**12 Essential Steps After Installing SQL Server**

By [Sadequl Hussain](http://www.sql-server-performance.com/author/sadequlhussain/)

FROM: <http://www.sql-server-performance.com/2016/post-installation-steps/>

Rolling  
out SQL Servers is a key task for DBAs.  While for some DBAs this can be a  
frequent exercise, others will have to face it one time or other throughout  
their career in an organisation.

Successfully  
planning an installation requires a fair bit of knowledge about SQL Server and  
a basic understanding of the operating system. Once the installation is  
complete, the system usually needs some changes to default configuration before  
production databases can be rolled out. In this article I list some common steps  
that I take after installing a SQL Server. This can be used as a “checklist” to  
ensure consistency for all the SQL Servers that you roll out.

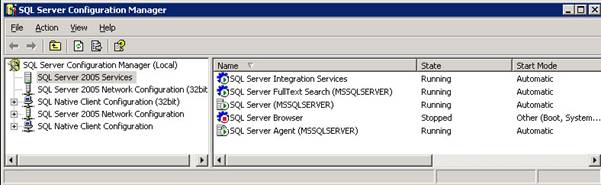
**Step 1:  
Install the Service Pack, Hotfixes and Cumulative Updates**

This  
is the first step you should take after a successful installation. You should  
always install the latest service pack available unless there is a clear  
business cause not to do so. At the time of this writing, SQL Server 2005 is in  
service pack level 3 and SQL Server 2008 has service pack 1. Service packs are  
freely downloadable from the Microsoft web site. If there are hotfixes or  
cumulative updates available after the last service pack has been released, you  
should apply them as well. This ensures that your server is ready with the  
latest version of the database software. Once all the patches are installed,  
make a backup of your system databases and reboot the server.

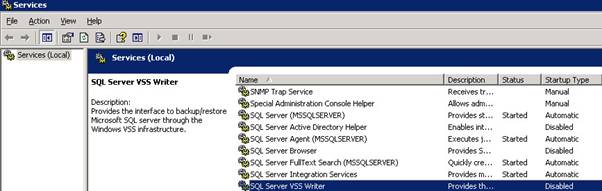
**Step  
2: Configure SQL Services**

Although  
your planning phase should determine what SQL services you are going to install,  
there may be occasions when a service is rolled out but not needed immediately.  
Perhaps your business requires you to install a SQL Server and although  
reporting is not going to be part of it, management insists you install  
Reporting Services – “just in case”. If a service is not going to be needed  
immediately (or needed at all), disable it.

One  
example of a service that you can disable is the Browser service. This service  
is installed by default. If you are not running named instances or planning to  
install multiple instances in one machine, you can disable this service.  You  
can disable the Browser service from the SQL Server Configuration Manager.

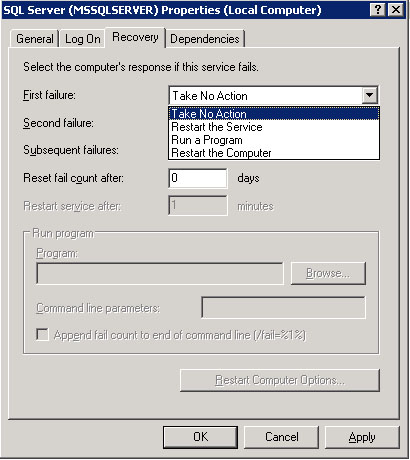
**

Another  
service that you can disable is the SQL Server VSS Writer. Unless you are using  
applications that make use of the Windows Volume Shadow Copy infrastructure to  
backup SQL databases, you can disable this service safely. Note that SQL VSS  
Writer is not available from the Configuration Manager. You can access it from  
the Control Panel > Services applet (see below).

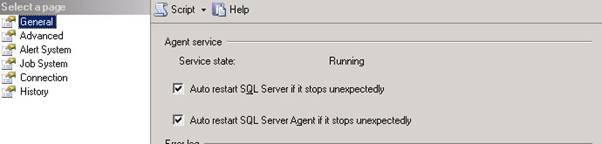
**

While  
non-essential services should be disabled, other services like the Agent are  
supposed to be running continuously. This is because scheduled backups,  
maintenance plans, replication and jobs are dependent on the Agent service.  
Similarly, if your server is going to execute Integration Service packages,  
ensure the SSIS service starts automatically every time the server boots up.  
You can configure the start-up property of a service from the SQL Server  
Configuration Manager.

One  
thing that is not available in the Configuration Manager is the recovery  
behaviour of SQL Services – i.e. what should the services do if they fail unexpectedly.  
To access this behaviour, you can bring up the service’s property from the  
Control Panel’s “Services” applet. The “Recovery” tab of the service property  
allows you to define what the computer should do in case the service fails once,  
twice or subsequent times. From here, you can specify that the service should  
start automatically if it fails.



The  
same type of behaviour can be set in the SQL Server Agent’s property page. Here,  
you can specify that SQL or Agent service should restart automatically if they  
stop unexpectedly.



**Step  
3: Configure Default Directories**

Everything  
has its place in this world and when it comes to files and folders, SQL Server  
is no exception. A number of directories are created during installation – you  
have one directory for storing data and log files, another directory for  
replication, another for full text indexes, and yet another for backup files.

Where  
SQL places its files is determined mostly during the installation process. This  
happens because either you provide this information explicitly or SQL uses a default  
set of locations. The directory path information you provide during the setup  
should really be coming from your planning phase. Once the installation is  
complete, it is still possible to fine-tune some of these component locations.

If  
you have followed the best practice, your data files should be residing in a  
separate physical drive and directory.  The transaction log files of user  
databases should also be in a separate drive and folder. You can configure both  
the locations from the server’s properties:

**

Similar  
to the data and log directories, you can specify the default locations for:

a)     
Backup  
files

b)     
Replication  
files

c)     
Full  
text index files

d)     
Job  
output files

SQL  
Server creates the folders for these components in the same location where data  
files are placed. To ensure data files are given the maximum possible dedicated  
space as far as disk activity is concerned, I try to move these folders to a  
separate drive.

Unfortunately,  
you cannot use a GUI front-end to configure these alternate locations. The path  
information is saved as registry values and that means you either have to  
manually change registry entries or use an extended stored procedure like  
xp\_regwrite.  Also, the location of the registry keys will be different between  
named instances. They will also vary depending on your version of SQL Server (2005  
or 2008). Generally, you will be looking for a registry key called “Working  
Directory”.

The  
script below shows how I changed the default locations for a default instance of  
SQL Server 2005 installed in a Windows Server 2003 system:

[Configuring\_Default\_Directories.sql](http://www.sql-server-performance.com/code/Configuring_Default_Directories.sql)

And  
of course, the paths you specify as registry key values must exist – that means  
you need to create (or move) the folders.

To  
ensure the system is aware of your changes, restart SQL services after making  
the changes.

**Step  
4: Configure Default Database Properties**

We  
all know it – model database works as the “template” for new databases. If you  
want to set some common properties across all your user databases, you can set  
them in the model. A number of properties can be set here to take advantage of  
this approach:

a)     
Recovery  
Model: perhaps you are installing SQL Server for a data warehouse that will host  
a number of data marts that would not require full recovery model. You can set  
the recovery model of the model database to “simple” mode in such cases.

b)     
File  
Properties: by default, the data files are configured to grow by 1 MB and the  
log files by 10% of their size. This means that a large transaction log files  
will grow by a large proportion when it needs more space. For example, every  
time a 6 GB log file needs to grow, it will take up 600 MB of disk space in one  
gallop. For a write intensive database, the data file’s incremental growth by 1  
MB can result in fragmentation. As a general rule of thumb, the log files  
should be quarter or half the size of the data files. If you have an idea about  
how big your data and log files are going to be and how they should grow, you  
can set up the properties in the model database. I generally set the growth  
increments in megabytes rather than leaving it as a percentage.

Another area where you can make use of the model database is filegroups. If you  
want your user tables to be created on secondary data files and leave the primary  
filegroups with system objects only, you can create secondary data files in the  
model database and assign them to a user defined filegroup and make that  
filegroup default.

c)     
Database  
Options: depending on your requirements you can set up other database options  
in the model database. Examples of such options could be torn page detection,  
ANSI defaults etc.

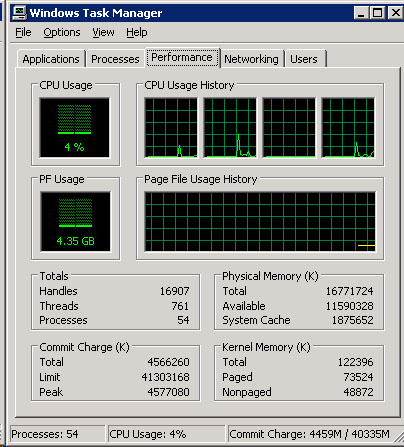
d)     
Custom  
Objects: if you have custom objects that need to be present in each user  
database, you can put them in the model database.

**Step  
5: Configure tempdb Database**

The  
temporary database is used heavily in SQL 2005 and latter versions. Best  
practice suggests that for optimal performance, SQL Server’s tempdb database  
files should be placed in a separate dedicated disk drive. This should be  
followed wherever possible, but even if your server’s current disk  
configuration does not allow this, you can still make some changes to ensure  
tempdb is optimally tuned. Part of this tuning means you should allocate a  
healthy amount of space to this database.

The  
number of data files you should allocate in the tempdb database will depend on  
the number of CPUs (logical or physical) present in the machine. When SQL  
Server accesses a database, it starts a scheduler thread for each data file present.  
So if you have eight CPUs in your physical server, the tempdb database should  
have eight data files to “load balance” the threads.

The  
easiest way you can check the number of CPUs present in the machine is by  
looking at the Windows Task Manager’s Performance tab.

**

Also,  
data files in a database are used in a “proportional fill” algorithm. This means  
that if you have two data files in a database – one 4 GB and another 8 GB, SQL  
will try to stripe the data across the two files in a 1:2 ratio. For each  
stripe of data written to the 4 GB file, SQL will write two stripes in the 8GB file  
– hence the term “proportional fill”.

Now  
if you configure tempdb to have the same number of data files as the number of CPUs  
present and specify each data file to have the same size and growth, you are  
effectively ensuring the CPU load is evenly distributed across tempdb.

The  
following script shows one such configuration done from the query window.

[*Configuring\_tempdb\_Data\_Files.sql*](http://www.sql-server-performance.com/code/Configuring_tempdb_Data_Files.sql)

As  
you can see from the code, we are adding three new data files to tempdb (since  
there are four CPUs present and there is already one data file) and ensuring  
the four files to have the same size and growth characteristics.

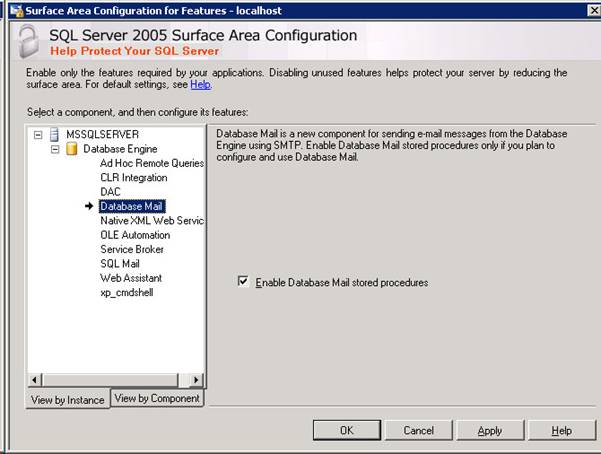
Once  
you have completed the tempdb configuration, you will need to restart the SQL  
instance for the changes to take effect.

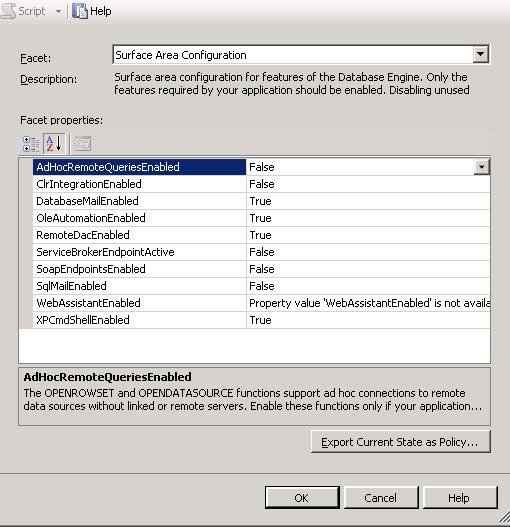
**Step  
6: Configure Server Properties and Features**

Of  
all the SQL Server configurations, this one is probably the most important and  
has a direct effect on how the server behaves. What properties you configure  
and what values you set here will depend on your organisation’s business needs,  
SOEs etc. There are a number of properties you can set for the database server  
engine and there is more than one way you can set some of them. I like to  
divide these properties into three categories:

**Configuring  
Features:**

If  
it is SQL 2005, use the Surface Area Configuration tool for enabling some of  
the SQL Server features. For SQL 2008, use the Surface Area Configuration Facet  
from the Server properties. Both these versions list the same features and by  
default all of these features are disabled.

**

**

What  
you enable here will depend on what you want your server to do. You may want to  
enable Database Mail (discussed later). If your stored procedures are going to  
use the xp\_cmdshell utility, you will need to switch it on here. If you think  
there will be managed code in your database application, you need to enable  
CLR.

**Configuring  
Parameter Values:**

a)     
Memory:  
Set the minimum and maximum server memory parameters. You can do so from the server  
properties dialogue box or using the sp\_configure system stored procedure. If  
AWE is required, enable that too.

b)     
Fill  
factor: Set a default value for index fill factor. This can save page splitting  
in indexes in future.

c)     
Processor:  
Configure max. degree of parallelism, cost threshold for parallelism as  
necessary.

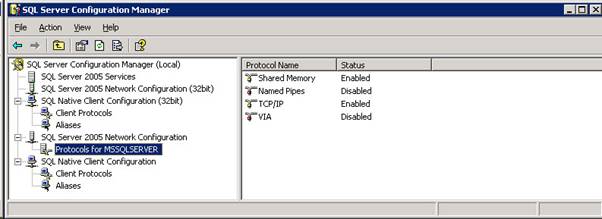
d)     
Network  
connections: Configure remote query timeouts if you want to override the  
default behaviour of 600 seconds.

e)     
Security:  
Although you can change the authentication mode (Trusted or Mixed) at this  
stage – this is a choice you probably have made during the installation. You  
can also enable C2 audit mode as part of security configuration.

f)      
Trace  
Flags: If you are installing SQL Server 2005, you may wish to enable trace  
flags as start-up parameters for detecting and reporting deadlocks. The trace  
flag 1204 can be used for such purposes.

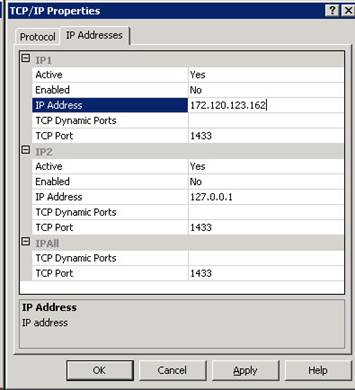
**Configuring  
SQL Server Network:**

This  
is where you define how SQL will use the network to talk to other computers. The  
first thing you need to do here is enabling or disabling the network protocols SQL  
will use to listen for incoming traffic. You set this from the Configuration  
Manager. The picture below shows how I have disabled Named Pipes.

**

Next,  
you need to see if your applications require server aliases. Some of the older  
apps may require you to define a server alias for a named instance.

Finally,  
you need to decide if you will assign non-default TCP port number for your  
instance. The default instance uses a default port number of 1433 and the named  
instances are usually assigned dynamic port numbers at start up. If you want to  
assign a non-default port number to SQL Server for TCP connections, you can set  
it here.

**

Most  
of the server property changes would require restarting the service.

**Step  
7: Configure Security**

When  
it comes down to SQL Server security, companies usually take one of two  
approaches – some follow a rigorous data security policy, others just leave it  
all to defaults (and to potential attacks). If you are working for a company  
like the former, you are in luck – you will know what to do. If your company  
does not have any specific procedures for implementing SQL Server security  
(written or unwritten), it is up to you how you define it. Once you have  
completed this part though, you will have something to refer back to in future.

First  
and foremost, ensure there are as few members in the sysadmin server role as  
possible. The Local Administrators Windows group is a member of the sysadmin  
role by default. Unless you want to have your Windows Administrators to have  
access to the SQL Server, you should get rid of this group from the sysadmin  
fixed server role. The next step would be to revoke the login for this group.

If  
you have a pre-defined account that you use for database administration, add  
that to the sysadmin server role.

You  
can also create Credentials and Proxy accounts at this stage if you want your  
jobs to run SSIS packages or operating system commands under those accounts.

**Step  
8: Configure Error Logs**

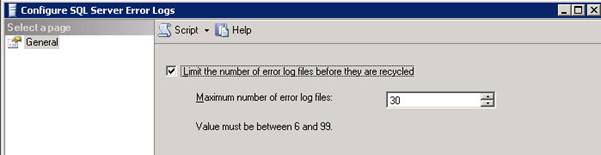
SQL  
Server error log is the first port of call for DBAs when they troubleshoot server  
related issues. By default, the server keeps six logs as archive and one log as  
current. Every time SQL service starts, the system begins a new log file. The  
currently active log file name is ERRORLOG. The log file before restart  
(previous ERRORLOG) becomes archived as ERRORLOG.1, the log file before that  
(named ERRORLOG.2) becomes ERRORLOG.3 and so on.

One  
of the annoying things with default error log behaviour is that any one file  
can contain a number of days of entries. When you try to view a file, it can  
take some time to load. If you are interested in a particular day’s events, you  
will need to sift through the entries or use a filter.

Also,  
log files are overwritten in a rolling fashion after six files are created. So  
that means you won’t have any history before the oldest log file (ERRORLOG.6).

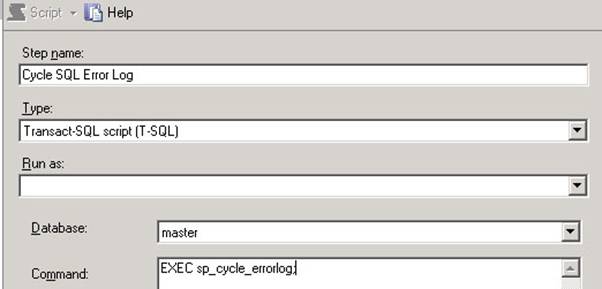
What  
you can do here is to configure SQL Server to keep a predefined number of log files.  
You can also create a scheduled job to “recycle” the error logs regularly.  
Recycling means that SQL Server will start a new log file without restarting  
the service.

To  
keep a pre-defined number of archived logs, you will need to configure the SQL error  
log properties.

**

As  
you can see, I would recommend keeping at least the last thirty log files.

Once  
the number of error log files is configured, you can create a scheduled job to  
run every day to reinitialise the log. The job will have only one step that  
will call the sp\_cycle\_errorlog system stored procedure. This procedure starts  
a new error log file without restarting the SQL service. I usually schedule it  
to run at 12:00 AM every morning.

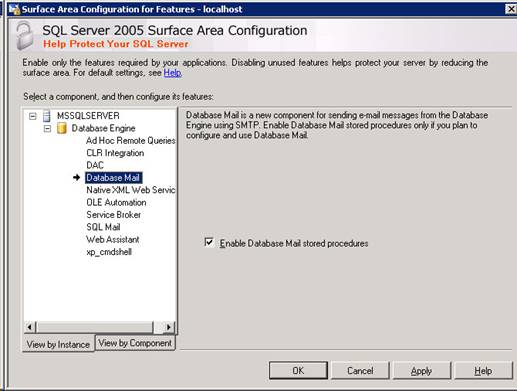
**

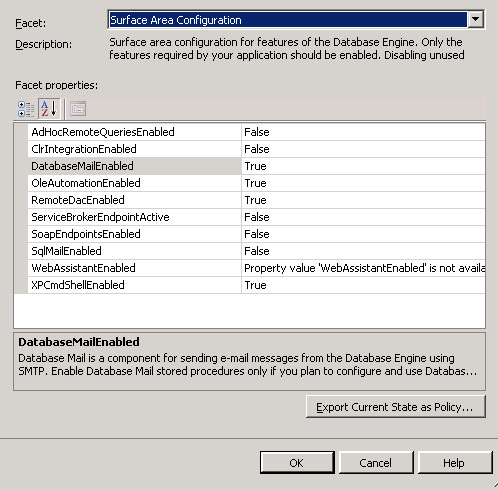
The  
combined effect of these two changes is that you will have one new log file  
created every day at midnight. If your server does not restart in between, you  
will have the last one month’s log entries in disk with each day’s log in one separate  
file.

The  
same kind of job can be defined for SQL Agent error log files. The latest Agent  
error log is named SQLAGENT.OUT. Executing the msdb database’s  
sp\_cycle\_agent\_errorlog procedure from a job will create a new Agent log file  
and archive the previous one. Unlike SQL Server error logs though, you cannot use  
a dialogue box to specify the number of archived Agent error logs to keep.

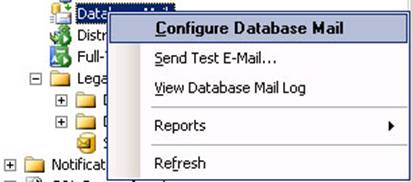
**Step  
9: Configure Database Mail**

This  
step is optional. If your databases are going to send e-mails, you will first need  
to enable database mail from the Surface Area Configuration tool (SQL 2005) or  
the Server Properties’ Surface Area Configuration Facet (for SQL 2008).

**

**

Once  
enabled, you can easily configure it using a wizard in the Management Studio.

**

You  
can also enable the mail profile for SQL Agent service.

If  
your apps are not going to use the mail feature – don’t enable or configure it.

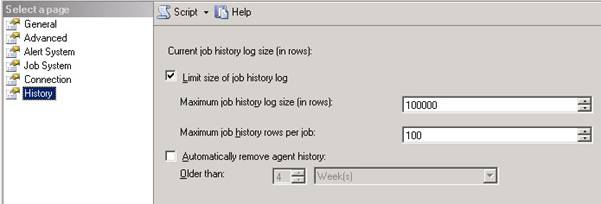
**Step  
10: Configure SQL Agent & Operator**

We  
are almost at the end of our configuration process. It is now worth having a look  
at the SQL Agent properties.  The three most important things that can be done here  
are:

a)     
Configuring  
SQL and Agent services to restart when they stop unexpectedly (discussed  
before)

b)     
Enabling  
the mail profile for SQL Agent. You can do it from the SQL Agent’s Alert System  
properties

c)     
Increasing  
the size of the job history log in the msdb database

**

Changing  
the size of the job history log in the msdb database is usually a good idea if  
your server is going to host a large number of scheduled jobs.

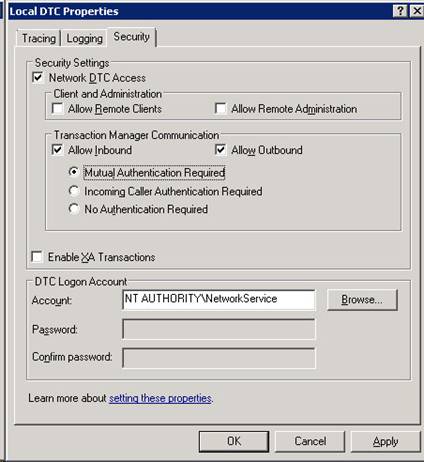
**Define  
DBA Operators:**

Every  
database server should have one standard DBA operator defined. This operator  
will be notified via e-mail when jobs fail or alerts fire. Do not create  
yourself as this operator – if you leave the company, your e-mail account is  
likely to be deleted and those who come after you will need to spend time for  
reconfiguring everything as the notifications will fail. That’s why you should  
ask your system administrators to create a standard DBA account and assign a  
mailbox to it.

**Step  
11: Configure Operating System**

You  
can also make some OS level changes to give some extra power to SQL. This works  
if you have administrative privilege on the Local Windows server.

If  
the database application is going to participate in distributed transactions  
through linked servers, ensure that Network DTC is enabled.

**

To  
reach this screen in a Windows Server 2003 system,

1.     
Start  
the Component Service applet (Start > Administrative Tools > Component  
Services)

2.     
Expand  
Component Services node and then Computers node in the left side pane

3.     
Right  
Click on My Computer and choose Properties

4.     
Choose  
the MS DTC tab

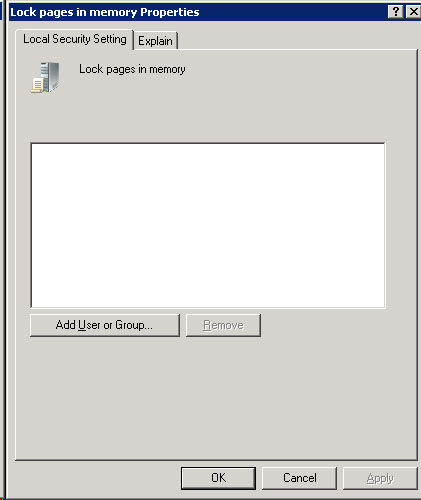
5.     
Click  
on Security Configuration screen

The  
steps are similar for Windows Server 2008

You  
should also give your SQL Service account the right to “Lock pages in memory”.  
This ensures that Windows will not swap out SQL pages from the memory on to the  
paging file on disk. In fact from version 2005, SQL Server can detect a memory  
pressure from OS signal and can dynamically allocate / de-allocate pages  
itself.

To  
configure this property, assign the SQL service account to this privilege from  
the Local Security Policy.

**



**Step  
12: Set up Maintenance Plans for System Databases**

Your  
server is now ready for production database rollouts. To wrap it up all, take a  
moment to create a maintenance plan for the system databases. This maintenance  
plan should reorganise the indexes, run DBCC CHECKS and update the statistics.  
It should also back up the master, model and the msdb databases.  As a best  
practice, the plan should backup the system databases once every day. When you  
have created the maintenance plan, run each of the tasks manually to see if  
they succeeded.

**Conclusion**

Although  
it is not possible to describe every possible installation scenario, the steps  
discussed here should be common in every installation. You should customise  
this list to incorporate any extra configuration necessary. For example, you  
may need to configure Reporting Services after SQL configuration is complete. Perhaps  
there are custom jobs or DBA stored procedures that you roll out in every SQL  
Server in the company. In each case, these extra steps will form parts of your  
post-installation configuration.

[General DBA](http://www.sql-server-performance.com/category/articles/dba/)

**6 Responses to “12 Essential Steps After Installing SQL Server”**

1. 

Henrik Staun Poulsen [Reply](http://www.sql-server-performance.com/2016/post-installation-steps/?replytocom=3501#respond) [June 4, 2013 at 12:54 pm](http://www.sql-server-performance.com/2016/post-installation-steps/#comment-3501)

I think I would also configure Instant file initialization

1. 

Viren [Reply](http://www.sql-server-performance.com/2016/post-installation-steps/?replytocom=3738#respond) [July 4, 2013 at 11:48 am](http://www.sql-server-performance.com/2016/post-installation-steps/#comment-3738)

Hi All, need your comments here. I have just did installation of SQL 2008 R2 SP2 and have not installed any CU reason being Microsoft does not recommend installing hotfix unless you encouter any specific issue. These guidelines are there with each CU so do you guys still suggest to install CU on fresh installation too??

————-

Hotfix Request  
Important  
•A hotfix is intended to correct a specific problem.  
•Apply the hotfix only to systems that are experiencing the specific problem.  
•Installing the incorrect hotfix can cause damage to your system.  
•If you are not sure whether the hotfix is the correct one for your system, do not install it.  
•Hotfixes are included in subsequent service packs that are safer to install through Microsoft Update.

<http://support.microsoft.com/hotfix/KBHotfix.aspx?kbnum=2797460&kbln=en-us>

1. 

Sadequl Hussain [Reply](http://www.sql-server-performance.com/2016/post-installation-steps/?replytocom=3832#respond) [August 6, 2013 at 7:14 am](http://www.sql-server-performance.com/2016/post-installation-steps/#comment-3832)

Hi Viren,

I believe CUs are all rolled up into SPs when they are released, while hotfixes are problem specific. I have installed SQL Server 2008, 2008 R2 and applied CUs when no SPs existed for them (or have applied latest SP + latest CU). Nothing ever broke in the servers I installed them on.

1. 

Kaunab naprico [Reply](http://www.sql-server-performance.com/2016/post-installation-steps/?replytocom=6156#respond) [May 25, 2014 at 5:07 pm](http://www.sql-server-performance.com/2016/post-installation-steps/#comment-6156)

The following two links given in the article are not valid.

Configuring\_Default\_Directories.sql  
Configuring\_tempdb\_Data\_Files.sql

would appreciate if linkage is corrected. Thanks

1. 

Golam Kabir [Reply](http://www.sql-server-performance.com/2016/post-installation-steps/?replytocom=28619#respond) [June 25, 2014 at 8:11 pm](http://www.sql-server-performance.com/2016/post-installation-steps/#comment-28619)

Is a very good one – thanks and keep posting

1. 

Eric [Reply](http://www.sql-server-performance.com/2016/post-installation-steps/?replytocom=431795#respond) [November 22, 2014 at 6:16 am](http://www.sql-server-performance.com/2016/post-installation-steps/#comment-431795)

I just think disable of VSS Writer of SQL Server is a rather risky choice.

In VSS architecture, any VSS writer is being called upon indirectly by every Backup program which try to take a snapshot of the involved drive. without the VSS Writer, the DB file image as such backup program taken may not be consistent as SQL Server is not participated in it.