Best Practices for Running Siemens Teamcenter on SQL Server

Maximize value and operational efficiency in support of world-class product lifecycle management

Microsoft Corporation
Published: May 2010

Technical Reviewers:
Art Rask – Mentor, Solid Quality Mentors
Richard Waymire – Mentor, Solid Quality Mentors
Christopher Gill – Teamcenter Centers of Excellence, Siemens PLM Software

Abstract

Product lifecycle management (PLM) is an enterprise, business, and information strategy that enables development and delivery of world-class products. Siemens PLM Software's Teamcenter® powers innovation and improves productivity by connecting people with the product and process knowledge they need to effectively function in a globally oriented product lifecycle. Teamcenter's proven lifecycle management solutions are built on an open PLM foundation.

This white paper provides best practices for configuring and running Teamcenter 8.1 on the Microsoft SQL Server database platform. This paper complements the detailed support documentation provided on the Siemens support Web site. Implementing these best practices can help you avoid or minimize common problems and optimize the performance of Siemens Teamcenter on SQL Server.

Microsoft

Copyright Information

Table of Contents

Overview

Product lifecycle management (PLM) is an enterprise, business, and information strategy that enables companies to establish Global Information Networks, essential for developing and delivering world-class products in today's highly competitive international marketplace.

Siemens Teamcenter powers innovation and improves productivity by *connecting people with the product and process knowledge* they need to effectively function in a globally oriented product lifecycle. Teamcenter's proven digital lifecycle management solutions are built on an open PLM foundation.

The performance and availability of Teamcenter—and of the underlying Microsoft SQL Server database platform—is critical to the success of the PLM platform. Responsiveness to users and reliability of the PLM solution are integral to realizing the business benefits of an integrated product development system.

This white paper, intended for database administrators (DBAs), presents the best practices that Siemens and Microsoft recommend for configuring and running Teamcenter on the SQL Server database platform. The paper can help DBAs optimize the performance of Teamcenter, while also preventing or minimizing potential problems.

Note that Teamcenter supports Microsoft® SQL Server® 2005 and SQL Server® 2008 database software. Unless there is a specific difference, this document uses "SQL Server" to refer to all supported versions.

Siemens Teamcenter: The Solution for Product Lifecycle Management

You can leverage PLM to provide total visibility into workflows and decision making at all stages in the product lifecycle. PLM provides unique opportunities to:

- Maximize innovation throughout your product lifecycle, which translates into higher revenues, greater market share, faster time to market, and improved portfolio success rates.
- Transform the decision making processes you use to determine what products you should offer and how these products should be brought to market.
- Increase the value of your product knowledge by managing this information as an intellectual asset on an enterprise basis and leveraging it across multiple programs, projects, and revenue-generating initiatives.
- Minimize lifecycle cost by replacing time-consuming manual processes with accelerated, fully automated solutions.

Teamcenter is *the world's most widely used PLM system*. It is backed by Siemens PLM Software's leadership in delivering Global Innovation Networks that enable companies to make unified, information-driven decisions at every stage in the product lifecycle.

SQL Server: An Enterprise-Ready Database Platform for Teamcenter

Microsoft SQL Server provides an ideal database platform for Teamcenter. SQL Server is a high-performance, integrated database and business intelligence solution for data management and analysis. This easy-to-implement, easy-to-support foundation provides a multifunctional solution for large-scale online transaction processing (OLTP), data warehousing, and e-commerce applications and a solution for data integration, analysis, and reporting.

SQL Server can help companies manage large volumes of mission-critical data and run software applications—such as Siemens Teamcenter—to optimize their business performance. SQL Server can extract and transform data from a variety of sources, including XML data files, flat files, and relational data sources, and then load it into one or more destinations. In addition to rapid data mining, analysis, processing, and reporting capabilities, SQL Server has built-in features that give you a secure, reliable, and productive data management environment that truly protects your data—helping reduce the potential risk of corruption. With its scalable infrastructure, SQL Server has the capability to grow with your business and keep up with your toughest data challenges.

Better Together: Teamcenter on the SQL Server Database Platform

Running Siemens Teamcenter on SQL Server delivers measureable value by channeling product and process data into a manageable, globally integrated resource. Benchmarking tests confirm that SQL Server scales to meet the performance needs of even the largest enterprise customers, while providing lower initial costs and licensing fees.

The Teamcenter Architecture

Best practices for Teamcenter on the SQL Server database platform start with configuration of the system. It is helpful to review and have a thorough understanding of the Teamcenter architecture (Figure 1).

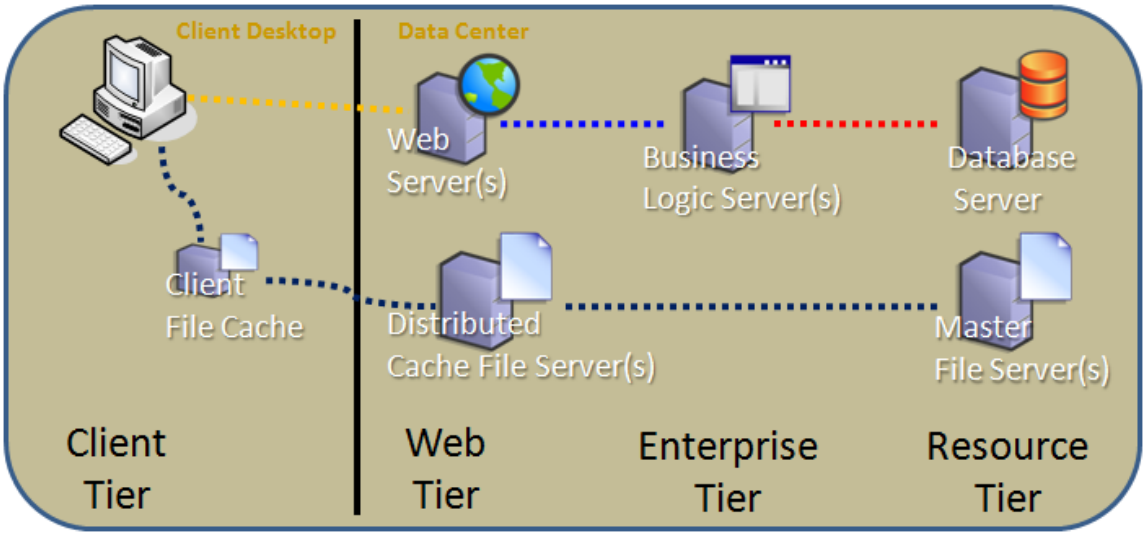

Figure 1 - Teamcenter Runtime Architecture

The Teamcenter suite is a service-oriented architecture (SOA)-based distributed application suite that uses a four-tier architecture.

Client Application Tier

Users interact with Teamcenter through desktop and browser-based clients. Web browser clients access UI and business services from the Web server in the Web Application Tier via HTTP/S. The preferred model for desktop clients is to access the Web application tier through WSDL-based SOA interfaces, also over HTTP/S. Based on HTTP/S, both types of client can effectively access Teamcenter across low or high latency network connections—as well as securely traverse corporate firewalls without needing to open additional application-specific ports. Some deployments with older clients, however, use non-SOA interfaces to interact with the Enterprise Tier directly.

Web Application Tier

Teamcenter deployments use Web application servers to expose the SOA service endpoints for all client types. Both REST-style (standard HTTP POST) and SOAP-style requests are supported by Teamcenter components executing on industry-standard Web servers, such as Microsoft IIS, as well as servers based on Java and J2EE technology. SOA Framework components running on these application servers are responsible for normalizing the request into a single common format, which is passed directly onto Teamcenter's Business Logic Server in the Enterprise Tier.

Enterprise Tier

Teamcenter's Business Logic Server, along with Teamcenter's SOA server-side components, resides in the Enterprise Tier. The Business Logic Server is primarily implemented in C++.

Resource Tier

The Resource Tier houses the Teamcenter SQL Server database as well as bulk file repositories, usually on distinct servers. The Microsoft SQL Server server(s) and database(s) hosted in this tier are the focus of this paper.

Two-Tier Teamcenter Deployments

Some Teamcenter deployments use a two-tier architecture. This is usually seen when Teamcenter has been migrated up from older versions. In a two-tier deployment, the Teamcenter client and an instance of the Business Logic server reside together on the end user's computer. The shared database and file repositories remain in a server-based Resource Tier.

All the database best practices discussed in this paper apply equally to two-tier and four-tier Teamcenter deployments.

Best Practices for Running Teamcenter on SQL Server

Among the strengths of SQL Server as a database platform is the relative simplicity of managing and tuning the database engine. There are, nevertheless, a range of best practices for managing SQL Server production deployment and for getting the best performance and reliability from the platform. Many of these are standard, irrespective of the Teamcenter application, while some specific guidance is driven by the nature of the Teamcenter database and application.

The following sections discuss a range of best practices and recommended choices for Teamcenter deployments on SQL Server.

Planning and Installing the SQL Server Instance

Use a Dedicated Database Server and SQL Instance

Because Teamcenter is a database-intensive application, Siemens recommends hosting the SQL Server database on a server dedicated to that purpose. This will prevent the Teamcenter database workload from having to compete for CPU, memory, network, and I/O with other processes.

Select the Appropriate SQL Server Edition

Teamcenter 8.1 can run on either the Standard or Enterprise Edition of SQL Server 2005 or 2008. Regardless of version and edition, you should always run on the latest SQL Server service pack, which you can find in the Downloads section of the SQL Server Developer Center.

For most functional aspects of Teamcenter, SQL Server Standard and Enterprise editions are equivalent. However, the Enterprise Edition of SQL Server includes a variety of higher-end features, including the ability to use more than four CPUs in a database server. The table below summarizes some of the key distinctions between editions.

Choose Standard Edition if:	Choose Enterprise Edition if:
Your database server has <= 4 CPU sockets	Your server has > 4 CPU sockets
Your high availability requirements can be met with a 2-node failover cluster	Your high availability requirements are best met with a failover cluster with 3+ nodes
	You will use SQL Server Data Compression
	You will use SQL Server Backup Compression
	Any of these other features are important for your

	deployment of Teamcenter: • Native encryption of database files on disk with Transparent Data Encryption • Rich, rigorous, and application-independent auditing of data access with SQL Server Audit • Managing server capacity with Resource Governor • Online index rebuilds • Hot add of RAM and CPU
	Your server uses an IA64 chip architecture

SQL Server Enterprise Edition also has a wide array of advanced data warehousing features for analytics, data mining, and ETL.

For more information, go online to read about SQL Server 2008 Editions or the SQL Server 2005 Feature Comparison.

Prior to installation, it is also very important to verify that the copy of SQL Server setup matches the chip architecture of your server and operating system. SQL Server 2008 has distinct setup programs for 32-bit (x86), 64-bit (x64), and Itanium 64-bit (IA64). Verify the chip architecture of the host operating system before installation and ensure that you are installing from the correct setup program. In particular, it is possible to install 32-bit SQL Server on a 64-bit operating system; making this mistake could severely impact the performance and scalability of your Teamcenter deployment.

Fortunately, SQL Server 2008 makes this verification easy. The main SQL Server 2008 setup wizard has an Options page to specify which architecture will be installed (Figure 2). If the host system is 64-bit (which is the recommended platform for the Teamcenter database), either the x64 or the ia64 option will be enabled. Ensure that the 32-bit variant (x86) is *not* selected.

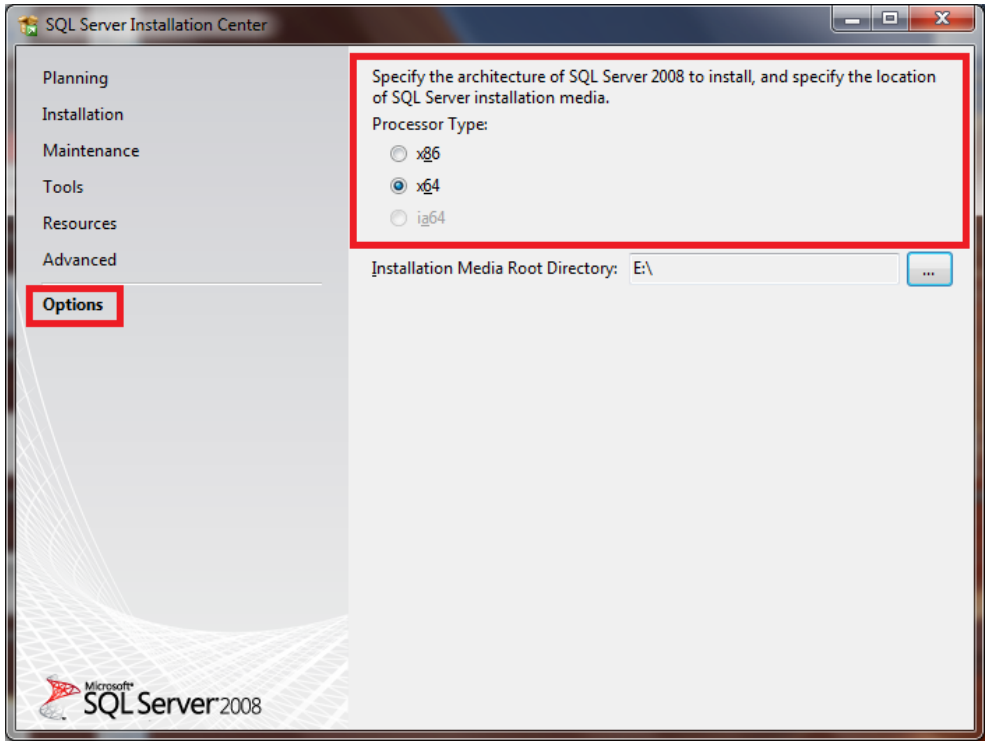

Figure 2 - Verifying the Architecture to be Installed (SQL Server 2008)

Options for High Availability

SQL Server has several approaches for maintaining database availability in the event of a system failure. The most sophisticated methods, and the only ones that provide automatic switchover to a standby server in the case of failure, are failover clustering and database mirroring. Other options, which require more manual intervention by administrators, include log shipping and SQL Server replication.

Siemens recommends a high availability strategy based on failover clustering as the preferred approach. Because Teamcenter 8.1 does not support SQL Server database mirroring in automatic failover mode, failover clustering is the only option that provides seamless redirection of traffic to a standby server in the event of a database server failure.

The use of database mirroring (without automatic failover) and log shipping are viable options as well. However, administrators must be prepared for the need to manually intervene to redirect Teamcenter to a standby server if the primary server experiences a failure.

Which Features to Install

SQL Server includes a rich array of components for data management and business intelligence. A minimal installation for Teamcenter requires only the SQL Server Database Engine and Management Tools. These selections are made in the Feature Selection step of SQL Server 2008 setup. The choices shown in Figure 3 are the minimum features for a Teamcenter database installation.

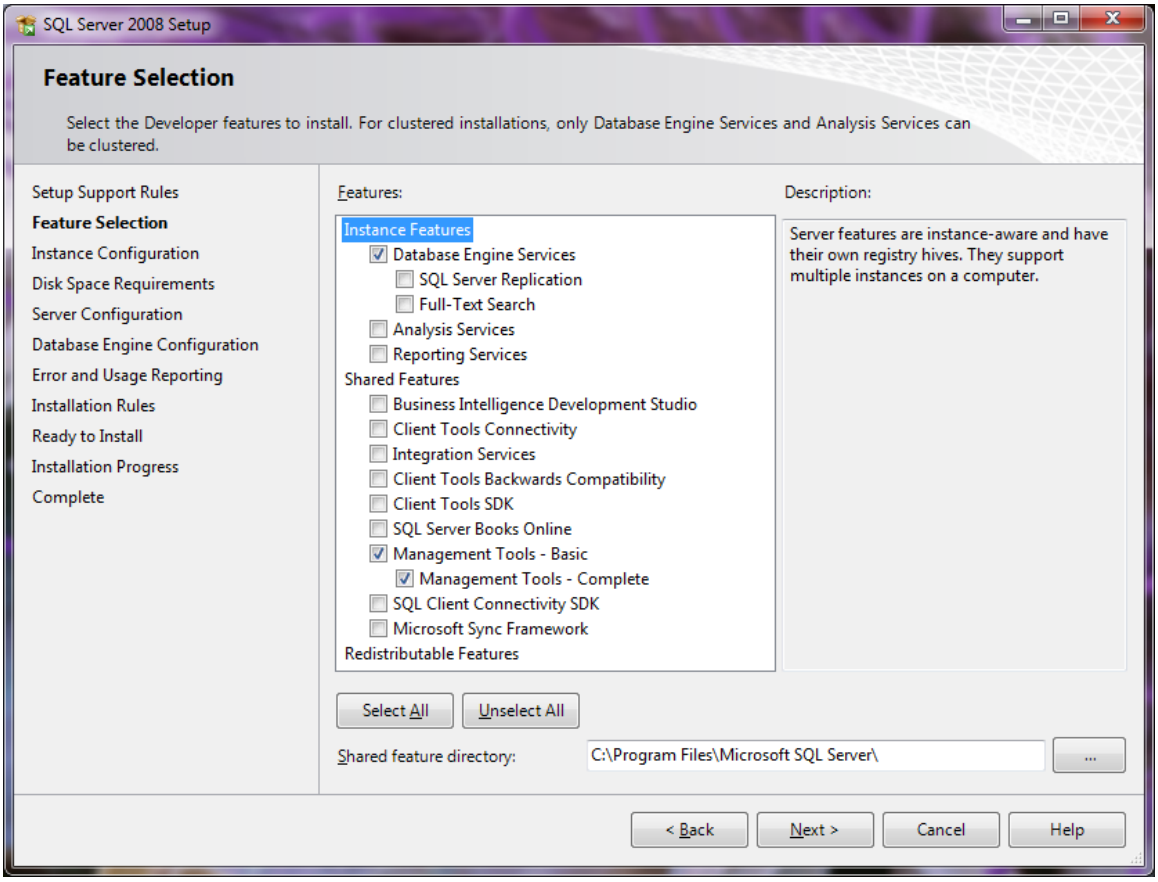

Figure 3 - SQL Server Features Required for a Teamcenter Installation (SQL Server 2008)

Many DBAs will also find it useful to install the product documentation (SQL Server Books Online) and Integration Services.

Setting the Service Account for SQL Server

The SQL Server Database Engine runs as a Windows service. Like all Windows services, it must execute in the context of a specific Windows user account. The service account for the SQL Server Database Engine is set during installation and can be changed afterwards using SQL Server Configuration Manager. Changing the service account for any SQL Server-related service directly through the Windows Services management snap-in is not supported.

Teamcenter will function properly if the Database Engine service account is set to LOCAL SYSTEM. However, this is not considered a good security practice because it gives the database engine much higher privileges on the operating system than it requires. According to the principle of least privilege, a service should be granted only the minimal permissions it needs to function correctly.

Therefore, Siemens recommends using a local machine account or a domain account for the Database Engine service. The same applies for the SQL Server Agent service. As shown in Figure

4, you configure the service account during SQL Server 2008 setup at the Server Configuration step.

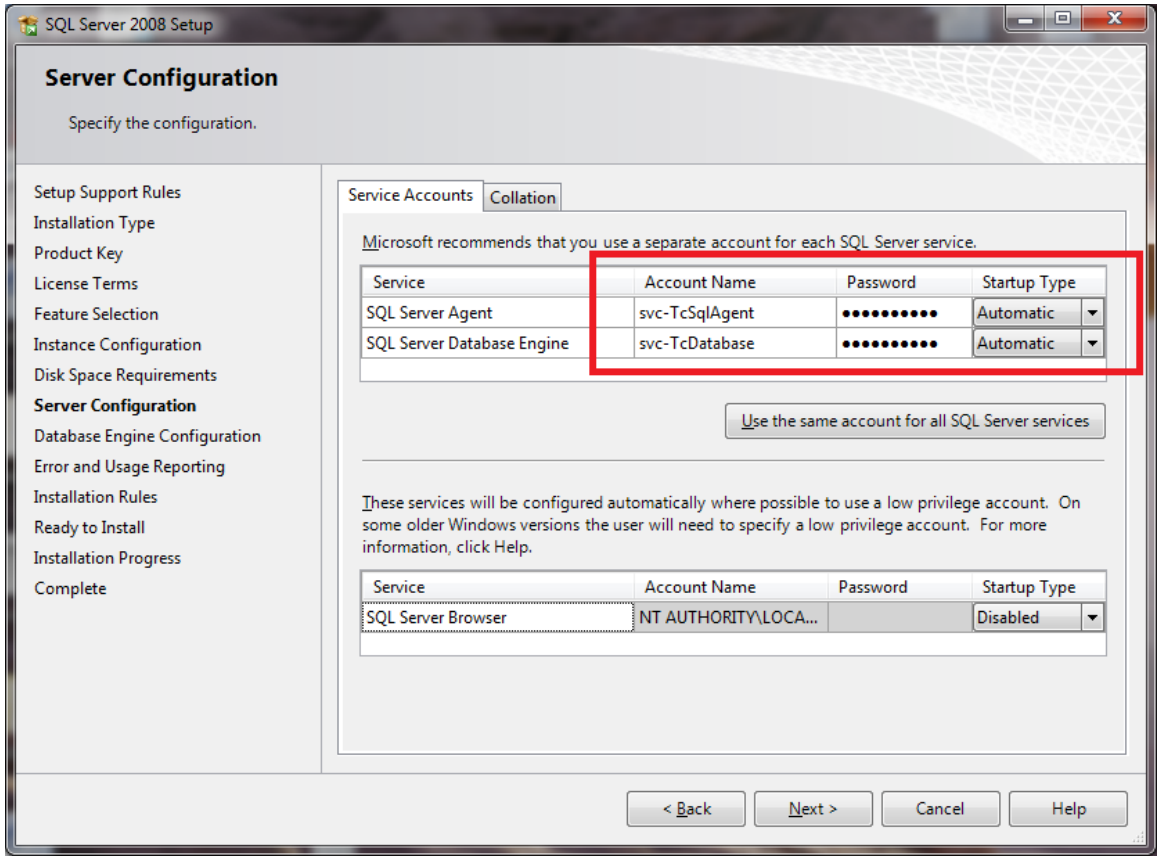

Figure 4 - SQL Server Service Accounts (SQL Server 2008)

In Figure 4, svc-TcSqlAgent and svc-TcDatabase are accounts on the Active Directory domain of which the server is a member.

Ensuring the Database Engine Can Use Fast File Initialization

SQL Server is able to initialize its data files instantly, without arduously zero-ing out the files on disk. This saves time when creating new database files and when growing existing files. For large databases, the impact can be significant. To use this capability, however, the Database Engine service requires a specific permission in the Windows operating system. Siemens recommends granting this permission to the service account to get the performance benefits of fast file initialization.

The permission required to use fast file initialization is called **Perform Volume Maintenance Tasks**. To grant this permission to the Database Engine service account, follow these steps.

1. Open the Local Security Policy management snap-in by executing secpol.msc.

2. In the left task pane, expand the tree to expose Security Settings > Local Policies > User Rights Assignment and select the User Rights Assignment node. Scroll though the list in the right pane to display the item named *Perform volume maintenance tasks* (as Figure 5 shows).

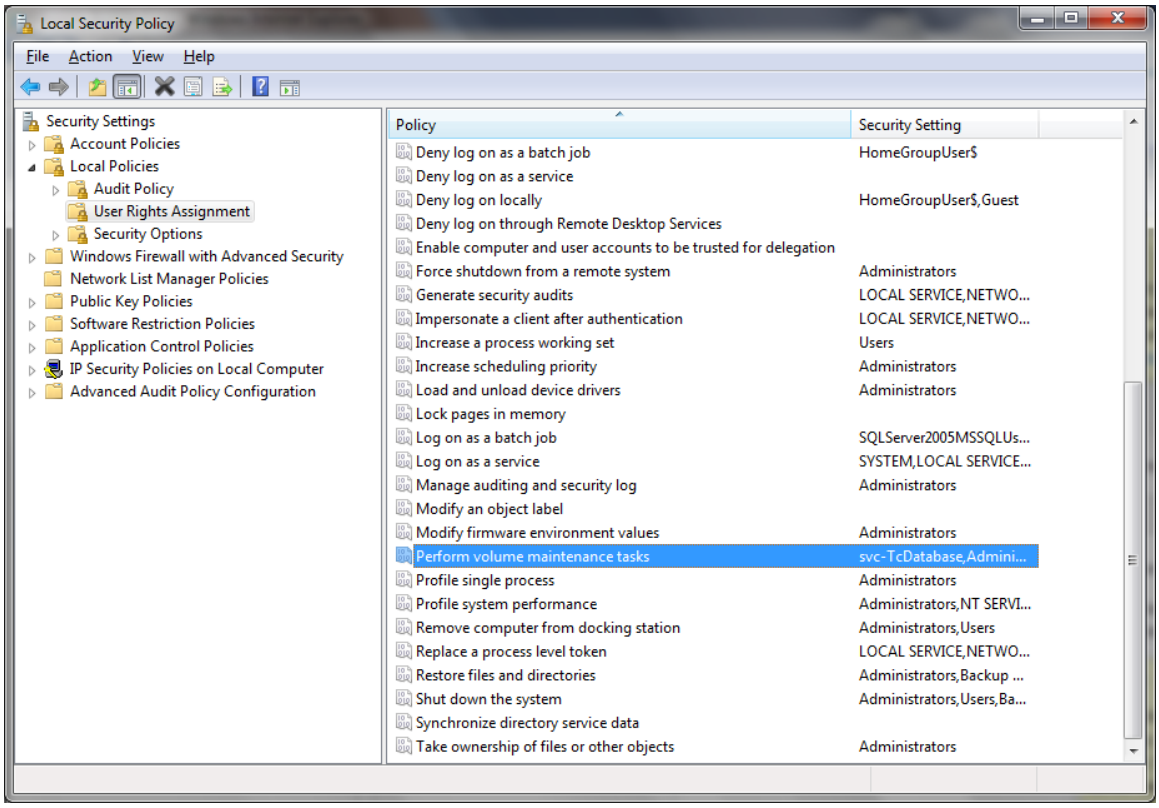

Figure 5 - Managing Windows User Rights to Allow Fast File Initialization

3. Double-click *Perform volume maintenance tasks* to display the Properties dialog box. Use this dialog box to add the service account used for the SQL Server Database Engine. Figure 6 shows svc-TcDatabase (our example service account) added to the list of accounts having this permission.

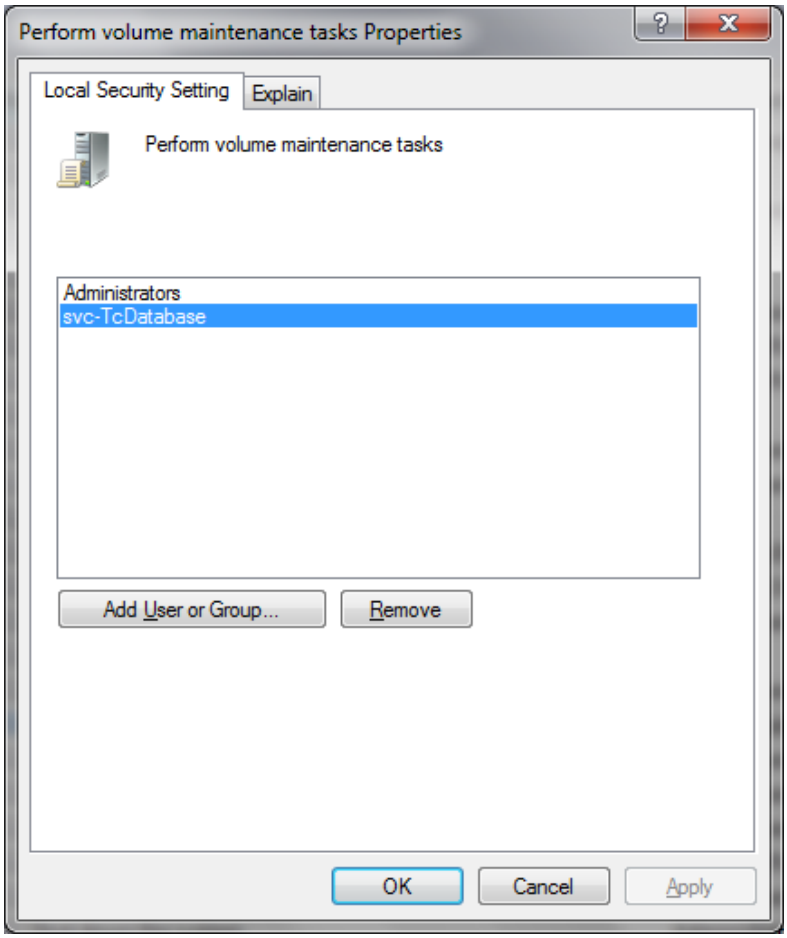

Figure 6 - Granting SE_MANAGE_VOLUME_NAME to SQL Server Database Engine Service Account

Granting this permission can be done any time before or after SQL Server is installed. If SQL Server is already installed, however, you will need to restart the Database Engine service to see the effects of this change.

Setting the Server Collation at Install Time

The collation setting in SQL Server affects how the database software handles sorting and comparisons of text data. Different collations allow the Database Engine to account for various expectations of sorting and comparing text across languages and cultures. It also allows a particular installation to be configured to use case sensitive vs. insensitive comparison (e.g., does “smith” = “Smith”) and accent sensitive vs. insensitive comparison (e.g., does “elan” = “elán”). SQL Server allows collation to be configured at the instance and database levels, as well as at more granular levels within databases.

Siemens Teamcenter is designed to use the Latin1_General collation with Binary sort order with SQL Server. Siemens recommends setting this collation at the SQL Server instance level to ensure that it affects all databases used by Teamcenter. It is strongly recommended that this be

done at the time SQL Server is installed. Changing the server collation after installation is a difficult and involved procedure.

Collation is set during SQL Server 2008 installation at the Server Configuration step (Figure 7). Select the Collation tab and click the Customize… button for Database Engine collation.

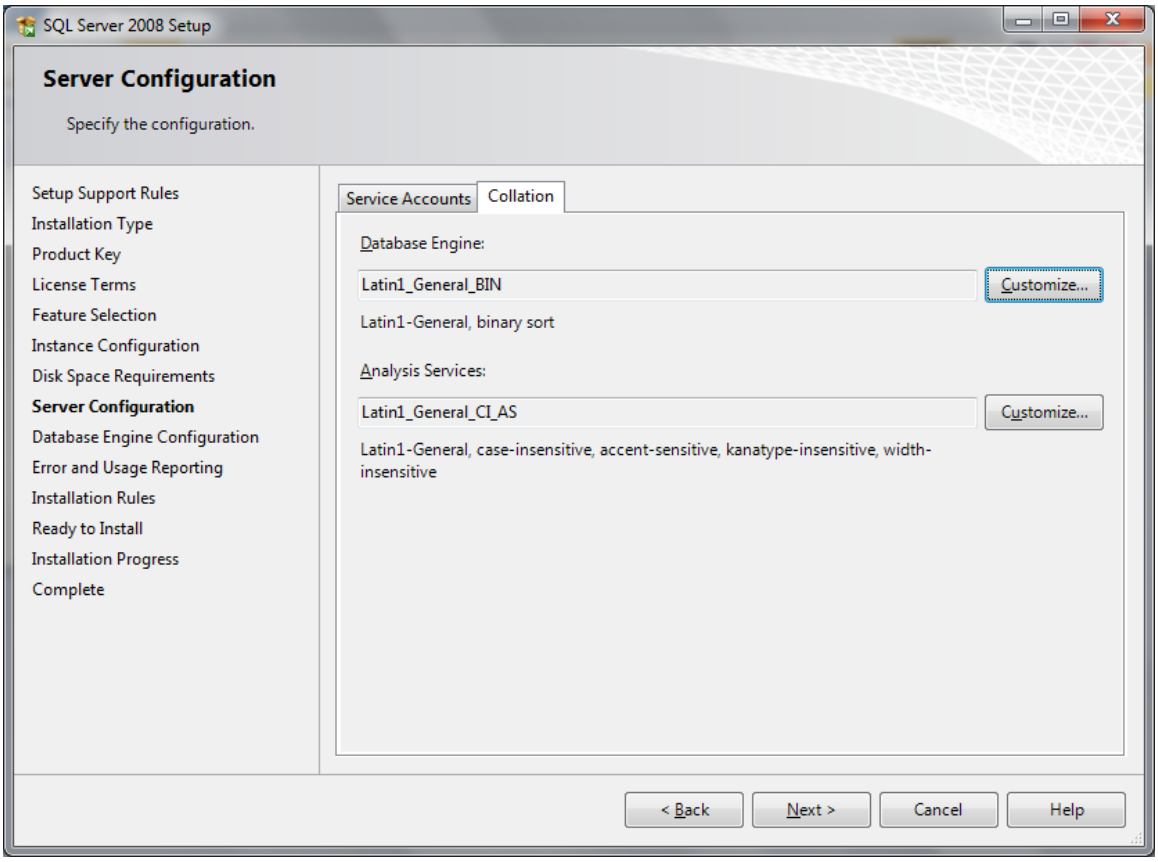

Figure 7 - Server Collation for the Database Engine (SQL Server 2008)

After clicking the Customize… button, the dialog box shown in Figure 8 will appear. Select the *Windows collation designator and sort order* option button. In the Collation designator drop-down list, select Latin1_General, and then check the check box labeled Binary. Click OK to return to the Setup wizard.

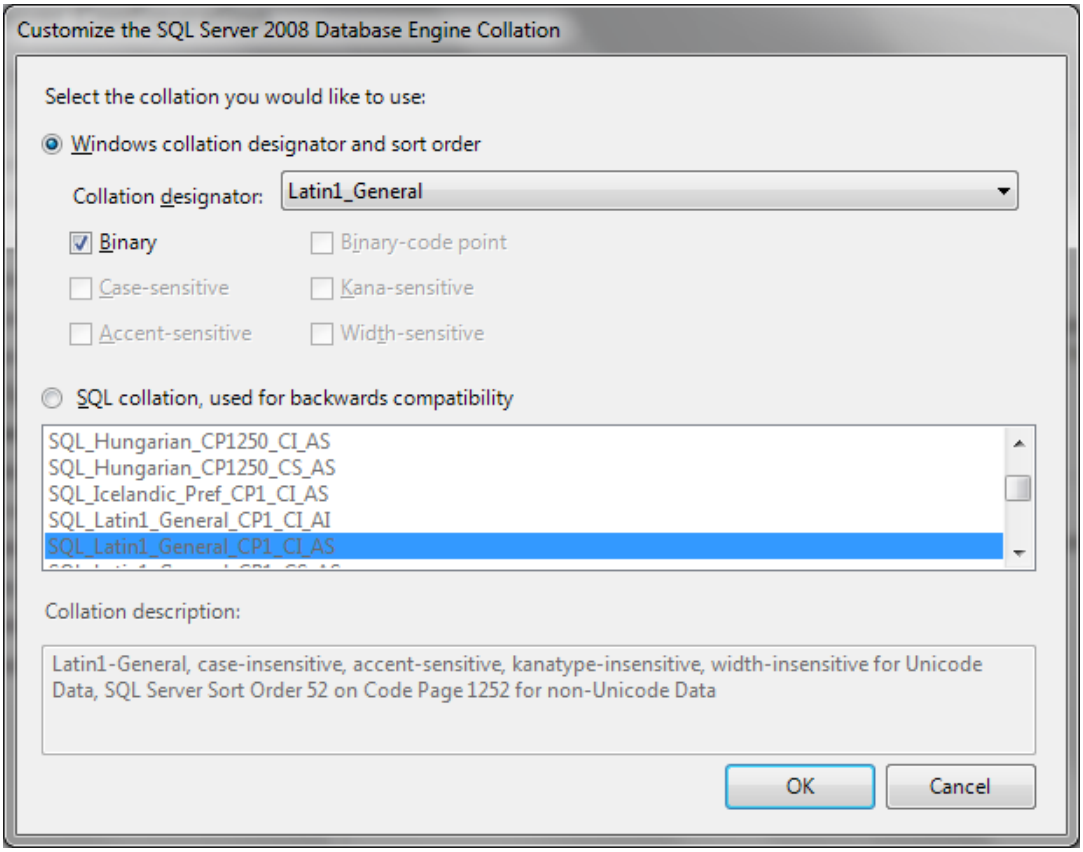

Figure 8 - Choosing Server Collation for a Dedicated Teamcenter Instance

Choosing the Security Mode

Teamcenter requires that the SQL Server instance be in Mixed authentication mode, because the Business Logic Server connects using a SQL Server login. You configure the server authentication mode during SQL Server 2008 setup in the Database Engine Configuration step, as shown in Figure 9.

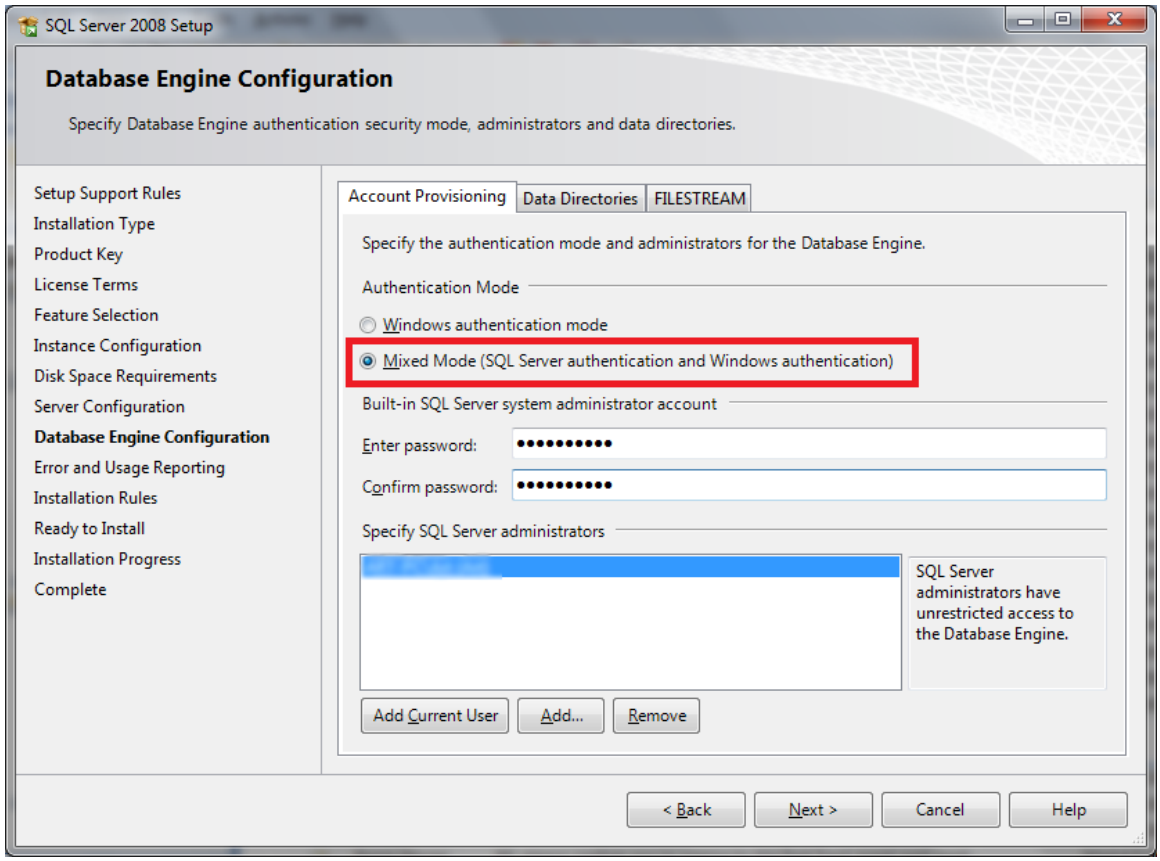

Figure 9 - Setting the Server Authentication Mode (SQL Server 2008)

When choosing Mixed Mode, you are required to provide a strong password for the built-in system administrator (sa) account.

You can also modify the authentication mode for a SQL Server instance that is already installed. This is done inside SQL Server Management Studio from the Object Explorer pane. With the relevant SQL Server instance displayed in the Object Explorer, right-click the server icon and select Properties from the context menu. In the Properties window, select the Security page. The authentication mode can be configured at the top of the Security page.

Deciding Which Services to Auto-Start

At minimum, the SQL Server Database Engine must be configured with a Startup Type of Automatic. The SQL Server Agent service is a job scheduling engine that is useful for running regular maintenance or health-checking tasks. Therefore, Siemens recommends also configuring SQL Server Agent with a Startup Type of Automatic for production database servers.

The service startup types can be configured in SQL Server 2008 setup at the Server Configuration step, which is shown in Figure 4 under the discussion of service accounts. After installation, service startup types can be changed in SQL Server Configuration Manager.

Proper Placement of the tempdb Database

Tempdb is a built-in database used by SQL Server as a workspace for temporary objects. These objects could be temporary tables resulting from application code, internal sort tables used by SQL Server, or a variety of other things. Depending on the load and usage patterns in a given environment, the configuration of tempdb can become a factor in overall performance. For this reason, there are some important best practices for managing this database.

These tempdb best practices can be boiled down to a few key points:

1. Move tempdb from its default location to one or more high-performance, fault-tolerant disk volumes.
2. If possible and practical given your disk resources, give tempdb files dedicated use of their disk(s).
3. To minimize auto growth, configure the startup size of tempdb based on your server's needs. Frequent auto growth can hurt performance.
4. Create multiple data files for tempdb to reduce contention in the engine's internal allocation routines.

We'll review the guidelines around each of these and then provide an example that brings it all together.

Moving tempdb

By default, tempdb files are located in the same directory as SQL Server's other system databases. For busy servers, this is usually sub-optimal. Tempdb should be moved to a high-performance, fault-tolerant disk volume—a RAID 1 or RAID 10 volume on fast disks, for example. Tempdb should not be placed on a RAID 5 volume because of the inferior write performance of RAID 5.

Tempdb is moved using the ALTER DATABASE statement. For example, to move tempdb to the G: drive and the log file to the H: drive, execute the following commands in a SQL Server Management Studio query window:

```
USE master;
GO

ALTER DATABASE tempdb
MODIFY FILE (NAME = tempdev, FILENAME = 'G:\SQLData\tempdb.mdf');
GO

ALTER DATABASE tempdb
MODIFY FILE (NAME = templog, FILENAME = 'H:\SQLData\templog.ldf');
GO
```

SQL Server must be restarted for this change to take effect. Note that SQL Server creates fresh copies of tempdb data and log files every time the service starts, so there is no need to move any files yourself. Just restart the service, and you will see the files in the new location(s) you have specified.

Using Multiple, Dedicated Disks

If your database server has sufficient disk resources and your server is heavily utilized, typical best practices for database file placement should be followed for tempdb. The following table summarizes these best practices.

If You Have…	Then…
One disk volume available for tempdb	Place all tempdb data and log files here, preferably with no other heavily utilized files on this volume.
Two disk volumes available for tempdb	Place the tempdb log on one volume. Place all tempdb data files on the other volume.
More than two disk volumes available for tempdb	Place the tempdb log alone on one volume. Distribute tempdb data files evenly across the remaining volumes.

Sizing tempdb

Tempdb is used very dynamically by SQL Server. In other words, the server is constantly adding and removing structures in tempdb. At any given time, some operation may cause a (relative) spike in the required size of the database. If tempdb starts out too small given your workload, the result may be a large number of frequent auto growth operations in tempdb. This has two negative side effects. First, performance can be hurt when the auto growth actually occurs. Second, many small auto growths will result in heavily fragmented data files, which hurts tempdb I/O performance from that time forward.

To avoid these problems, the size of tempdb should be set proactively as appropriate for the environment. There is no fixed rule for the size of tempdb—it depends on the environment. Two potential approaches are recommended here. The first is to use an arbitrary rule of thumb, and the second is to observe your environment in action and derive appropriate sizing.

Proposed sizing guidelines for tempdb in Teamcenter deployments are shown in the table below. These are reasonable rules of thumb based on experience in previous deployments. It should be understood, however, that many variables affect the size requirements for tempdb. Therefore, these are at best good starting points. DBAs should monitor usage over time to see if configuration changes become needed.

Environment Size	Description	Database Size (MB)	Log Size (MB)
Small	Teamcenter DB < 100GB 1-100 Users	2,048	1,024
Medium	Teamcenter DB 100–500GB 100-1000 Users	10,240	5,120

Large	Teamcenter DB > 500GB 1000+ Users	20,480	10,240

The second approach—observation of the environment in action—starts with leaving tempdb sizes at their default level with auto growth enabled. After a reasonable period of production activity (one or more weeks, for example), the DBA should check the space used by tempdb. The size of tempdb can be viewed in SQL Server Management Studio by using the Standard Reports accessible through the Object Browser pane (see Figure 10).

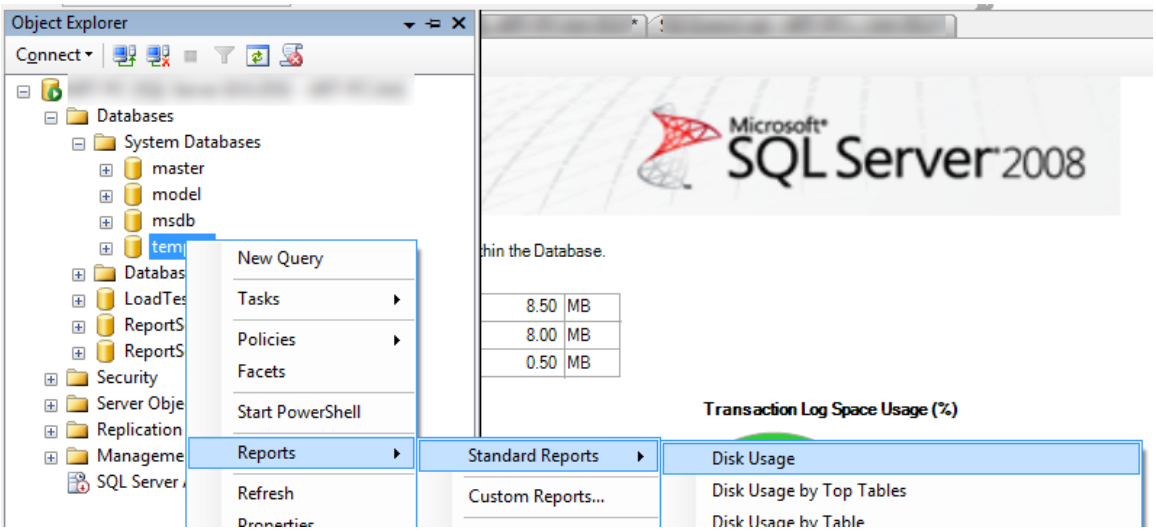

Figure 10 - Disk Usage Report Shows the Current Size of tempdb

The size of tempdb can also be obtained by using a SQL query like the following, taken from SQL Server 2008 Books Online:

```
SELECT name AS FileName,
    size*1.0/128 AS FileSizeinMB,
    CASE max_size
        WHEN 0 THEN 'Autogrowth is off.'
        WHEN -1 THEN 'Autogrowth is on.'
        ELSE 'Log file will grow to a maximum size of 2 TB.'
    END,
    growth AS 'GrowthValue',
    'GrowthIncrement' =
        CASE
            WHEN growth = 0 THEN 'Size is fixed - will not grow.'
            WHEN growth > 0 AND is_percent_growth = 0
                THEN 'Growth value is in 8-KB pages.'
            ELSE 'Growth value is a percentage.'
        END
FROM tempdb.sys.database_files;
```

The DBA can use the size to which tempdb data and log files have grown as a good indicator of target size. Adding an additional 20% buffer can help ensure that auto growth is minimized.

With this target size in mind, the starting size of tempdb is set using the ALTER DATABASE command, as in this example, which sets the database at 12GB and the log file at 10GB:

```
ALTER DATABASE tempdb
MODIFY FILE (NAME = 'tempdev', SIZE = 12288);

ALTER DATABASE tempdb
MODIFY FILE (NAME = 'templog', SIZE = 10240);
```

Finally, whatever size you choose for tempdb, it is a good idea to leave auto growth enabled on the database. Minimizing auto growth is a goal, but you want to leave auto growth enabled so that tempdb does not run out of space in an emergency.

More discussion of sizing tempdb can be found on MSDN at Capacity Planning for tempdb and Optimizing tempdb Performance. Additional, in-depth information can be found in Troubleshooting Insufficient Disk Space in tempdb and Working with tempdb in SQL Server 2005.

Adding Data Files to tempdb

By default, the tempdb database has one data file (.mdf) and one log file (.ldf). A best practice for large, multi-core servers is to increase the number of files in tempdb. This can reduce contention in SQL Server's internal allocation routines related to tempdb. There are essentially four points to consider when doing this:

1. Aim for one tempdb data file per processor core available to SQL Server[1]. For systems with more than 16 CPUs, it is safe to relax this ratio closer to 0.5 files per core.
2. Ensure that the data files are equally sized. This allows SQL Server's proportional fill algorithms to run at their most efficient.
3. The data files need not all be on separate disk volumes. The purpose of this recommendation is not to split I/O throughput across devices, but to exploit some aspects of the Database Engine's internal I/O and allocation algorithms. This is a bit counterintuitive and differs from the guidance for user databases, but it is intentional.
4. There is no need for multiple log files.

An Example

To pull these recommendations together, let's consider an example. Say that you are setting up a new SQL Server Enterprise Edition server for Teamcenter. The server's disk configuration

[1] Usually this is the same as the physical cores in the system, unless an affinity mask or WSRM is used to limit the processors used by SQL Server.

allows you to dedicate three RAID 1 volumes to tempdb (drives G, H, and K). The SQL Server software was installed on the C drive. The server has eight (8) processor cores, and all are in use by SQL Server. You have decided that tempdb should be sized with 10GB for data and have a 5GB log.

You would run the following T-SQL in SQL Server Management Studio:

```
USE master;
GO

--Move and size the log
ALTER DATABASE tempdb
MODIFY FILE (NAME = templog, FILENAME = 'K:\SQLData\templog.ldf', SIZE = 5120);
GO

--Move and size the default data file
ALTER DATABASE tempdb
MODIFY FILE (NAME = tempdev, FILENAME = 'G:\SQLData\tempdb.mdf', SIZE = 1280);
GO

--Create additional data files
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev2, FILENAME = 'G:\SQLData\tempdev2.ndf', SIZE = 1280);
GO
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev3, FILENAME = 'G:\SQLData\tempdev3.ndf', SIZE = 1280);
GO
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev4, FILENAME = 'G:\SQLData\tempdev4.ndf', SIZE = 1280);
GO
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev5, FILENAME = 'H:\SQLData\tempdev5.ndf', SIZE = 1280);
GO
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev6, FILENAME = 'H:\SQLData\tempdev6.ndf', SIZE = 1280);
GO
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev7, FILENAME = 'H:\SQLData\tempdev7.ndf', SIZE = 1280);
GO
ALTER DATABASE tempdb
ADD FILE (NAME = tempdev8, FILENAME = 'H:\SQLData\tempdev8.ndf', SIZE = 1280);
GO
```

After running this T-SQL, you must restart SQL Server for the changes to take effect.

Configuring Memory for SQL Server

RAM on 64-bit Systems

SQL Server can dynamically manage memory without intervention by administrators. In many cases, this is fine. For systems that will host large databases with heavy user traffic, however, it may be necessary to set some limits on memory usage to prevent performance degradation.

A key indicator that this may be necessary is if the size of the Teamcenter database (actual data space used) is significantly greater than the amount of RAM on the database server. Over time,

memory pressure may develop; SQL Server will want to continue growing its in-memory data cache to avoid reading from disk, but the operating system will push back, trying to allocate memory for itself and other processes. The result can be excessive memory paging and a negative effect on performance.

If your Teamcenter database (data + indexes) is or is expected to be much larger than the total RAM on the database server, Siemens recommends configuring the min and max server memory options in SQL Server. Use the recommendations in the following table to select values appropriate for your environment.

Physical RAM	SQL Min Server Memory (MB)	SQL Max Server Memory (MB)
8GB	4,096	5,120 (5GB)
16GB	8,192	12,288 (12GB)
24GB	12,288	18,432 (18GB)
32GB	16,384	25,600 (25GB)
48GB	32,768	39,936 (39GB)
64GB	49,152	56,320 (55GB)
96GB	73,728	88,064 (86GB)
128GB	104,448	120,832 (118GB)

The min/max server memory options can be set in the Server Properties dialog box in SQL Server Management Studio or by using T-SQL, as in the following example:

```
exec sp_configure 'show advanced options', 1
RECONFIGURE
GO
exec sp_configure 'min server memory', 16384
RECONFIGURE
GO
exec sp_configure 'max server memory', 25600
RECONFIGURE
GO
```

In-depth discussion of sizing SQL Server memory can be found at Askperf: SQL and the Working Set and in this entry in Slava Oks' Blog.

Another option to consider is granting the **Lock pages in memory** permission to the SQL Server service account. This permission allows SQL Server to lock its buffer memory and refuse to page it out in response to requests from the operating system. This can give the Database Engine additional ability to preserve its caches and maintain performance, especially if other processes are competing for resources on the server. However, Siemens recommends hosting the Teamcenter database on a server dedicated to this purpose. If this recommendation is followed, the importance of the **Lock pages in memory** permission is lessened. In addition, improvements

in memory management in both SQL Server 2008 and Windows 2008 make it less likely that this setting will make a big difference. For additional information, see How To: Enable the Lock Pages in Memory Option.

RAM on 32-bit Systems

32-bit database servers are increasingly rare. Siemens strongly recommends running your Teamcenter database on a 64-bit server running a 64-bit Windows operating system, with a 64-bit edition of SQL Server. The key advantage of a 64-bit system for database workloads is the ability to efficiently use much larger amounts of RAM.

If you are using a 32-bit system to host the Teamcenter database, Siemens recommends configuring SQL Server to use as much RAM as possible. By default, a 32-bit process (i.e., an x86 install of SQL Server) can use at most 2GB of system RAM. If the database server has 4GB or more of RAM, SQL Server should be configured to use AWE memory. In brief form, the process for enabling AWE memory for SQL Server is as follows:

1. Add the /PAE switch to the Windows boot.ini file. An example of this can be found at http://support.microsoft.com/kb/283037. A system reboot is required for this change to take effect.
2. Grant the **Lock pages in memory** permission to the SQL Server service account. This is required for SQL Server to use AWE memory. Refer to How To: Enable the Lock Pages in Memory Option if necessary.
3. Execute the following commands in a query window in SQL Server Management Studio:

```
exec sp_configure 'show advanced options', 1
RECONFIGURE
GO
exec sp_configure 'awe enabled', 1
RECONFIGURE
GO
exec sp_configure 'min server memory', 8192
RECONFIGURE
GO
exec sp_configure 'max server memory', 12288
RECONFIGURE
GO
```

   Substitute appropriate values (in MB) for the min and max server memory settings. This example uses values that might be appropriate for a server with 16GB of RAM.

4. Restart SQL Server.

More information and background on large memory support in 32-bit systems can be found at the following links:

Memory Architecture

Using AWE
Enabling Memory Support for Over 4 GB of Physical Memory
Enabling AWE Memory for SQL Server
Server Memory Options
PAE and /3GB and AWE

Maximum Degree of Parallelism

Siemens recommends that the Max Degree of Parallelism option in SQL Server be set to 1. This is preferable given the type of query workload that the Teamcenter application places on its database.

As shown in Figure 11, this value is set in the Server Properties dialog box in SQL Server Management Studio.

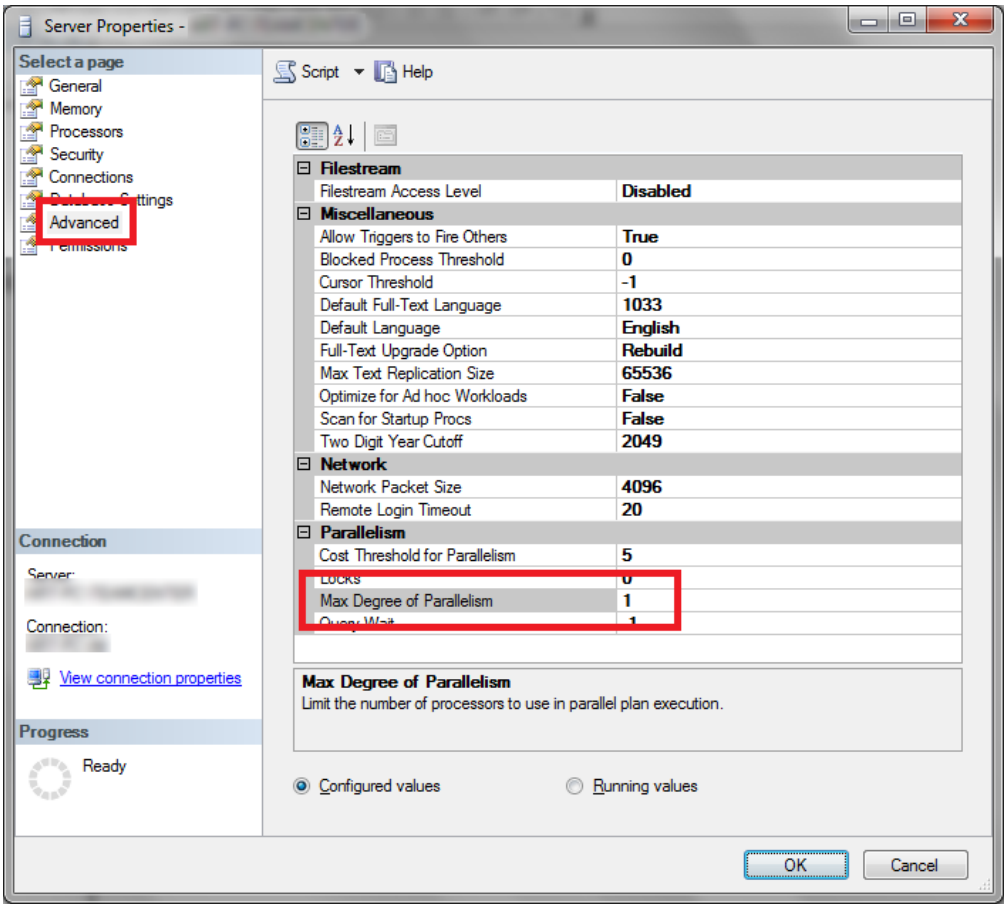

Figure 11 - Set Max Degree of Parallelism to 1 in SSMS

Max Degree of Parallelism can also be set using T-SQL, as in this example:

```
exec sp_configure 'show advanced options', 1;
RECONFIGURE WITH OVERRIDE;
```

```
GO
exec sp_configure 'max degree of parallelism', 1;
RECONFIGURE WITH OVERRIDE;
GO
```

A restart is *not* required for this setting to take effect.

Installing Teamcenter Databases

File Placement on the Disk Subsystem

Before installing the Teamcenter database, Siemens recommends planning the placement of the files that make up the database. In the default configuration, the Teamcenter database consists of two files: a data file and a log file. It is general SQL Server best practice to place these files on separate physical disk volumes to get the best performance from the overall system. These disk volumes should also not be used for any other heavily accessed files or for the Windows operating system.

It is critical that the disk volumes on which you place your database files be fault-tolerant RAID sets. The ultimate safety of your Teamcenter data depends on the data and log files residing on redundant physical disk storage. Siemens recommends RAID 10 (also called RAID 1 + 0) for the data file. RAID 5 is acceptable. For the log file, RAID 10 or RAID 1 is recommended. The log file should not be placed on a RAID 5 volume because RAID 5 has lower write performance than the other RAID levels.

The following table shows a summary of recommendations for where to place the Teamcenter database files.

File Type	Placement	Recommended RAID Level	Disk Speed
Data (.mdf, .ndf)	Physically separate disks or LUN from the log Disk(s) should not be used for other heavy file I/O	RAID 10 or RAID 5	10,000 RPM or faster
Log (.ldf)	Physically separate disks or LUN from data files Disk(s) should not be used for other heavy file I/O	RAID 1 or RAID 10 RAID 5 is discouraged for log	10,000 RPM or faster

Note that when placing data and log files on separate disk volumes, they should be truly physically separate. The drive letters on which the files are placed should actually map to distinct physical disk drive sets. Placement on different partitions/LUNs on a single disk drive set is not helpful for performance or fault tolerance.

Sizing Data Files Up Front

Most Teamcenter installations will see significant database growth over time. For best long-term performance, Siemens recommends creating data and log files with a size that matches the long-term size expectations of the Teamcenter database.

If no explicit size is provided, SQL Server creates data and log files based on the size of a special template database called *model*. Usually, this is just a few MB in size. As data flows in to the Teamcenter database, SQL Server automatically extends the data and log files as needed. For large, heavily used databases, this eventually produces data files that are highly fragmented on disk. This fragmentation has a negative effect on performance. There is also a small run-time performance penalty every time a file is extended.

These problems can easily be avoided by sizing files appropriately when the database is created. The following guidelines are recommended for setting the initial size of database files.

1. Estimate long-term size of the Teamcenter database (e.g., 1-3 years).
2. Add 50% for small-sized databases (< 100GB). Add 35% for medium-sized databases (> 100GB). Add 25% for larger databases (> 500GB). This is the recommended data file size.
3. Size the transaction log file at approximately 20% of the data file size.

The table below illustrates these calculations for a new Teamcenter database.

	Example 1	Example 2
Estimated Long-Term Size of Data + Indexes (1-3 years)	50GB	400GB
Add buffer	25GB	140GB
Initial Data File Size	75GB	540GB
Initial Log File Size	15GB	100GB

Automatic file growth should be enabled for the database, even though the initial size ought to be sufficient for some time. This is simply to avoid a production problem if the data files run out of space. As a rule, though, file size for the Teamcenter database should be managed proactively. Siemens recommends configuring data files for 5GB growth increments and log files for 1GB growth increments. In particular, avoid using large percent-based growth settings on large log files, because the initialization of new log segments can temporarily block activity on the database.

Spread the PRIMARY Filegroup across Files on Multiple Disks (optional)

For heavily used Teamcenter databases, overall performance can be affected by the throughput limitations of working with a single data file on a single disk volume. With SQL Server, you can easily spread I/O load for your Teamcenter database across multiple disk volumes or RAID sets by mapping SQL Server filegroups to multiple files. The extra administrative work of setting this

up should only be done if you have confirmed that disk I/O is a bottleneck for your system. The monitoring techniques discussed later in this paper can help in making that determination.

If you choose to add additional files to spread I/O across disks, there are a few things to consider. First, Teamcenter 8.1 creates all data objects (except optional audit logs) on the PRIMARY filegroup, so this is the one you will want to map to the new files. Second, SQL Server balances I/O most effectively when the files used by a filegroup are of equal size. Try to create all the files that will host the PRIMARY filegroup so that they are the same size.

The Teamcenter 8.1 database also contains a filegroup called ILOG. In the default database configuration, ILOG is hosted in its own file. Objects in the ILOG filegroup are created and used only if Teamcenter audit logging is enabled and configured to log to the database (rather than text files). If your deployment is set up to log to the database, you can expect heavy activity on this file. Therefore, if you use the database for audit logging, Siemens recommends placing the ILOG data file on a separate disk volume from the main data file and the transaction log.

Use Data Compression (Enterprise Edition only)

SQL Server Enterprise Edition includes the ability to natively compress its data pages on disk. This can provide significant space savings on disk. Because it reduces the amount of disk I/O done for a given amount of Teamcenter data, it also tends to improve performance. When data compression is enabled, data pages are also compressed in the buffer pool (i.e., RAM). This provides the additional benefit of increasing the effective amount of application data that can be accessed from the data cache rather than disk.

Benchmarking with Teamcenter and SQL Server has shown significant performance increases when data compression is used for the Teamcenter database.

Siemens recommends using data compression if your edition of SQL Server allows it.

Adjusting SQL Server and Database Settings

Verify Database Collation

When the Teamcenter database is created, it will assume the server collation (set when SQL Server was installed) unless otherwise specified. It is a good idea to verify that the collation for the database is the one the Teamcenter software expects. As shown in Figure 12, this can be verified in the Database Properties dialog box in SQL Server Management Studio. The collation displayed in this dialog box should be Latin1_General_BIN.

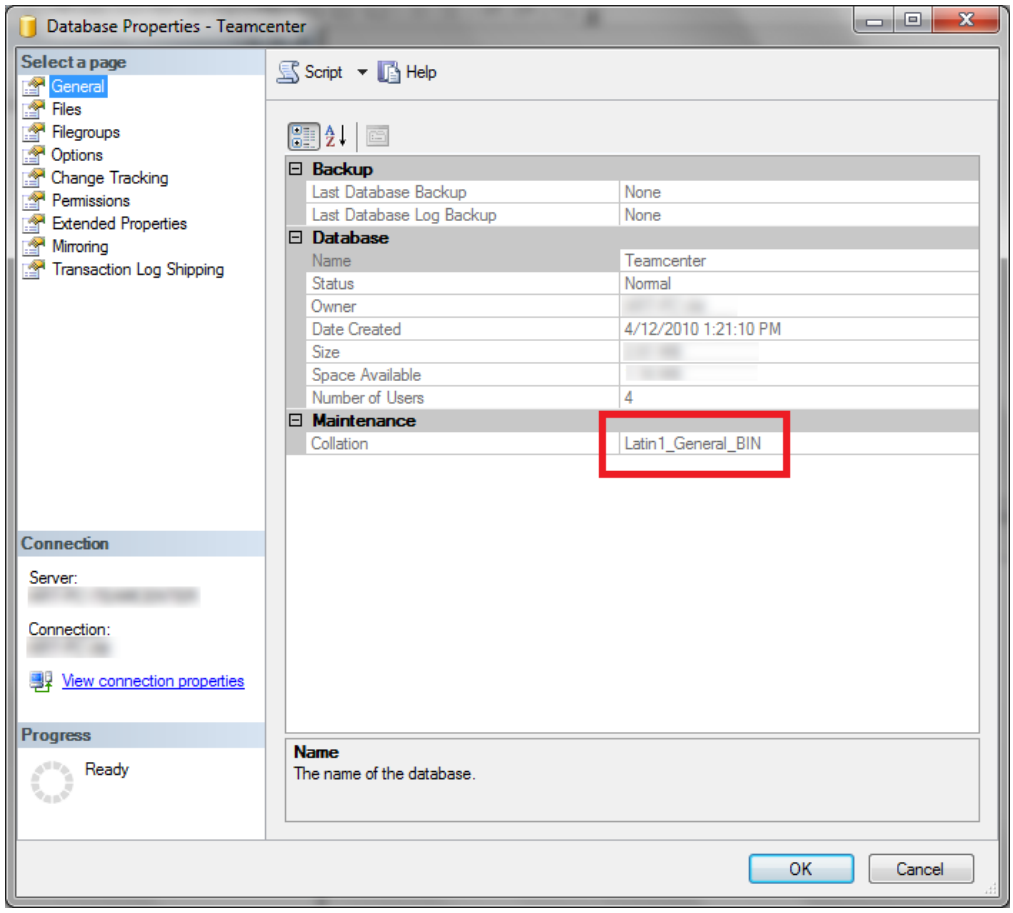

Figure 12 - Verify the Database Collation for Teamcenter

Automatic Statistic Updates, Auto Shrink, and Other Options

SQL Server uses internal statistics about the data stored in tables to decide which indexes, if any, to use for most efficiently executing a query. As table data changes over time, these statistics need to be kept current. Each SQL Server database has the option for these statistics to be updated automatically (statistics update can also be initiated manually, if needed). Siemens recommends that auto-update of statistics be enabled for the Teamcenter database.

Figure 13 shows the Database Properties dialog box with the Options page selected. The highlighted section lets you view and configure the settings for automatic statistics. Auto Create Statistics and Auto Update Statistics should be set to True. Auto Update Statistics Asynchronously can be tuned to your preference.

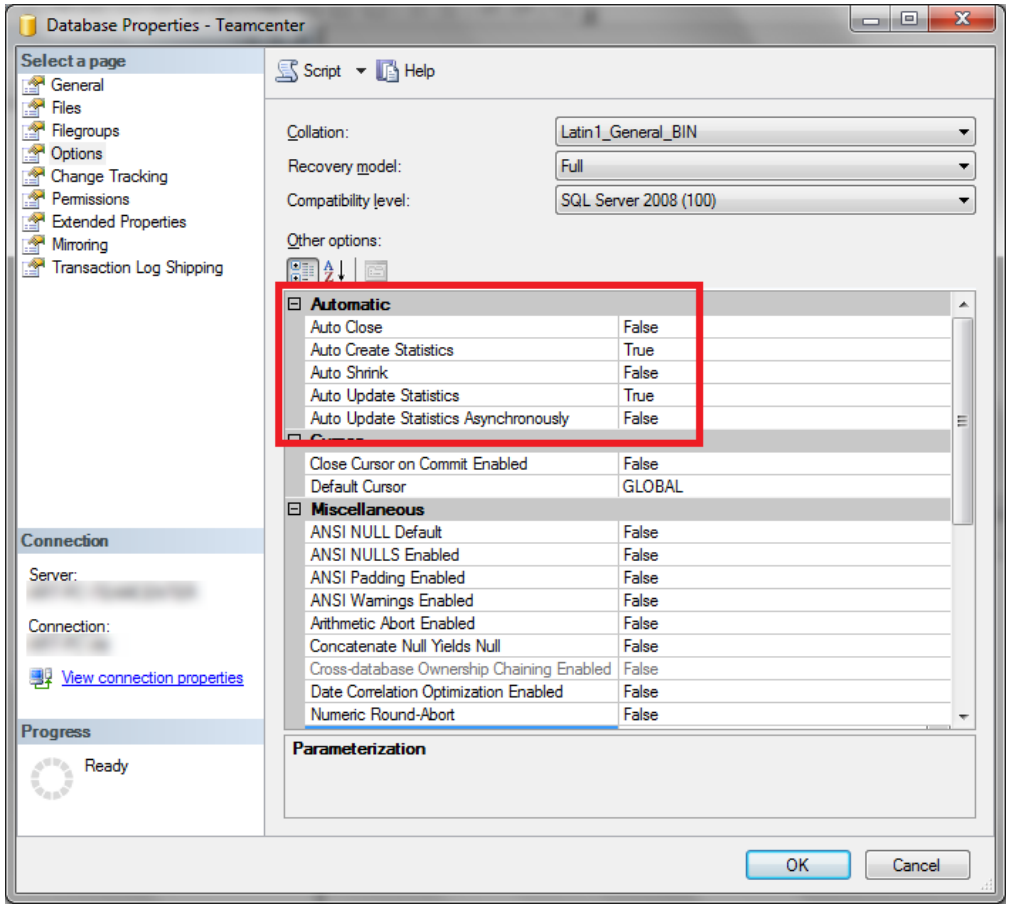

Figure 13 - Auto Statistics Management Should be Enabled; Auto-Close and Auto-Shrink Disabled

Two other options are also shown in the highlighted section of Figure 13, with their recommended settings selected. Both Auto-Close and Auto-Shrink should be set to False.

Choosing a Database Recovery Model

The recovery model of a SQL Server database determines how changes are recorded and stored in the database's transaction log. The three options for the recovery model setting on your Teamcenter database are Full, Bulk-logged, and Simple:

- **Full recovery model.** This is the best model for preventing critical data loss and restoring data to a specific point in time, and it is generally used by enterprise production systems. If the transaction log is available, it is possible to get up-to-the-minute recovery and point-in-time restore if the end of the transaction log is backed up and restored. The trade-off for the Full recovery model is that it requires more administrator attention and a more complex backup/restore strategy.
- **Bulk-logged model.** This model is for databases that have critical data, but for which the DBA wants to minimize storage or performance impact of bulk operations (e.g., bulk

data loads, table copies, index maintenance). It provides nearly the recovery power of the Full model, but with some limitations.

- **Simple recovery model.** This model is appropriate if the data backed up is not critical, data is static or does not change often, or if data loss is not a concern for the organization. If an active database is lost with the Simple recovery model, the organization loses all transactions since the last full or last differential backup. This model is typical for test environments or production databases that are not mission critical or have very light activity.

In general, Teamcenter databases are frequently updated and hold information that is of high importance to the organization. Especially for large Teamcenter databases, full backups may not be feasible with high frequency (e.g., daily or more often). For these reasons, Siemens recommends using the Full recovery model with Teamcenter databases.

Administrators must be aware of the management implications of the Full and Bulk-logged recovery models. In particular, under these recovery models, transaction logs must be backed up regularly to prevent them from filling up or exhausting available disk space.

Ongoing Management

The Database Backup Plan

Perhaps the single most important ongoing maintenance task for a business-critical database such as Teamcenter is managing backups. Siemens recommends you carefully plan, document, and test the backup *and recovery* strategy for the Teamcenter database.

The types of database backups allowed in SQL Server are:

- **Full database backup.** Contains all the data in the database, plus enough transaction log information to allow for recovering that data
- **Differential database backup.** Contains all the data changed since the last full backup. Differential backups allow a smaller, faster backup to be taken, which can later be combined with the full backup to restore the database to the time of the differential backup.
- **Transaction log backup.** Contains all the database changes since the last (full or differential) database or log backup. Periodically backing up the log allows you to restore the last database backup followed by subsequent log backups to recover to the latest point in time needed.
- **Database file or filegroup backup.** Contains only data from specified file(s) or filegroups(s) to allow quick, piecemeal restores if only part of the database is lost. These types of backup are not recommended for Teamcenter deployments.

The different types of backup allow you to back up with the frequency appropriate to your business requirements, even for databases that are very large. For example, a typical backup strategy for a large database (e.g., several hundred gigabytes) might include a weekly full backup, nightly differential backups, and transaction log backups every 30 minutes. If the database were lost, the DBA would:

1. Restore the last full backup.
2. Restore all differential backups since the full backup, in sequence.
3. Restore all the transaction log backups since the last differential backup, in sequence.

If properly executed, this would return the database to operation with at most 30 minutes of work lost.

For small databases, it is perfectly valid to simply perform full backups one or more times per day. There is nothing inherently wrong with this approach, and it has the virtue of simplicity. But it is only practical for databases that can be backed up very quickly.

Keep in mind also that for databases with the Full or Bulk-logged recovery model, regular transaction log backups are required to keep the log from growing out of control.

The right backup strategy is dependent on the database size and level of change activity in your installation. This can vary greatly, even across deployments of the Teamcenter product, so a one-size-fits-all recommendation can't be made here. However, keep the following in mind as you plan your backup strategy:

- If you choose the Full recovery model, the transaction log can grow very quickly and should be scheduled for backup at multiple times throughout the day. Performing multiple backups ensures that the log is truncated frequently and provides for improved data recoverability.
- If you choose the Simple recovery model, schedule full database backups for at least once per day. With this method, data recoverability is limited to the last full database backup.
- Database backups should be stored off site when possible and appropriate.
- Back up a database immediately after you make extensive changes or perform non-logged operations.
- Be sure to include regular backups of the master and msdb databases as part of your overall backup strategy. The master database contains information about the databases on your system, and the msdb database stores information about scheduled jobs.
- 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.

Finally, remember that what you really need in an emergency is a *restore*, not a backup. Therefore, plan restore with care, and test restoring your database from actual backups to validate that you have covered everything.

Synchronization of Teamcenter Database and File Backups

Another point to consider when managing overall backup of the Teamcenter application is the interaction between the SQL Server database and the file repository. As shown in the architecture diagram in Figure 1, the Resource Tier includes both the Teamcenter database and the Teamcenter file server(s). A full and consistent backup of the Resource Tier requires both database and file backups.

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

Siemens recommends choosing one of three approaches to doing this:

- **Option 1**: Take cold backups—that is, take the system offline to halt user activity, back up both the database and file repository, and put the system 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.
- **Option 3**: Use Teamcenter's hot backup mode. In this mode, Teamcenter suspends writing files directly to the file repository (they are held in a temporary location called a Blobby Volume). File backups are taken, and the system is then returned to normal mode. If a restore is necessary, SQL Server's point-in-time restore capability[2] can be used to restore the database to the point in time when the system was placed in hot backup mode. This will ensure consistency between the database and the file repository.

More information on this advanced topic is available from Siemens. You can talk with your Siemens support professional, or refer to documentation at the Siemens Teamcenter GTAC Support Site.

[2] Point-in-time restore requires the Full or Bulk-logged recovery model and a chain of regular transaction log backups.

Monitor Index Fragmentation and Defragment When Necessary

SQL Server uses indexes to provide fast access to information when users or applications request it. These indexes are maintained by the Database Engine as the table data grows and/or changes. Over time, the indexes can become fragmented; especially in databases that handle heavy insert, update, and delete activity. An index is "fragmented" when the physical ordering on disk does not match the logical order of the data (as defined by the index key) or when data pages that contain the index are dispersed across non-adjacent sections of the disk.

Fragmentation of an index can reduce the speed of data access and result in slower application performance. It can also cause more disk space to be used than is actually necessary. Index fragmentation can be corrected by reorganizing or rebuilding the index.

You can tell which indexes, if any, have fragmentation problems by using the sys.dm_db_physical_stats() system function. This function provides lots of details about the physical layout of the index. However, the most important result column for tracking fragmentation is avg_fragmentation_in_percent. This column indicates how fragmented the index is on disk. A low number means low fragmentation (good); a high number means high fragmentation (bad).

For example, this query returns index physical stats for all the indexes in the current database:

```
SELECT OBJECT_NAME(object_id),
       index_id,
       page_count,
       index_type_desc,
       avg_fragmentation_in_percent,
       fragment_count
FROM sys.dm_db_index_physical_stats(db_id(),NULL,NULL,NULL,'LIMITED')
```

To identify the indexes by name, you can join against the sys.indexes system view.

Similar information is also available in the Standard Reports in SQL Server Management Studio. To view this report, right-click the Teamcenter database, select Reports > Standard Reports, and select the Index Physical Statistics report on the fly-out menu.

The following table indicates general guidelines for interpreting the avg_fragmentation_in_percent value.

Fragmentation	Recommended Action
< 5%	Do nothing
5% to 30%	Reorganize with ALTER INDEX REORGANIZE
> 50%	Rebuild with ALTER INDEX REBUILD WITH (ONLINE=ON) or CREATE INDEX with DROP_EXISTING=ON

Reorganizing an index does not block user access to the index while underway. However, rebuilding or re-creating the index does prevent user access to the index. The exception to this is if the ALTER INDEX REBUILD statement is used with the ONLINE = ON option. Note that online index rebuild requires the Enterprise Edition of SQL Server 2008.

Periodically checking index fragmentation and taking any necessary corrective action is important to maintaining the performance of your Teamcenter deployment. The rate at which fragmentation may occur depends on user activity, but as a general rule, Siemens recommends checking index fragmentation at least monthly.

For more information about reorganizing and rebuilding indexes, see Reorganizing and Rebuilding Indexes.

Run the Teamcenter Index-Verifier Tool

Siemens provides an index_verifier utility, packaged with all versions of Teamcenter that can analyze your Teamcenter database for missing indexes. If missing indexes are found, the index_verifier utility will provide SQL output that you can use to create the needed indexes.

Siemens best practice is to run this utility at least monthly.

Siemens also recommends adding additional indexes you may require by using the *install* command-line utility provided with Teamcenter. Doing so will register the index as part of the Teamcenter schema, at which point the index_verifier utility can check for its existence. If you manually add indexes to the Teamcenter database, index_verifier will have no knowledge of them, and it will be your responsibility to ensure that those indexes are carried forward and maintained.

Full documentation of these utilities can be found in the *Teamcenter Utilities Reference Manual*, available on the Siemens Teamcenter GTAC Support Site.

Periodically Check Database Integrity

For all the sophisticated data management techniques embedded in the SQL Server Database Engine, there is still the possibility of some corruption occurring in a database, most notably as the result of a hardware glitch. To head off the potential impact of such problems, you should regularly check the integrity of the database. The statement for doing this in T-SQL is DBCC CHECKDB.

It is best to include database integrity checks in a scheduled job that executes DBCC CHECKDB, or do it with a scheduled Maintenance Plan that includes the Check Database Integrity task.

If any problems are detected, you can restore from backups (usually the best option) or use one of the REPAIR options on DBCC CHECKDB.

Monitor Space Used

Over time, the space used in your Teamcenter database for data and indexes will increase. Regardless of the initial space allocations you have made, you will need to occasionally check on the amount of space remaining in data and log files as well as on the host disk volumes.

Space consumed and remaining for the Teamcenter database can easily be checked using the Standard Reports in SQL Server Management Studio. The Disk Usage report displays the currently reserved space in data and log files and the amount that is in use. This is also a handy place to check for any recent data or log file growths. If these are occurring, it could be a sign that a rearrangement or expansion of data files is necessary.

Details on space usage can also be obtained via T-SQL by using the sp_spaceused stored procedure. See sp_spaceused (Transact-SQL) in SQL Server Books Online for details.

Use SQL Server Agent Jobs and Alerts

Many of the ongoing database management and monitoring tasks required to administer your Teamcenter database can and should be automated. SQL Server includes SQL Server Agent, a job scheduling and alerting service that is tightly intertwined with the SQL Server database engine. SQL Server Agent can be used to schedule regular backups, database integrity checks, index reorganization jobs, and many other routine tasks. It can also be used to check database engine metrics and alert someone if they are outside normal bounds.

SQL Server Maintenance Plans also provide a neatly packaged way to define and schedule routine management tasks for a database or for multiple databases.

Siemens recommends making use of these tools to configure automated maintenance and monitoring tasks for the Teamcenter database.

Tools for Monitoring Database Server Performance

Once your Teamcenter database is in production, it is important to establish a baseline of its normal performance characteristics and to check its health and status periodically. The following sections discuss some of the tools and techniques available for monitoring database server performance.

SQL Server Activity Monitor

A good, easy-to-use starting point for monitoring your SQL Server instance is the Activity Monitor in SQL Server Management Studio. This pulls several key pieces of information together into one view. Although it does not provide highly detailed insight on its own, it can be a good part of your SQL Server monitoring toolbox.

To open Activity Monitor, right-click the server icon in the Object Explorer pane of SQL Server Management Studio and select the Activity Monitor item. There is also a button for Activity

Monitor on the Standard toolbar in SQL Server Management Studio. The Activity Monitor will open as a new tab in the main Management Studio window.

The Overview section, at the top, shows summary graphs of four key metrics for getting a sense of the load on the server. % Processor Time is an average of CPU utilization across all processors. Waiting Tasks shows the number of tasks waiting on CPU, memory, disk, locks, or other resources. The Database I/O graph shows overall disk I/O throughput of SQL Server. Batch Requests/sec shows the number of individual T-SQL batches executed per second (e.g., SQL statements or stored procedures calls from applications).

The other sections (also called panes) in the Activity Monitor window provide detailed snapshots about different aspects of the Database Engine's current workload. A full discussion of these sections and how to interpret the information provided is well beyond the scope of this paper. However, a brief introduction to each section is provided here.

The Processes section presents a list of the active user connections to the server. For each connection, useful troubleshooting information is provided, including the current database the connection is using, what resource the connection is waiting on (if any), and whether the connection is blocked by another connection. The Processes section is thus useful for getting an idea of the number of active connections and what sort of performance roadblocks they may be experiencing.

The Resource Waits section provides a summary of the "waits" the Database Engine is observing in current activity. Examples of waits are network I/O for clients to pull results across the wire, disk I/O for the server to pull data from disk into buffer memory, or lock waits while one connection waits on a transaction lock held by another connection. Besides the type of wait, this section shows the cumulative and recent amount of time spent on this type of wait and the number of connections experiencing it. This information is just a coarse summary of a highly detailed set of tuning information available from SQL Server wait statistics. It can be quite useful, though, to get an indication of system bottlenecks at a glance. Much more information about using wait stats to understand the performance of SQL Server can be found in SQL Server Waits and Queues.

The Data File I/O section lists all data and transaction log files in use by databases on the current SQL Server instance. With each file, recent read and write throughput (in MB/sec) is reported. The last column in this section is particularly interesting. The Response Time (ms) column reports disk latency for I/O against the file. This is a good number to look at to begin an investigation of whether a workload is I/O bound.

Finally, the Recent Expensive Queries section lists some of the most expensive queries the engine has recently executed. Besides the query text, average execution metrics are listed for each.

SQL Server Books Online has more information about Activity Monitor.

Standard Database Reports in Management Studio

Another easy-to-use resource is the set of database Standard Reports in SQL Server Management Studio. These reports are accessible by right-clicking the icon for the Teamcenter database in the Object Explorer pane and selecting Reports > Standard Reports. The reports available on the fly-out menu provide basic and advanced information on many aspects of the database.

Performance Monitor

Windows Performance Monitor[3], a management tool included with Windows, is a key tool for understanding the performance characteristics of your database server. Performance Monitor allows you to observe a wealth of metrics about the activity of the operating system and applications such as SQL Server. Performance Monitor can be used to observe these metrics interactively in real time or to log them to disk for later review.

It is good practice to establish a baseline understanding of the performance of your database system after it is deployed. This will allow you to better understand issues if your system runs into performance or other problems later. By comparing observed behavior in a problem situation to your baseline expectations, you'll more easily be able to focus in on the root cause.

The following list of Performance Monitor counters is a condensed set of valuable measurements to use to understand your database server's activity. It only scratches the surface of the SQL Server monitoring that is possible, but is a useful starting point.

Type	Object	Counter	Instance	Description
CPU	Processor	% Processor Time	-	Total CPU use on the server
CPU	Process	% Processor Time	sqlservr	CPU use by SQL Server alone
I/O	Physical Disk	Disk sec/Read	*All volumes used by SQL*	Latency of reads from disk. If this is high, server may be I/O bound.
I/O	Physical Disk	Disk sec/Write	*All volumes used by SQL*	Latency of writes to disk. If this is high, server may be I/O bound.
Memory	Memory	Page Faults/sec	-	Rate at which Windows could not find the page it needed at the expected location in memory.

[3] Also known as System Monitor in some editions of Windows

Memory	Memory	Pages/sec	-	Rate at which Windows read or wrote to the system paging file. If the ratio of this to Page Faults/sec is more than a few percentage points, memory may be insufficient.
Memory	SQL Server: Buffer Manager	Buffer cache hit ratio	-	% of the time data is found in SQL Server's memory cache, rather than read from disk. Should be > 90%.
Memory	SQL Server: Memory Manager	Total Server Memory (KB)	-	Total memory used by the SQL Server process. If this is near total system memory, more memory may be needed, or you may need to reduce 'max server memory'.
SQL	SQL Server: SQL Statistics	Batch Requests/sec	-	Number of SQL batches executed by SQL Server per second. An easy gauge of how actively the server is servicing client activity.
SQL	SQL Server: General Statistics	User Connections	-	Number of open connections to SQL Server.

Note that if you are using a named instance of SQL Server, the name of the object in Performance Monitor will be slightly different. For default (not named) instances, the object name is SQL Server: <object>. For a named instance, the object name is MSSQL$*InstanceName*: <object>. An example of this for an instance named TEAMCENTER is shown in Figure 14.

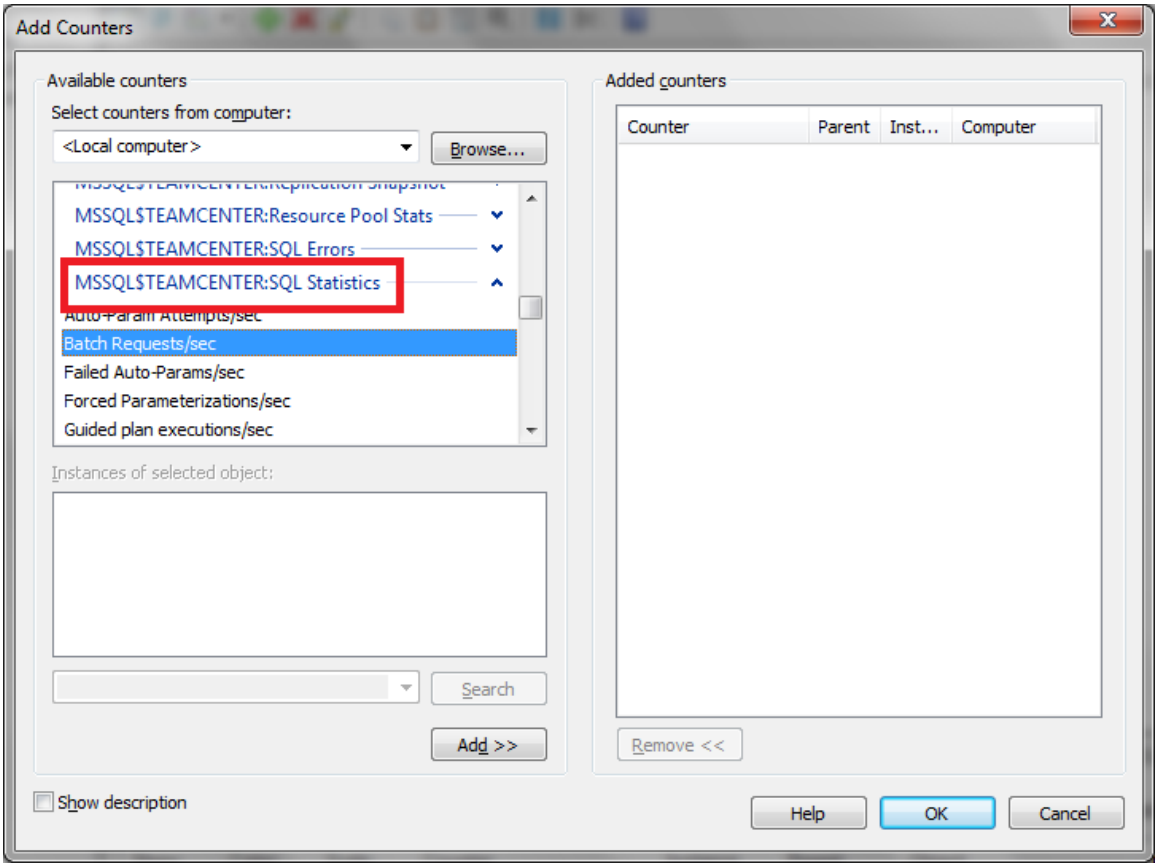

Figure 14 - Performance Monitor Counters for a Named Instance of SQL Server

Checklist

For convenience, the following checklist summarizes and consolidates the best practices discussed in this paper. After familiarizing yourself with the preceding in-depth discussion, you can use this as a reminder to cover all the recommended best practices.

	Use a dedicated database server and instance.
	Choose the right SQL Server Edition (Standard or Enterprise) for your needs.
	Use 64-bit SQL Server if your hardware allows. Consider acquiring 64-bit hardware if it does not.
	Choose a high availability strategy.
	Use a low-privilege Windows account for the SQL Server and SQL Agent services.
	Grant the *Perform volume maintenance tasks* permission to the SQL Server service account to allow fast file initialization.
	Set the collation to Latin1_General_BIN when installing SQL Server.
	Set the server Authentication Mode to Mixed.
	Configure the SQL Server Database Engine service and the SQL Server Agent service to Auto-Start.
	Size and place tempdb per best practices (as described in detail in this document).
	Set the maximum server memory for SQL Server per recommendations.
	Set Maximum Degree of Parallellism to 1.
	Place the data and log files on physically distinct disk volumes.
	Use RAID 10 or RAID 5 for data files; use RAID 10 or RAID 1 for log files.
	Use large initial size for data and log files, based on projected growth.
	Leave automatic file growth enabled “just in case.”
	Spread the PRIMARY filegroup across equal-sized files on multiple disk volumes (optional).
	Use SQL Server data compression (SQL Server Enterprise Edition only).
	Verify that the Teamcenter database is using the Latin1_General_BIN collation.
	On the Teamcenter database, use the following settings: ➢ Auto Create Statistics = True ➢ Auto Update Statistics = True ➢ Auto Close = False ➢ Auto Shrink = False
	Use the Full recovery model for the Teamcenter database.
	Ensure frequent transaction log backups are scheduled to prevent the log file from growing out of control.
	Plan and implement an effective backup strategy.
	Test database restore using live backups.
	Be sure to back up the master and msdb databases in addition to the Teamcenter database.
	Monitor index fragmentation, and defragment when necessary.
	Periodically check database integrity.
	Monitor space used.
	Use SQL Server Agent jobs and alerts to automate and schedule maintenance and monitoring.
	Get familiar with SQL Server monitoring tools, and establish a baseline performance profile for your system.

Summary

Siemens Teamcenter PLM software, coupled with Microsoft SQL Server database software, provides your enterprise with the platform to build an agile, globally competitive product development process.

SQL Server provides an ideal database platform for Teamcenter. SQL Server is an enterprise-ready, comprehensive, integrated data management and analysis platform that makes it possible for organizations to reliably manage large, mission-critical workloads and complex business applications. SQL Server provides rapid data integration and data mining and fast, intuitive analysis and reporting capabilities. Its built-in features deliver reliability and security on a scalable foundation that includes a powerful database engine.

Using the best practices discussed in this paper can help you optimize performance of Siemens Teamcenter for product lifecycle management and can help you avoid and minimize problems. The links in the following section provide even more resources to ensure a successful implementation of Teamcenter on SQL Server.

Links for Further Information

For general information, visit the Siemens PLM Software home page.

SQL Server information can be found in Books Online:

- SQL Server 2008 Books Online
- SQL Server 2005 Books Online

See the SQL Server Best Practices portal for technical white papers, the SQL Server Best Practices Toolbox, Top 10 Lists, and other resources.

For Siemens and Microsoft news, events, and further information, see the Siemens/Microsoft Alliance page.

Following is a list of technical white papers that were tested and validated by the SQL Server development team. These can help you learn more about specific SQL Server topics.

- Storage Top 10 Best Practices (for SQL Server)
- Predeployment I/O Best Practices
- Tuning the Performance of Change Data Capture in SQL Server 2008
- The Data Loading Performance Guide
- Best Practices for Migrating Non-Unicode Data Types to Unicode
- The Impact of Changing Collations and of Changing Data Types from Non-Unicode to Unicode
- XML Best Practices for Microsoft SQL Server 2005
- SQL Server 2005 Security Best Practices - Operational and Administrative Tasks
- Partial Database Availability
- Comparing Tables Organized with Clustered Indexes versus Heaps
- SQL Server 2005 Deployment Guidance for Web Hosting Environments
- Implementing Application Failover with Database Mirroring
- TEMPDB Capacity Planning and Concurrency Considerations for Index Create and Rebuild
- Getting Started with SQL Server 2008 Failover Clustering
- SQL Server 2008 Failover Clustering
- Database Mirroring Best Practices and Performance Considerations
- SQL Server 2005 Waits and Queues
- Troubleshooting Performance Problems in SQL Server 2005