FROM: <http://www.mssqltips.com/sqlservertip/1584/auto-generate-sql-server-restore-script-from-backup-files-in-a-directory/>

**Auto generate SQL Server restore script from backup files in a directory**

|  |  |  |
| --- | --- | --- |
| http://www.mssqltips.com/images/GregRobidoux.jpg | By: [Greg Robidoux](http://www.mssqltips.com/sqlserverauthor/37/greg-robidoux/) | [Read Comments (24)](http://www.mssqltips.com/sqlservertip/1584/auto-generate-sql-server-restore-script-from-backup-files-in-a-directory/#comments) | [Print](http://www.mssqltips.com/tipprint.asp?tip=1584)   Greg is the President of [Edgewood Solutions](http://www.edgewoodsolutions.com/) and a co-founder of MSSQLTips.com.   Related Tips: [1](http://www.mssqltips.com/sqlservertip/1243/auto-generate-sql-server-database-restore-scripts/) | 2 | [3](http://www.mssqltips.com/sqlservertip/1611/auto-generate-sql-server-restore-scripts-after-each-backup-completes/) | [4](http://www.mssqltips.com/sqlservertip/2287/automate-a-database-restore/) | [More](http://www.mssqltips.com/sql_server_dba_tips.asp) | [http://www.mssqltips.com/images/MSSQLTipsTurnsSix_Giveaway.gif](http://www.mssqltips.com/giveaway.asp) |

**Problem**One of the ongoing challenges of a DBA is to backup and restore databases.  Backups are done on an automated schedule, but restores can take on many different versions, you may need to restore a production database, restore a development or test database or just create another copy of the database somewhere else.  There are several ways of automating the restore process and creating a script, but this approach shows a way this can be done by just reading the contents of a directory for the backup files that exist.

**Solution**The following is one simple approach of reading the contents of a directory and creating the restore commands that need to be issued to restore the database.  This script will work for full, differential and transaction log backups.

Before we get started the script below assumes the following:

1. The restored database will have the same name as the backed up database
2. The restored database will be restored in the same location as the backed up database
3. The files have the following naming format
   * dbName\_YYYYMMDDHHMM.xxx
4. File extensions are as follows
   * Full backup – BAK
   * Differential backup – DIF
   * Transaction log backup – TRN
5. XP\_CMDSHELL is enabled
6. There are no missing transaction logs that may break the restore chain

So let's say we are creating our backups on the following schedule:

* Full backups at midnight
* Differential backups every 3 hours starting at 3:15am
* Log backups every 30 minutes starting at 1am

At 9am we would have the following backup files created for September 10, 2008 for the "Customer" database following the rules above.

* Customer\_200809100000.BAK
* Customer\_200809100100.TRN
* Customer\_200809100130.TRN
* Customer\_200809100200.TRN
* Customer\_200809100230.TRN
* Customer\_200809100300.TRN
* Customer\_200809100315.DIF
* Customer\_200809100330.TRN
* Customer\_200809100400.TRN
* Customer\_200809100430.TRN
* Customer\_200809100500.TRN
* Customer\_200809100530.TRN
* Customer\_200809100600.TRN
* Customer\_200809100615.DIF
* Customer\_200809100630.TRN
* Customer\_200809100700.TRN
* Customer\_200809100730.TRN
* Customer\_200809100800.TRN
* Customer\_200809100830.TRN
* Customer\_200809100900.TRN

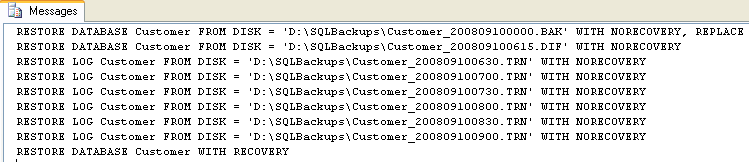
If we wanted to do a restore of the latest Full, Differential and Transaction Log backups to 9am we would need to restore the following files:

* Customer\_200809100000.BAK
* Customer\_200809100615.DIF
* Customer\_200809100630.TRN
* Customer\_200809100700.TRN
* Customer\_200809100730.TRN
* Customer\_200809100800.TRN
* Customer\_200809100830.TRN
* Customer\_200809100900.TRN

The script below will read through the directory and create the restore script for us.  The only two parameters that would need to change are the **@dbName** and the **@backupPath**.

|  |
| --- |
| USE Master;  GO   SET NOCOUNT ON   -- 1 - Variable declaration  DECLARE @dbName sysname  DECLARE @backupPath NVARCHAR(500)  DECLARE @cmd NVARCHAR(500)  DECLARE @fileList TABLE (backupFile NVARCHAR(255))  DECLARE @lastFullBackup NVARCHAR(500)  DECLARE @lastDiffBackup NVARCHAR(500)  DECLARE @backupFile NVARCHAR(500)   -- 2 - Initialize variables  SET @dbName = 'Customer'  SET @backupPath = 'D:\SQLBackups\'   -- 3 - get list of files  SET @cmd = 'DIR /b ' + @backupPath   INSERT INTO @fileList(backupFile)  EXEC master.sys.xp\_cmdshell @cmd   -- 4 - Find latest full backup  SELECT @lastFullBackup = MAX(backupFile)   FROM @fileList   WHERE backupFile LIKE '%.BAK'      AND backupFile LIKE @dbName + '%'   SET @cmd = 'RESTORE DATABASE ' + @dbName + ' FROM DISK = '''          + @backupPath + @lastFullBackup + ''' WITH NORECOVERY, REPLACE'  PRINT @cmd   -- 4 - Find latest diff backup  SELECT @lastDiffBackup = MAX(backupFile)   FROM @fileList   WHERE backupFile LIKE '%.DIF'      AND backupFile LIKE @dbName + '%'     AND backupFile > @lastFullBackup   -- check to make sure there is a diff backup  IF @lastDiffBackup IS NOT NULL  BEGIN     SET @cmd = 'RESTORE DATABASE ' + @dbName + ' FROM DISK = '''          + @backupPath + @lastDiffBackup + ''' WITH NORECOVERY'     PRINT @cmd     SET @lastFullBackup = @lastDiffBackup  END   -- 5 - check for log backups  DECLARE backupFiles CURSOR FOR      SELECT backupFile      FROM @fileList     WHERE backupFile LIKE '%.TRN'      AND backupFile LIKE @dbName + '%'     AND backupFile > @lastFullBackup   OPEN backupFiles    -- Loop through all the files for the database   FETCH NEXT FROM backupFiles INTO @backupFile    WHILE @@FETCH\_STATUS = 0   BEGIN      SET @cmd = 'RESTORE LOG ' + @dbName + ' FROM DISK = '''          + @backupPath + @backupFile + ''' WITH NORECOVERY'     PRINT @cmd     FETCH NEXT FROM backupFiles INTO @backupFile   END   CLOSE backupFiles   DEALLOCATE backupFiles    -- 6 - put database in a useable state  SET @cmd = 'RESTORE DATABASE ' + @dbName + ' WITH RECOVERY'  PRINT @cmd |

If you run the above code in a query window, assuming the listed files above existed, you will get the following output.  At this point you can copy and paste this code into another query window and run the query to do the actual restore.



As you can see it does a Full restore, the latest Differential restore and all Transaction Logs after that.  The script also does a WITH RECOVERY at the end to put the database in a useable state.

**Next Steps**

* This is a pretty straight forward and simple approach.  As mentioned above it restores using the same name and also restores to the same file location.  Try making some modifications to restore it to another database name as well as restoring the files to a different location by incorporating the RESTORE FILELISTONLY command
* This script will work on any server where the files exists and you can run a SQL Server query.  So you can copy the files from one server to another, run this script and then have your restore script ready to go.
* Check out these other restore scripts:
  + [Automate Restoration of Log Shipping Databases for Failover in SQL Server 2000](http://www.mssqltips.com/sqlservertip/1516/automate-restoration-of-log-shipping-databases-for-failover-in-sql-server/)
  + [Auto generate SQL Server database restore scripts](http://www.mssqltips.com/sqlservertip/1243/auto-generate-sql-server-database-restore-scripts/)