## http:\www.cryer.co.uk/brian/index.htm

## Cry How to...

# Add a column to a table unless it already exists

To add a column to a SQL Server database table, checking first that the column does not already exist:

if not exists (select \* from syscolumns  
  where id=object\_id('<*table\_name*>') and name='<*column\_name*>')  
    alter table <*table\_name*> add <*column\_name*> <*column\_definition*>

where: <table\_name> is the name of the table, <column\_name> is the name of the column and <column\_definition> is the definition of the column used when creating the column.

For example:

if not exists (select \* from syscolumns  
  where id=object\_id('Employees') and name='MaidenName')  
    alter table Employees add MaidenName varchar(64) NULL  
go

These notes have been tested against SQL Server 7 and SQL Server 2000.

# Create a table unless it already exists

To create a table in SQL Server, unless it has already been created:

if not exists (select \* from sysobjects  
  where name='<*table\_name*>' and xtype='U')  
    create table <*table\_name*>' ...

where <table\_name> is the name of the table. The 'xtype' field holds the type of object, 'U' denoting a user defined table.

For example:

if not exists (select \* from sysobjects where name='cars' and xtype='U')  
    create table cars (  
        Name varchar(64) not null  
    )  
go

# Create a temporary table

Creating a temporary table is virtually the same as creating a normal table. The main exception is the naming of the table. A hash ('#') character as the first character in the table name denotes that it is a temporary table.

There are two types of temporary tables, local and global.

A local temporary table has a single hash ('#') at the start of its name. A local temporary table is visible only to the user who created it and is destroyed automatically when that user disconnects.

A global temporary table is denoted by a name starting with two hashes (i.e. '##'). A global temporary table is visible to all users and is deleted automatically when the last user who has referenced the table disconnects.

An example of creating a local temporary table:

create table #foo  
(  
    CarIndex  smallint,  
    CarType   varchar(20)  
)

An example of creating a global temporary table:

create table ##baz  
(  
    CarIndex  smallint,  
    CarType   varchar(20)  
)

# Create an index on a table unless it already exists

To create an index on a table in SQL Server, unless the index has already been created:

if not exists (select \* from sysindexes  
  where id=object\_id('<*table\_name*>') and name='<*index\_name*>')  
    create index <*index\_name*> on <*table\_name*>(<*field\_names*>)

where <table\_name> is the name of the table, <index\_name> is the name of the index and <column\_names> are the field names used when creating the index. For example:

if not exists (select \* from sysindexes  
  where id=object\_id('Employees') and name='IE1Employee')  
    create index IE1Employee on Employees(name)  
go

# Determine the name of the database you are connected to

To get the name of the current database:

select db\_name(dbid) from master..sysprocesses where spid=@@SPID

alternately:

sp\_who @@SPID

will give the name of the current database and other information about the current user.

# Determine the name of the user for the current session

SQL Server provides a number of ways of getting information about the current user:

o   sp\_who @@SPID

will give the user name and other session information about the current user session.

o   The database variable 'SYSTEM\_USER' returns the domain and login name of the user IF windows authentication is used. However if current user is logged on using SQL Server Authentication then 'SYSTEM\_USER' returns the SQL Server login identification name. For example:

print SYSTEM\_USER

might yield 'PRO\_BRAIN\Brian Cryer', if the user 'Brian Cryer' were logged on using NT authentication from a workstation called 'PRO\_BRAIN' (note 'XP' allows spaces in the user name whilst NT does not)  
or 'sa' if connected using the 'sa' account using SQL Server Authentication.

o   The database variable 'CURRENT\_USER' (or 'SESSION\_USER' - the two always give the same value) returns the database authorization name of the user that made the connection.

o   Other information can be found by interrogating the table 'master..sysprocesses' directly, for example:

select nt\_username, hostname, nt\_domain, loginame, login\_time, program\_name from master..sysprocesses where spid=@@SPID

# Determine which service pack is installed

To determine which SQL Server service pack is installed simply obtain the current version number of SQL Server and from this the service pack number can be looked up.

To obtain the version number using SQL:

select @@version

Alternately, for SQL Server 7 and SQL Server 2000, using SQL Server Enterprise Manager expand the list of servers, right click the server and select ‘Properties’.

Then look the version number up in one of the tables below (which cover SQL Server 2000, SQL Server 7, SQL Server 6.5 and SQL Server 6.).

For SQL Server 2000:

| **@@version** | **Version and Service Pack** |
| --- | --- |
| 8.00.760 | SQL Server 2000 with Service Pack 3a |
| 8.00.760 | SQL Server 2000 with Service Pack 3. |
| 8.00.532 | SQL Server 2000 with Service Pack 2 |
| 8.00.384 | SQL Server 2000 with Service Pack 1 |
| 8.00.194 | SQL Server 2000 with no service pack applied |

For SQL Server 7:

| **@@version** | **Version and Service Pack** |
| --- | --- |
| 7.00.1063 | SQL Server 7 with Service Pack 4 |
| 7.00.961 | SQL Server 7 with Service Pack 3 |
| 7.00.842 | SQL Server 7 with Service Pack 2 |
| 7.00.699 | SQL Server 7 with Service Pack 1 |
| 7.00.623 | SQL Server 7 with no service pack applied |

For SQL Server 6.5:

| **@@version** | **Version and Service Pack** |
| --- | --- |
| 6.50.479 | SQL Server 6.5 with Service Pack 5a (updated) |
| 6.50.416 | SQL Server 6.5 with Service Pack 5a. |
| 6.50.415 | SQL Server 6.5 with Service Pack 5 |
| 6.50.281 | SQL Server 6.5 with Service Pack 4 |
| 6.50.258 | SQL Server 6.5 with Service Pack 3 |
| 6.50.240 | SQL Server 6.5 with Service Pack 2 |
| 6.50.213 | SQL Server 6.5 with Service Pack 1 |
| 6.50.201 | SQL Server 6.5 with no service pack applied |

For SQL Server 6:

| **@@version** | **Version and Service Pack** |
| --- | --- |
| 6.00.151 | SQL Server 6 with Service Pack 3 |
| 6.00.139 | SQL Server 6 with Service Pack 2 |
| 6.00.124 | SQL Server 6 with Service Pack 1 |
| 6.00.121 | SQL Server 6 with no service pack applied |

SQL Server Service packs can be downloaded from the Microsoft web site.

See also Microsoft Knowledge Base article [321185](http://support.microsoft.com/?kbid=321185).

# Export Data to a Comma Separated File

To export data to a comma separated file (csv), use the bcp command line utility:

bcp [<database\_name.>[<owner>].]<table\_name> out <csv-file> [/U<username>] [/P<password>] [/S<server\_name>] /c /t**,**

where:

| <database\_name> | is an optional parameter specifying the name of the database. It can only be omitted if the desired database is the default database. |
| --- | --- |
| <owner> | the name of the table owner. |
| <table\_name> | the name of the table. |
| <csv-file> | The name of the comma separated file to generate. |
| <username> | The SQL Server username to use. This can be left blank if NT authentication is used. |
| <password> | The password to use for the account. |
| <server\_name> | The name of the server on which the database resides. This can be omitted if the database is on the local machine (i.e. the same machine. as the bcp command is issued from). |

for example:

bcp mydb.dbo.sales out sales.csv /U /P /Ssales\_svr /c /t,

or:

bcp mydb..resellers out resellers.csv /U /P /Ssales\_svr /c /t,

# Import Data from a Comma Separated File

To import data from a comma separated file (csv), use bcp:

bcp [<database-name**.**>][<owner**.**>]<table-name> in <csv-file> [/U<username>] [/P<password>] [/S<server-name>] /c /t**,**

where:

| <database-name> | is an optional parameter specifying the name of the database. It can only be omitted if the desired database is the default database. |
| --- | --- |
| <owner> | the name of the table owner. |
| <table-name> | the name of the table. |
| <csv-file> | The name of the comma separated file to generate. |
| <username> | The SQL Server username to use. This can be left blank if NT authentication is used. |
| <password> | The password to use for the account. |
| <server-name> | The name of the server on which the database resides. This can be omitted if the database is local. |

Note:

∙        The flags are case sensitive.

∙        The table should expect the fields in the same order as they are saved in the file.

for example:

bcp mydb.dbo.sales in sales.csv /U /P /Ssales\_fs /c /t,

# List all currently connected users

The simplest way to list all users and processes is to use the system stored procedure:

sp\_who

to list all active users:

sp\_who 'active'

to list details about a particular user:

sp\_who '*login-name*'

the same information (and more) can be found by interrogating the system table master..sysprocesses, for example the following is equivalent to 'sp\_who':

select spid, status, loginame, hostname, blocked, db\_name(dbid), cmd from master..sysprocesses

but the table master..sysprocesses also provides additional information such as the login\_time, program\_name and others. For full details of the master..sysprocesses table refer to SQL Server Books Online.

# List all columns in a table

To list all the columns in a table the simplest way is:

exec sp\_help <table-name>

unfortunatly this returns more than just the columns in the table. Alternatly, to just list the columns use:

select name from syscolumns where id=object\_id('<*table\_name*>')

where <*table\_name*> is the name of the table. For example:

select name from syscolumns where id=object\_id('Employees')

# List all filegroups

To get a list of all the filegroups, use:

select groupname from sysfilegroups

# List all indexes on a table

To list all the indexes on a table use:

exec sp\_helpindex <*table\_name*>

where <*table\_name*> is the name of the table for which the list of indexes are required.

This lists the index name, the type (e.g. clustered, non clustered, unique etc) and which columns the index is created on. Further details can be found by consulting the SQL server documentation.

Alternatly, just to list the names of the indexes (with no further information about the index) you could use:

select name from sysindexes where id=object\_id('<*table\_name*>')

for example:

exec sp\_helpindex Employees

or

select name from sysindexes where id=object\_id('Employees')

both produce the same list of index names, but the first also provides information about each of those indexes.

# List all tables

To get a list of all the tables in a SQL Server database, use either:

sp\_tables

or alternately to list only the user tables use:

select name, user\_name(uid) from sysobjects where type='U'

# List all tables or indexes within a filegroup

To list all the tables in a filegroup use:

select distinct(object\_name(id)) from sysindexes  
where groupid=filegroup\_id('<filegroup-name>')

To list all indexes in a filegroup use:

select name from sysindexes  
where groupid=filegroup\_id('<filegroup-name>')  
and indid > 0

where:

<filegroup-name>

is the name of the filegroup for which the list of tables or indexes is required.

for example, to list all the tables in a file group called "ConfigurationData":

select distinct(object\_name(id)) from sysindexes  
where groupid=filegroup\_id('ConfigurationData')

Note:

* Be sure to first select the database that contains the file group.
* If nothing is listed then it may be because the file group that has been specified does not exist - check spelling and which database is currently connected.

# List all tables that contain a given column name

To list all the tables in an SQL Server database that contain a given column name, use the following SQL:

select sysobjects.name, \* from syscolumns, sysobjects

where syscolumns.name='*MyColumn*'

and sysobjects.id = syscolumns.id

and (sysobjects.xtype='U' or sysobjects.xtype='S')

Note:

∙        Replace 'MyColumn' with the name of the column you wish to search on.

∙        This query will return the names of tables that contain the specified column for the current database only.

# List table information

To get information about a table in an SQL Server database, use:

sp\_help <*table-name*>

where: <table-name> is the name of the table for which details are required.

This lists the table name, columns, indexes and other information about the table. Further details can be found by consulting the SQL Server documentation.

# Move a database file

\* Important \*

Before attempting to move a database file please ensure that you have a good backup to go back to should things go wrong.

The procedure for moving a database file varies depending on whether you want to:

* Move a Master database file
* Move a Temp database file
* Move any other database file

## Move a database file

This procedure can be used for any database except 'tempdb' or 'master':

To move one or more database files for a given database:

1. Consider backing up the database before you start, in case anything goes wrong.
2. Get a list of all the filenames for files currently used in the database. Do this using:

Use MyDatabase

Go

Exec sp\_helpfile

This step is essential - you can only successfully reattach the database by specifying each and every file.

1. Detach the database. This prevents others from using it while the files are being moved. Do this using:

Exec sp\_detach\_db 'MyDatabase'

You will not be able to detach a database whilst it is in use.

1. Move the database files to their new location, or rename them, as  appropriate to your needs.
2. Reattach the database, explicitly specifying the full pathname of every file that constitutes the database. This includes any files that were not moved or renamed. For example:

Exec sp\_attach\_db 'MyDatabase',

    'E:\MsSql7\NewHome\MyDatabase\_Data.mdf',

    'E:\MsSql7\NewHome\MyDatabase\_Log.ldf'

Notes:

* 'sp\_attach\_db' can only be used with up to 16 files. If the database has more than 16 files then instead use 'Create Database' with the 'For Attach' clause.
* If the detached database was enabled for replication and is attached to a DIFFERENT server then 'sp\_removedbreplication' should be run to remove replication from the database.
* Only members of the 'sysadmin' server role can execute 'sp\_detach\_db' and 'sp\_attach\_db'.

## Move master database files

Procedure to move the master database:

1. Take a full backup that you are confident you can recover from, incase anything goes wrong.
2. Stop MS Sql Server.
3. Move the files that constitute the master database to their new location and/or rename them, as appropriate.
4. Change the start-up parameters for SQL Server to reflect the change.
5. The easiest way to do this is to edit the parameters in the registry, under:

HKEY\_LOCAL\_MACHINE

    \SOFTWARE

        \Microsoft

            \MSSQLServer

                \MSSQLServer

                    \Parameters

Typical parameters are:

SQLArg0 -dc:\MSSQL7\data\master.mdf

SQLArg1 -ec:\MSSQL7\log\ERRORLOG

SQLArg2 -lc:\MSSQL7\data\masterlog.ldf

1. Restart MS Sql Server.
2. Whilst it should be unnecessary, for confidence you may wish to check the database:

dbcc checkdb (master) with no\_infomsgs

## Move temporary database files

To move a file that constitutes part of the temporary database use:

Alter database tempdb modify file

(name=<logical-name>, filename='<full-pathname>')

For example:

Alter database tempdb modify file

(name=templog, filename='e:\mssql\data\templog.ldf')

This change will take affect when the MS SQL Server is next restarted.

# Obtain record field size information

To see the minimum, average and maximum storage requirements of a field in a given table:

Select min(datalength(<field>)),  
    avg(datalength(<field>)),  
    max(datalength(<field>)) from <table>;

where:

 <field> is the name of the field of interest

 <table> the name of the table.

Be aware that the total storage requirements for a record are more than the sum of the size of the individual fields. For example datalength on a varchar field will return the number of characters and will not include the overhead for storing that number.

# Restore an NT4 system and SQL Server 7 databases from a full backup

These notes are provided as is, for the purpose of assisting with the development of a procedure for the restoration of an NT 4 system and SQL Server 7 databases. No guarantee is stated or implied. A backup cannot be relied upon until the restore has been tested and demonstrated successfully.

These notes are applicable only to NT 4 running SQL Server 7. See also:

* [How to script a full NT 4 system and SQL Server 7 backup](http://www.cryer.co.uk/brian/sqlserver/howtofullnt4sql7bkp.htm)
* [How to script a full Windows 2000 and SQL Server 2000 backup](http://www.cryer.co.uk/brian/sqlserver/howtofullw2ksql2000bkp.htm)
* [How to restore a Windows 2000 system and SQL Server 2000 databases from a backup](http://www.cryer.co.uk/brian/sqlserver/howtorestorew2ksql2000bkp.htm)

## Pre-requisites

* **These notes assume that a backup has been performed to a single tape in accordance with the notes '**[**How to script a full NT 4 system and SQL Server 7 backup**](http://www.cryer.co.uk/brian/sqlserver/howtofullnt4sql7bkp.htm)**'**. Those notes described a process to backup the NT system and SQL Server 7 databases to a single tape.
* Do not approach any restore operation lightly. Be particularly careful restoring onto a working system, because a restore operation that goes wrong could reck both the system and your day.
* These notes assume a reasonable knowledge of NT.

## Introduction

The notes on this page cover how to recover the NT system and SQL Server 7 databases following a catastrophic failure requiring everything to be restored. They assume that the steps covered in '[How to script a full NT 4 system and SQL Server 7 backup](http://www.cryer.co.uk/brian/sqlserver/howtofullnt4sql7bkp.htm)' have been followed to produce the backup that is to be restored from.

Steps covered here:

1. Write protect the backup tape
2. Reload NT 4
3. Restore NT 4 from backup
4. Rebuild SQL Server 7 MASTER database
5. Restore SQL Server 7 DISTRIBUTION and Replicated Databases
6. Restore SQL Server 7 MASTER database
7. Restore SQL Server 7 MODEL, MSDB databases
8. Restore other SQL Server 7 databases

Each step should be considered in order. Depending on the recovery required it may be appropriate to skip some of these steps.

## Write Protect The Backup Tape

It may seem trivial but it is vitally important that before you start any restore operation that you write-protect the tape containing the backup. This is normally done by sliding a tab over on the tape cartridge. This protects the backup against it being accidentally overwritten. Many horror stories would have been avoided if this simple precaution was taken.

## Reload NT 4

In order to be able to recover from the backup NT must be up and running. If NT is already running then there is no need to perform this step. **Only reload NT if the system will not boot because the disk has been wiped or replaced.**

Reloading NT will require access to the original media, be this NT 4 Workstation or NT 4 Server. When reloading NT it does not matter what name is given to the server, domain, network settings etc, because all of these will be recovered as part of the backup process.

If reloading onto a new disk, it is vital that you know the original disk partitioning information and drive letter assignments. The information is not recovered as part of the restore process. Exact disk partiton sizes *may* not matter provided each partition is recreated large enough to hold the original data.

Be sure to load:

* NT itself
* Any SCSI drivers (assuming the tape drive is a SCSI device)
* Tape drivers (so that the tape can be accessed). These are loaded via Control Panel > Tape Devices.

Only proceed with the next step (the restore of NT) when 'ntbackup' can see the tape drive.

## Restore NT 4 From Backup

1. Load the backup tape (write-protected) into the tape drive.
2. Start NT backup, either from the command line ('ntbackup') or from the Start menu (Start > Programs > Administrative Tools (Common) > Backup).
3. In the tape window, select each drive listed (there will only be one if only the C drive was backed up).
4. Click on the [ Restore ] button, and for the restore specify:

| Restore to drive: | C   (i.e. to the original location) |
| --- | --- |
| [/] | Restore Local Registry |
| [/] | Restore File Permissions |
| Logging options do not matter | |

1. Click [ OK ] to start the restore.

* If it asks to replace any files then select 'Yes to all'.
* It will need a reboot to complete the restore of some files.

When the restore is complete (and the system rebooted) all files, accounts, applications, names and other settings should be back to their original state at the time of the backup. The NT 4 component of the restore is now complete.

## Rebuild SQL Server 7 MASTER Database

The MASTER database only needs to be rebuilt if it is missing or totally corrupt. If the MASTER database exists then skip to the next step of restoring it.

1. Start a command session.
2. Ensure that SQL Server is not running, the simplest means is to type:  
       net stop MSSQLServer  
   This may say that this will also stop another service, in which case allow it to continue. It will also indicate if the service is already stopped.
3. Type:  
       rebuildm
4. This will require the original SQL Server cd. (If it auto-runs then close it down.) The 'data' directory should point to the original location of the MASTER database and should not need changing. The source directory should point to '*<cd-drive*>:\X86\Data'. Click '[ Rebuild ]'.

This will reset the MASTER, MODEL and MSDB databases to their original status when SQL Server was first installed. Typically this will be with a blank password for the 'sa' account.

## Restore SQL Server 7 DISTRIBUTION and Replicated Databases

If the database has a DISTRIBUTION database then that (together with the databases involved in replication) must be restored next. If the database is not involved in replication or is not a publisher then this step should be skipped.

Although it may be counter intuitive to restore the DISTRIBUTION and associated replicated databases at this point, I have yet to be able to successfully restore them except as the next step after rebuilding the MASTER database.

Before restoring the distribution database, be sure that the share used as the snapshot folder exists and has the necessary share permissions on it. If this was on a disk that has already been restored as part of the NT restore then this can be taken as read. If it is on a separate disk then be sure to recreate it. Generally, SQLServerAgent on the server requires full access and agents on remote servers require read only access.

SQL Server should be running in order to restore the distribution database. If unsure, issue the following command (in a command window):

net start MSSQLServer

The distribution database can only be restored if you have its exclusive use. Unfortunatly it cannot be set to single user if it is currently currupt or suspect. Check using SQL Server Enterprise Manager whether there are any users other than 'system' or 'sa'. If there are then disconnect them. (SQLServerAgent is a prime suspect here.)

Now enter and run the following script (use SQL Query Analyzer and connect as 'sa'):

use MASTER  
go  
Restore Database DISTRIBUTION from Tape='\\.\Tape0' with File=*N1*, NOUNLOAD  
Restore Database MYDB from Tape='\\.\Tape0' with File=*N2*  
Restore Database YOURDB from Tape='\\.\Tape0' with File=*N3*  
go

Note:

* Be sure to check that the file numbers are the correct for your system. If unsure then the list of file numbers can be found by issuing the command:  
      Restore HeaderOnly from Tape='\\.\Tape0' with nounload  
  the two fields to look at in the result set are the 'BackupName' and 'Position' which is the file number to be used in the above.
* The distribution database is called 'Distribution' by default, but it may be different on your system.
* The two other databases shown in this script are fictitious. Replace them with the names (and file numbers) of the real databases involved in replication for your system.
* If you are unsure whether a database is involved in replication then, to be safe, restore it at this point.
* There is a high likelihood that with transactional replication that the subscribers may hold a more recent update that will have been lost to the publisher. Merge replication seems to normally recover without any problem.

## Restore SQL Server 7 MASTER Database

The MASTER database can only be restored when the database is in single-user mode. To place it in single user mode:

1. Start a command session.
2. To ensure that SQL Server is not running type:  
       net stop MSSQLServer  
   This may say that this will also stop another service, in which case allow it to continue. It will also indicate if the service is already stopped.
3. To start SQL Server in single user mode type:  
       sqlservr -m

To restore the MASTER database from the backup use the following script (run in SQL Server Query Analyzer):

use MASTER  
go  
Restore Database MASTER From Tape='\\.\Tape0' With File=2, NOUNLOAD  
go

Note: While SQL Server is in single user mode you should still be able to connect from SQL Server Query Analyzer if you connect using the "sa" account.

The above assumes that the file number is 2 for the MASTER database. If unsure on the file number then the list of file numbers can be found by issuing the command:

Restore HeaderOnly from Tape='\\.\Tape0' with nounload

the two fields to look at in the result set are the 'BackupName' and 'Position' which is the file number to be used in the above.

Once the MASTER database has been restored start the database by issuing the following command at the command line:

net start MSSQLServer

Remember that the password for the 'sa' account will now be what it was at the time of the original backup.

## Restore SQL Server 7 MODEL, MSDB Databases

After the MASTER database has been restored, the MODEL and MSDB databases must also be restored.

1. Start SQL Server Query Analyzer (Start > Programs > Microsoft SQL Server 7.0 > Query Analyzer)
2. Enter and run the following script:

use MASTER  
go  
Restore Database MODEL from Tape='\\.\Tape0' with File=3, NOUNLOAD  
Restore Database MSDB  from Tape='\\.\Tape0' with File=4  
go

Note:

* Be sure to check that the file numbers are the correct for your system. (To determine the file numbers see the notes on restoring the MASTER database above.)

## Restore Other SQL Server 7 Databases

Each of the remaining (application) databases can now be restored. The simplest procedure is to use a script such as the following:

use MASTER  
go  
Restore Database MYDB from Tape='\\.\Tape0' with File=*N1*, NOUNLOAD  
Restore Database YOURDB from Tape='\\.\Tape0' with File=*N2*  
Restore Database ANODB from Tape='\\.\Tape0' with File=*N3*  
go

In the above replace the fictitious database names with the real database names for the system. Also, be sure to check that the file numbers are the correct for your system.

For replicated databases only: Do not restore any databases that were restored at the same time as the DISTRIBUTION database.

I would recommend completing the restore process by rebooting the server, this will ensure that any other processes which rely on the database should be able to start.

These notes have been tested against SQL Server 7 running under NT 4 on a database using merge and transactional replication. These notes are provided as is, to assist with the development of backup and restore scripts and procedures. Be sure to test the backup and restore of any system. No guarantee stated or implied.

# Restore a W2K system and SQL Server 2000 databases from a full backup

These notes are provided as is, for the purpose of assisting with the development of a procedure for the restoration of an Windows 2000 system and SQL Server 2000 databases. No guarantee is stated or implied. A backup cannot be relied upon until the restore has been tested and demonstrated successfully.

These notes are applicable only to Windows 2000 running SQL Server 2000. See also:

* [How to script a full NT 4 system and SQL Server 7 backup](http://www.cryer.co.uk/brian/sqlserver/howtofullnt4sql7bkp.htm)
* [Restore an NT4 system and SQL Server 7 databases from a full backup](http://www.cryer.co.uk/brian/sqlserver/howtorestorent4sql7bkp.htm)
* [How to script a full Windows 2000 and SQL Server 2000 backup](http://www.cryer.co.uk/brian/sqlserver/howtofullw2ksql2000bkp.htm)

## Pre-requisites

* **These notes assume that a backup has been performed to a single tape in accordance with the notes '**[**How to script a full Windows 2000 and SQL Server 2000 backup**](http://www.cryer.co.uk/brian/sqlserver/howtofullw2ksql2000bkp.htm)**'**. Those notes described a process to backup the Windows 2000 system and SQL Server 2000 databases to a single tape.
* Do not approach any restore operation lightly. Be particularly careful restoring onto a working system, because a restore operation that goes wrong could reck both the system and your day.
* These notes assume a reasonable knowledge of Windows 2000.

## Introduction

The notes on this page cover how to recover a Windows 2000 system and SQL Server 2000 databases following a catastrophic failure requiring everything to be restored. They assume that the steps covered in '[How to script a full Windows 2000 and SQL Server 2000 backup](http://www.cryer.co.uk/brian/sqlserver/howtofullw2ksql2000bkp.htm)' have been followed to produce the backup that is to be restored from.

The restoration procedure described here has not been developed or tested for databases involved in replication. If the SQL Server database is involved in replication then it may be wise to read the notes on this page in conjunction with those in the note '[Restore an NT4 system and SQL Server 7 databases from a full backup](http://www.cryer.co.uk/brian/sqlserver/howtorestorent4sql7bkp.htm)', which has been tested with respect to restoring replicated SQL Server 7 databases.

Steps covered here:

1. Write protect the backup tape
2. Reload Windows 2000
3. Restore Windows 2000 from backup
4. Rebuild SQL Server 2000 MASTER database
5. Restore SQL Server 2000 MASTER database
6. Restore other SQL Server 2000 MODEL, MSDB and other databases

Each step should be considered in order. Depending on the recovery required it may be appropriate to skip some of these steps.

## Write Protect The Backup Tape

It may seem trivial but it is vitally important that before you start any restore operation that you write-protect the tape containing the backup. This is normally done by sliding a tab over on the tape cartridge. This protects the backup against it being accidentally overwritten. Many horror stories would have been avoided if this simple precaution was taken.

## Reload Windows 2000

In order to be able to recover from the backup Windows 2000 must be up and running. If Windows 2000 is already running then there is no need to perform this step. **Only reload Windows 2000 if the system will not boot because the disk has been wiped or replaced.**

Reloading Windows 2000 will require access to the original media, be this Windows 2000 Workstation or Windows 2000 Server. When reloading Windows 2000 it does not matter what name is given to the server, domain, network settings etc, because all of these will be recovered as part of the backup process.

If reloading onto a new disk, it is vital that you know the original disk partitioning information and drive letter assignments. The information is not recovered as part of the restore process. Exact disk partition sizes *may* not matter provided each partition is recreated large enough to hold the original data.

Be sure to load:

* Windows 2000 itself
* Any SCSI drivers (assuming the tape drive is a SCSI device)
* Tape drivers (so that the tape can be accessed). These are loaded via Control Panel > Tape Devices.

Only proceed with the next step (the restore of Windows 2000) when the backup utility can see the tape drive.

## Restore Windows 2000 From Backup

Warning: Be aware that a full restore should only be done onto a machine with the same hardware. Differences in some peripherals (e.g. different types of SCSI cards/drivers) may make the restored system unbootable on a different machine. If you find that the machine will not start (blue screen or hangs) after doing a restore then reload Windows 2000 and consider only restoring the individual files that you require. This does not affect SQL Server (although drives designations must be the same).

If the active directory service is running (it will not be if Windows 2000 has been freshly installed) then:

1. Reboot the server.
2. Press F8 when Windows 2000 starts to boot.
3. From the Windows 2000 Advanced Options menu select 'Directory Services Restore Mode'. This will start the server without active directory, and will boot into 'safe mode'.

The above can be ignored if the active directory has not yet been installed.

Note: If there are any drives other than the C drive then ensure that these have been partitioned and formatted.

To perform the restore:

1. Run the backup utility (Start > Accessories > System Tools > Backup)
2. Change the restore options to allow it to always replace files, this is under Tools > Options, select on 'Restore' tab and set the option 'Always replace the file on my computer'. (This allows it to replace an existing file with an older one from the tape - this is essential otherwise you will encounter problems when trying to restore SQL Server.)
3. Insert the tape. The presence of the tape should be recognised and a message window appear titled 'New Import Media'. (This may take a few minutes.)
4. Select 'Allocate this media to Backup' and click '[OK]'.
5. On the 'Restore' tab, highlight the type of drive folder (e.g. '4mm DDS') and the media should be listed. Right click the media and select 'catalog'.
6. Expand the tape folder and select the tape media just catalogued. Check each drive letter listed (which will be just C: if only C: was backed up) and check the 'system state'. (There will be SQL Server backup sets also listed and these can be ignored at this stage but if unsure check everything - the SQL server backup sets will be ignored during the restore at this stage.)
7. Click the 'Start Restore' button. There will be a warning if you are restoring the system state.
8. On the confirm dialog box click 'Advanced' and check all the options. (Optional)
9. After okaying the dialogs the restore will commence. When the restore is completed allow it to reboot.

When the restore is complete (and the system rebooted) all files, accounts, applications, names and other settings should be back to their original state at the time of the backup. The Windows 2000 component of the restore is now complete.

Note: If you get the following error from SQL Server when logging on after rebooting:

The ordinal 29 could not be located in the dynamic link library odbcbcp.dll

then most probably you did not reset the restore options to always replace files. See step 2 above and repeat the restore.

## Rebuild SQL Server 2000 MASTER Database

The MASTER database only needs to be rebuilt if it is missing or totally corrupt. If the MASTER database exists then skip to the next step of restoring it. If restoring onto a clean system then this step is required.

1. Start a command session.
2. There is a bug in the version of 'rebuildm' which ships with SQL Server. (Rebuildm is the utility for rebuilding the master database. The bug is that it leaves the read-only file attribute set on the database files which means the rebuild fails - for further details see the Microsoft knowledge base article Q273572.) To work around this bug you will need to copy the contents of '*<cd-drive*>:\X86\Data' from the original Microsoft SQL Server 2000 cd to a directory on the server (it can be deleted once the rebuild process has been completed). Clear the read-only file property on each of these files.
3. Type:  
       rebuildm
4. The 'data' directory should point to the original location of the MASTER database and should not need changing (unless the database was originally installed or needs to be installed to a different location). The source directory should point to the directory into which the '*<cd-drive*>:\X86\Data' files were copied. Then click '[ Rebuild ]'.

This will reset the MASTER, MODEL and MSDB databases to their original status when SQL Server was first installed. Typically this will be with a blank password for the 'sa' account. You can now restart the server or manually restart the SQL Server service.

If the rebuild ends with the error:

Rebuild Master Failed with error: -1

The error occurred during server configuration. Refer to install\cnfgsvr.out and the log\errorlog files in C:\ Program Files\Microsoft SQL Server\MSSQL for diagnostic information.

Then delete all the files in the 'data' directory and try again. (You may be able to get away with just deleting 'master\*', 'model\*' and 'msdb\*' but I have not tested this.) It is most likely that one of the files in the data directory was read-only or newer than the ones that rebuild needs to copy in.

## Restore SQL Server 2000 MASTER Database

The MASTER database can only be restored when the database is in single-user mode. To place it in single user mode:

1. Start a command session.
2. To ensure that SQL Server is not running type:  
       net stop MSSQLServer  
   This may say that this will also stop another service, in which case allow it to continue. It will also indicate if the service is already stopped.
3. To start SQL Server in single user mode type:  
       sqlservr -m

If this is not recognised then cd to 'C:\Program Files\Microsoft SQL Server\MSSQL\BINN' and try again.

To restore the MASTER database from the backup use the following script (run in SQL Server Query Analyzer):

use MASTER  
go  
Restore Database MASTER From Tape='\\.\Tape0' With File=3, NOUNLOAD  
go

Note: While SQL Server is in single user mode you should still be able to connect from SQL Server Query Analyzer if you connect using the "sa" account.

The above assumes that the file number is 3 for the MASTER database. If unsure on the file number then the list of file numbers can be found by issuing the command:

Restore HeaderOnly from Tape='\\.\Tape0' with nounload

the two fields to look at in the result set are the 'BackupName' and 'Position' which is the file number to be used in the above.

Once the MASTER database has been restored (the restore will have caused the database to be shutdown) start the database by issuing the following command at the command line:

net start MSSQLServer

Remember that the password for the 'sa' account will now be what it was at the time of the original backup.

## Restore Other SQL Server 2000 MODEL, MSDB and other databases

Each of the databases can now be restored:

1. Start SQL Server Query Analyzer (Start > Programs > Microsoft SQL Server > Query Analyzer)
2. Enter and run the following script:

use MASTER  
go  
Restore Database MODEL from Tape='\\.\Tape0' with File=4, NOUNLOAD  
Restore Database MSDB from Tape='\\.\Tape0' with File=5  
Restore Database MYDB from Tape='\\.\Tape0' with File=*N1*  
Restore Database YOURDB from Tape='\\.\Tape0' with File=*N2*  
Restore Database ANODB from Tape='\\.\Tape0' with File=*N3*  
go

Note:

* Be sure to check that the file numbers are the correct for your system. (To determine the file numbers see the notes on restoring the MASTER database above.)
* You must restore the MODEL and MSDB databases.
* In the above replace the ficticious database names ('MYDB', 'YOURDB' and 'ANODB') with the real database names for the system. Also, be sure to check that the file numbers are the correct for your system.
* The 'nounload' only needs to be specified once.

I would recommend completing the restore process by rebooting the server, this will ensure that any other processes which rely on the database should start normally.

These notes have been tested against SQL Server 2000 running under Windows 2000. These notes are provided as is, to assist with the development of backup and restore scripts and procedures. Be sure to test the backup and restore of any system. No guarantee stated or implied.

# Run dbcc checkdb on each database (using cursors)

The simplest way to run 'dbcc checkdb' (to check the structural integrity of an SQL Server database) on each database in the system is to write a simple script that includes each database by name, for example:

dbcc checkdb(MASTER) with no\_infomsgs  
dbcc checkdb(MODEL) with no\_infomsgs  
.  
.  
dbcc checkdb(TEMP) with no\_infomsgs

The disadvantage of this approach is that the script needs to be updated each time a new database is added or a database deleted. It is also unlikely to be portable from one server to another without being rewritten.

An alternate and more adaptable approach is to let the script determine what databases are present and to run dbcc checkdb on each in turn:

declare database\_cursor CURSOR for select name from master..sysdatabases  
declare @database\_name sysname  
  
open database\_cursor  
fetch next from database\_cursor into @database\_name  
while @@FETCH\_STATUS=0  
begin  
  print @database\_name  
  dbcc checkdb(@database\_name) with no\_infomsgs  
  fetch next from database\_cursor into @database\_name  
end  
  
close database\_cursor  
deallocate database\_cursor

The advantage of this approach is that the script does not need updating should a new database be added or one removed and it can be copied from one server to another and executed without change.

# SQL Server Replication

## SQL Server Replication Topics

* Capabilities of SQL Server Replication
* SQL Server Replication Terminology
* Which type(s) of replication do you need?
* How to set up a distribution database
* How to set publication options
* How to create a new publication
* How to subscribe to a publication

These notes assume SQL Server 7.

## Capabilities of SQL Server Replication

SQL Server 6.5 only allows read-only copies of data to be replicated. However, with SQL Server 7, updateable copies of data can be replicated. This means that copies of the same data can be available on two different servers and should the data on one server be updated then that change will be replicated to (i.e. duplicated on) the other server.

| Back to Top |
| --- |

## SQL Server Replication Terminology

An understanding of SQL Server Replication terminology is useful, but not essential, for setting up and maintaining a replicated system. The following is a list of the terms used with SQL Server Replication.

Article

An article can be an entire table, select rows from a table, specified columns from a table or a stored procedure. An article is what is to be replicated. See also publication.

Distribution Agent

The distribution agent is a process that moves transactions and snapshots held in the distribution database to subscribers. See also Snapshot replication and Transaction replication.

Distribution Database

A distribution database tracks the changes (to publications) that need to be replicationed (from publishers). A publisher can act as its own distributor, or a remote database server can be used to hold the distribution database.

For transactional replication, the distribution database tracks changes made to publications.

For merge replication it stores only synchronization history.

Distributor

The server that contains the distribution database.

Immediate Transactional Consistency

All sites are guaranteed to always see the same data as the publisher, with no time lag. With immediate transactional consistency each site must simultaneously commit the change. Immediate Transactional Consistency therefore has a performance impact, because of which it is unlikely to be suitable for high performance databases or where replicating over a slow LAN or a WAN.

Merge Agent

The merge agent is a process that merges changes between publication and subscription databases where merge replication is used. See also Merge Replication.

Merge Replication

Type of replication that allows changes to the data to be made at any site. Changes to publications are then merged with the copies of that publication held at the other server(s). Merge replication cannot guarantee transactional consistency because the same or related records can be updated on different servers at the same time. See also snapshot replication and transactional replication.

Publication

The data to be replicated is contained with a publication. A publication can contain a selection of tables and stored procedures. A table included in a publication is called an article. A publication therefore defines the set of data that is to be replicated. Each publication can be replicated as a snapshot publication (using snapshot replication), a transactional publication (using transactional replication) or a merge publication (using merge replication). See also article.

Publisher

A publisher is a server that makes data available for other servers, i.e. it is the server that makes data available for replication.

Pull Subscription

With pull subscription the subscriber asks the publisher for periodic updates. With pull subscription each subscriber can specify the best time for the update. Pull subscription is generally considered the better form of replication when a large number of subscribers are involved, or where the load on the LAN needs to be distributed.

Push Subscription

With push subscription the publisher pushes any changes made to the data out to the subscribers. Typically push subscription is used where changes need to be replicated to the subscribers shortly after the change occurs, i.e. where the the replication of changes is required in near real time.

Replication

The capability to copy (i.e. replicate) data and changes to that data from one database to another.

Snapshot replication

Type of replication where a snapshot is taken of the current publication data. The subscribers copy of the data is then replaced with the entire contents of that snapshot. Snapshot replication requires less processor overhead than either merge or transaction replication because it does not require continuous monitoring of data changes. Probably not suitable for replicating large volumes of data (i.e. large tables) because of the network traffic involved. See also transaction replication and merge replication.

Subscriber

A subscriber is a server that receives updates to the data. Each subscriber is associated with a publisher.

Transactional Consistency

With transactional consistency all sites are guaranteed to have the same data as the publisher. Sites may lag behind the publisher, but the view at each subscriber will be the same as that at the publisher at some point in time.

Transactional Replication

Type of replication where the copies of the transactions are replicated to each subscriber. Transactional replication uses the transaction log to capture changes. These changes are then sent to subscribers and applied in the same order. This guarantees transactional consistency. Transactional replication is well suited where near real-time updates are required. See also snapshot replication and merge replication.

| Back to Top |
| --- |

## Which type(s) of replication do you need?

The three different types of replication (merge, snapshot and transaction) cannot be mixed within a publication, i.e. for each publication only one type of replication can be used. However, given that there is no limit on the number of publications that can be defined this should not be a problem (if you need a table to participate in a different type of replication from other tables simply place it in another publication).

Each type of replication is suited to different requirements, as illustrated below:

|  | **Merge** | **Snapshot** | **Transaction** |
| --- | --- | --- | --- |
| Edit anywhere | Yes | No | No |
| Only changes replicated | Yes | No | Yes |
| Suited for large quantities of data | Yes | No | Yes |
| Replicate stored procedures | No | Yes | Yes |
| Read only at subscriber | No | Yes | Yes |
| Consistency guaranteed | No | Yes | Yes |
| Complete refresh of data | No | Yes | No |

It would be wise to plan for each article/table which form of replication would be most appropriate. As a general rule, I would recommend merge replication for tables that need to be updateable at any server and transaction replication where updates are only to be made at the publisher.

| Back to Top |
| --- |

## How to set up a distribution database

The distribution database tracks what information needs to be replicated from the publisher to the subscribers. It can be held on a separate server to the publisher or on the same server. (Personally I think it makes more sense for it to be on the same server as the publisher.)

Be aware that replication requires heavy use of the distribution database. It is suggested that the initial size of the data device should be at least 30MB and for the log device at least 15MB.

To set up a distribution database:

     i.        Run Enterprise Manager

    ii.        From the server manager window select a server that is to hold the distribution database.

  iii.        From the ‘Tools’ menu select ‘Replication’, then from the drop down menu that appears select ‘Configure Publishing and Subscribers...’.

  iv.        If distribution database has not already been created then the ‘Configure Publishing and Distribution Wizard’ will now run. Click '[Next >]' on the first screen.

   v.        On the window ‘Choose Distributor’ - I would recommend using the publisher-server as its own distributor, but another server may be selected at this point. Click ‘[Next >]’.

  vi.        On the window ‘Use Default Configuration’ - I would recommend using the default settings. Any servers already registered with the server will be listed as potential subscribers. Subscribing servers can be added later if need be, but if you already know what servers you want to replicate to then it would be as well to register them with Enterprise Manager prior to creating the distribution database. If you are not happy with the default list of servers or the location for the distribution database then do not choose the default settings. Click ‘[Next >]’ to progress to the next screen.

 vii.        If you selected the default configuration then you can select ‘[Finish]’ now. Otherwise you can configure the distribution database name and location, which server is to act as publisher and which databases are to participate in replication.

| Back to Top |
| --- |

## How to set publication options

The publication options cover all the options relating to the publication of data, short of defining the data itself. It provides a means of setting or changing:

∙        The distribution database, and its properties.

∙        The publishers

∙        The databases to be published

∙        The subscribers - including the subscription schedule, i.e. the frequency at which replication will take place.

An initial set of publication options will normally have been set up when the distribution database was configured. To change or reconfigure the publication options:

     i.        Run Enterprise Manager

    ii.        From the server manager window select the publisher-server.

  iii.        From the ‘Tools’ menu select ‘Replication’ , and from the sub-menu select ‘Configure Publishing, Subscribers and Distribution...’

The distributor, publisher, published database and subscriber options are each held on different tabs.

To configure when or the frequency at which each subscriber will receive any updates, select the 'Subscribers' tab. Double clicking the subscriber of interest will show its subscription properties. One of the tabs is 'schedules'. On this the timing and frequency of updates for replication can be set.

If the publisher and subscriber are in different NT domains then some thought must be given to which login account the replication agents are to use on the subscriber. To show or configure which accounts are used, select ‘Replication’ from the ‘Tools’ menu, from the sub-menu select ‘Configure Publishing, Subscribers and Distribution...’, click the ‘Subscribers’ tab and double click the subscriber. If the account used by the replication agents must be a valid account at the subscriber. If only transaction (or snapshot) replication is to be used then it must have ‘sysadmin’ or ‘db\_owner’ roles for the database being replicated. If merge replication is to be used then it must have the 'sysadmin' role for the database being replicated.

| Back to Top |
| --- |

## How to create a new publication

A publication defines the data that is to be replicated, and the method of replication. Before a publication is created the distribution database must already have been setup and configured.

To create (or modify) a publication:

     i.        Run Enterprise manager

    ii.        From the server manager window select the publication server.

  iii.        From the ‘Tools’ menu select ‘Replication’, and from the sub-menu select ‘Create and Manage Publications’. The ‘Create and Manage Publications on {server}’ dialog box should now appear.

  iv.        To create a new publication first select the database that contains the data to publish and then click ‘[Create Publication...]’. The 'Create Publication Wizard' should now run.

Note:

∙        Unless you absolutely need every server to be completely in step with every other don’t use ‘immediate-updating subscriptions’. This option will cripple performance, because no transaction can complete unless it can complete on every other server.

| Back to Top |
| --- |

## How to subscribe to a publication

To subscribe to a publication:

     i.        Run Enterprise Manager

    ii.        From the server manager window select the publisher-server.

  iii.        From the ‘Tools’ menu select ‘Replication’ , and from the sub-menu select ‘Create and Manage Publications...’.

  iv.        Expand the database of interest and select the publication that is to be subscribed to.

   v.        Click ‘[Properties and Subscriptions]’

  vi.        Select the ‘Subscriptions’ tab.

 vii.        Click the ‘[Push New...]’ button to add a new subscriber. This will cause the ‘Push Subscription Wizard’ to run.

# Script a full NT4 and SQL Server 7 backup

These notes are provided as is, for the purpose of assisting with the development of scripts to perform a full NT4 and SQL Server 7 backup. No guarantee is stated or implied. A backup system cannot be relied upon until the restore has been tested and demonstrated successfully.

These notes are applicable only to NT 4 running SQL Server 7. See also:

* [How to restore an NT 4 system and SQL Server 7 databases from a full backup.](http://www.cryer.co.uk/brian/sqlserver/howtorestorent4sql7bkp.htm)
* [How to script a full Windows 2000 and SQL Server 2000 backup.](http://www.cryer.co.uk/brian/sqlserver/howtofullw2ksql2000bkp.htm)
* [How to restore a Windows 2000 system and SQL Server 2000 databases from a backup.](http://www.cryer.co.uk/brian/sqlserver/howtorestorew2ksql2000bkp.htm)

## Pre-requisites

A backup requires a working tape device. It is possible to backup to another server across a network, but these notes assume a working tape drive. These notes also assume that the tape device has sufficient capacity.

## Backup Scripts

To create a script that will perform a full backup of the NT 4 system together with the SQL Server 7 databases onto a single tape, first create a command file containing the following:

ntbackup backup c:\ /d "NT Complete Backup" /t normal /b /hc:on  
osql -U sa -P "psswrd" -i SqlFullBackup.sql

This can be done with a text editor such as notepad. Name this file suitably, such as 'FullNT4SqlBackup.cmd'. The first line ('ntbackup ...') performs a backup of the NT 4 system. The line, as shown, assumes that only the C drive needs to be backup up, add other drives as appropriate for your system. Other switches on the line are:

| backup | That this is a backup, not a restore or eject. |
| --- | --- |
| /d "NT Complete Backup" | Sets a brief description of the backup set. |
| /t normal | Sets the type of backup. |
| /b | Includes the registry in the backup. |
| /hc:on | Use hardware compression. |

The 'osql' statement runs a script that backs up the SQL Server 7 database. The '-P' flag specifies the password for the 'sa' account, and this will need to be specified appropriately for each site. Create the 'SqlFullBackup.sql' script using notepad, use the following as a template:

-- This script will perform a full backup of all the SQL Server 7  
-- databases, except TEMPDB. This is because TEMPDB  
-- is recreated each time SQL Server starts.  
  
-- Start by truncating the logs.  
BACKUP LOG MASTER WITH TRUNCATE\_ONLY  
BACKUP LOG MODEL  WITH TRUNCATE\_ONLY  
BACKUP LOG MSDB   WITH TRUNCATE\_ONLY  
BACKUP LOG MYDB   WITH TRUNCATE\_ONLY  
BACKUP LOG YOURDB WITH TRUNCATE\_ONLY  
BACKUP LOG ANODB  WITH TRUNCATE\_ONLY  
GO  
  
-- Now backup each database in turn.  
BACKUP DATABASE MASTER TO TAPE='\\.\TAPE0' WITH NAME='MASTER COMPLETE', NOUNLOAD  
BACKUP DATABASE MODEL  TO TAPE='\\.\TAPE0' WITH NAME='MODEL COMPLETE'  
BACKUP DATABASE MSDB   TO TAPE='\\.\TAPE0' WITH NAME='MSDB COMPLETE'  
BACKUP DATABASE MYDB   TO TAPE='\\.\TAPE0' WITH NAME='MYDB COMPLETE'  
BACKUP DATABASE YOURDB TO TAPE='\\.\TAPE0' WITH NAME='YOURDB COMPLETE'  
BACKUP DATABASE ANODB  TO TAPE='\\.\TAPE0' WITH NAME='ANODB COMPLETE'  
GO

The default behaviour is for SQL Server to eject the tape after it has backed up each database. This is overridden by the 'NOUNLOAD' option against the first database. To have the tape ejected at the end of the script, add ', UNLOAD' to the end of the last 'BACKUP ...' line, e.g.:

BACKUP DATABASE ANODB TO TAPE='\\.\TAPE0' WITH NAME='ANODB COMPLETE', UNLOAD

## Putting it all together

The above scripts should be sufficient to perform a full backup of the NT 4 system together with all the SQL Server 7 databases. The only steps remaining to complete the task are:

1. Review it.  
   The scripts presented here may be suitable for your site with only the disk drives and database names amended, or you may require something more sophisticated. Review the process to be confident with it. If things go wrong it will be YOU that bears the responsibility.
2. Test it.  
   To be relied upon the backup script must be tested. Part of this testing must involve the restore process. [The restore process is covered by a separate note here](http://www.cryer.co.uk/brian/sqlserver/howtorestorent4sql7bkp.htm). Preferably the restore should be to a different machine (or the same machine with the original disks replaced) otherwise restoring to the same machine may hide problems that only come to light when you need to do a complete restore because of a catastrophic error such as disk failure or a fire.
3. Automate the backup.  
   The simplest strategy is to have the task scheduler run the backup every night or once a week. All you then have to do is remember to change the tapes.
4. Safe storage.  
   The backup tape produced is very important. It is your sole means of recovery. Store if safely - away from the server, preferably in a fire safe or off-site.

# Script a full W2K and SQL Server 2000 backup

These notes are provided as is, for the purpose of assisting with the development of scripts to perform a full Windows 2000 and SQL Server 2000 backup. No guarantee is stated or implied. A backup system cannot be relied upon until the restore has been tested and demonstrated successfully.

These notes are applicable only to Windows 2000 running SQL Server 2000. See also:

* [How to script a full NT4 and SQL Server 7 backup](http://www.cryer.co.uk/brian/sqlserver/howtofullnt4sql7bkp.htm)
* [How to restore an NT 4 system and SQL Server 7 databases from a full backup.](http://www.cryer.co.uk/brian/sqlserver/howtorestorent4sql7bkp.htm)
* [How to restore a Windows 2000 system and SQL Server 2000 databases from a backup.](http://www.cryer.co.uk/brian/sqlserver/howtorestorew2ksql2000bkp.htm)

## Pre-requisites

A backup requires a working tape device. It is possible to backup to another server across a network, but these notes assume a working tape drive. These notes also assume that the tape device has sufficient capacity.

## Backup Scripts

To create a script that will perform a full backup of the NT 4 system together with the SQL Server 7 databases onto a single tape, first create a command file containing the following:

ntbackup backup systemstate c:\ /p "4mm DDS" /d "W2K Complete Backup" /um /hc:on  
osql -E -i SqlFullBackup.sql

This can be done with a text editor such as notepad. Name this file suitably, such as 'FullW2K4SqlBackup.cmd'. The first line ('ntbackup ...') performs a backup of the Windows 2000 system. The line, as shown, assumes that only the C drive needs to be backup up, add other drives as appropriate for your system. Other switches on the line are:

| backup | That this is a backup, not an eject. |
| --- | --- |
| systemstate | Include a backup of the system state. |
| c:\ | Backup the C drive. Add 'D:\' to also back up the D drive and so on. |
| /d "W2K Complete Backup" | Sets a brief description of the backup set. |
| /p "4mmDDS" | The media pool. This assumes that the tape backup is a 4mm DDS drive, this may need to be changed for your system. |
| /um | Use the first available media. You may wish to review the use of this flag for your system. |
| /hc:on | Use hardware compression. |

The 'osql' statement runs a script that backs up the SQL Server 2000 database. The '-E' flag specifies that a trusted connection is to be used. (Alternately use something like '-U sa -P "psswrd"' to specify that the 'sa' account is to be used with the password given - this will need to be specified appropriately for each site. You will need to specify the server name (-S server) if the database is not running on the local machine.) Create the 'SqlFullBackup.sql' script using notepad, use the following as a template:

-- This script will perform a full backup of all the SQL Server 2000  
-- databases, except TEMPDB. This is because TEMPDB is  
-- recreated each time SQL Server starts.  
  
-- Start by truncating the logs.  
BACKUP LOG MASTER WITH TRUNCATE\_ONLY  
BACKUP LOG MODEL  WITH TRUNCATE\_ONLY  
BACKUP LOG MSDB   WITH TRUNCATE\_ONLY  
BACKUP LOG MYDB   WITH TRUNCATE\_ONLY  
BACKUP LOG YOURDB WITH TRUNCATE\_ONLY  
BACKUP LOG ANODB  WITH TRUNCATE\_ONLY  
GO  
  
-- Now backup each database in turn.  
BACKUP DATABASE MASTER TO TAPE='\\.\TAPE0' WITH NAME='MASTER COMPLETE', NOUNLOAD  
BACKUP DATABASE MODEL  TO TAPE='\\.\TAPE0' WITH NAME='MODEL COMPLETE'  
BACKUP DATABASE MSDB   TO TAPE='\\.\TAPE0' WITH NAME='MSDB COMPLETE'  
BACKUP DATABASE MYDB   TO TAPE='\\.\TAPE0' WITH NAME='MYDB COMPLETE'  
BACKUP DATABASE YOURDB TO TAPE='\\.\TAPE0' WITH NAME='YOURDB COMPLETE'  
BACKUP DATABASE ANODB  TO TAPE='\\.\TAPE0' WITH NAME='ANODB COMPLETE'  
GO

The default behaviour is for SQL Server to eject the tape after it has backed up each database. This is overridden by the 'NOUNLOAD' option against the first database. To have the tape ejected at the end of the script, add ', UNLOAD' to the end of the last 'BACKUP ...' line, e.g.:

BACKUP DATABASE ANODB TO TAPE='\\.\TAPE0' WITH NAME='ANODB COMPLETE', UNLOAD

## Putting it all together

The above scripts should be sufficient to perform a full backup of the NT 2000 system together with all the SQL Server 2000 databases. The only steps remaining to complete the task are:

1. Review it.  
   The scripts presented here may be suitable for your site with only the disk drives and database names amended, or you may require something more sophisticated. Review the process to be confident with it. If things go wrong it will be YOU that bears the responsibility.
2. Test it.  
   To be relied upon the backup script must be tested. Part of this testing must involve the restore process. [The restore process is covered by a separate note here](http://www.cryer.co.uk/brian/sqlserver/howtofullw2ksql2000bkp.htm). Preferably the restore should be to a different machine (or the same machine with the original disks replaced) otherwise restoring to the same machine may hide problems that only come to light when you need to do a complete restore because of a catastrophic error such as disk failure or a fire.
3. Automate the backup.  
   The simplest strategy is to have the task scheduler run the backup every night or once a week. All you then have to do is remember to change the tapes.
4. Safe storage.  
   The backup tape produced is very important. It is your sole means of recovery. Store if safely - away from the server, preferably in a fire safe or off-site.

# Test if Trigger exists

To test if a trigger exists:

if exists (select \* from sysobjects where name='<*trigger\_name*>' and xtype='TR')  
    print 'Trigger exists'  
else  
    print 'Trigger does not exist'

where <trigger\_name> is the name of the trigger. For example:

if exists (select \* from sysobjects where name='tD\_Employee' and xtype='TR')  
    print 'Trigger exists'  
else  
    print 'Trigger does not exist'

Note:

* The 'xtype' field in sysobjects denotes the type of object. 'TR' denoting a trigger.