SQL Server 2005 Express
Surface Area Configuration (SAC)
expressmaint script settings

Automating Database maintenance in SQL 2005 Express Edition Part I[1] — "demonstrates a couple of different approaches to writing a maintenance utility that mimics some of the behavior of the sqlmaint utility that is included with SQL Server 2000." The SQL Server 2000's sqlmaint utility, which can be accessed command-line or through the SQL Enterprise Manager, is used to configure and run manual/automated SQL backup jobs.

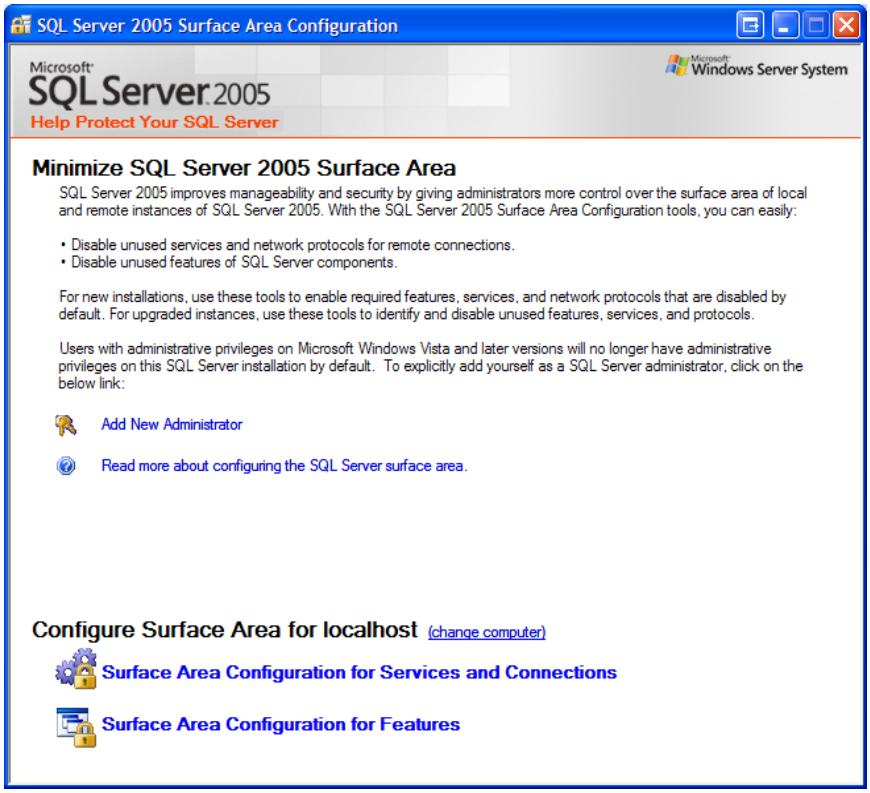
Depending on the default security or current configuration of the Surface Area Configuration (SAC), it may be necessary to adjust the SAC settings to install and run the expressmaint script. This script creates a stored procedure that provided the following operations:

- Full Database Backup
- Differential Database Backup
- Log Backup
- Housekeeping of backup files
- Database Integrity Checks
- Database Index Rebuilds
- Database index Reorganization
- Report Creation

Once the expressmaint stored procedure has been installed, the Windows Scheduler can be used to automate backup jobs.

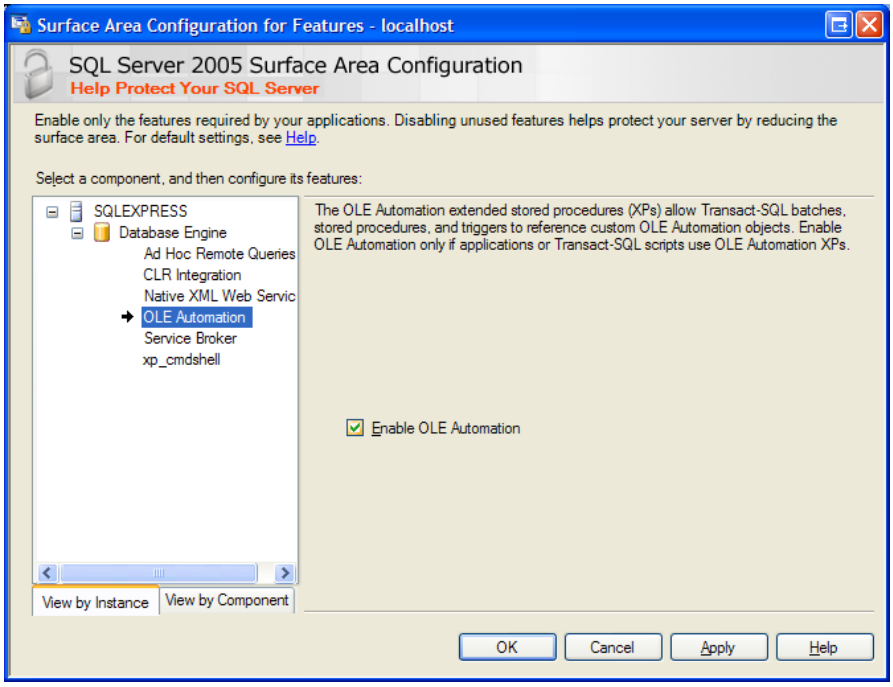
A clean install of SQL Server 2005 Express required the following changes to the default settings of the SAC to run the expressmaint script:

1. Run SQL Server 2005 Surface Area Configuration

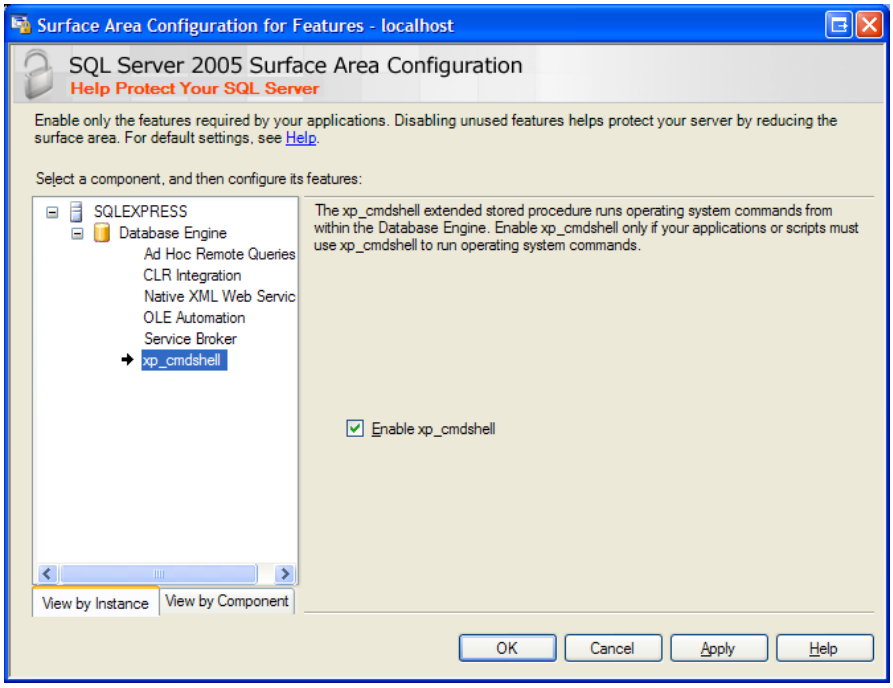


[1] **Automating Database maintenance in SQL 2005 Express Edition Part I**
http://www.sqldbatips.com/showarticle.asp?ID=27
by Jasper Smith, July 31, 2004 for SQL2005

2. Select Surface Area Configuration for Features
3. Enable OLE Automation by selecting OLE Automation for the Database Engine in the SQL Instance (default instance is “SQLExpress”).



4. Enable xp_cmdshell by selecting xp_cmdshell for the Database Engine in the SQL Instance (default instance is “SQLExpress”).



5. Press OK to accept the changes.
6. Close Run SQL Server 2005 Surface Area Configuration

7. Load the stored procedure (expressmaint.sql) into SQL Express using the **sqlcmd**.

   You can copy the file "SQL Server 2005 Express Backup Automation Script.sql" located on the distribution disk and rename it expressmaint.sql or copy the SQL script code which starts on page 9 of this document and past it into an empty notepad file which can be saved as espressmaint.sql.

   A. Select Start | Run and enter cmd.exe then press OK. A command window will be displayed.

   B. At the command prompt enter the following (assuming a named instance called SQLExpress):

```
Sqlcmd -E -S .\SQLExpress -i c:\expressmaint.sql
```

8. To perform a backup copy the expressmaint instructions to a SQL script file and execute the script via sqlcmd.

```
sqlcmd -E -S.\SQLExpress -i"c:\backup scripts\userfullbackup.sql"
```

   Where "c:\backup scripts\userfullbackup.sql" contains the expressmaint instructions to execute (See page 8 for expressmaint script syntax), examples follow:

   A. To perform a Full Database Backup of all user databases to c:\backups, verify the backups and report to c:\reports keeping backups for 1 week and reports for 1 week use the following script:

```
exec expressmaint
   @database      = 'ALL_USER',
   @optype        = 'DB',
   @backupfldr    = 'c:\backups',
   @reportfldr    = 'c:\reports',
   @verify        = 1,
   @dbretainunit  = 'weeks',
   @dbretainval   = 1,
   @rptretainunit = 'weeks',
   @rptretainval  = 1,
   @report        = 1
```

   B. To perform a rebuilding of the indexes for the DB_ASTRA database and report to c:\reports keeping reports for 1 week use the following script:

```
exec expressmaint
   @database      = 'DB_ASTRA',
   @optype        = 'REINDEX',
   @reportfldr    = 'c:\reports',
   @rptretainunit = 'days',
   @rptretainval  = 1,
   @report        = 1
```

   C. To use a single script and pass parameters to it for the type of task, the following script and sqlcmd would perform a Full Database Backup of all user databases to c:\backups, verify the backups and report to c:\reports keeping backups for 1 week and reports for 1 week:

```
exec expressmaint
   @database      = '$(DB)',
   @optype        = 'DB',
   @backupfldr    = '$(BACKUPFOLDER)',
   @reportfldr    = 'c:\reports',
   @verify        = 1,
   @dbretainunit  = '$(DBRETAINUNIT)',
   @dbretainval   = '$(DBRETAINVAL)',
   @rptretainunit = 'weeks',
   @rptretainval  = 1,
   @report        = 1
```

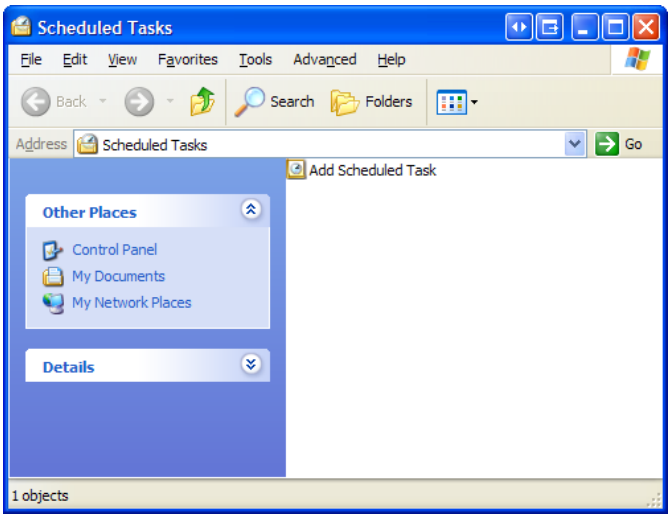
   The sqlcmd would be as follows (no line breaks):

```
sqlcmd -S .\SQLExpress -i"c:\backup scripts\userfullbackup.sql"
    -v DB="ALL_USER"
    -v BACKUPFOLDER="c:\backups"
    -v DBRETAINUNIT="weeks"
    -v DBRETAINVAL="1"
```

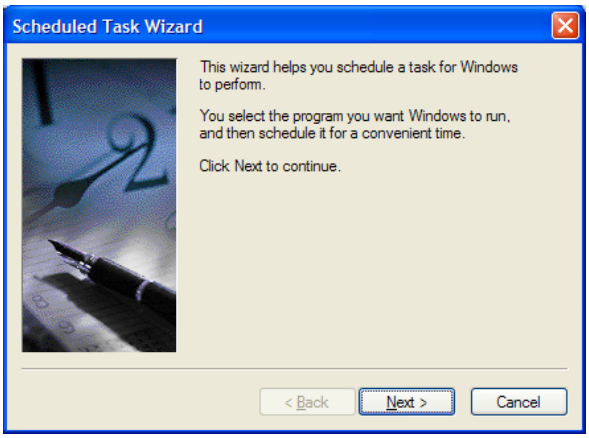
Scheduling Backups (Windows XP)

To schedule backups, use Windows Scheduled Tasks.

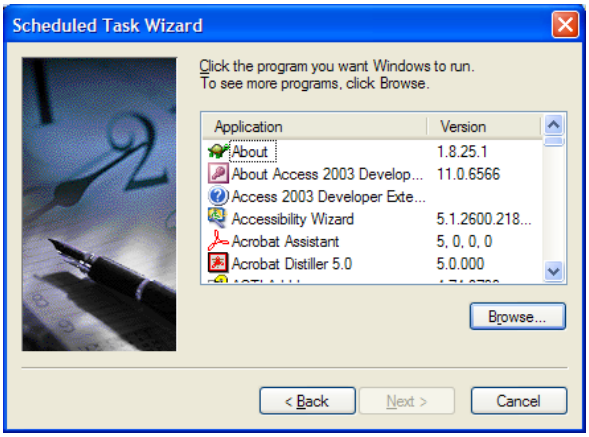
1. Open the Scheduled Tasks window by pressing the Start button, then selecting All Programs, Accessories, System Tools, Scheduled Tasks.



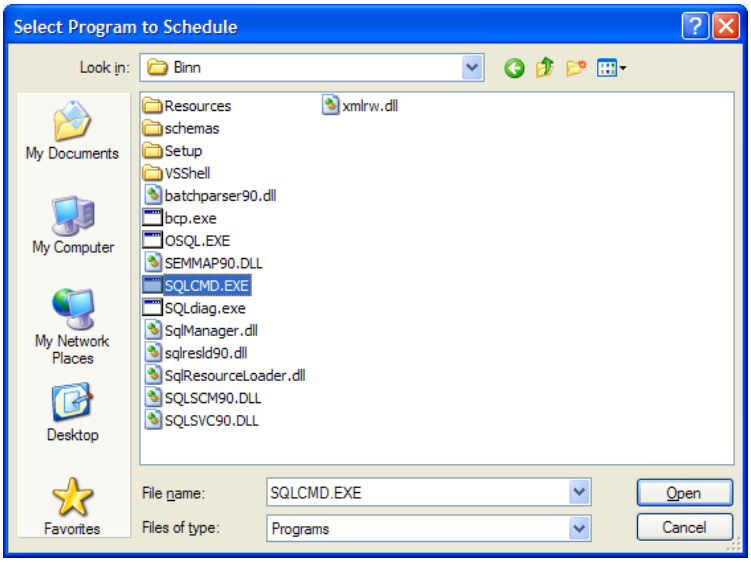
2. Double-click Add Scheduled Task to start the Scheduled Task Wizard



3. Press Next

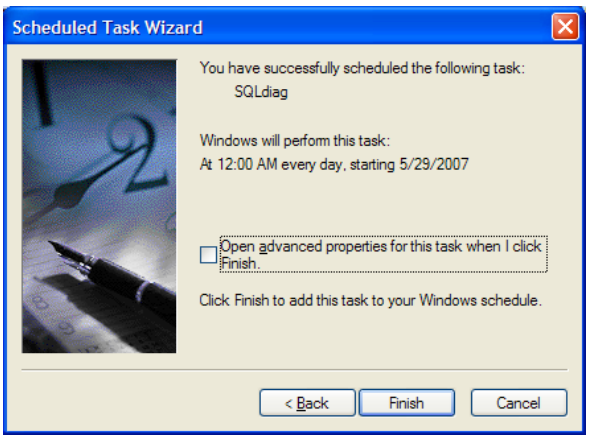


4. Press Browse and navigate to the folder containing sqlcmd.exe, select the file and press Open. The default installation folder is “c:\Program Files\Microsoft SQL Server\90\Tools\Binn”



5. Configure when the task should occur.

6. Press Finish to save the scheduled task.



7. Edit the task, right-click the task name and select Properties. The run command on the Task tab will contain the following:

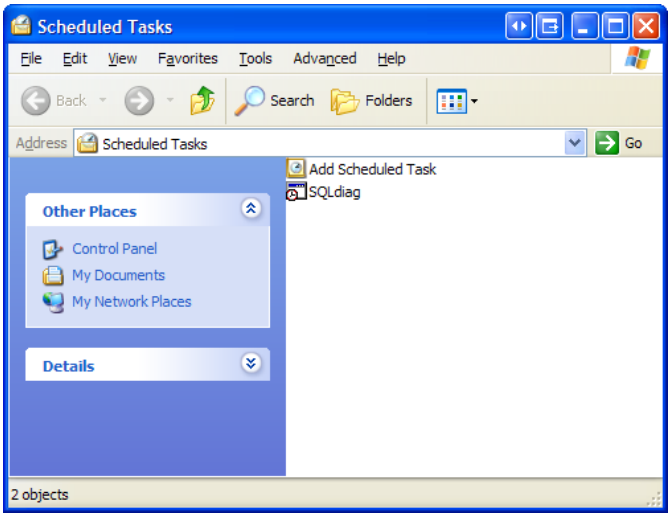
   "C:\Program Files\Microsoft SQL Server\90\Tools\Binn\SQLCMD.EXE"

   A. Add a space after the closing quote and add the sqlcmd parameters. For example:

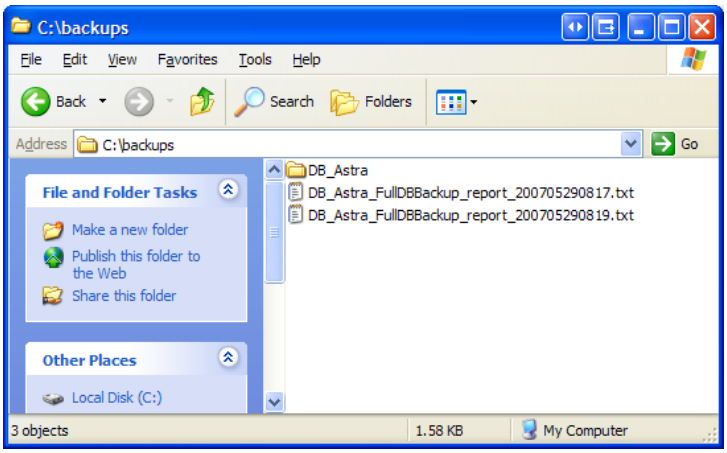
      "C:\Program Files\Microsoft SQL Server\90\Tools\Binn\SQLCMD.EXE" -E -S.\SQLExpress -i"c:\userfullbackup.sql"

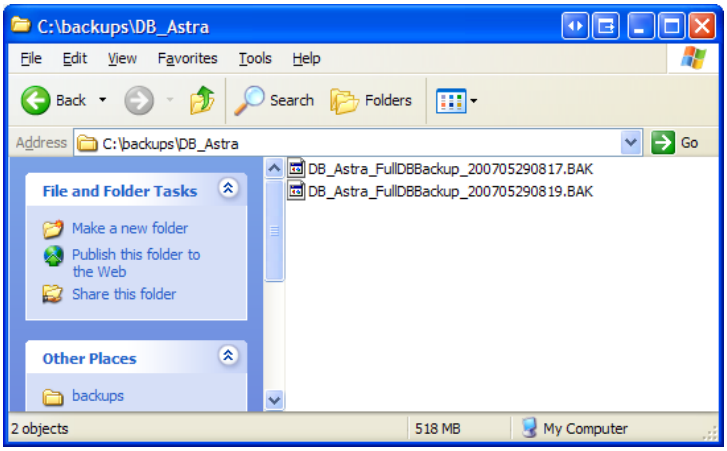
   B. Press OK to save and close the task.

8. Test the new task, right-click the task name (SQLdiag in this example) and select run.



9. View the report file in the specified report folder. Verify the existence of the backup file in the specified folder. The example show below used the same folder for the backup and report files (c:\backup) and performed two full back ups on the DB_Astra database.





Expressmaint Script Syntax

Parameter	Required	Default	Description
@database	Y	NONE	The target database for the maintenance operation. Valid values are a single database name, ALL_USER which will process all user databases and ALL_SYSTEM which will process all system databases
@optype	Y	NONE	The type of maintenance operation to be performed. Valid values are • DB - Full Database Backup • DIFF - Differential Database Backup • LOG - Log Backup • CHECKDB - Database Integrity Check • REINDEX - Rebuild all indexes • REORG - Reorganize all indexes
@backupwith	N	NULL	Specify additional backup options as documented in BOL for the BACKUP WITH command
@backupfldr	N	NULL	The base folder to write the backups to. Sub folders will be created for each database
@verify	N	1	Indicates whether to verify the backup file. Valid values are 1 and 0 with 1 = TRUE and 0 = FALSE
@verifywith	N	NULL	Specify additional verify options as documented in BOL for the VERIFY WITH command
@dbretainunit	N	NULL	The unit of measure for the @dbretainval parameter. Valid values are minutes, hours, days, weeks, months and copies. The combination of these two parameters determines how long or how many copies of old backup files are kept
@dbretainval	N	1	The time period or number of copies of old backups to keep
@report	N	1	Indicates whether to produce a report of the maintenance carried out. Valid values are 1 and 0 with 1 = TRUE and 0 = FALSE
@reportfldr	N	NULL	The folder where maintenance reports are written to if @report = 1
@rptretainunit	N	NULL	The unit of measure for the @rptretainval parameter. Valid values are minutes, hours, days, weeks, months and copies. The combination of these two parameters determines how long or how many copies of old reports are kept
@rptretainval	N	1	The time period or number of copies of old reports to keep
@checkattrib	N	0	Indicates whether to check the archive bit on a backup file before deleting it. This is a safety check to prevent deletion of files that have not been backed up onto tape. Valid values are 1 and 0 with 1 = TRUE and 0 = FALSE
@delfirst	N	0	Indicates whether to delete old backups prior to doing the current backup. This is not advisable but can be useful if disk space is limited. Valid values are 1 and 0 with 1 = TRUE and 0 = FALSE
@debug	N	0	Indicates whether print out debug information such as the commands generated and the contents of the temporary tables used in the procedure. Valid values are 1 and 0 with 1 = TRUE and 0 = FALSE

expressmaint.sql Script Code

The expressmaint script code is included below. Copy everything after the separator to the end of the document and paste into a text file name “expressmaint.sql” to recreate the script file. Page headers and line numbers for the remainder of the document have been omitted for copying the script text.

```
use master
GO

if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[expressmaint]')
           and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[expressmaint]
GO

CREATE PROCEDURE expressmaint
(
   @database      sysname,                  -- database name | ALL_USER | ALL_SYSTEM
   @optype        varchar(7),               -- LOG | DB | DIFF | REINDEX | REORG | CHECKDB
   @backupwith    varchar(500) = NULL,      -- additional backup options
   @backupfldr    varchar(200) = NULL,      -- folder to write backup to
   @reportfldr    varchar(200) = NULL,      -- folder to write text report
   @verify        bit = 1,                  -- verify backup
   @verifywith    varchar(500) = NULL,      -- additional verify options
   @dbretainunit  varchar(10)  = NULL,      -- minutes | hours | days | weeks | months | copies
   @dbretainval   int = 1,                  -- specifies how many retainunits to keep backup
   @report        bit = 1,                  -- flag to indicate whether to generate report
   @rptretainunit varchar(10)  = NULL,      -- minutes | hours | days | weeks | months | copies
   @rptretainval  int = 1,                  -- specifies how many retainunits to keep reports
   @checkattrib   bit = 0,                  -- check if archive bit is cleared before deleting
   @delfirst      bit = 0,                  -- delete before backup (handy if space issues)
   @debug         bit = 0                   -- print commands to be executed
)
AS
/*
   expressmaint

   see http://www.sqldbatips.com/showarticle.asp?ID=27 for documentation

   Date           Author                  Notes
   24/07/2004     Jasper Smith            Initial release

*/
SET NOCOUNT ON
SET DATEFORMAT YMD

/************************
   VARIABLE DECLARATION
************************/

   DECLARE @fso             int
   DECLARE @file            int
   DECLARE @reportfilename  varchar(500)
   DECLARE @backupfilename  varchar(500)
   DECLARE @delfilename     varchar(500)
   DECLARE @cmd             varchar(650)
   DECLARE @backupfldrorig  varchar(200)
   DECLARE @databaseorig    sysname
   DECLARE @table           nvarchar(600)
   DECLARE @exists          varchar(5)
   DECLARE @err             int
   DECLARE @start           datetime
   DECLARE @finish          datetime
   DECLARE @runtime         datetime
   DECLARE @output          varchar(200)
   DECLARE @errormsg        varchar(210)
   DECLARE @datepart        nchar(2)
   DECLARE @execmd          nvarchar(1000)
   DECLARE @delcmd          nvarchar(1000)
   DECLARE @exemsg          varchar(8000)
   DECLARE @filecount       int             ; SET @filecount    = 0
   DECLARE @delcount        int             ; SET @delcount     = 0
   DECLARE @hr              int             ; SET @hr           = 0
```

```
   DECLARE @ret             int              ; SET @ret          = 0
   DECLARE @cmdret          int              ; SET @cmdret       = 0
   DECLARE @delbkflag       int              ; SET @delbkflag    = 0
   DECLARE @delrptflag      int              ; SET @delrptflag   = 0
   DECLARE @filecrt         int              ; SET @filecrt      = 0
   DECLARE @user            sysname          ; SET @user         = SUSER_SNAME()
   DECLARE @jobdt           datetime         ; SET @jobdt        = GETDATE()
   DECLARE @jobstart        char(12)         ;
   DECLARE @stage           int              ; SET @stage        = 1

   SET @jobstart = CONVERT(char(8),@jobdt,112)+LEFT(REPLACE(CONVERT(char(8),@jobdt,108),':',''),4)
   IF RIGHT(@reportfldr,1)<>'\' SET @reportfldr = @reportfldr + '\'
   IF RIGHT(@backupfldr,1)<>'\' SET @backupfldr = @backupfldr + '\'
   SET @backupfldrorig = @backupfldr
   SET @databaseorig = @database

   CREATE TABLE #files(filename varchar(255))
   CREATE TABLE #exists(exist int,isdir int,parent int)
   CREATE TABLE #databases(dbname sysname)

/*********************************
     INITIALIZE FSO IF @report = 1
*********************************/

   IF @report = 1
   BEGIN
      EXEC @hr=sp_OACreate 'Scripting.FileSystemObject',@fso OUT
      IF @hr <> 0
      BEGIN
         EXEC sp_OAGetErrorInfo @fso
         RAISERROR('Error creating File System Object',16,1)
         SET @ret = 1
       GOTO CLEANUP
      END
   END

/************************
       CHECK INPUT
************************/

   -- check SQL2005 or higher
   IF (select SUBSTRING(@@version,(CHARINDEX('-',@@version)+2),1))<9
        BEGIN
        RAISERROR('SQL2005 or higher is required for sp_expressmaint',16,1)
      SET @ret = 1
        GOTO CLEANUP
        END

   -- check sysadmin
   IF IS_SRVROLEMEMBER('sysadmin') = 0
        BEGIN
        RAISERROR('The current user %s is not a member of the sysadmin role',16,1,@user)
      SET @ret = 1
        GOTO CLEANUP
        END

   -- check database exists and is online
   IF @database NOT IN ('ALL_USER','ALL_SYSTEM')
   BEGIN
      IF (DB_ID(@database) IS NULL) OR ((select state from sys.databases where name = @database) <>0)
        BEGIN
        RAISERROR('Database %s is invalid or database status is not ONLINE',16,1,@database)
          SET @ret = 1
        GOTO CLEANUP
        END
   END

   -- check @optype is valid
   IF UPPER(@optype) NOT IN ('LOG','DB','DIFF','REINDEX','REORG','CHECKDB')
        BEGIN
        RAISERROR('%s is not a valid option for @optype',16,1,@optype)
      SET @ret = 1
        GOTO CLEANUP
        END

   -- check recovery mode is correct if trying log backup
   IF @database NOT IN ('ALL_USER','ALL_SYSTEM')
   BEGIN
```

```
      IF (@optype = 'LOG' and ((select recovery_model from sys.databases where name = @database) = 3))
        BEGIN
        RAISERROR('%s is not a valid option for database %s because it is in SIMPLE recovery
mode',16,1,@optype,@database)
          SET @ret = 1
        GOTO CLEANUP
        END
   END

   -- no log backups for system databases
   IF @database = 'ALL_SYSTEM'
   BEGIN
      IF @optype = 'LOG'
        BEGIN
        RAISERROR('%s is not a valid option for the option ALL_SYSTEM',16,1,@optype)
          SET @ret = 1
        GOTO CLEANUP
        END
   END

   -- check that @backupfldr exists on the server
   IF @optype NOT IN ('REINDEX','CHECKDB','REORG')
   BEGIN
      IF @report = 1
      BEGIN
         EXEC sp_OAMethod @fso,'FolderExists',@exists OUT,@backupfldr
         IF @exists <> 'True'
        BEGIN
               RAISERROR('The folder %s does not exist on this server',16,1,@backupfldr)
            SET @ret = 1
               GOTO CLEANUP
        END
      END
      ELSE
      BEGIN
         INSERT #exists
         EXEC master.dbo.xp_fileexist @backupfldr
         IF (SELECT MAX(isdir) FROM #exists)<>1
        BEGIN
               RAISERROR('The folder %s does not exist on this server',16,1,@backupfldr)
            SET @ret = 1
               GOTO CLEANUP
        END
      END
   END

   -- check that @reportfldr exists on the server
   IF @reportfldr IS NOT NULL or @report = 1
   BEGIN
      IF @report = 1
      BEGIN
         EXEC sp_OAMethod @fso,'FolderExists',@exists OUT,@reportfldr
         IF @exists <> 'True'
        BEGIN
               RAISERROR('The folder %s does not exist on this server',16,1,@reportfldr)
            SET @ret = 1
               GOTO CLEANUP
        END
      END
      ELSE
      BEGIN
         DELETE #exists
         INSERT #exists
         EXEC master.dbo.xp_fileexist @reportfldr
         IF (SELECT MAX(isdir) FROM #exists)<>1
        BEGIN
               RAISERROR('The folder %s does not exist on this server',16,1,@reportfldr)
            SET @ret = 1
               GOTO CLEANUP
        END
      END
   END

   -- check @dbretainunit is a vaild value
   IF @optype NOT IN ('REINDEX','CHECKDB','REORG')
   BEGIN
      IF UPPER(@dbretainunit) NOT IN ('MINUTES','HOURS','DAYS','WEEKS','MONTHS','COPIES')
        BEGIN
```

```
        RAISERROR('%s is not a valid value for @dbretainunit (''minutes | hours | days | weeks | months |
copies'')',16,1,@dbretainunit)
          SET @ret = 1
        GOTO CLEANUP
        END
   END

   --check @dbretainval is a vaild value
   IF @dbretainval<1
        BEGIN
        RAISERROR('%i is not a valid value for @dbretainval (must be >0)',16,1,@dbretainval)
      SET @ret = 1
        GOTO CLEANUP
        END

   -- check @rptretainunit is a vaild value if present
   IF UPPER(@rptretainunit) NOT IN ('MINUTES','HOURS','DAYS','WEEKS','MONTHS','COPIES') and @rptretainunit IS
NOT NULL
        BEGIN
        RAISERROR('%s is not a valid value for @rptretainunit (''minutes | hours | days | weeks | months |
copies'')',16,1,@rptretainunit)
      SET @ret = 1
        GOTO CLEANUP
        END

   --check @rptretainval is a vaild value
   IF @rptretainval<1
        BEGIN
        RAISERROR('%i is not a valid value for @rptretainval (must be >0)',16,1,@rptretainval)
      SET @ret = 1
        GOTO CLEANUP
        END

/***********************************
   list of databases to process
***********************************/

   IF @database IN ('ALL_USER','ALL_SYSTEM')
   BEGIN
      IF @database = 'ALL_USER'
         INSERT #databases(dbname)
         SELECT [name] from sys.databases where database_id > 4
         AND (@optype <> 'LOG' OR recovery_model <> '3')
      ELSE
         INSERT #databases(dbname)
         SELECT [name] from sys.databases where database_id in (1,3,4)
   END
   ELSE
      INSERT #databases(dbname) SELECT @database


/***********************************
   INITIALIZE REPORT IF @report = 1
***********************************/

   -- generate report filename
   SELECT @reportfilename = @reportfldr + REPLACE(@database,' ','_') +
   CASE WHEN UPPER(@optype) = 'DB'   THEN '_FullDBBackup_report_'
        WHEN UPPER(@optype) = 'DIFF' THEN '_DiffDBBackup_report_'
        WHEN UPPER(@optype) = 'LOG'  THEN '_LogBackup_report_'
        WHEN UPPER(@optype) = 'REINDEX'  THEN '_Reindex_report_'
        WHEN UPPER(@optype) = 'REORG'  THEN '_Reorg_report_'
        WHEN UPPER(@optype) = 'CHECKDB'  THEN '_CheckDB_report_'
   END + @jobstart + '.txt'

   -- if no report just set @reportfilename to NULL
   IF @report = 0 SET @reportfilename = NULL

   IF @debug = 1
   BEGIN
      PRINT '@reportfilename = ' + ISNULL(@reportfilename,'NULL')
   END

   IF @report = 1
   BEGIN
      -- create report file
      EXEC @hr=sp_OAMethod @fso, 'CreateTextFile',@file OUT, @reportfilename
      IF (@hr <> 0)
```

```
      BEGIN
         EXEC sp_OAGetErrorInfo @fso
         RAISERROR('Error creating log file',16,1)
         SET @ret = 1
       GOTO CLEANUP
      END
      ELSE
         -- set global flag to indicate we have created a report file
         SET @filecrt = 1

      -- write header
      EXEC sp_OAMethod @file,'WriteLine',NULL,''
      SET @output = 'Expressmaint utility, Logged on to SQL Server [' + @@SERVERNAME + '] as ' + '[' + @user +
']'
      IF @debug = 1 PRINT @output
      EXEC sp_OAMethod @file,'WriteLine',NULL,@output


      IF UPPER(@optype) NOT IN ('REINDEX','CHECKDB','REORG')
      BEGIN
         SET @output = 'Starting backup on ' + convert(varchar(25),getdate(),100)
      END
      IF UPPER(@optype) = 'CHECKDB'
      BEGIN
         SET @output = 'Starting CheckDB on ' + convert(varchar(25),getdate(),100)
      END
      IF UPPER(@optype) IN ('REINDEX','REORG')
      BEGIN
         SET @output = 'Starting Reindex on ' + convert(varchar(25),getdate(),100)
      END

      IF @debug = 1 PRINT @output
      EXEC sp_OAMethod @file,'WriteLine',NULL,@output
      EXEC sp_OAMethod @file,'WriteLine',NULL,''
   END

/************************
     BACKUP ACTIONS
************************/

   IF  UPPER(@optype) = 'CHECKDB' GOTO CHECK_DB
   IF  UPPER(@optype) IN ('REINDEX','REORG') GOTO REINDEX

   -- if @delfirst = 1  we need to delete prior backups that qualify
   IF @delfirst = 1 GOTO DELFIRST

   -- this label is so that we can return here after deleting files if @delfirst = 1
   DOBACKUP:

   DECLARE dcur CURSOR LOCAL FAST_FORWARD
   FOR SELECT dbname FROM #databases ORDER BY dbname
   OPEN dcur
   FETCH NEXT FROM dcur into @database
   WHILE @@FETCH_STATUS=0
   BEGIN

      -- set backup start time
      SET @start = GETDATE()

      -- write to text report
      IF @report = 1
      BEGIN
         SET @output = '[' + CAST(@stage as varchar(10)) + '] Database ' + @database + ': ' +
                       CASE WHEN UPPER(@optype) = 'DB'   THEN 'Full Backup '
                            WHEN UPPER(@optype) = 'DIFF' THEN 'Differential Backup '
                            WHEN UPPER(@optype) = 'LOG'  THEN 'Log Backup '
                       END + 'starting at ' + CONVERT(varchar(25),@start,100)
         IF @debug = 1 PRINT @output
         EXEC sp_OAMethod @file,'WriteLine',NULL,@output
      END

      -- backup subfolder
      SET @execmd = 'IF NOT EXIST "' + @backupfldrorig + @database + '\" MKDIR "' + @backupfldrorig + @database
+ '\"'
      EXEC master.dbo.xp_cmdshell @execmd,no_output
      SET @backupfldr = @backupfldrorig + @database + '\'

      SELECT @backupfilename = @backupfldr + REPLACE(@database,' ','_') +
```

```
      CASE WHEN UPPER(@optype) = 'DB'   THEN '_FullDBBackup_'
           WHEN UPPER(@optype) = 'DIFF' THEN '_DiffDBBackup_'
           WHEN UPPER(@optype) = 'LOG'  THEN '_LogBackup_'
      END + @jobstart +
      CASE WHEN UPPER(@optype) = 'LOG' THEN '.TRN' ELSE '.BAK' END

      /************************
            FULL BACKUP
      ************************/

      IF UPPER(@optype) = 'DB'
      BEGIN

         SET @execmd = N'BACKUP DATABASE [' + @database + '] TO DISK = ''' + @backupfilename + '''' +
                       CASE WHEN @backupwith IS NULL THEN '' ELSE (' WITH ' + @backupwith) END

         BEGIN TRY

            EXEC(@execmd)

         END TRY
         BEGIN CATCH -- backup failure
            SELECT @err = @@ERROR,@ret = @err
            SELECT @errormsg = 'Full backup of database ' + @database + ' failed with error : ' +  CAST(@err as
varchar(10))
            SET @output = SPACE(4) + '*** ' + @errormsg + ' ***'
            IF @debug = 1 PRINT @output
            IF @report = 1
            BEGIN
               EXEC sp_OAMethod @file,'WriteLine',NULL,@output
               SET @output = SPACE(4) + 'Refer to SQL Error Log and NT Event Log for further details'
               EXEC sp_OAMethod @file,'WriteLine',NULL,@output
               EXEC sp_OAMethod @file,'WriteLine',NULL,''
            END
            CLOSE dcur
            DEALLOCATE dcur
            GOTO CLEANUP

         END CATCH

         -- backup success
         SET @finish = GETDATE()
         SET @output = SPACE(4) + 'Database backed up to ' + @backupfilename
         IF @debug = 1 PRINT @output
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
         END

         --calculate backup runtime
         SET @runtime = (@finish - @start)
         SET @output = SPACE(4) + 'Full database backup completed in '
                     + CAST(DATEPART(hh,@runtime) as varchar(2)) + ' hour(s) '
                     + CAST(DATEPART(mi,@runtime) as varchar(2)) + ' min(s) '
                     + CAST(DATEPART(ss,@runtime) as varchar(2)) + ' second(s)'
         IF @debug = 1 PRINT @output
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
         END

      END


      /************************
         DIFFERENTIAL BACKUP
      ************************/

      IF UPPER(@optype) = 'DIFF'
      BEGIN

         SET @execmd = N'BACKUP DATABASE [' + @database + '] TO DISK = ''' +
                        @backupfilename + ''' WITH DIFFERENTIAL' +
                       CASE WHEN @backupwith IS NULL THEN '' ELSE (' , ' + @backupwith) END

         BEGIN TRY
```

```
            EXEC(@execmd)

         END TRY
         BEGIN CATCH -- backup failure

            SELECT @err = @@ERROR,@ret = @err
            SELECT @errormsg = 'Differential backup of database ' + @database + ' failed with error : ' +
CAST(@err as varchar(10))
            SET @output = SPACE(4) + '*** ' + @errormsg + ' ***'
            IF @debug = 1 PRINT @output
            IF @report = 1
            BEGIN
               EXEC sp_OAMethod @file,'WriteLine',NULL,@output
               SET @output = SPACE(4) + 'Refer to SQL Error Log and NT Event Log for further details'
               EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            END
            CLOSE dcur
            DEALLOCATE dcur
            GOTO CLEANUP

         END CATCH

         -- backup success
         SET @finish = GETDATE()
         SET @output = SPACE(4) + 'Database backed up to ' + @backupfilename
         IF @debug = 1 PRINT @output
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
         END

         --calculate backup runtime
         SET @runtime = (@finish - @start)
         SET @output = SPACE(4) + 'Differential database backup completed in '
                     + CAST(DATEPART(hh,@runtime) as varchar(2)) + ' hour(s) '
                     + CAST(DATEPART(mi,@runtime) as varchar(2)) + ' min(s) '
                     + CAST(DATEPART(ss,@runtime) as varchar(2)) + ' second(s)'
         IF @debug = 1 PRINT @output
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
         END

      END

      /************************
            LOG BACKUP
      ************************/

      IF UPPER(@optype) = 'LOG'
      BEGIN

         SET @execmd = N'BACKUP LOG [' + @database + '] TO DISK = ''' + @backupfilename + '''' +
                       CASE WHEN @backupwith IS NULL THEN '' ELSE (' WITH ' + @backupwith) END

         BEGIN TRY

            EXEC(@execmd)

         END TRY
         BEGIN CATCH -- backup failure

            SELECT @err = @@ERROR,@ret = @err
            SELECT @errormsg = 'Log backup of database ' + @database + ' failed with error : ' +  CAST(@err as
varchar(10))
            SET @output = SPACE(4) + '*** ' + @errormsg + ' ***'
            IF @debug = 1 PRINT @output
            IF @report = 1
            BEGIN
               EXEC sp_OAMethod @file,'WriteLine',NULL,@output
               SET @output = SPACE(4) + 'Refer to SQL Error Log and NT Event Log for further details'
               EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            END
            CLOSE dcur
            DEALLOCATE dcur
            GOTO CLEANUP
```

```
         END CATCH

         -- backup success
         SET @finish = GETDATE()
         SET @output = SPACE(4) + 'Log backed up to ' + @backupfilename
         IF @debug = 1 PRINT @output
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
         END

         --calculate backup runtime
         SET @runtime = (@finish - @start)
         SET @output = SPACE(4) + 'Log backup completed in '
                     + CAST(DATEPART(hh,@runtime) as varchar(2)) + ' hour(s) '
                     + CAST(DATEPART(mi,@runtime) as varchar(2)) + ' min(s) '
                     + CAST(DATEPART(ss,@runtime) as varchar(2)) + ' second(s)'
         IF @debug = 1 PRINT @output
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
         END

      END

      SET @stage = (@stage + 1)

      FETCH NEXT FROM dcur into @database
   END

   CLOSE dcur
   DEALLOCATE dcur

   /************************
         VERIFY BACKUP
   ************************/

   IF @verify = 1
   BEGIN

      DECLARE dcur CURSOR LOCAL FAST_FORWARD
      FOR SELECT dbname FROM #databases ORDER BY dbname
      OPEN dcur
      FETCH NEXT FROM dcur into @database
      WHILE @@FETCH_STATUS=0
      BEGIN

         SELECT @backupfilename = @backupfldrorig + @database + '\' + REPLACE(@database,' ','_') +
         CASE WHEN UPPER(@optype) = 'DB'   THEN '_FullDBBackup_'
              WHEN UPPER(@optype) = 'DIFF' THEN '_DiffDBBackup_'
              WHEN UPPER(@optype) = 'LOG'  THEN '_LogBackup_'
         END + @jobstart +
         CASE WHEN UPPER(@optype) = 'LOG' THEN '.TRN' ELSE '.BAK' END

         SET @start = GETDATE()

         -- write to text report
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
            SET @output = '[' + CAST(@stage as varchar(10)) + '] Database ' + @database + ': Verify Backup
File...'
            IF @debug = 1 PRINT @output
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
         END

         SET @execmd = N'RESTORE VERIFYONLY FROM DISK = ''' + @backupfilename + '''' +
                       CASE WHEN @verifywith IS NULL THEN '' ELSE (' WITH ' + @verifywith) END

         BEGIN TRY

            EXEC(@execmd)

         END TRY
         BEGIN CATCH

            SELECT @err = @@ERROR,@ret = @err
```

```
            SET @errormsg = 'Verify of ' + @backupfilename + ' failed with Native Error : ' + CAST(@err as
varchar(10))
            SET @output = SPACE(4) + '*** ' + @errormsg + ' ***'
            IF @debug = 1 PRINT @output
            IF @report = 1
            BEGIN
               EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            END
            CLOSE dcur
            DEALLOCATE dcur
            GOTO CLEANUP

         END CATCH

         -- verify success
         SET @finish = GETDATE()
         SET @output = SPACE(4) + 'Backup file ' + @backupfilename + ' verified'
         IF @debug = 1 PRINT @output

         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
         END

         --calculate verify runtime
         SET @runtime = (@finish - @start)
         SET @output = SPACE(4) + 'Verify backup completed in '
                     + CAST(DATEPART(hh,@runtime) as varchar(2)) + ' hour(s) '
                     + CAST(DATEPART(mi,@runtime) as varchar(2)) + ' min(s) '
                     + CAST(DATEPART(ss,@runtime) as varchar(2)) + ' second(s)'
         IF @debug = 1 PRINT @output
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
         END

         SET @stage = (@stage + 1)
         FETCH NEXT FROM dcur into @database
      END

      CLOSE dcur
      DEALLOCATE dcur
   END

/************************
    DELETE OLD FILES
************************/


   -- we have already deleted files so skip to the end
   IF @delfirst = 1 GOTO CLEANUP

   -- this label is so that we can delete files prior to backup if @delfirst = 1
   DELFIRST:

   /************************
      DELETE OLD BACKUPS
   ************************/

   SET @datepart = CASE
      WHEN UPPER(@dbretainunit) = 'MINUTES' THEN N'mi'
      WHEN UPPER(@dbretainunit) = 'HOURS'   THEN N'hh'
      WHEN UPPER(@dbretainunit) = 'DAYS'    THEN N'dd'
      WHEN UPPER(@dbretainunit) = 'WEEKS'   THEN N'ww'
      WHEN UPPER(@dbretainunit) = 'MONTHS'  THEN N'yy'
   END

   IF @debug = 1 PRINT '@datepart for backups = ' + @datepart

   -- write to text report
   IF @report = 1
   BEGIN
      EXEC sp_OAMethod @file,'WriteLine',NULL,''
   END
   SET @output = '[' + CAST(@stage as varchar(10)) + '] Delete Old Backup Files...'
   IF @debug = 1 PRINT @output
   IF @report = 1
   BEGIN
```

```
      EXEC sp_OAMethod @file,'WriteLine',NULL,@output
   END

   DECLARE dcur CURSOR LOCAL FAST_FORWARD
   FOR SELECT dbname FROM #databases ORDER BY dbname
   OPEN dcur
   FETCH NEXT FROM dcur into @database
   WHILE @@FETCH_STATUS=0
   BEGIN

      SET @backupfldr = + @backupfldrorig + @database + '\'
      SELECT @backupfilename = @backupfldr + REPLACE(@database,' ','_') +
      CASE WHEN UPPER(@optype) = 'DB'   THEN '_FullDBBackup_'
           WHEN UPPER(@optype) = 'DIFF' THEN '_DiffDBBackup_'
           WHEN UPPER(@optype) = 'LOG'  THEN '_LogBackup_'
      END + @jobstart +
      CASE WHEN UPPER(@optype) = 'LOG' THEN '.TRN' ELSE '.BAK' END

      -- load files in @backupfldr
      IF @checkattrib = 1
         SET @cmd = 'dir /B /A-D-A /OD "' + @backupfldr + REPLACE(@database,' ','_') +
         CASE WHEN UPPER(@optype) = 'DB' THEN '_FullDBBackup_'
              WHEN UPPER(@optype) = 'DIFF' THEN '_DiffDBBackup_'
              WHEN UPPER(@optype) = 'LOG'  THEN '_LogBackup_' END + '*' +
         CASE WHEN UPPER(@optype) = 'LOG' THEN '.TRN' ELSE '.BAK' END + '"'
      ELSE
         SET @cmd = 'dir /B /A-D /OD "' + @backupfldr + REPLACE(@database,' ','_') +
         CASE WHEN UPPER(@optype) = 'DB' THEN '_FullDBBackup_'
              WHEN UPPER(@optype) = 'DIFF' THEN '_DiffDBBackup_'
              WHEN UPPER(@optype) = 'LOG'  THEN '_LogBackup_' END + '*' +
         CASE WHEN UPPER(@optype) = 'LOG' THEN '.TRN' ELSE '.BAK' END + '"'

      IF @debug = 1 PRINT '@cmd = ' + @cmd

      DELETE #files
      INSERT #files EXEC master.dbo.xp_cmdshell @cmd
      DELETE #files WHERE filename IS NULL or filename =
ISNULL(REPLACE(@backupfilename,@backupfldr,''),'nothing')

      IF @debug = 1 SELECT * FROM #files

      -- get count of files that match pattern
      SELECT @filecount = COUNT(*) from #files
      WHERE PATINDEX('%File Not Found%',filename) = 0
      AND PATINDEX('%The system cannot find%',filename) = 0

      -- remove files that don't meet retention criteria if there are any files that match pattern
      IF UPPER(@dbretainunit) <> 'COPIES'
      BEGIN
         IF @filecount>0
         BEGIN
            SET @delcmd = N'DELETE #files WHERE DATEADD(' + @datepart + N',' + CAST(@dbretainval as
nvarchar(10)) + N',' +
                  'CONVERT(datetime,(SUBSTRING(SUBSTRING(filename,((LEN(filename)-
CHARINDEX(''_'',REVERSE(filename)))+2),12),7,2) +''/''
                  + SUBSTRING(SUBSTRING(filename,((LEN(filename)-
CHARINDEX(''_'',REVERSE(filename)))+2),12),5,2) +''/''
                  + SUBSTRING(SUBSTRING(filename,((LEN(filename)-
CHARINDEX(''_'',REVERSE(filename)))+2),12),1,4) +'' ''
                  + SUBSTRING(SUBSTRING(filename,((LEN(filename)-
CHARINDEX(''_'',REVERSE(filename)))+2),12),9,2) +'':''
                  + SUBSTRING(SUBSTRING(filename,((LEN(filename)-
CHARINDEX(''_'',REVERSE(filename)))+2),12),11,2)),103)) > ''' + CAST(@jobdt as nvarchar(25)) + N''''

            IF @debug = 1 PRINT '@delcmd=' + @delcmd
            EXEC master.dbo.sp_executesql @delcmd

            SELECT @delcount = COUNT(*) from #files
         END
         ELSE
         BEGIN
            SELECT @delcount = 0
         END
      END
      ELSE  -- number of copies not date based (include current backup that's not in #files)
      BEGIN
         IF @filecount>0
         BEGIN
```

```
            IF @dbretainval>1
            BEGIN
               SET @delcmd = N'DELETE #files WHERE filename IN(SELECT TOP ' + CAST((@dbretainval-1) as
nvarchar(10)) +
                             N' filename FROM #files ORDER BY substring(filename,((len(filename)+2)-
charindex(''_'',reverse(filename))),12) DESC)'

               IF @debug = 1 PRINT '@delcmd=' + @delcmd
               EXEC master.dbo.sp_executesql @delcmd
            END

            SELECT @delcount = COUNT(*) from #files

         END
         ELSE
         BEGIN
            SELECT @delcount = 0
         END
      END

      IF @debug = 1 PRINT '@delcount = ' + STR(@delcount)

      -- if there are any matching files
      IF @filecount>0
      BEGIN
         -- are there any files that need deleting
         IF @delcount>0
         BEGIN
            DECLARE FCUR CURSOR FORWARD_ONLY FOR
            SELECT * FROM #files
            OPEN FCUR
            FETCH NEXT FROM FCUR INTO @delfilename
            WHILE @@FETCH_STATUS=0
            BEGIN
               SET @cmd = 'DEL /Q "' + @backupfldr + @delfilename + '"'
               EXEC @cmdret = master.dbo.xp_cmdshell @cmd,no_output

               -- log failure to delete but don't abort procedure
               IF @cmdret<>0
               BEGIN
                  SET @output = SPACE(4) + '*** Error: Failed to delete file ' + @backupfldr + @delfilename + '
***'
                  IF @debug = 1 PRINT @output
                  IF @report = 1
                  BEGIN
                     EXEC sp_OAMethod @file,'WriteLine',NULL,@output
                  END
                  SELECT @delbkflag = 1 , @cmdret = 0, @delcount = (@delcount-1)
               END
               ELSE
               BEGIN
                  SET @output = SPACE(4) + 'Deleted file ' + @backupfldr + @delfilename
                  IF @debug = 1 PRINT @output
                  IF @report = 1
                  BEGIN
                     EXEC sp_OAMethod @file,'WriteLine',NULL,@output
                  END
               END

               FETCH NEXT FROM FCUR INTO @delfilename
            END
            CLOSE FCUR
            DEALLOCATE FCUR
         END
      END

      -- write to text report
      SET @output = SPACE(4) + CAST(@delcount as varchar(10)) + ' file(s) deleted.'
      IF @debug = 1 PRINT @output
      IF @report = 1
      BEGIN
         EXEC sp_OAMethod @file,'WriteLine',NULL,@output
         EXEC sp_OAMethod @file,'WriteLine',NULL,''
      END

      FETCH NEXT FROM dcur into @database
   END
```

```
   CLOSE dcur
   DEALLOCATE dcur

   -- clear temporary table and variables
   DELETE #files
   SET @cmd = ''
   SET @delcmd = ''
   SET @delfilename = ''
   SET @datepart = ''
   SET @filecount = 0
   SET @delcount = 0
   SET @cmdret = 0
   SET @stage = @stage + 1


   /************************
      DELETE OLD REPORTS
   ************************/

   DELREPORTS:

   IF @rptretainunit IS NOT NULL
   BEGIN
      SET @datepart = CASE
         WHEN UPPER(@rptretainunit) = 'MINUTES' THEN N'mi'
         WHEN UPPER(@rptretainunit) = 'HOURS'   THEN N'hh'
         WHEN UPPER(@rptretainunit) = 'DAYS'    THEN N'dd'
         WHEN UPPER(@rptretainunit) = 'WEEKS'   THEN N'ww'
         WHEN UPPER(@rptretainunit) = 'MONTHS'  THEN N'yy'
   END

   IF @debug = 1 PRINT '@datepart for reports = ' + @datepart

   -- write to text report
   SET @output = '[' + CAST(@stage as varchar(10)) + '] Delete Old Report Files...'
   IF @debug = 1 PRINT @output
   IF @report = 1
   BEGIN
      EXEC sp_OAMethod @file,'WriteLine',NULL,@output
   END

   -- load files in @reportfldr
   SET @cmd = 'dir /B /A-D /OD "' + @reportfldr + REPLACE(@databaseorig,' ','_') +
   CASE WHEN UPPER(@optype) = 'DB' THEN '_FullDBBackup_report_'
        WHEN UPPER(@optype) = 'DIFF' THEN '_DiffDBBackup_report_'
        WHEN UPPER(@optype) = 'REINDEX'  THEN '_Reindex_report_'
        WHEN UPPER(@optype) = 'CHECKDB'  THEN '_CheckDB_report_'
        WHEN UPPER(@optype) = 'REORG'  THEN '_Reorg_report_'
        WHEN UPPER(@optype) = 'LOG'  THEN '_LogBackup_report_' END + '*.txt"'

   IF @debug = 1 PRINT '@cmd = ' + @cmd

   INSERT #files EXEC master.dbo.xp_cmdshell @cmd
   DELETE #files WHERE filename IS NULL

   IF @debug = 1 SELECT * FROM #files

   -- get count of files that match pattern
   SELECT @filecount = COUNT(*) from #files
   WHERE PATINDEX('%File Not Found%',filename) = 0
   AND PATINDEX('%The system cannot find%',filename) = 0

   -- remove files that don't meet retention criteria if there are any files that match pattern
   IF UPPER(@rptretainunit) <> 'COPIES'
   BEGIN
      IF @filecount>0
      BEGIN
         SET @delcmd = N'DELETE #files WHERE DATEADD(' + @datepart + N',' + CAST(@rptretainval as nvarchar(10))
+ N',' +
               'CONVERT(datetime,(SUBSTRING(SUBSTRING(filename,((LEN(filename)-
CHARINDEX(''_'',REVERSE(filename)))+2),12),7,2) +''/''
               + SUBSTRING(SUBSTRING(filename,((LEN(filename)-CHARINDEX(''_'',REVERSE(filename)))+2),12),5,2)
+''/''
               + SUBSTRING(SUBSTRING(filename,((LEN(filename)-CHARINDEX(''_'',REVERSE(filename)))+2),12),1,4)
+'' ''
               + SUBSTRING(SUBSTRING(filename,((LEN(filename)-CHARINDEX(''_'',REVERSE(filename)))+2),12),9,2)
+'':''
```

```
                + SUBSTRING(SUBSTRING(filename,((LEN(filename)-
CHARINDEX(''_'',REVERSE(filename)))+2),12),11,2)),103)) > ''' + CAST(@jobdt as nvarchar(25)) + N''''

         IF @debug = 1 PRINT '@delcmd=' + @delcmd
         EXEC master.dbo.sp_executesql @delcmd

         SELECT @delcount = COUNT(*) from #files
      END
      ELSE
      BEGIN
         SELECT @delcount = 0
      END
   END
   ELSE  -- number of copies not date based
   BEGIN
      IF @filecount>0
      BEGIN
         SET @delcmd = N'DELETE #files WHERE filename IN(SELECT TOP ' + CAST(@rptretainval as nvarchar(10)) +
                       N' filename FROM #files ORDER BY substring(filename,((len(filename)+2)-
charindex(''_'',reverse(filename))),12) DESC)'

         IF @debug = 1 PRINT '@delcmd=' + @delcmd
         EXEC master.dbo.sp_executesql @delcmd

         SELECT @delcount = COUNT(*) from #files
      END
      ELSE
      BEGIN
         SELECT @delcount = 0
      END
   END

   IF @debug = 1 PRINT STR(@delcount)

   -- if there are any matching files
   IF @filecount>0
   BEGIN
      -- are there any files that need deleting
      IF @delcount>0
      BEGIN
         DECLARE FCUR CURSOR FORWARD_ONLY FOR
         SELECT * FROM #files
         OPEN FCUR
         FETCH NEXT FROM FCUR INTO @delfilename
         WHILE @@FETCH_STATUS=0
         BEGIN
            SET @cmd = 'DEL /Q "' + @reportfldr + @delfilename + '"'
            EXEC @cmdret = master.dbo.xp_cmdshell @cmd,no_output

            -- log failure to delete but don't abort procedure
            IF @cmdret<>0
            BEGIN

               SET @output = SPACE(4) + '*** Error: Failed to delete file ' + @reportfldr + @delfilename + '
***'
               IF @debug = 1 PRINT @output
               IF @report = 1
               BEGIN
                  EXEC sp_OAMethod @file,'WriteLine',NULL,@output
               END
               SELECT @delrptflag = 1 , @cmdret = 0, @delcount = (@delcount-1)
            END
            BEGIN
               SET @output = SPACE(4) + 'Deleted file ' + @reportfldr + @delfilename
               IF @debug = 1 PRINT @output
               IF @report = 1
               BEGIN
                  EXEC sp_OAMethod @file,'WriteLine',NULL,@output
               END
            END

            FETCH NEXT FROM FCUR INTO @delfilename
         END
         CLOSE FCUR
         DEALLOCATE FCUR
      END
   END
```

```
   -- write to text report
   SET @output = SPACE(4) + CAST(@delcount as varchar(10)) + ' file(s) deleted.'
   IF @debug = 1 PRINT @output
   IF @report = 1
   BEGIN
      EXEC sp_OAMethod @file,'WriteLine',NULL,@output
      EXEC sp_OAMethod @file,'WriteLine',NULL,''
   END

   -- update stage
   SET @stage = @stage + 1
   END
   -- if we got here due to @delfirst = 1 go back and do the backups
   IF @delfirst = 1
      GOTO DOBACKUP
   ELSE
      GOTO CLEANUP


/************************
        CHECKDB
************************/

   CHECK_DB:

   IF @optype = 'CHECKDB'
   BEGIN

      DECLARE dcur CURSOR LOCAL FAST_FORWARD
      FOR SELECT dbname FROM #databases ORDER BY dbname
      OPEN dcur
      FETCH NEXT FROM dcur into @database
      WHILE @@FETCH_STATUS=0
      BEGIN

         -- write to text report
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
            SET @output = '[' + CAST(@stage as varchar(10)) + '] Database ' + @database + ': Check Data and
Index Linkage...'
            IF @debug = 1 PRINT @output
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
         END

         -- set backup start time
         SET @start = GETDATE()

         SET @execmd = N'DBCC CHECKDB([' + @database + N']) WITH NO_INFOMSGS'
         IF @debug = 1 PRINT 'DBCC Command : ' + @execmd

         BEGIN TRY

            EXEC(@execmd)

         END TRY
         BEGIN CATCH

            SELECT @err = @@ERROR,@ret = @err
            SET @errormsg = 'CheckDB of ' + @database + ' failed with Native Error : ' + CAST(@err as
varchar(10))
            SET @output = SPACE(4) + '*** ' + @errormsg + ' ***'
            PRINT @output
            IF @report = 1
            BEGIN
               EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            END
            CLOSE dcur
            DEALLOCATE dcur
            GOTO CLEANUP

         END CATCH

         SET @finish = GETDATE()

         --calculate checkdb runtime
         SET @runtime = (@finish - @start)
         SET @output = SPACE(4) + 'CheckDB completed in '
```

```
                    + CAST(DATEPART(hh,@runtime) as varchar(2)) + ' hour(s) '
                    + CAST(DATEPART(mi,@runtime) as varchar(2)) + ' min(s) '
                    + CAST(DATEPART(ss,@runtime) as varchar(2)) + ' second(s)'
         IF @debug = 1 PRINT @output
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
         END
         SET @stage = (@stage + 1)
         FETCH NEXT FROM dcur into @database

      END

      CLOSE dcur
      DEALLOCATE dcur

      -- delete reports
      IF @report = 1
      BEGIN
         EXEC sp_OAMethod @file,'WriteLine',NULL,''
      END
      GOTO DELREPORTS
   END

/************************
     REINDEX/REORG
************************/

   REINDEX:

   IF @optype in ('REINDEX','REORG')
   BEGIN

      DECLARE dcur CURSOR LOCAL FAST_FORWARD
      FOR SELECT dbname FROM #databases ORDER BY dbname
      OPEN dcur
      FETCH NEXT FROM dcur into @database
      WHILE @@FETCH_STATUS=0
      BEGIN

         -- write to text report
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
            IF @optype = 'REINDEX'
               SET @output = '[' + CAST(@stage as varchar(10)) + '] Database ' + @database + ': Index Rebuild
(using original fillfactor)...'
            ELSE
               SET @output = '[' + CAST(@stage as varchar(10)) + '] Database ' + @database + ': Index
Reorganize...'

            IF @debug = 1 PRINT @output
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
         END

         -- set start time
         SET @start = GETDATE()

         -- all user tables
         CREATE TABLE #tables(tablename sysname)
         EXEC(N'INSERT #tables(tablename) SELECT DISTINCT(''['' + s.[name] + ''].['' + t.[name] + '']'') FROM ['
+ @database + N'].sys.tables t ' +
              N'JOIN [' + @database + N'].sys.schemas s on t.schema_id=s.schema_id ' +
              N'JOIN [' + @database + N'].sys.indexes i on t.object_id=i.object_id ' +
              N'WHERE t.is_ms_shipped = 0 AND i.type>0')

         DECLARE tcur CURSOR LOCAL FAST_FORWARD
         FOR SELECT tablename FROM #tables ORDER BY tablename
         OPEN tcur
         FETCH NEXT FROM tcur INTO @table
         WHILE @@FETCH_STATUS = 0
         BEGIN

            IF @report = 1
            BEGIN
               IF @optype = 'REINDEX'
```

```
                  SET @output = SPACE(4) + N'Rebuilding indexes for table ' + @table
               ELSE
                  SET @output = SPACE(4) + N'Reorganizing indexes for table ' + @table

               EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            END


            IF @optype = 'REINDEX'
               SET @execmd = N'ALTER INDEX ALL ON [' + @database + N'].' + @table + N' REBUILD'
            ELSE
               SET @execmd = N'ALTER INDEX ALL ON [' + @database + N'].' + @table + N' REORGANIZE'

            IF @debug = 1 PRINT 'Reindex Command : ' + @execmd

            BEGIN TRY

               EXEC(@execmd)

            END TRY
            BEGIN CATCH

               SELECT @err = @@ERROR,@ret = @err
               SET @errormsg = 'Rebuild of indexes on [' + @database + N'].' + @table + ' failed with Native
Error : ' + CAST(@err as varchar(10))
               SET @output = SPACE(4) + '*** ' + @errormsg + ' ***'
               PRINT @output
               IF @report = 1
               BEGIN
                  EXEC sp_OAMethod @file,'WriteLine',NULL,@output
               END
               CLOSE tcur
               DEALLOCATE tcur
               DROP TABLE #tables
               GOTO CLEANUP

            END CATCH

            FETCH NEXT FROM tcur INTO @table
         END

         CLOSE tcur
         DEALLOCATE tcur

         SET @finish = GETDATE()

         --calculate runtime
         SET @runtime = (@finish - @start)
         SET @output = SPACE(4) + 'Index maintenance completed in '
                     + CAST(DATEPART(hh,@runtime) as varchar(2)) + ' hour(s) '
                     + CAST(DATEPART(mi,@runtime) as varchar(2)) + ' min(s) '
                     + CAST(DATEPART(ss,@runtime) as varchar(2)) + ' second(s)'
         IF @debug = 1 PRINT @output
         IF @report = 1
         BEGIN
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
            EXEC sp_OAMethod @file,'WriteLine',NULL,@output
            EXEC sp_OAMethod @file,'WriteLine',NULL,''
         END

         DROP TABLE #tables

         SET @stage = (@stage + 1)
         FETCH NEXT FROM dcur into @database

      END

      CLOSE dcur
      DEALLOCATE dcur

      -- delete reports
      GOTO DELREPORTS
   END


/************************
         CLEAN UP
************************/
```

```
   CLEANUP:

   DROP TABLE #files
   DROP TABLE #exists
   DROP TABLE #databases

   -- if we encountered errors deleting old backups return failure
   IF @delbkflag<>0
   BEGIN
      SET @errormsg = 'Expressmaint encountered errors deleting old backup files' + CHAR(13)
                    + CASE WHEN @report = 1 THEN ('Please see ' + @reportfilename + CHAR(13) + ' for further
details') ELSE '' END
      RAISERROR(@errormsg,16,1)
      SET @ret = 1
   END

   -- if we encountered errors deleting old reports return failure
   IF (@delrptflag<>0 AND @delbkflag = 0)
   BEGIN
      SET @errormsg = 'Expressmaint encountered errors deleting old report files' + CHAR(13)
                    + CASE WHEN @report = 1 THEN ('Please see ' + @reportfilename + CHAR(13) + ' for further
details') ELSE '' END
      RAISERROR(@errormsg,16,1)
      SET @ret = 1
   END

   -- if we created a file make sure we write trailer and destroy object
   IF @filecrt = 1
   BEGIN
      -- write final part of report
      EXEC sp_OAMethod @file,'WriteLine',NULL,''
      SET @output = 'Expressmaint processing finished at ' + CONVERT(varchar(25),GETDATE(),100)
                  + ' (Return Code : ' + CAST(@ret as varchar(10)) + ')'
      IF @debug = 1 PRINT @output
      EXEC sp_OAMethod @file,'WriteLine',NULL,@output
      EXEC sp_OAMethod @file,'WriteLine',NULL,''

      -- destroy file object
      EXEC @hr=sp_OADestroy @file
      IF @hr <> 0 EXEC sp_OAGetErrorInfo @file
   END

   IF @report = 1
   BEGIN
      EXEC @hr=sp_OADestroy @fso
      IF @hr <> 0 EXEC sp_OAGetErrorInfo @fso
   END

RETURN @ret
GO
```