Why Does the Transaction Log Keep Growing or Run Out of Space?

This one seems to be a common question in most forums and all over the web, it is asked here in many formats that typically sound like this:

In SQL Server -

- What are some reasons the transaction log grows so large?
- Why is my log file so big?
- What are some ways to prevent this problem from occurring?
- What do I do when I get myself on track with the underlying cause and want to put my transaction log file to a healthy size?

A Shorter Answer:
You probably either have a long running transaction running (Index maintenance? Big batch delete or update?) or you are in the "default" (more below on what is meant by default) recovery mode of Full and have not taken a log backup (or aren't taking them frequently enough).

There could be other reasons but these are the most common. This answer begins to dive into the most common two reasons and gives you some background information on the why and how behind the reasons as well as explores some other reasons.

A Longer Answer:

What Scenarios can Cause the Log to Keep Growing?

There are many reasons, but usually these reasons are of the following two patterns:

1.) Not Understanding Recovery Models

(**Being in Full Recovery Mode and Not Taking Log Backups - This is the most common reason - the vast majority of those experiencing this issue are.**)
This answer is not a deep dive in SQL Server recovery models, the topic of recovery models is critical to this problem. In SQL Server, there are three recovery models - `Full`, `Bulk-Logged` and `Simple`. We'll ignore `Bulk-Logged` for now we'll sort of say it is a hybrid model and most people who are in this model are there for a reason and understand recovery models. The two we care about and their confusion are the cause of the majority of the cases of people having this issue are `Simple` and `Full`.
Before we talk about Recovery Models - Let's talk about recovery in general. If you want to go even deeper with this topic, just read Paul Randal's blog and as many posts on it as you want. For this question, though:

1. The transaction log file is there for crash/restart recovery. For the rolling forward and rolling back of work that was either done (rolling forward/redo) before a crash or restart and the work that was started but not finished after a crash or restart (rolling back/undo). It is the job of the transaction log to see that a transaction started but never finished (rolled back or crash/restart happened before the transaction committed). In that situation It is the log's job to say "hey.. this never really finished, let's roll it back" during recovery. It is also the log's job to see that you did finish something and that your client application was told it was finished (even if it hadn't yet hardened to your data file) and say "Hey.. this really happened, let's roll it forward, let's make it like the applications think it was" after a restart. Now there is more but that is the main purpose.

2. The other purpose for a transaction log file is to be able to give us the ability to recover to a point in time due to an "oops" in a database or to guarantee a recovery point in the event of a hardware failure involving the data and/or log files of a database. If this transaction log contains the records of transactions that have been started and finished for recovery, SQL Server can and does then use this information to get a database to where it was before an issue happened. But that isn't always an available option for us. For that to work we have to have our database in the right recovery model, and we have to take log backups.

Onto the recovery models:
Simple Recovery Model - So with the above introduction, it is easiest to talk about `Simple Recovery`model first. In this model, you are telling SQL Server - I am fine with you using your transaction log file for crash and restart recovery (You really have no choice there.. Look up ACID properties and that should make sense quickly), but once you no longer need it for that crash/restart recovery purpose, go ahead and reuse the log file.
SQL Server listens to this request in Simple Recovery and it only keeps the information it needs to do crash/restart recovery. Once SQL Server is sure it can recover because data is hardened to the data file (more or less), the data that has been hardened is no longer necessary in the log and is marked for truncation - which means it gets re-used.

Full Recovery Model - With Full Recovery, you are telling SQL Server that you want to be able to recover to a specific point in time, as long as your log file is available or to a specific point in time that is covered by a log backup. In this case when SQL Server reaches the point where it would be safe to truncate the log file in Simple Recovery Model, it will not do that. Instead **It lets the log file continue to grow** and will allow it to keep growing, ***until you take a log backup*** (or run out of space on your log file drive) under normal circumstances.
There are rules and exceptions here: We'll talk about long running transactions below, but one to keep in mind for in full mode - If you just switch into Full Recovery mode, but never take an initial Full Backup, SQL Server will not honor your request to be in Full Recovery model. Your transaction log will continue to operate as it has in simple until you switch to Full Recovery Model **AND** Take your first Full Backup.
So, that's the most common reason for uncontrolled log growth: *Being in Full Recovery mode without having any log backups.* This happens **all** the time to people.
(***Why does it happen all the time?*** *Because each new database gets its initial recovery model setting by looking at the model database. Model's initial recovery model setting is always Full Recovery Model - until and unless someone changes that. So you could say the "default Recovery*

Model" is Full. Many people are not aware of this and have their databases running in Full Recovery Model with no log backups, and therefore a transaction log file much larger than necessary. This is why it is important to change defaults when they don't work for your organization and its needs)
You can also get yourself in trouble here by not taking log backups frequently enough. - Taking a log backup a day may sound fine, it makes a restore require less restore commands, but keeping in mind the discussion above, that log file will continue to grow and grow until you take log backups. You need to consider your log backup frequency with two things in mind:

1. **Recovery Needs** - This should hopefully be first. In the event that the drive housing your transaction log goes bad or you get serious corruption that affects your log backup, how much data can be lost? If that number is no more than 10-15 minutes, then you need to be taking the log backup every 10-15 minute, end of discussion.
2. **Log Growth** - If your organization is fine to lose more data because of the ability to easily recreate that day you may be fine to have a log backup much less frequently than 15 minutes. Maybe your organization is fine with every 4 hours. But you have to look at how many transactions you generate in 4 hours. Will allowing the log to keep growing in those four hours make too large of a log file? Will that mean your log backups take too long?

2.) Long Running Transactions

(**"My recovery model is fine! The log is still growing!**)
This can also be a cause of uncontrolled and unrestrained log growth. No matter the recovery model, but it often comes up as "But I'm in Simple Recovery Model - why is my log still growing?!"

The reason here is simple, if SQL is using this transaction log for recovery purposes as I described above, then it has to see back to the start of a transaction. If you have a transaction that takes a long time or does a lot of changes, the log cannot truncate on checkpoint for any of the changes that are still in open transactions or that have started since that transaction started.

This means that a big delete, deleting millions of rows in one delete statement is one transaction and the log cannot do any truncating until that whole delete is done. In Full Recovery Model, this delete is logged and that could be a lot of log records. Same thing with Index optimization work during maintenance windows. It also means that poor transaction management and not watching for and closing open transactions can really hurt you and your log file.

You can save yourself here by: - Properly sizing your log file to account for the worst case scenario - like your maintenance or known large operations. And when you grow your log file you should look to thisguidance (and the two links she sends you to) by Kimberly Tripp. Right sizing is super critical here. - Watching your usage of transactions. Don't start a transaction in your application server and start having long conversations with SQL Server and risk leaving one open too long. - Watch your implied transactions in your DML statements. `UPDATE TableName Set Col1 = 'New Value'` is a transaction. I didn't put a `BEGIN TRAN` there and I don't have to, it is one transaction that just automatically commits when done. So if doing operations on large numbers of rows, consider batching those operations up into more manageable chunks and giving the log time to recover. Or consider the right size to deal with that, or perhaps look into changing recovery models during a bulk load window.

What about Log Shipping?

"I'm using log shipping, so my log backups are automated... Why am I still seeing transaction log growth?"
Log shipping is just what it sounds like - you are shipping your transaction log backups to another server for DR purposes. The process is fairly simple - after the initialization - a job to back up the log on one server, a job to copy that log backup and a job to restore it without recovery (either `NORECOVERY` or`STANDBY`) on the destination server. There are also some jobs to monitor and alert if things don't go as you have them planned.
In some cases, you may only want to do the log shipping restore once a day or every 3rd day or once a week. That is fine. But if you make this change on all of the jobs (including the log backup and copy jobs) that means you are waiting all that time to take a log backup. That means you will have a lot of log growth - because you are in full recovery mode without log backups, and it also likely means a large log file to copy across. You should only modify the restore job's schedule and let the log backups and copies happen on a more frequent basis, otherwise you will suffer from the first issue described in this answer.

Getting Info on Your Cause

There are reasons other than these two, but these are the most common. Regardless of the cause - there is a way you can analyze your reason for this unexplained log growth/lack of truncation and see what they are:

By Querying the `[sys.databases][5]` catalog view you can see information describing the reason your log file may be waiting on truncate/reuse. There is a column called `log_reuse_wait`with a lookup ID of the reason code and a `log_reuse_wait_desc` column with a description of the wait reason. From the referenced books online article are the majority of the reasons (the ones you are likely to see and the ones we can explain reasons for. The missing ones are either out of use or for internal use) with a few notes about the wait in *italics*:

```
Select
name,database_id,user_access_desc,state_desc,recovery_model_desc,log_reuse_wa
it,log_reuse_wait_desc
From Sys.databases
where name like 'CNG%'
```

- 0 = Nothing - *What it sounds like.. Shouldn't be waiting*
- 1 = Checkpoint - *Waiting for a checkpoint to occur. This should happen and you should be fine - but there are some cases to look for here for later answers or edits.*
- 2 = Log backup - *You are waiting for a log backup to occur. Either you have them scheduled and it will happen soon, or you have the first problem described here and you now know how to fix it*
- 3 = Active backup or restore - *A backup or restore operation is running on the database*
- 4 = Active transaction - * There is an active transaction that needs to complete (either way - `ROLLBACK` or `COMMIT`) before the log can be backed up. This is the second reason described in this answer.
- 5 = Database mirroring *Either a mirror is getting behind or under some latency in a high performance mirroring situation or mirroring is paused for some reason*

- 6 = Replication - *There can be issues with replication that would cause this - like a log reader agent not running, a database thinking it is marked for replication that no longer is and various other reasons. You can also see this reason and it is perfectly normal because you are looking at just the right time, just as transactions are being consumed by the log reader*
- 7 = Database snapshot creation *You are creating a database snapshot, you'll see this if you look at just the right moment as a snapshot is being created*
- 8 = Log Scan *I have yet to encounter an issue with this running along forever. If you look long enough and frequently enough you can see this happen, but it shouldn't be a cause of excessive transaction log growth, that I've seen.*
- 9 = An AlwaysOn Availability Groups secondary replica is applying transaction log records of this database to a corresponding secondary database. *About the clearest description yet..*

Credit: Mike Walsh

http://dba.stackexchange.com/questions/29829/why-does-the-transaction-log-keep-growing-or-run-out-of-space

Manage the Size of the Transaction Log File

http://msdn.microsoft.com/en-us/library/ms365418.aspx

SHRINK Transaction Log file in SQL for CNG_Main database

Step 1: Login to SQL Instance that contains CNG_Main Database
Step 2: Make Full Database backup (SQL Instance>Database>CNG_Main>Tasks>Back Up…)
Step 2: Create a new query
Step 3: Paste the following code in the query window

```
-- START SHRINK Query
USE CNG_Main;
GO
-- Truncate the log by changing the database recovery model to SIMPLE.
ALTER DATABASE CNG_Main
SET RECOVERY SIMPLE;
GO
-- Shrink the truncated log file to 2 MB.
DBCC SHRINKFILE (CNG_Main_Log, 2);
GO
-- Reset the database recovery model.
ALTER DATABASE CNG_Main
SET RECOVERY FULL;
GO
-- END SHRINK Query
```

Step 4: Edit the code if necessary for the Database name, log name and the shrink value.

Please look at database file settings below for additional information.

NOTE: Use caution when running this or any query. Always make a backup of the database before you run any query. If in doubt ask for help.

Database and transaction log backups using scripts or from SQL Management Studio.

A few minutes spent on a backup, verification and a recovery plan will go a long way in keeping your sanity when you actually need your backed up data.

Why Backup?:

Two main reasons:

1) Disaster recovery
2) Prevent transaction logs from getting large (For Recovery Mode: Full)

What to Backup?:

SQL Data:

We strongly urge you to make database backup for:

1. Main database (generally named CNG_Main)
2. Forms database (Always named CNGForms)
3. Transaction Log files for both #1 and #2 (For Recovery Mode: Full)

CNG Repository Data:

The CNG document files are stored on the hard drive separate from SQL. These have to be backed up in synch with the SQL backup. (The repository data backup procedure is not covered in this article, but should be pretty straight forward to implement.)

Backup Verification:

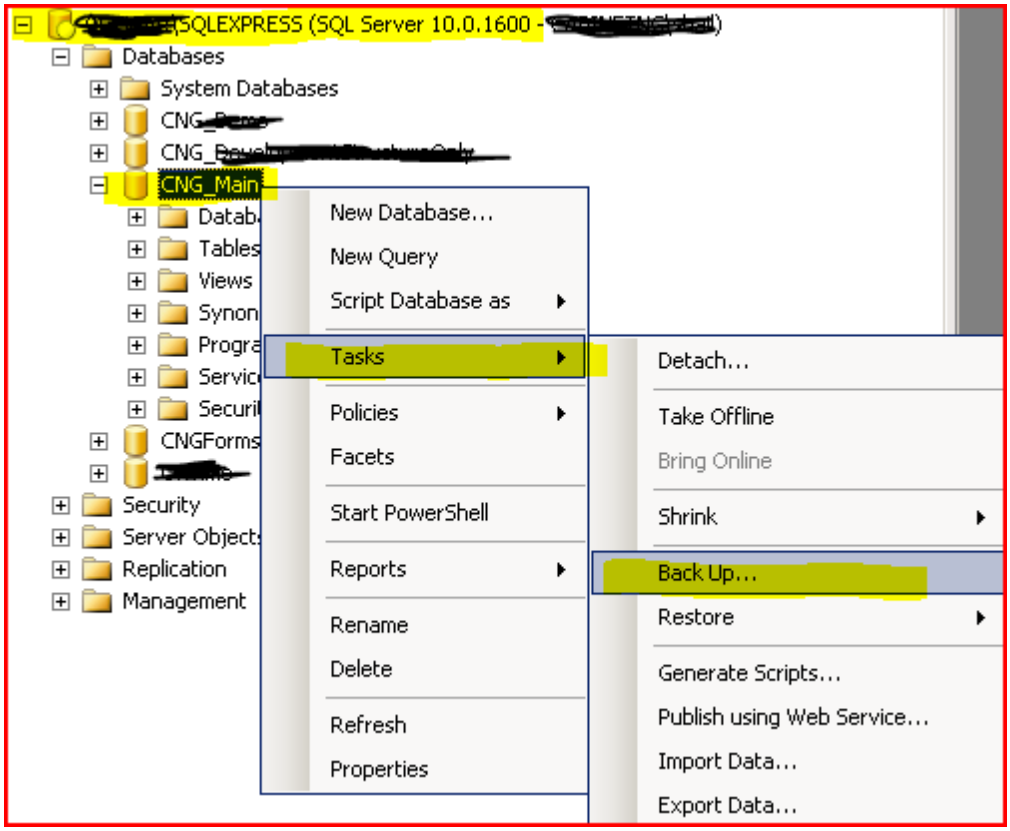
Once your backup process is finalized: scripts are generated and scheduled to run, or a backup job is created in SQL\ backup utility, It is very essential to test the process to see if it is backing up your data correctly. It is also a good idea to periodically check this process, as settings, passwords or data locations may change over time. It is also a good idea to have a backup in a location different from the original data(CNG & SQL) location.

Methodology:

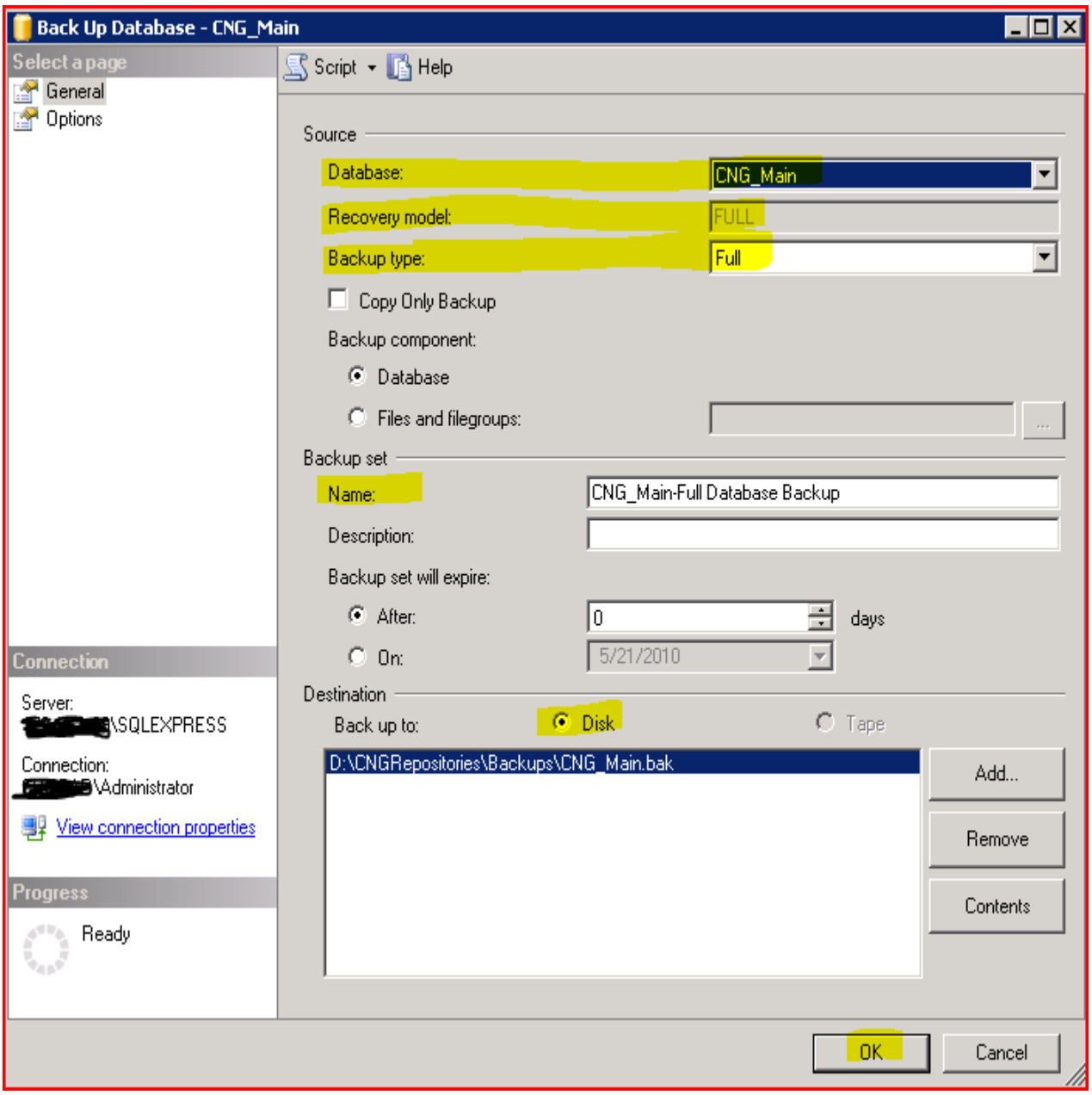
Databases and transaction log files can be backed up using a variety of ways. **Your IT personnel should help you set these up**. Methods and steps listed below should give you an idea of what databases need to be backed up and how it can be done for Recover Mode: Full

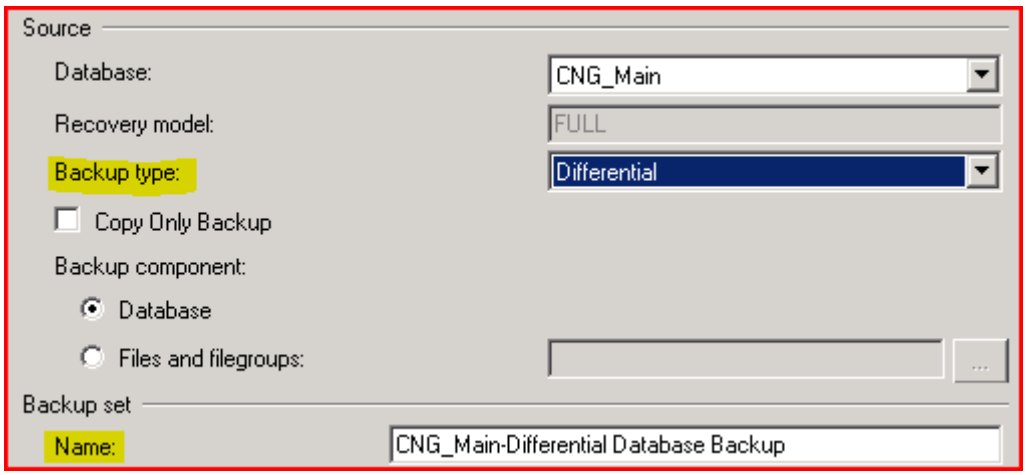
1) **SQL Management studio**: Steps are as follows…
   Step 1: Select Server Instance

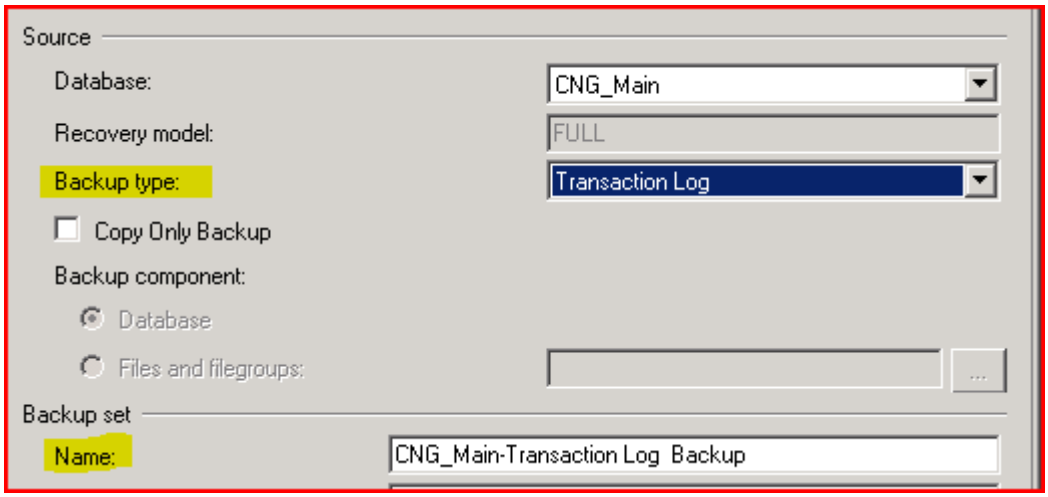
   Step 2: Select Databases
   Step 3: Select Database of interest (CNG_Main or CNGForms)
   Step 4: Right Click database (CNG_Main)
   Step 5: Select Tasks
   Step 6: Select Backup…
   Step 7: Configure backup (3 times for each of the following…)
      a) Full
      b) Differential (if you want to do this)
      c) Transaction Log
   Figures:



Note: Read about Recovery Model at the end of this write up.


Back Up Database - CNG_Main
Select a page
General
Options
Script
Help
Source
Database:
CNG_Main
Recovery model:
FULL
Backup type:
Full
Copy Only Backup
Backup component:
Database
Files and filegroups:
Backup set
Name:
CNG_Main-Full Database Backup
Description:
Backup set will expire:
After:
0
days
On:
5/21/2010
Destination
Back up to:
Disk
Tape
D:\CNGRepositories\Backups\CNG_Main.bak
Add...
Remove
Contents
Connection
Server:
[redacted]\SQLEXPRESS
Connection:
[redacted]\Administrator
View connection properties
Progress
Ready
OK
Cancel

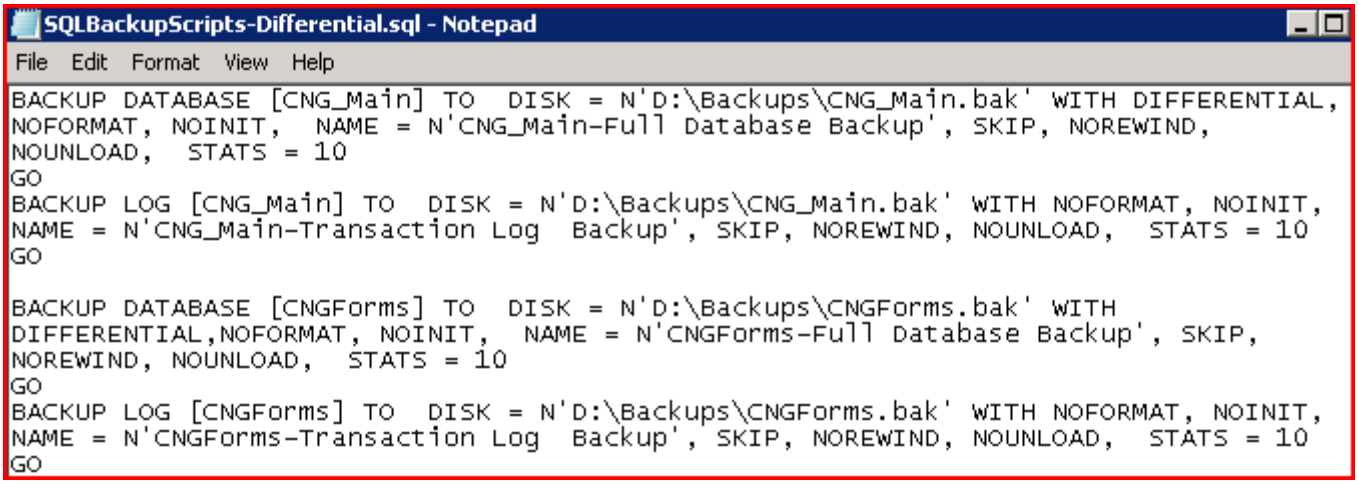




2) **External Back Agent**. Use instructions that come with it. Backup the following:
   a) Databases CNG_Main and CNGForms
   b) Transaction logs for both the databases

3) **Running script on task scheduler**:
   a) **Scripts**: The backup can be done with different backup types. One suggestion would be to write a script for full, Sequential and transaction log. Then run the sequential everynight of the week and the full backup once every week over the weekend. (Sample screen shots are provided for both tasks and scripts) NOTE: The scripts provided include both the main database (CNG_Main) and the forms database (CNGForms). Each script will include the backup of both database and transaction log.

Differential Backup Script:



Script to Copy and paste. (You can create a text file and then do a file saveas to save as a .sql file.)

######### START: **DIFFERENTIAL** - Edit the path and database name if necessary. File name: **SQLBackupScripts-Differential.sql** #############

BACKUP DATABASE [CNG_Main] TO DISK = N'D:\Backups\CNG_Main.bak' WITH DIFFERENTIAL, NOFORMAT, NOINIT, NAME = N'CNG_Main-Full Database Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

GO

BACKUP LOG [CNG_Main] TO DISK = N'D:\Backups\CNG_Main.bak' WITH NOFORMAT, NOINIT, NAME = N'CNG_Main-Transaction Log Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

GO

BACKUP DATABASE [CNGForms] TO DISK = N'D:\Backups\CNGForms.bak' WITH DIFFERENTIAL,NOFORMAT, NOINIT, NAME = N'CNGForms-Full Database Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

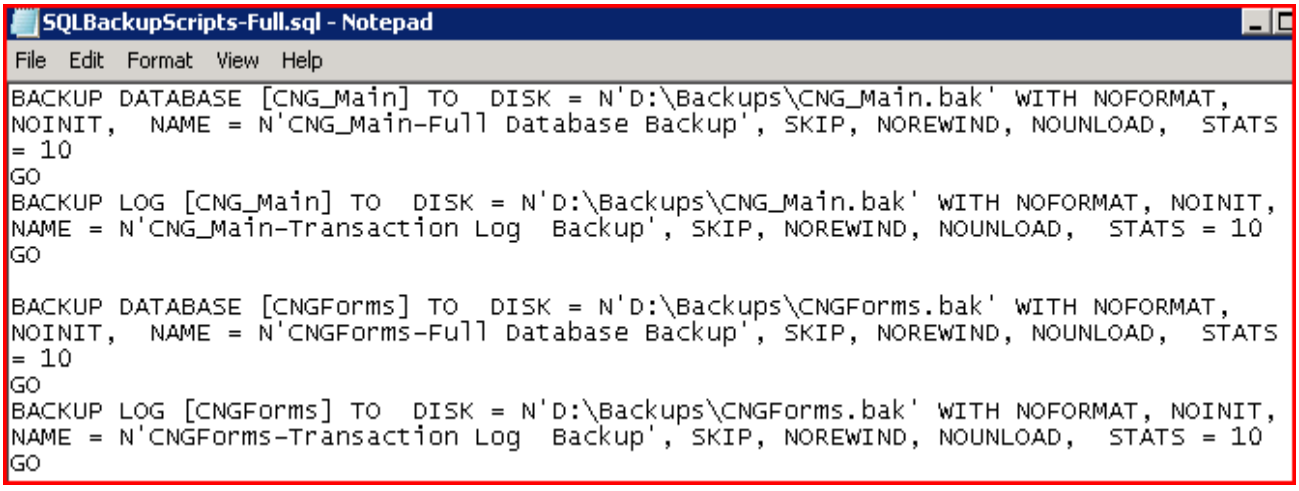
GO

BACKUP LOG [CNGForms] TO DISK = N'D:\Backups\CNGForms.bak' WITH NOFORMAT, NOINIT, NAME = N'CNGForms-Transaction Log Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

GO

######### END DIFFERENTIAL Script to Copy and paste. ##

Full Back up Script:



Script to Copy and paste. (You can create a text file and then do a file saveas to save as a .sql file.)

######### START: **FULL** - Edit the path and database name if necessary File name: **SQLBackupScripts-Full.sql** #############

BACKUP DATABASE [CNG_Main] TO DISK = N'D:\Backups\CNG_Main.bak' WITH NOFORMAT, NOINIT, NAME = N'CNG_Main-Full Database Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

GO

BACKUP LOG [CNG_Main] TO DISK = N'D:\Backups\CNG_Main.bak' WITH NOFORMAT, NOINIT, NAME = N'CNG_Main-Transaction Log Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

GO

BACKUP DATABASE [CNGForms] TO DISK = N'D:\Backups\CNGForms.bak' WITH NOFORMAT, NOINIT, NAME = N'CNGForms-Full Database Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

GO

BACKUP LOG [CNGForms] TO DISK = N'D:\Backups\CNGForms.bak' WITH NOFORMAT, NOINIT, NAME = N'CNGForms-Transaction Log Backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

GO

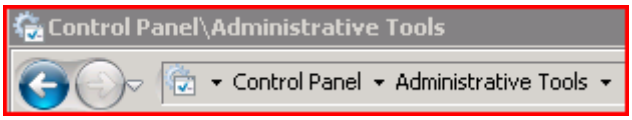
######### END FULL Script to Copy and paste.
##

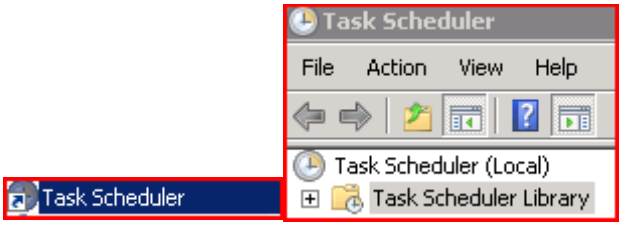
b) **Task Scheduler**:

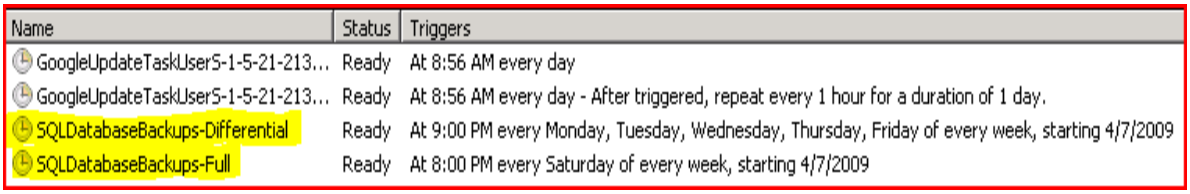
Once the scripts are written they need to be run via the task scheduler. Following steps will help you configure the tasks.

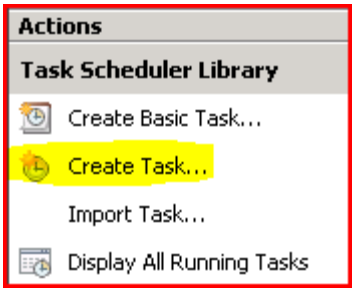
1. Start Task Scheduler
2. Create new Task
3. Configure the tasks

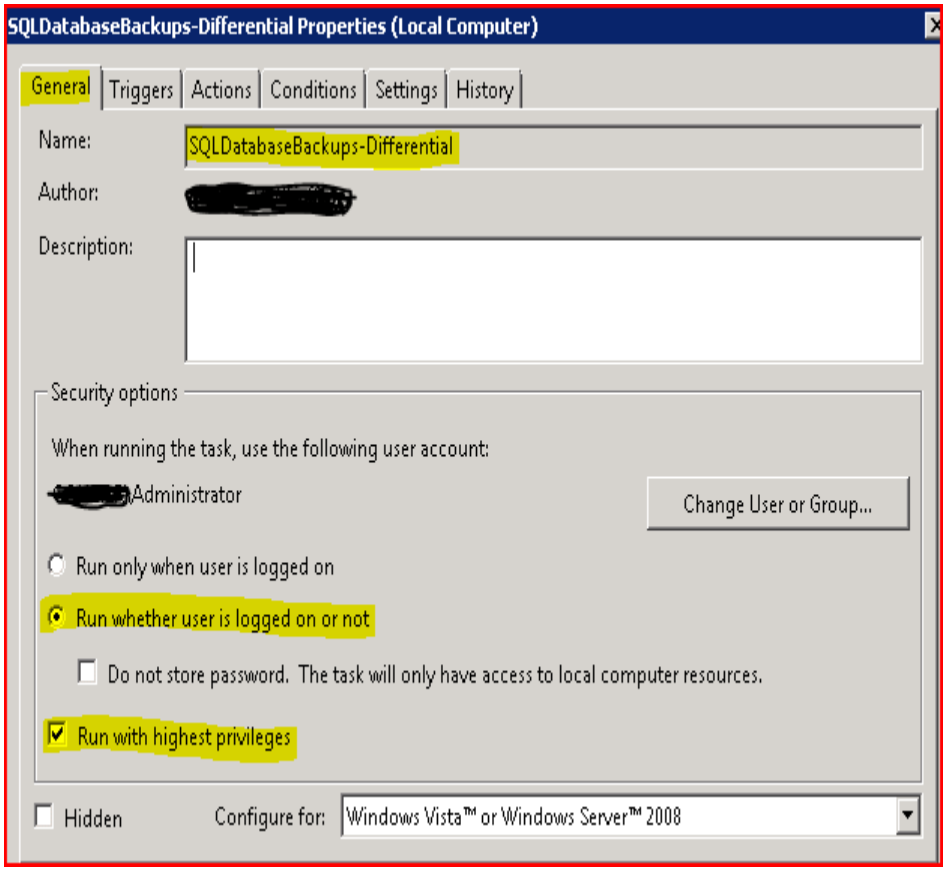
Note: Please change the path to the program appropriately and set the correct SQL username and password where needed.

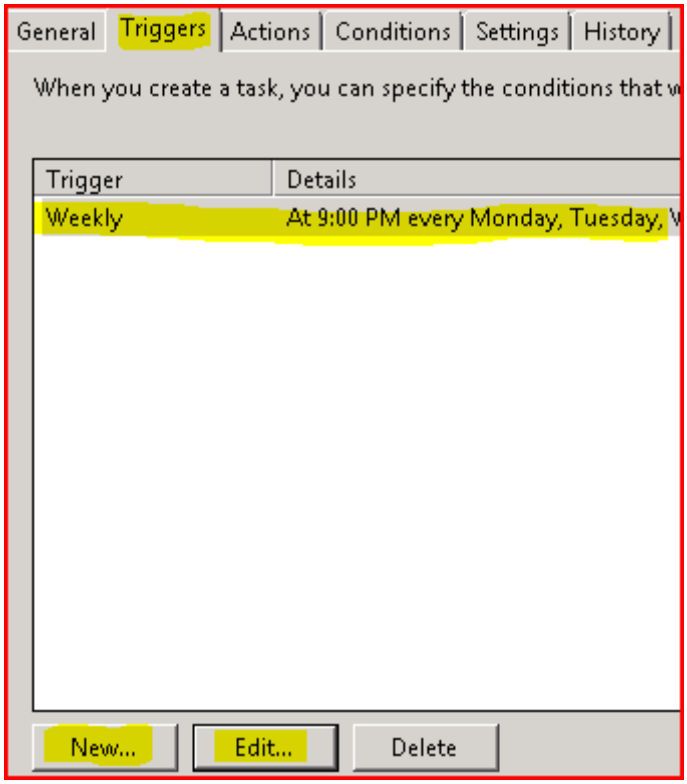




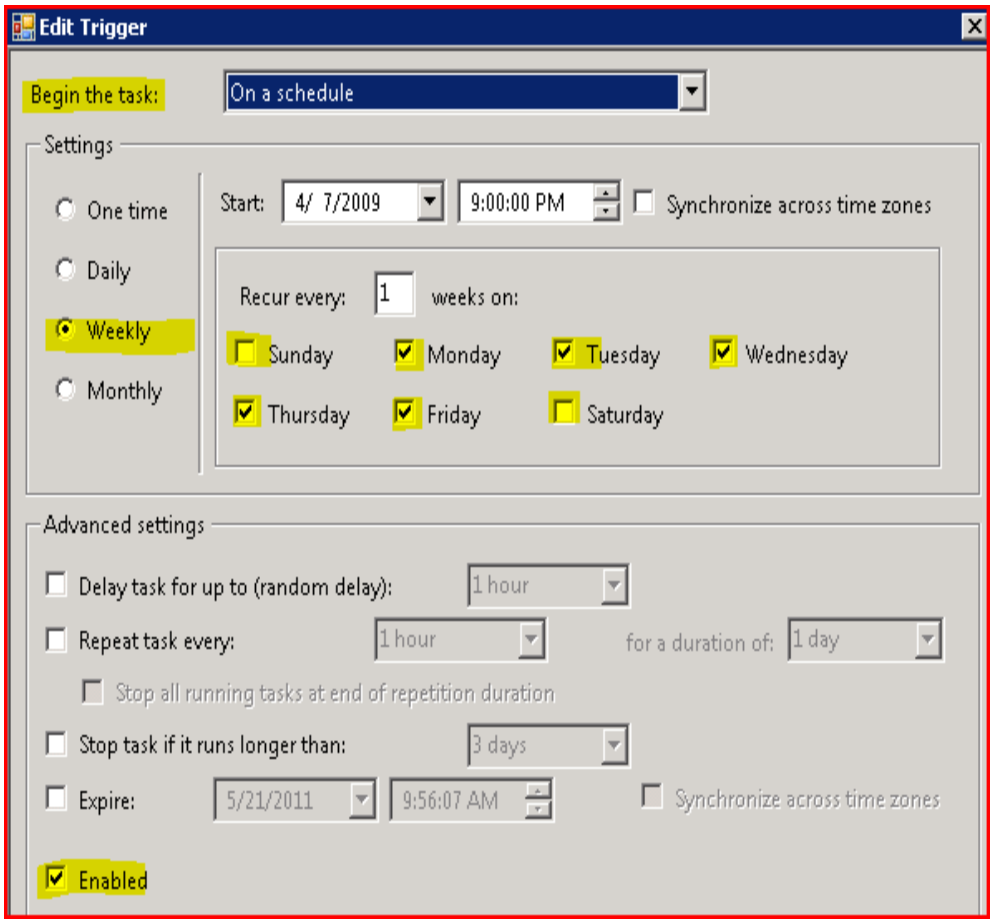




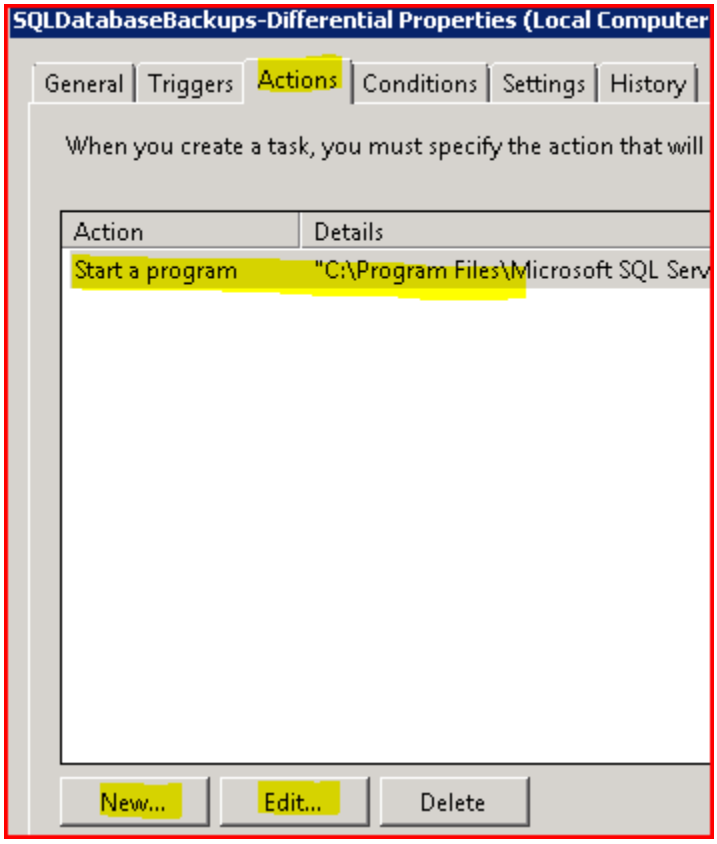

SQLDatabaseBackups-Differential Properties (Local Computer)
General
Triggers
Actions
Conditions
Settings
History
Name:
SQLDatabaseBackups-Differential
Author:
Description:
Security options
When running the task, use the following user account:
Administrator
Change User or Group...
Run only when user is logged on
Run whether user is logged on or not
Do not store password. The task will only have access to local computer resources.
Run with highest privileges
Hidden
Configure for:
Windows Vista™ or Windows Server™ 2008


General
Triggers
Actions
Conditions
Settings
History
When you create a task, you can specify the conditions that
Trigger
Details
Weekly
At 9:00 PM every Monday, Tuesday,
New...
Edit...
Delete

For Differential, set to week days only.



☺ You are doing good. Just a little more to go…



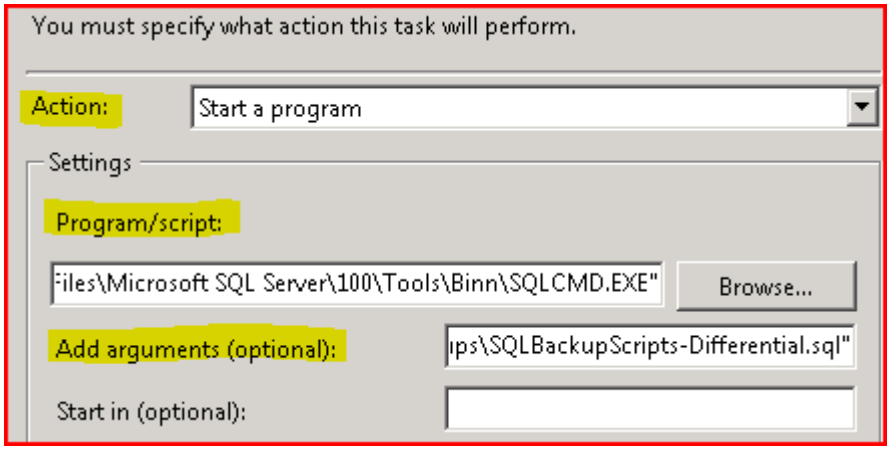
The paths may be different on your machine depending on the location of install and SQL version.

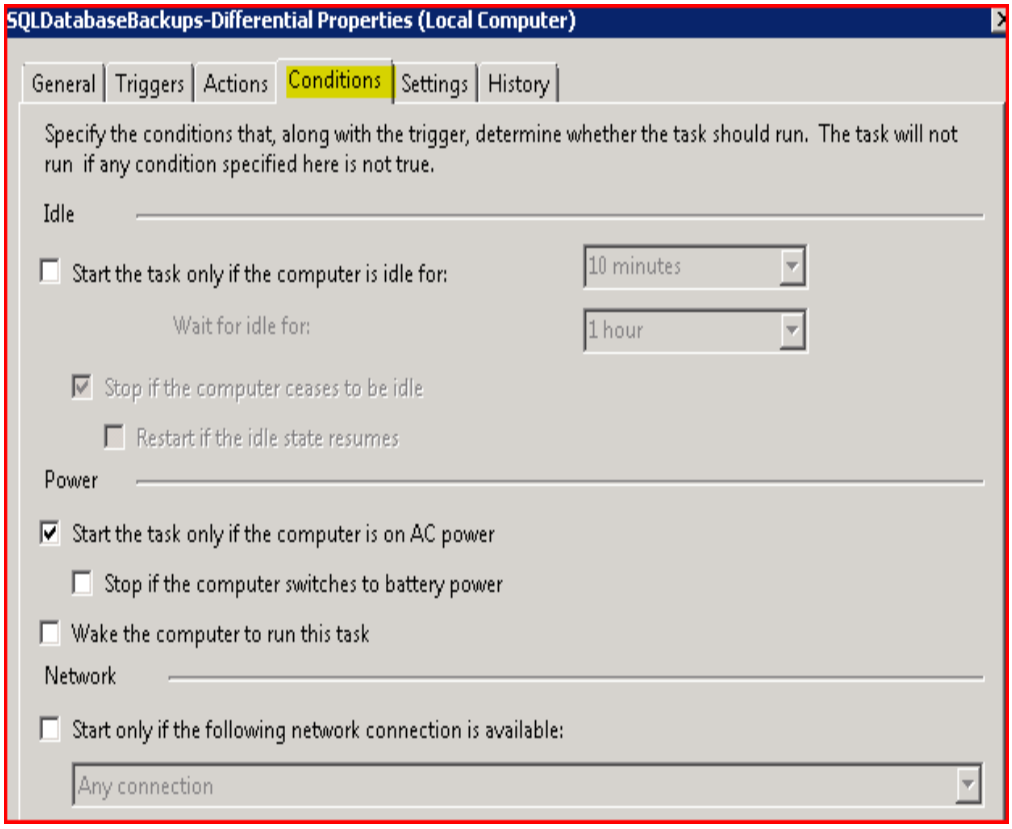
Arguments are CASE SENSITIVE.

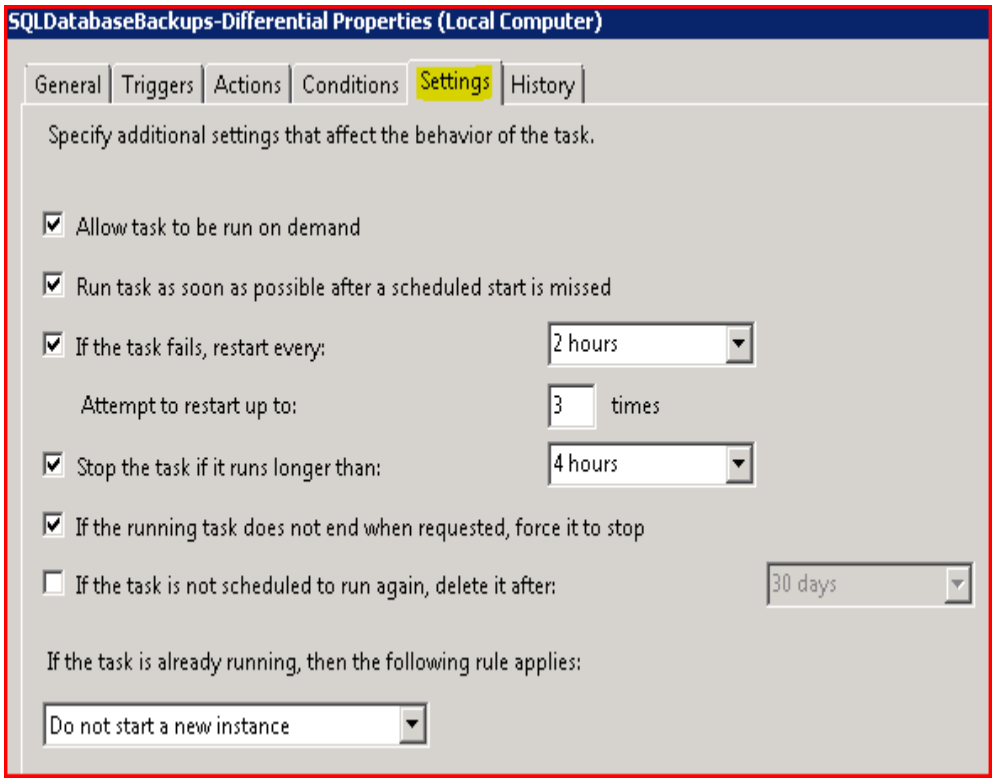
Program/Script: **"C:\Program Files\Microsoft SQL Server\100\Tools\Binn\SQLCMD.EXE"**

Add arguments (Optional): **-S \SQLEXPRESS -U username -P password -i "D:\Backups\SQLBackupScripts-Differential.sql"**

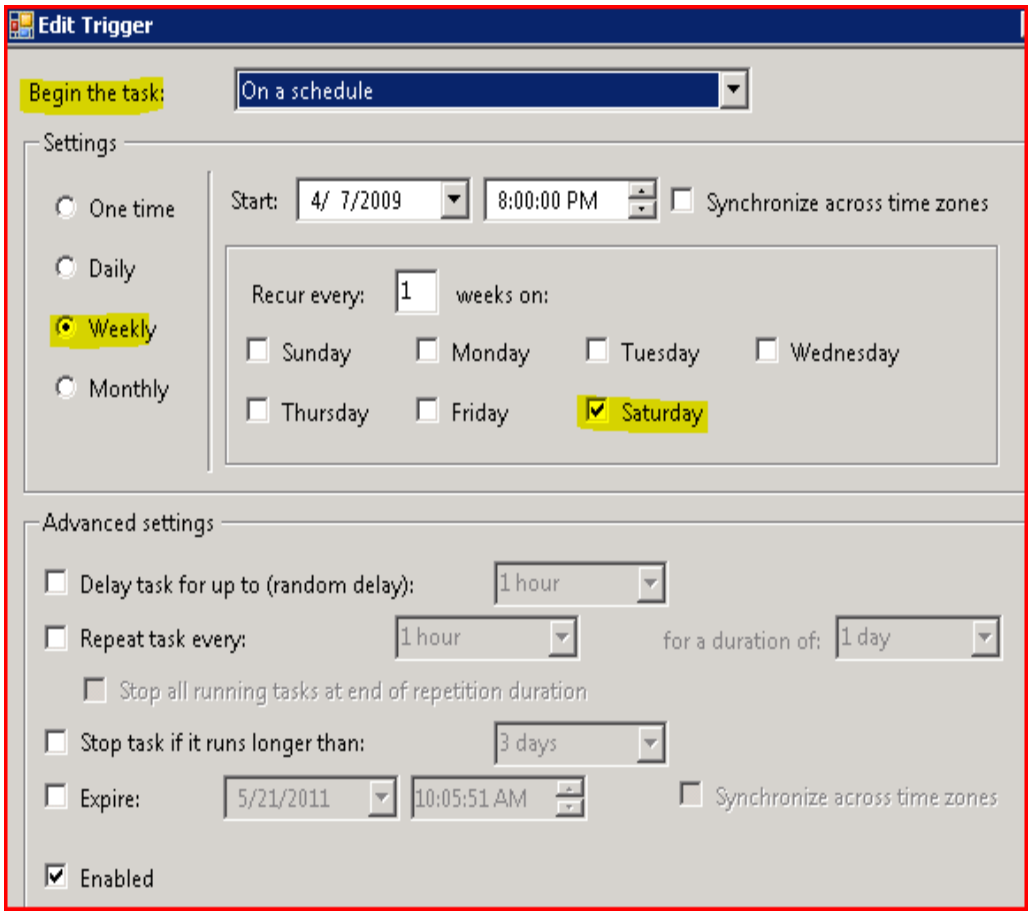
Note: **\SQLEXPRESS can be anything. It is just the name of the SQL Server instance**

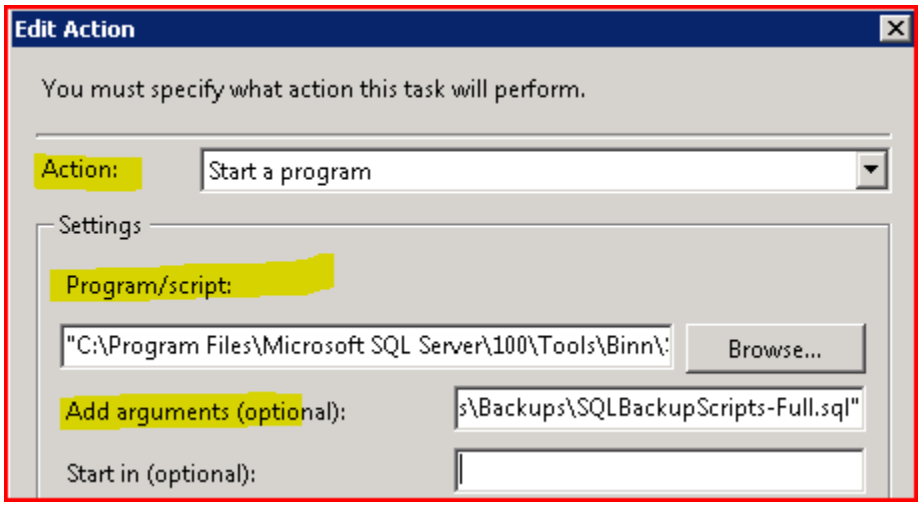

You must specify what action this task will perform.
Action:
Start a program
Settings
Program/script:
Files\Microsoft SQL Server\100\Tools\Binn\SQLCMD.EXE"
Browse...
Add arguments (optional):
ips\SQLBackupScripts-Differential.sql"
Start in (optional):


SQLDatabaseBackups-Differential Properties (Local Computer)
General
Triggers
Actions
Conditions
Settings
History
Specify the conditions that, along with the trigger, determine whether the task should run. The task will not run if any condition specified here is not true.
Idle
Start the task only if the computer is idle for:
10 minutes
Wait for idle for:
1 hour
Stop if the computer ceases to be idle
Restart if the idle state resumes
Power
Start the task only if the computer is on AC power
Stop if the computer switches to battery power
Wake the computer to run this task
Network
Start only if the following network connection is available:
Any connection

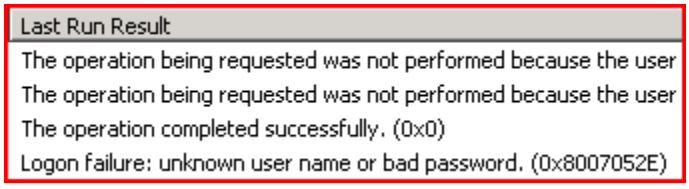


Full Backup….Same as sequential with the exception of the Trigger and the Script to run.





Finally…Test your scheduled task to see if it runs correctly



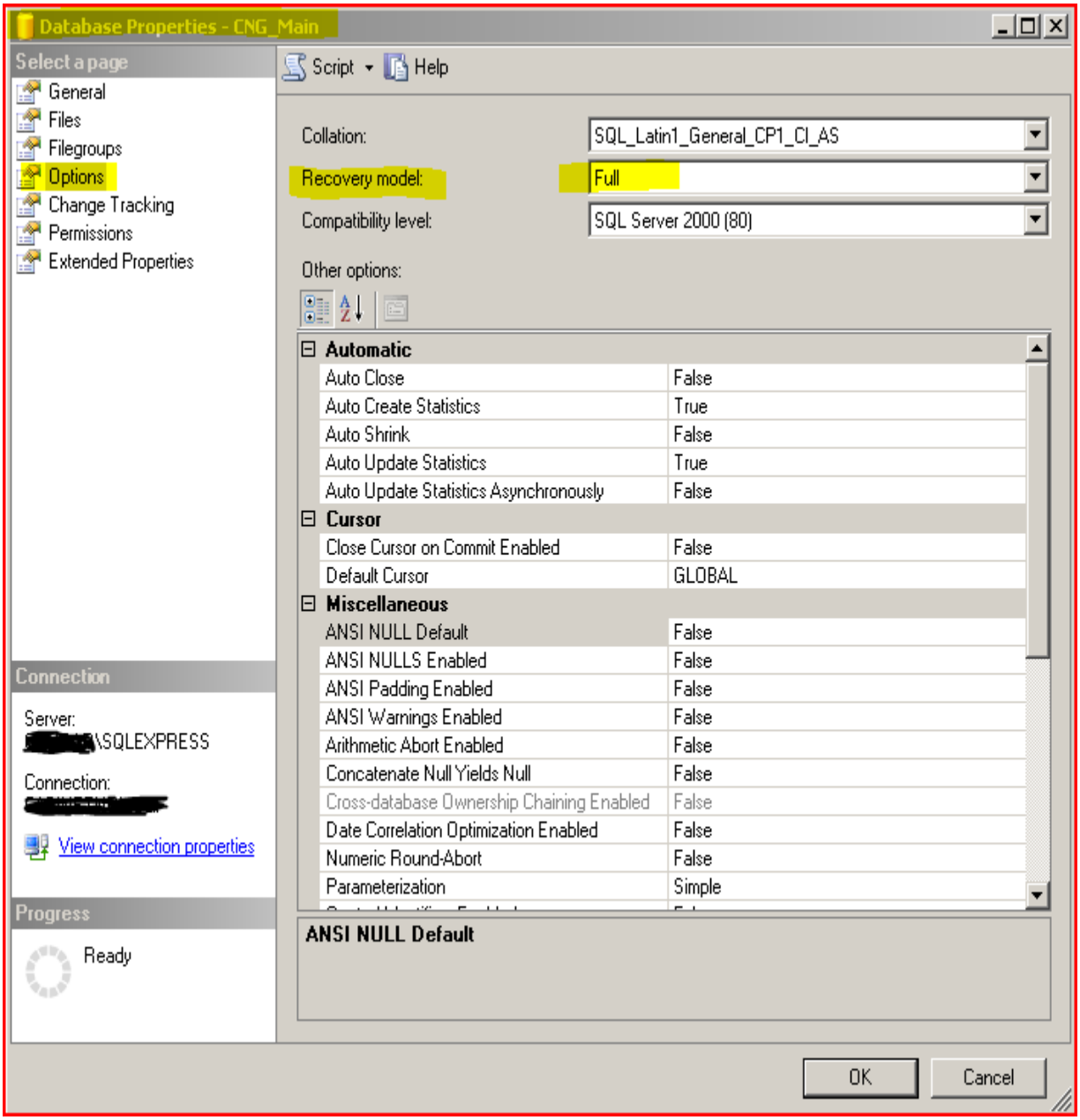
You would like to see “The operation completed successfully.” At this point if the backup file has been created and if it already exists, check its modified date time stamp.

Default outputs will look like this…

Recovery Model:

This can be set by doing the falling:

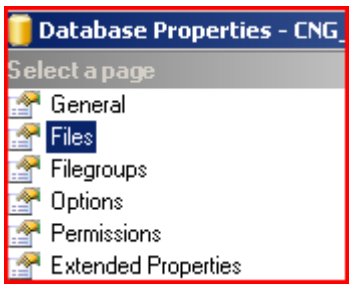
1. Select Database (CNG_Main)
2. Right Click and Select Properties.
3. Select Options
4. Set Recovery Model: to Full or Simple based on your needs.

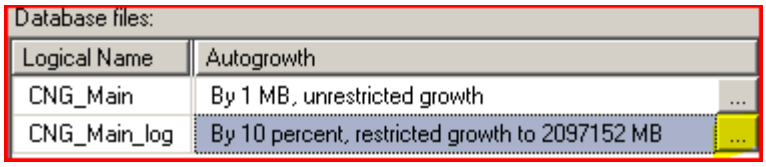


Database File Settings:

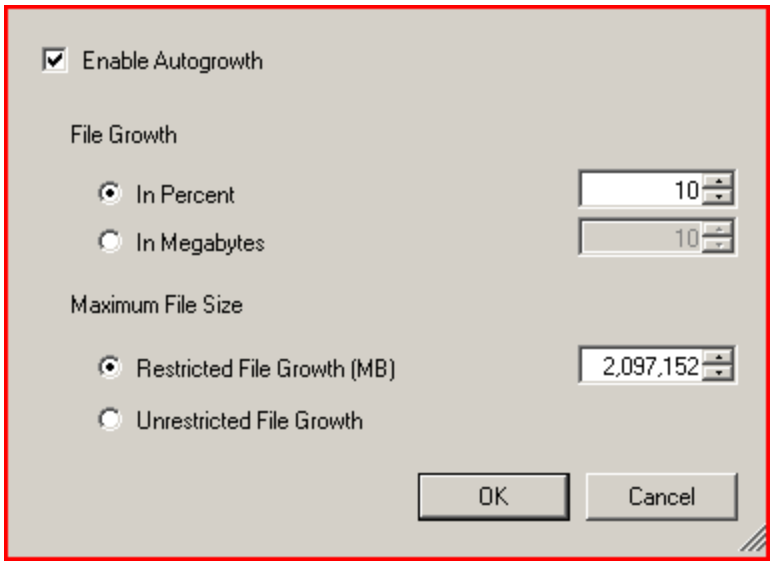
Settings can be set by doing the following:

1. Select Database (CNG_Main)
2. Right Click and Select Properties.
3. Select Files
4. Click the ellipses button of Autogrowth and set what is appropriate for your database.





Settings for the Log file:



Note: Restricted file growth is in MB. Set what is appropriate for your needs.