Enterprise PDM - Backup and Restore

Field Services - Best Practices

[Enterprise PDM 2010]

[September 2010]

[Revision 2]
Brief Overview:

This document describes factors influencing the processes of backup and restore for the PDM Enterprise product set. It first describes the general principles for SQL Server, and then applies this to the Enterprise PDM architecture.

There are few rigid rules in this area, a single strategy will not be recommended; however if all the available options are understood, it will help you to define the best strategy for the specific requirements of the customer.

It is intended that this document will be expanded and enhanced over a number of versions, as field experience provides feedback on the most useful areas.

Notes for VAR

(i) This best practice document was written as a guideline and should not be used without proper alignment to the customer requirements and situations. Not all the alternatives discussed in this document will be suitable for all customers.

(ii) When specific commands, or screen images are given these will refer to SQL Server 2008 (unless specifically noted otherwise). In most cases the operations will also apply to SQL Server 2005, although some details may vary.

Additional reading material

Various Microsoft Knowledgebase articles are available to support SQL Server; some of these are listed in the References section.

The list will be expanded in subsequent versions of this document.
Backup Strategies for SQL Server

Your data is critical to your business. You need to be able to reliably back up your data systems, and then easily restore them if there is some sort of failure. This is especially true in an environment that is running a PDM system such as Enterprise PDM.

To some extent, your strategy will be determined by the recovery model you use. There are factors which influence the best model to support Enterprise, but there are other organizational factors which will influence your choice of strategy.

SQL Server itself provides four recovery models, each with a different approach toward the trade-offs between conserving disk space and providing granularity.

Simple Recovery Model

As the name implies, this is the simplest of the recovery models. If you choose this model, you can restore only full or differential backups. You cannot restore to any specific point in time. SQL Server truncates the transaction logs when the database reaches a transaction checkpoint, and it leaves no log entries for disaster recovery. You lose any modifications to the data that may have been made between the last full or differential backup and the failure. In some scenarios, particularly those in which read-only databases are used, this potential loss can be an acceptable trade-off for the savings in disk space, time used, and the simplicity of the process.

Bulk-Logged Recovery Model

This recovery model is recommended to be used on databases that are already using Full recovery and performing bulk load operations (i.e., data warehouse load).

If you choose the bulk-logged database recovery model, you have the ability to take transaction log backups. However, you are not able to restore to a specific point in time with these log backups. You must restore the entire transaction log backup. This is better than in the simple recovery model, because at least you won't lose any data. But, you probably don't need to use the bulk-logged recovery model unless you are continually performing data loads.

This method is not particularly applicable to Enterprise PDM environments, so it will not be considered in later discussions.

Full Recovery Model

The full recovery model gives you the most flexibility on the types of backups that you can take. You can use full database backups, file backups, differential backups, and transaction log backups. The choice of the backup strategy in this scenario will vary greatly based on your database size, the backup medium, and what level of recoverability you want to have for the data.
Database file or filegroup backup

There is a technique where copies are taken from specified file(s) or filegroups(s) to allow quick, piecemeal restores if only part of the database is lost. For example a complete .mdf file could be replaced. However this strategy is very imprecise, it does not always take account of the interactions between the various different files. This backup/restore strategy is not recommended for Enterprise PDM deployments, and will not be discussed further.
The steps in making the backup

Database administrators (DBSs) are charged with not only protecting your data but also being able to recover it in case of emergencies. For that reason, knowing the different types of backups that are available to you in SQL Server, how they work, and how to properly manage them is a critical skill set to possess.

SQL server offers three recovery models, one of which can be selected when the database is created, or can be set later.

- Full recovery mode
- Bulk-logged mode (not discussed here)
- Simple recovery mode

Full Recovery Mode

In the Full Recovery mode, SQL Server logs every part of every operation. When a transaction occurs, the transaction log keeps this record until a transaction log backup occurs, which then clears the transaction log.

A database in Full recovery mode allows you to do "point-in-time recovery." This model requires the most administration because all operations are logged and the transaction log is not truncated until a transaction log backup is performed.

Be careful not to run out the disk space. Databases are usually set to Full recovery mode by default. If the database is left in Full recovery mode and regular transaction log backups (or full backups) are not taken, the log will fill up and grow until disk space runs out.

The only way to clear a transaction log is to perform a transaction log backup. So, when you are using the Full Recovery mode, you should take transaction log backups as often as you can afford to. To start taking transaction log backups, you must start what is referred to as the "backup chain" by performing a Full backup of the database. To do this, follow these steps:
1. In Microsoft SQL Server Management Studio, expand the SQL Server instance node.
2. Locate the database that you want to back up.

Right-click the database, select Tasks, and then select Back Up. The Back Up Database dialog box opens.
3. Make sure that **Full** is selected in the Backup type box.

4. In the Destination area of the Back Up Database dialog box, select the location where you want the backup to reside. By default, the location that is specified is `%SQL Server Installation Path%\Backup\<database name>.bak`. But this is not ideal, it is better to define an area specifically suited for the purpose. (Preferably on another disk drive.)

5. Click **OK**. This will initiate a full backup of the database. When it is complete, you will get a confirmation message. Click **OK**.

After the first full backup is complete and the backup chain has been established, you can perform a transaction log backup. To do this, repeat the steps above. However, this time, click **Transaction Log** in the Backup type box. Leave everything else the same, and click **OK** to start the backup.
**Simple Recovery Mode**

The Simple recovery mode is the easiest to both backup and restore. In the Simple recovery mode, the logging works the same as in the Bulk-logged recovery mode. But, transaction log backups are not needed because the inactive transactions in the log are automatically truncated after a checkpoint.
So for Simple Recovery mode, you would take a FULL backup, as described above, then periodically you would take a differential backup.

If you are using SQL Server 2008 Enterprise Edition or later, use backup compression to reduce the size of backups and the time required to take them. You should always backup to disk, (preferably a separate disk drive) to ensure that the time taken is as short as possible.

In practice it is likely that the DBA will set up a maintenance plan which follows these steps and can be defined to run at specific days and times. The DBA should also include notification steps and error handling within the plan.
Backing up the rest of the Enterprise system

Synchronization of the database and Archive file backups

A critical point to consider when managing overall backup of the Enterprise PDM environment is the interaction between the SQL Server database and the archive file storage.

As shown in the architecture diagram, the Archive Server stores the files under the 16 folder structure.

When a Vault is initialised a single folder with the vault name is created in the "Root Folder" to represent to “top” of the Vault, then 16 sub-folders are created, (numbered 0-F in hexadecimal notation).

![Archive Server Diagram]

The Enterprise database contains references (essentially file pointers) to content in this file repository. To ensure maximum consistency between the database file references and the actual content of the file repository, serious attention must be paid to backing up both simultaneously.

Backing up the Archive Files

There are basically two approaches to doing this:

- **Option 1**: Take cold backups—that is, take the database system offline to halt user activity, back up both the database and file repository, and put the database back online.

- **Option 2**: Take online backups and accept some small potential inconsistency. The issues with this approach can be minimized if the backups are taken during periods of low activity.

You can take the entire Enterprise offline by taking the database itself offline. Users will not be able to access any Enterprise functions if the database is not available.

The backup of the Archive file set can be made using whichever tools are available at the customer site. Something as simple as a shell script using “XCOPY /S” could be used to make a full copy of the entire folder structure. More traditional backup tools would be able to
make scheduled full and incremental backups of the file set, and should certainly be recommended when the size of the Vault means that the duration of backup operation becomes a significant factor.

Since it is critical that the Vault (file) backup and the database for the vault are consistent and in synchronisation, this will influence the level of granularity that you can achieve in your backup/restore plan. For example if you only backup the Archive files once a day (during the night), then it is only necessary to be able to restore the database to that same timeframe. This is why, in most cases we recommend the Simple backup approach for SQL Server, it tends to match an overnight file backup strategy.

**Backing up the Archive Server Settings**

The archive server contains file vault settings such as passwords and defined login types. It is also the physical location of the Enterprise PDM vault archive files. Remember - Backing up the archive server settings will not back up the archive files.

To backup the archive server settings:

1. On the archive server, from the Windows Start menu, click **All Programs > SolidWorks Enterprise PDM > Archive Server Configuration**.
2. Select **Tools > Backup settings**.
3. In the Backup Settings dialog box:
   4. Select **Include all vaults**. (preferred setting)
Alternatively, you can select **Include selected vaults** and specify the file vaults for which settings will be backed up.

1. Specify or select the Backup location. The default location is the archive root folder.
2. To schedule an automatic backup, click **Schedule** and specify the schedule.
3. Type and confirm a password for the backup file. This password is required to restore settings.
4. Do one of the following:
   - To perform the backup immediately, click **Launch backup**. When a message confirms the backup, click **OK**.
   - To perform the backup at the scheduled time, click **OK**.

The backup file is saved in the specified location and called Backup.dat, and should be backed up along with the archive files. This information will only change when the Archive Server settings are changed, so occasional (e.g. weekly) backups may be sufficient.
Restore Strategies

In this section we will discuss the other half of the equation: restoring the database. After all, backups are only as good as the last restoration. We will cover the following topics:

- How to perform a basic restore operation
- How to restore to a point in time

Restoring a Full Database Backup

The first thing you should do to test your backups is to try to restore your full database backup. This type of database backup will work for every database recovery model. You should practice restoring from the full backup via every available mechanism. Use the graphical interface (SQL Server Management Studio), the Transact-SQL RESTORE command, or the Windows PowerShell scripting. You will not only have more confidence in your backups, but you'll also save valuable time when a crisis hits and you need to actually recover your database.

Recovery in addition to the Database Backup

Your database backup will not contain everything you need to recover from a server failure. You will also need backups of your system databases (at least the master databases). It's a good idea to have a backup of the SQL Server registry keys and installation settings. Additionally, you should have a record of the service account names and passwords that you used to install SQL Server.

Recovery is a nontrivial operation, and you will almost always be under stress when you are doing a real database recovery. The next sections discuss the restore process and what’s needed to ensure a smooth recovery of your databases.

How to Perform a Basic Restore Operation

Before you begin a restore operation, ensure that you have at least one good full backup of the database. This establishes the backup chain. To restore a database, follow these steps:

1. In SQL Server Management Studio, right-click the Databases node, and then click Restore Database.
2. In the **Source for restore** area of the **Restore Database** dialog box, select which database you want to restore. After you select a database to restore, the name of the database is automatically populated in the **To database** box in the **Destination for restore** area. For this example, we will restore the database to its original location with its original name.
3. In the **Select the backup sets to restore** area, click to select the check boxes for the backups that you want to restore.

The **Select the backup sets to restore** area displays all the various backups that exist for this database. The backups that you see are part of the backup chain. This backup chain contains the last full backup and any differential and transaction log backups that were taken after it.

![Image of Restore Database - PDMWE_MCADPDM](RestoreDatabase.png)

**Notes:**

- By default, SQL Server assumes that you want to restore the database to the point of the latest backup. When you select a database to restore, only the backups that are needed to perform the restore operation are selected.

- Because a full backup starts and ends a backup chain, you will only see one full backup in the list at a time.

4. Under Select a page, click Options.
5. In the Recovery state area, leave the following default option selected: Leave the database ready for use by rolling back the uncommitted transactions. Additional transaction logs cannot be restored. (RESTORE WITH RECOVERY) Note: On the Options page, you can select several important restore options. I'll discuss the other options in more detail later on.

6. Click OK to start the restore operation. After the database is restored, a confirmation lets you know that the database was restored successfully.

**How to Restore to a Point in Time**

Your backup configuration and restoration goals will determine the steps you need to take to recover your database fully. If you are using a backup strategy like the following one, you can select a recovery strategy to match:

- On Sunday, you take a full database backup.
- On Monday through Friday, you take a differential backup at 6 A.M. every day.
- You take transaction log backups every four hours, starting at 7 A.M. every day.

As practice, imagine that your database fails at 11:30 A.M. on a Tuesday. Your first step is to take a transaction log backup of the remaining transaction log entries. This is referred to as backing up the "tail of the log." You would then recover your Sunday full database backup, the differential backup from Tuesday, the transaction logs from 7 A.M. and 11 A.M., and the backup that you just completed for the tail end of the transaction log. Practicing this sort of recovery scenario at your leisure is much less stressful than attempting to perform a complicated recovery after hours or in a real emergency.
As mentioned earlier, by default, SQL Server Management Studio selects the most recent backup as the point of restoration. But, there are times when you want to restore to a specific point in time. The only way to perform point-in-time restorations is for the database to be in the Full Recovery mode.

To restore to a point in time, follow these steps:

1. First, you must perform a tail log backup:
2. In SQL Server Management Studio, right-click the database, point to Tasks, and then click Back Up.
3. Change the Backup type to Transaction Log.
4. Under Select a page, click Options.
5. In the Transaction log area, click Back up the tail of the log, and leave the database in the restoring state.

- Click OK to start the backup. After the backup is complete, click OK in the confirmation dialog box.
- The database is now in a Restoring state. Repeat steps 1 and 2 from the previous section.
• This time, in the Destination for restore area, Most recent possible is selected in the To a point in time box. To change this setting, click the ellipsis (…) button. The Point in time restore dialog box opens.

![Point in time restore dialog box](image)

• In the Restore to area, click A specific date and time, enter the date and time that you want to restore your database to, and then click OK.

• In the Select the backup sets to restore area, all the backup files that will be used to fulfil your restore request will be selected. Click OK to start the restore operation. After the database is restored, click OK in the confirmation dialog box.

**Restoring the Archive**

The archive file set needs to be brought back into synch with the database which has just been restored. It may be that you have had a hardware failure localised to the SQL Server computer, so the Archive Server(s) is still operational, so on the face of it you do not need to worry about the Archive Server.

But remember that the files MUST be synchronised with the database, if the system is to remain consistent. So you need to replace the Archive Server folder structure with the file backup taken at the same time as the database backup was made.

It would be sensible to MOVE the current files to some other temporary area, before copying the backup version of the files into place. This will provide you with the possibility of salvaging any files which changed between the backup and the failure. This is a manual process, but potentially still preferable to losing the work.

**What will we lose?**

If you have a system failure which can only be recovered by applying a backup, then typically you should expect some data to be lost. For example if you take backups at midnight and you have a disk crash at 11.00am, you will need to revert to the previous overnight backup. Once this is restored, the system will be back to the state it was in at midnight. All PDM work done since midnight will be lost, with one very important exception described below.
Important Note: If people have been working on local files after the checkpoint, then you restore the servers to the check-point, the users have not necessarily lost their work, since it is still available locally, however there may need to be some manual fixing-up to get the entire system back into synch.

As an example, imagine the user begins work by checking out his file(s) and has made some changes when the Enterprise system is taken down to be restored to the previous night’s backup checkpoint. When the system returns, our user finds that the system says that the files are not checked-out, his local files are not known to the Enterprise system. If he wishes to preserve the work he has just done, he needs to copy his local file(s) to a temporary area, check-out the 'original' files, and then overwrite the original files with his modified files. The system now has a consistent view of the users files, and little actual work has been lost.

Users who are working in Offline mode, during a crash/restore sequence users are even easier, since, the user’s local copy is defined as being ‘the master copy' anyway, so nothing will be lost.

Remember, however that database-centric operations, for example workflows, will be reset to their over-night checkpoint, so it is possible that users may receive notifications a second time, or need to approve once again a workflow transition that had been 'lost'.
Fire Drills: Practice Restoring a Database

Restoring a database is an essential part of any DBA's job running Enterprise. Thus, the topic of restoring a database seems well worth digging into more deeply.

A Quick Review of a Basic Restore Operation

We discussed recovering your database from the database backups you took previously. Your recovery strategy is only as good as your backup strategy. A "fire drill" can test to make sure that you've got the appropriate kinds of backups, that they were written to the appropriate kinds of media, and that they were taken at the appropriate frequency to match your recovery goals.

Once you have the backups, you should practice these tasks:

- Restore the databases.
- Apply differential backups.
- Apply differential logs.
- Try recovering to a specific point in time to understand how that works.
- Recover to a marked transaction point.
- Recover the logs over a period of several logs to understand approximately how long it actually takes to perform the restore operation.
- Try to use a variety of media (local hard drives, SAN storage, and tape), and discover the advantages and disadvantages with each type of media.

Where Should the Restored Database Go?

This question is not as strange as it appears at first. To get the best sense of how things will go in a real situation, you would want to recover your backups to your production server (preferably in place) during some downtime on the server. However, this can be a high-risk operation because you have the possibility of affecting your production databases. Additionally, you would be placing a load on the system.

You could possibly restore with a new database name on the same server, but that only slightly lowers the risk. Therefore, you will probably end up restoring to a test server. This will complicate the restoration, as your disk, files, and filegroup layouts may be different on another server and this may make the restore commands more complex. However, it significantly lowers your risk of accidentally damaging your production server. Additionally, if you make a critical mistake, you can always drop a test database and start the recovery all over.
Recovering other parts of the system

A great fire drill is to assume that your database server was either damaged or lost (perhaps it was stolen, a fire occurred, the HVAC failed and the computer will not start up anymore. Consider these questions?

- How would you go about recovering that server?
- Do you have copies of Windows Server handy?
- If this is not directly under your control, do you know who to call?
- How long will it take to get another computer with similar memory, storage, and processor capabilities? Do you have a backup computer available?
- What settings were used to configure the computer before SQL Server was installed?
- Are the service accounts local, or are they domain accounts?
- Who had administrator rights on the computer?

If you cannot answer these questions, you need to spend some time investigating and documenting this information so that it is readily available in a recovery situation.

After you correctly install and configure Windows, install SQL Server. Here are some questions to think about:

- What collation was used before?
- What service accounts will you need to set up?
- Do you remember the location of the system drives?
- Did you use any trace flags?
- Will you need other server-wide rights?
- Did you also install SQL Server Integration Services or other optional modules on that system?
- Do you have ability to restore Registry settings on this server?
- What about local management tools?
- Do you have backups of the system databases (master, msdb) etc.
- Do you have backups of the ConisioMasterDb database?
- Do you have scripts to re-create critical logons, security objects, and jobs?
- Did you use encryption, and do you have backups of any security keys and certificates?

As you can tell, going through this exercise will almost certainly lead you to discover items you have missed or do not have documented to a sufficient standard.
How to design the backup strategy

We have discussed the mechanisms involved in the database backup and restore operations, now we will examine how they can be applied to Enterprise PDM systems within various types of company.

The main issue is the ability to achieve a coordinated backup of the database and the archive files such that no changes are made to either database or files during the time it takes to make the backup copy.

Companies with down-time.

One practical strategy is to identify a time where there are no transactions happening in the Enterprise system, generally during the night, and then take both a file backup and a database backup at the same time. If there is no activity during an over-night period, then the two backups need not be done at precisely the same time, just both within the period of zero activity. In fact if you do not expect any activity, it is not even necessary to take the database offline. The time of this set of file copies represents the check-point to which you will be able to restore.

This will be suitable for a single-site organisation. It would also suit an organisation with several sites spread over adjacent time-zones. For example several sites across Europe may cover several time-zones, but still allow an overnight period where none of the sites is working.

If there is enough time, and sufficient disk space, make a FULL backup each night. This will allow a quicker and simpler recovery.

Companies that work around-the-clock.

Sometime it is not possible to identify any 'down period', probably because the organisation is spread world-wide and has a 'follow-the-sun' working practice. In such a case the best that can be done is to manage the working practices to provide a suitable backup opportunity.

There are no specific tools in Enterprise to switch it into a read-only mode or enforce a "zero-activity" period, but it may be possible to define a working practice to achieve the same effect. Remember in this context "zero-activity" means:

- No Check-in or check-out operations
- No "Get" operations
- No Workflow activity (no transitions, revisions etc.)
- No changes to meta-data (cards)
- No Rollback operations
Users are still allowed to:

- Work on their local files
- Perform Searches in Enterprise
- Refer to Enterprise (e.g. cards, computed or named BOMs)

It may be possible to identify a time when this "zero-activity" can be achieved without being intrusive for users, for example during a meal break, a team briefing, a shift-change etc. The approach here must be to minimise the time it takes to make the backup, so differential backups will help.

A practical approach is to schedule a FULL backup of files and database over a weekend, then you can perform a DIFFERENTIAL backup of files and database every night. This minimises the time necessary. It may be that you can limit the weekday backup period to a few minutes, so you could find a time when this would not disrupt the working practices.

Also ensure that the backup is disk to disk to tape, rather than disk to tape. In other words, do not backup from your archive files to tape, this will be slow, copy the archive files to another disk, and then copy to tape, this will minimise the down time of the working files.

**Restoring the Archive files**

Although the SQL Server backup/restore techniques provide tools to achieve precise 'point-in-time' restoration, this is only as useful as your Archive File backup granularity. If you only backup the files once a day, it becomes meaningless to be able to restore to any checkpoint other than that used for the archive files.

If the file backup takes a long time, you may wish to restore to a point in time corresponding to the end of file backup, (although this would imply DB activity during file backup – this is not a good thing to allow)

The most important parameters involved in planning your backup/restore strategy become how quickly you can perform the backup of your archive files, as well as how quickly you can backup the database.

This has two aspects:

- the time taken to perform a FULL backup of the files
- the time taken to perform a DIFFERENTIAL backup of the files (i.e. how fast the vault is growing)

**General factors**

There are many file backup technologies available, and it is beyond the scope of the document to recommend any in particular, but you should focus your attention on understanding and optimising the backup of your archive files. In addition may customer site
will already have an IT infrastructure that provides some backup tools or schedules, these need to be understood and integrated into any proposed strategy.

As we have discussed, SQL Server provides a comprehensive set of tools to help with backup and restore, these are well documented elsewhere, and the customer should be encouraged to familiarize themselves with the tool set.

The resource level and skill level of the customer IT staff will also have an influence on the practicality of any Backup/Restore strategy. If the customer already has SQL Server skills in-house, it may be appropriate to suggest a Full Recovery Model based strategy, however this assumes and requires a skilled DBA to implement properly. If the customer has a lower IT skill set it would be more responsible to recommend the Simple Recovery model, since this is more robust and self-contained.

**Summary**

Knowing the steps required and the time needed to restore your database is probably the most important part of this whole topic. The advantage of restoring databases on a regular basis is that it allows you to verify that your backups are still valid and can be used for a restore.

Another advantage is that it gives you an idea of how long it takes to restore a database during a disaster recovery scenario.

The ideal backup/restore procedure for a given organisation will be a compromise between perfect data security, and the level of intrusion into working practices and the investment in time and equipment needed to maintain the procedures.

While SQL Server itself offers a lot of tools for backup and restore, it must be remembered that the *entire* Enterprise system must be restored to a synchronised state, and this must be the ultimate target for any processes that you design.
References

There are many sources for technical information on these topics; here are some, but note that the list is not exhaustive:

Solutions

<table>
<thead>
<tr>
<th>Reference</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-038742</td>
<td>Does the Microsoft SQL Server need to communicate with the Domain Controllers to authenticate users when using the Windows® login option in Enterprise PDM?</td>
</tr>
<tr>
<td>S-042417</td>
<td>How is the firewall port opened for SQL Server on Windows® Server 2008?</td>
</tr>
<tr>
<td>S-042552</td>
<td>Is it possible or recommended to rebuild SQL Server database indexes in a SolidWorks Enterprise PDM file vault database?</td>
</tr>
<tr>
<td>S-042701</td>
<td>How can I verify or test that an Enterprise PDM client or archive server can communicate with the SQL server hosting the file vault database (via ODBC test)?</td>
</tr>
<tr>
<td>S-022452</td>
<td>What is the recommended procedure to make a backup of an existing PDMWorks® Enterprise vault?</td>
</tr>
<tr>
<td>S-013295</td>
<td>How is a PDMWorks® Enterprise SQL database backup in SQL Server 2005 management studio restored?</td>
</tr>
<tr>
<td>S-014843</td>
<td>Is it possible to stop or start the PDMWorks® Enterprise and SQL Server services from a command line or using a batch script?</td>
</tr>
<tr>
<td>S-016097</td>
<td>Is the backup / restore of the SQL transaction log supported in a PDMWorks® Enterprise?</td>
</tr>
<tr>
<td>S-021061</td>
<td>How is a manual backup of a PDMWorks® Enterprise file vault SQL database performed using SQL Server Management Studio? (2005)</td>
</tr>
<tr>
<td>S-025691</td>
<td>What are the best practices in setting disk configuration for a Microsoft® SQL Server?</td>
</tr>
<tr>
<td>S-026469</td>
<td>Does Enterprise PDM have any issues with a Microsoft® SQL Server setup for a dynamic port used by a named instance (using the SQL Server Browser service)?</td>
</tr>
<tr>
<td>S-027230</td>
<td>How can a SQL server 2005 named instance be correctly connected to an Enterprise PDM database server service?</td>
</tr>
<tr>
<td>S-029263</td>
<td>What is the difference between Simple Recovery and Full Recovery for the Microsoft® SQL Server Recovery Model option?</td>
</tr>
<tr>
<td>S-044156</td>
<td>How can a user shrink a Microsoft® SQL Server tempdb database file? <a href="http://support.microsoft.com/kb/307487">http://support.microsoft.com/kb/307487</a></td>
</tr>
<tr>
<td>S-044163</td>
<td>What is the Microsoft® SQL Server Transaction Log used for?</td>
</tr>
<tr>
<td>S-046380</td>
<td>How is Windows® Firewall configured to allow a Microsoft® SQL Server instance to use a dynamic port?</td>
</tr>
<tr>
<td>S-048542</td>
<td>How can I monitor Archive, Database and SQL Servers to be sure that the system is in a 'healthy' condition?</td>
</tr>
</tbody>
</table>

Microsoft Knowledgebase


A checkpoint occurs in the Simple recovery mode automatically when the number of log records reaches the lesser of these two values:

- The log becomes 70 percent full.
- The number of log records reaches the number that the Database Engine estimates it can process during the time that is specified in the recovery interval option.

For more information, see "Checkpoints and the Active Portion of the Log" http://msdn.microsoft.com/en-us/library/ms189573.aspx on MSDN.

For a detailed overview of the restore and recovery process, see the "Restore and Recovery Overview" http://msdn.microsoft.com/en-us/library/ms191253%28v=SQL.105%29.aspx topic in SQL Server Books Online on MSDN.