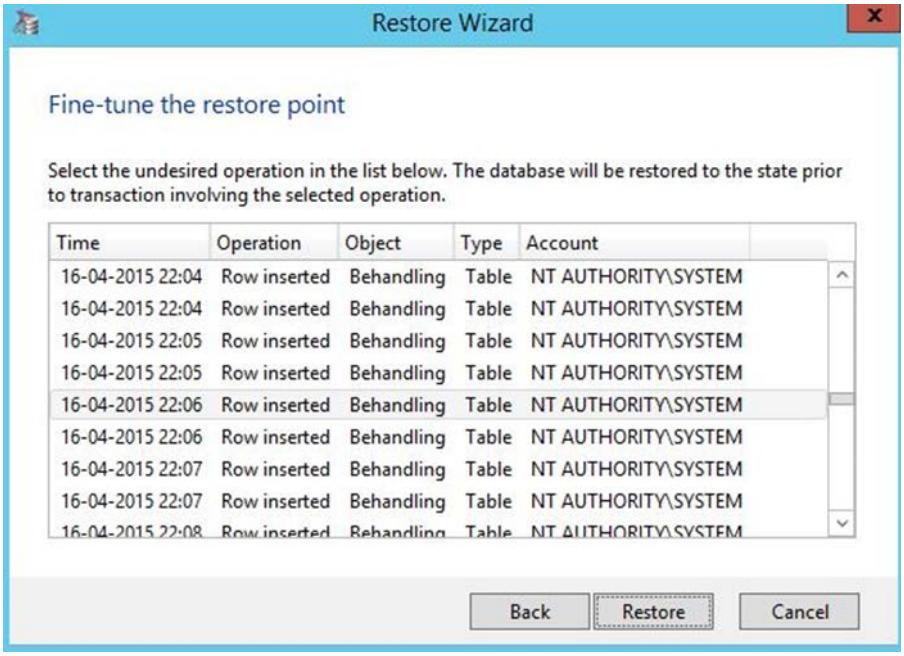
"Restore Database" will restore the selected database to the latest point in time, the selected point in time, or restore it onto a different SQL Server.

"Export Database" will create database files (mdf, ldf and possibly ndf) that you can copy to another SQL Server machine and attach to that SQL Server.

For either of the above two buttons, if you select the bottom two options (Restore/Export "point-in-time…" or "to another…") you have the option to select point in time and will be presented a dialog similar to below:



The top two options are self-explanatory. The "Perform restore to the specific transaction" will read the transaction backup file and based on that present to you the modifications reflected in there. You can select the operation and the database will be restored to the point in time just before this operation.



When is the local SQL Server required?

Some of the operations above require a local SQL Server (also known as "staging SQL Server"), installed on the Veeam server:

- Restore to the state before selected transaction (Fine-tune restore point)
- Export to selected point in time
- Export to the state before selected transaction (Fine-tune restore point)

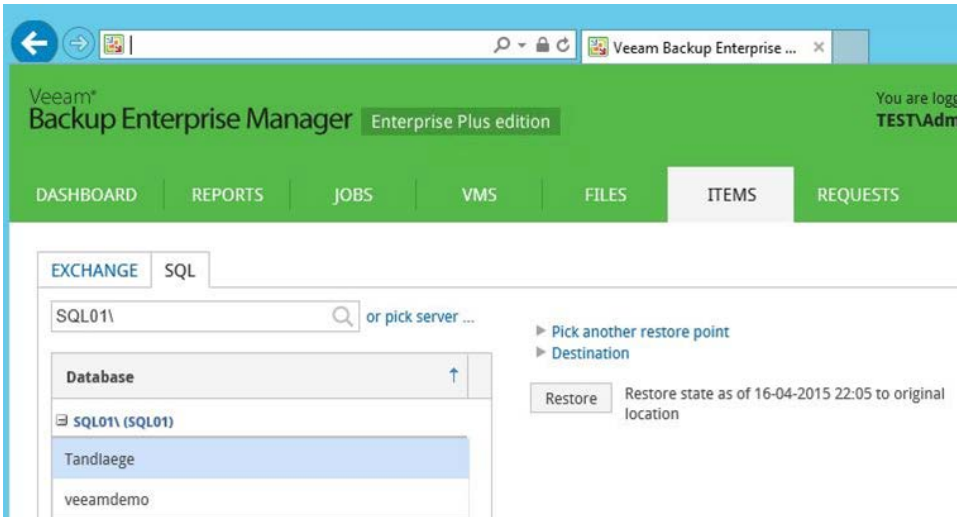
If features specific to a certain SQL Edition are used for the database, then the same Edition is required for the staging SQL Server. For example, the feature "Table and index partitioning" is supported only in Enterprise Edition. Furthermore, if the staging SQL Server is Express edition, then you will not be able to work with databases above the size limit for the Express edition (currently 10 GB).

Restore using Veeam Backup Enterprise Manager or Veeam Self Service File Restore

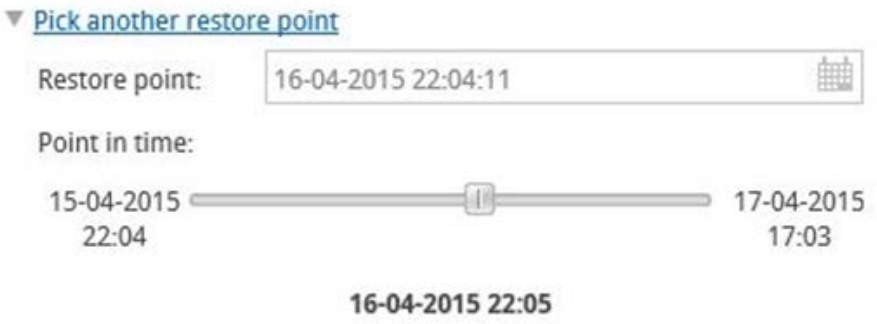
These two are really the same tool, where the Self Service tool is limited to restore only onto the machine where you are sitting. So, if you are on the SQL Server machine then you, using Self Service, can only restore onto this SQL Server machine.

The tool is designed to be easy-to-use. It is likely the SQL Server DBA will be more interested in this tool compared to the full-blown Veeam Backup & Replication™ tool. Also, since this tool is web-based, bits of the tool can be incorporated and customized into other tools and solutions (using a REST API), such as from a Service Provider.

Use "Items" in this page when you want to restore SQL Server databases, as shown below:

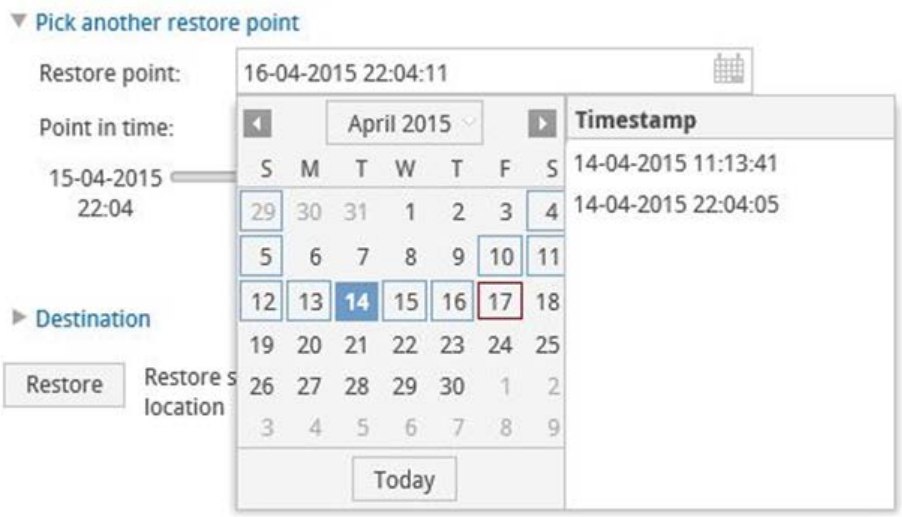


By default, you will restore based on the most recent restore point. However, since you might have performed transaction log backups for the database, you want to click "Pick another restore point" so you can dig a bit deeper into when you want to restore to:

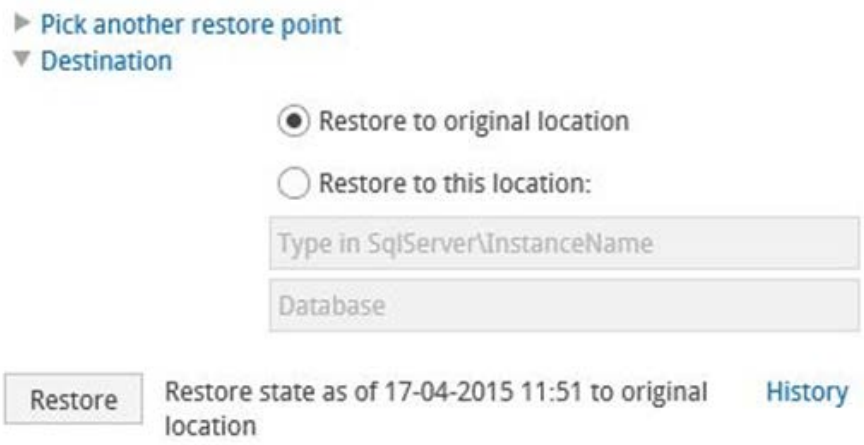


By clicking "Pick another restore point" you can select your restore point (your snapshot). You can use the slider to specify point-in-time restore if there are transaction log backups available for the selected database. The default is to restore to the exact restore point. If you want to restore to a more recent point (you have transaction log backups), then use the slider to specify that point or earlier. The range for the slider is based on the selected restore point, back to the prior restore point, up to the next restore point, or the most recent transaction log backup, if it is the last restore point.

Note that there might be several restore points per day, i.e. several snapshot were produced that day. You select to the right of the day selector which restore point to use:



You can either restore the database to its original location, or select a different SQL Server to restore to:



The credentials used for the restore commands are specified for the Guest OS in the Veeam Backup & Replication tool.

Hybrid strategies

You can combine the two strategies outlined above, the "Let SQL Server produce backup files" strategy and the "Use Veeam to backup your SQL Servers" strategy.

The DBA centric hybrid strategy

This strategy has already been described. As long as you make sure the machine snapshot is produced using the SQL Server VSS Writer Service (enable "Enable application-aware processing") then the machine snapshot produced by Veeam is a valid restore option for your SQL Server data. And then, of course you also have the backup files produced by your SQL Server, as described in that section.

Two distinct options for restore:

- Perform restore using Veeam. Use Veeam snapshots (Restore Points) only.
- Perform restore using SQL Server RESTORE command or GUI. Use whatever backup files you told SQL Server to produce (full, differential, log).

The backup operator centric hybrid strategy

This strategy means you will let Veeam produce its snapshot (restore point) and also transaction log backups. The question now is whether we can perform this while also performing SQL Server backups to file – and not have SQL Server somehow interfere with the Veeam snapshot and log backups?

The answer is simple… Yes, we can!

You can produce full backups any way you want – they won't interfere with the Veeam snapshot or log backups. But you don't want to use the COPY_ONLY option for these full backups, if you also want differential backups (see next paragraph).

If you want, you can now complement these full backups with differential backups. For instance, you perform a full backup every night and a differential backup every four hours.

Full backups or differential backup do not interfere in any way with the chain of log backups produced by Veeam.

However, in this scenario, you want to avoid allowing SQL Server to produce log backups. Remember that this scenario means that Veeam produces log backups.

Again, here are two distinct options for restore:

- Perform restore using Veeam. Use Veeam snapshots (Restore points) and the log backups produced by Veeam.
- Perform restore using SQL Server RESTORE command or GUI. Use whatever backup files you told SQL Server to produce (full, differential – but not log).

Summary

You can use Veeam for all of your SQL Server backup operations, or only to produce snapshots of your virtual machines. We suggest the person responsible for the overall backup strategy discuss with the DBA whether or not to use Veeam for your SQL Server data. There are several advantages for allowing Veeam to handle your SQL Server backups, including storage space saving on both the backup server and your SQL Server virtual machines, and a single interface for all of your backup and restore operations, etc.

Veeam uses supported measures to back up your SQL Server, and when you let Veeam also produce transaction log backups you have a single place for all of your restore operations. The backup operator will most likely be using the Veeam Backup & Replication tool for the restore operations, where the DBA might be more comfortable using the web-based interface. Regardless of which tool is used, you simply specify the date and time you want to restore to by specifying the restore point, and if you have log backups, then select exactly what time to restore to – and the restore tool will perform the restore operations for you.

And you can also restore onto a different SQL Server, or export to new database files which you yourself can copy to another SQL Server and attach.

About the Author

Tibor Karaszi

Tibor lives in Stockholm, Sweden. He has worked with Microsoft SQL Server since 1988. In 1997, Tibor earned the title Most Valuable Professional (MVP), as the 7th SQL Server MVP in the world. He has been honored with an MVP award each year since.

Tibor's SQL Server training experience dates back to 1991. Since then, he has delivered internal SQL Server training for Microsoft in Sweden and internationally.

He is the co-author of several books on SQL Server and was a regular contributor and editor for SQL Server Magazine. Tibor also speaks at conferences, seminars and workshops, including events organized by Microsoft. He co-manages the Swedish SQL Server user group, SQLUG.se, since its inception.

Tibor currently works as a consultant, focusing solely on Microsoft SQL Server. Assignments include training, reviewing both code and design, elaborating on issues as a discussion partner and other more hands-on consulting assignments.

About Veeam Software

Veeam® recognizes the new challenges companies across the globe face in enabling the Always-On Business™, a business that must operate 24/7/365. To address this, Veeam has pioneered a new market of *Availability for the Modern Data Center*™ by helping organizations meet recovery time and point objectives (RTPO™) of less than 15 minutes for all applications and data, through a fundamentally new kind of solution that delivers high-speed recovery, data loss avoidance, verified protection, leveraged data and complete visibility Veeam Availability Suite™, which includes Veeam Backup & Replication™, leverages virtualization, storage, and cloud technologies that enable the modern data center to help organizations save time, mitigate risks, and dramatically reduce capital and operational costs.

Founded in 2006, Veeam currently has 29,000 ProPartners and more than 135,000 customers worldwide. Veeam's global headquarters are located in Baar, Switzerland, and the company has offices throughout the world. To learn more, visit http://www.veeam.com.

VEEAM

IT JUST WORKS!™

AVAILABILITY™

for the Modern Data Center

RTPO <15 min for **ALL** applications and data

Veeam Availability Suite v8

High-Speed Recovery

Data Loss Avoidance

Verified Protection

Leveraged Data

Complete Visibility

To learn more, visit **www.veeam.com**