

The BACKUP command

Use the BACKUP command with the SQL Backup Pro -SQL parameter to back up one or more databases, transaction logs, or filegroups using the command line or extended stored procedure.

- [Syntax](#) provides the grammar for the BACKUP command.
- [Arguments](#) describes the arguments for the BACKUP command.
- [WITH options](#) describes the options that can be used in the BACKUP command.
- [Examples](#) provides examples of using the command line and extended stored procedure to create backups with a range of options.



When using the extended stored procedure, the parameter or set of parameters (such as -SQL) must be delimited by single quotes. Therefore, wherever a single quote is used for the arguments below, for the extended stored procedure you must use **two** single quotes so that SQL Server does not interpret it as a string delimiter. See [Using the extended stored procedure](#) for more information.

Syntax

The following arguments are available only with SQL Server 2005 and later:

- CHECKSUM / NO_CHECKSUM
- CONTINUE_AFTER_ERROR / STOP_ON_ERROR
- COPY_ONLY
- READ_WRITE_FILEGROUPS

Backing up a database to a single file or split into multiple files

```
BACKUP DATABASE { database_name }
[FILE = { 'logical_file_name' } [ ,...n ] | FILEGROUP = { 'logical_filegroup_name' } ] [ ,...n ]
TO { DISK } = { 'physical_backup_device_name' | '<AUTO>' } [ ,...n ]
[ WITH
    [ [ , ] { CHECK_PREFERRED_AG_REPLICA } ]
    [ [ , ] { CHECKSUM | NO_CHECKSUM } ]
    [ [ , ] COMPRESSION = { 0 | 1 | 2 | 3 | 4 } ]
    [ [ , ] { CONTINUE_AFTER_ERROR | STOP_ON_ERROR } ]
    [ [ , ] COPY_ONLY ]
    [ [ , ] COPYTO = { 'target_folder_name' } [ ,...n ] ]
    [ [ , ] COPYTO_HOSTED ]
    [ [ , ] DESCRIPTION = { 'text' } ]
    [ [ , ] DIFFERENTIAL ]
    [ [ , ] DISKRETRYCOUNT = { n } ]
    [ [ , ] DISKRETRYINTERVAL = { seconds } ]
    [ [ , ] ERASEFILES | ERASEFILES_ATSTART | ERASEFILES_PRIMARY = { days | hours{h} | except latest{b} } ]
    [ [ , ] ERASEFILES_REMOTE | ERASEFILES_SECONDARY = { days | hours{h} | except latest{b} } ]
    [ [ , ] FILECOUNT = { n } ]
    [ [ , ] FILEOPTIONS = { 1 | 2 | 3 | 4 | 5 | 6 | 7 } ]
    [ [ , ] INIT ]
    [ [ , ] KEYSIZE = { 128 | 256 } ]
    [ [ , ] LOG_ONERROR ]
    [ [ , ] LOG_ONERRORONLY ]
    [ [ , ] LOGTO = { 'target_folder_name' | 'file_name' } ]
    [ [ , ] MAILTO = { 'recipients' } ]
    [ [ , ] MAILTO_NOLOG ]
    [ [ , ] MAILTO_ONERROR = { 'recipients' } ]
    [ [ , ] MAILTO_ONERRORONLY = { 'recipients' } ]
    [ [ , ] MAXDATABLOCK = { 65536 | 131072 | ... | 2097152 } ]
    [ [ , ] MAXTRANSFERSIZE = { 65536 | 131072 | ... | 1048576 } ]
    [ [ , ] MIRRORFILE = { 'physical_backup_device_name' } ] [ ,...n ]
    [ [ , ] NAME = { 'backup_set_name' } ]
    [ [ , ] NOCOMPRESSWRITE | NOWRITE ]
    [ [ , ] NOLOG ]
    [ [ , ] PASSWORD = { 'password' | 'FILE:file_path' } ]
    [ [ , ] SECONDARY_REPLICA_COPY_ONLY ]
    [ [ , ] SINGLERESULTSET ]
    [ [ , ] SQLCOMPRESSION ]
    [ [ , ] THREADCOUNT = { n } ]
    [ [ , ] THREADPRIORITY = { 0 | 1 | 2 | 3 | 4 | 5 | 6 } ]
    [ [ , ] USEQUEUEDCOPY ]
    [ [ , ] USESIMPLECOPY ]
    [ [ , ] VERIFY ]
    [ [ , ] VERIFYINTERVAL = { n } ]
]
```

Backing up multiple databases to single files or split into multiple files

```
BACKUP
{
  [
    { [ ALL | SYSTEM | USER ] DATABASES [EXCLUDE [ list of databases ] ]
    /
    DATABASES [EXCLUDE] { [ list_of_databases ] }
  ]
}
TO { DISK } = { 'physical_backup_device_name' | '<AUTO>' } [ ,...n ]
[ WITH

  [ [ , ] { CHECK_PREFERRED_AG_REPLICA } ]
  [ [ , ] { CHECKSUM | NO_CHECKSUM } ]
  [ [ , ] COMPRESSION = { 0 | 1 | 2 | 3 | 4 } ]
  [ [ , ] { CONTINUE_AFTER_ERROR | STOP_ON_ERROR } ]
  [ [ , ] COPY_ONLY ]
  [ [ , ] COPYTO = { 'target_folder_name' } [ ,...n ] ]
  [ [ , ] COPYTO_HOSTED ]
  [ [ , ] DESCRIPTION = { 'text' } ]
  [ [ , ] DIFFERENTIAL ]
  [ [ , ] DISKRETRYCOUNT = { n } ]
  [ [ , ] DISKRETRYINTERVAL = { seconds } ]
  [ [ , ] ERASEFILES | ERASEFILES_ATSTART | ERASEFILES_PRIMARY = { days | hours{h} | except latest{b} } ]
  [ [ , ] ERASEFILES_REMOTE | ERASEFILES_SECONDARY = { days | hours{h} | except latest{b} } ]
  [ [ , ] FILECOUNT = { n } ]
  [ [ , ] FILEOPTIONS = { 1 | 2 | 3 | 4 | 5 | 6 | 7 } ]
  [ [ , ] FULLIFREQUIRED ]
  [ [ , ] INIT ]
  [ [ , ] KEYSIZE = { 128 | 256 } ]
  [ [ , ] LOG_ONERROR ]
  [ [ , ] LOG_ONERRORONLY ]
  [ [ , ] LOGTO = { 'target_folder_name' | 'file_name' } ]
  [ [ , ] MAILTO = { 'recipients' } ]
  [ [ , ] MAILTO_NOLOG ]
  [ [ , ] MAILTO_ONERROR = { 'recipients' } ]
  [ [ , ] MAILTO_ONERRORONLY = { 'recipients' } ]
  [ [ , ] MAXDATABLOCK = { 65536 | 131072 | ... | 2097152 } ]
  [ [ , ] MAXTRANSFERSIZE = { 65536 | 131072 | ... | 1048576 } ]
  [ [ , ] NAME = { 'backup_set_name' } ]
  [ [ , ] NOCOMPRESSWRITE | NOWRITE ]
  [ [ , ] NOLOG ]
  [ [ , ] PASSWORD = { 'password' | 'FILE:file_path' } ]
  [ [ , ] SECONDARY_REPLICA_COPY_ONLY ]
  [ [ , ] SINGLERESULTSET ]
  [ [ , ] SQLCOMPRESSION ]
  [ [ , ] THREADCOUNT = { n } ]
  [ [ , ] THREADPRIORITY = { 0 | 1 | 2 | 3 | 4 | 5 | 6 } ]
  [ [ , ] USEQUEUEDCOPY ]
  [ [ , ] USESIMPLECOPY ]
  [ [ , ] VERIFY ]
  [ [ , ] VERIFYINTERVAL = { n } ]
]
]
```

Creating a partial filegroup backup

```
BACKUP DATABASE { database_name }
  READ_WRITE_FILEGROUPS [ , FILEGROUP = { 'logical_filegroup_name' } [ ,...n ] ]
TO { DISK } = { 'physical_backup_device_name' | '<AUTO>' } [ ,...n ]
[ WITH
```

```

[[,]{CHECK_PREFERRED_AG_REPLICA}]
[[,]{CHECKSUM|NO_CHECKSUM}]
[[,]COMPRESSION={0|1|2|3|4}]
[[,]{CONTINUE_AFTER_ERROR|STOP_ON_ERROR}]
[[,]COPY_ONLY]
[[,]COPYTO={ 'target_folder_name' }[,...n]
[[,]COPYTO_HOSTED]
[[,]DESCRIPTION={ 'text' }]
[[,]DIFFERENTIAL]
[[,]DISKRETRYCOUNT={ n }]
[[,]DISKRETRYINTERVAL={ seconds }]
[[,]ERASEFILES|ERASEFILES_ATSTART|ERASEFILES_PRIMARY={ days|hours{h}|except latest{b} }]
[[,]ERASEFILES_REMOTE|ERASEFILES_SECONDARY={ days|hours{h}|except latest{b} }]
[[,]FILECOUNT={ n }]
[[,]FILEOPTIONS={ 1|2|3|4|5|6|7 }]
[[,]INIT]
[[,]KEYSIZE={ 128|256 }]
[[,]LOG_ONERROR]
[[,]LOG_ONERRORONLY]
[[,]LOGTO={ 'target_folder_name'| 'file_name' }]
[[,]MAILTO={ 'recipients' }]
[[,]MAILTO_NOLOG]
[[,]MAILTO_ONERROR={ 'recipients' }]
[[,]MAILTO_ONERRORONLY={ 'recipients' }]
[[,]MAXDATABLOCK={ 65536|131072|...|2097152 }]
[[,]MAXTRANSFERSIZE={ 65536|131072|...|1048576 }]
[[,]MIRRORFILE={ 'physical_backup_device_name' }[,...n]
[[,]NAME={ 'backup_set_name' }]
[[,]NOCOMPRESSWRITE|NOWRITE]
[[,]NOLOG]
[[,]PASSWORD={ 'password'| 'FILE:file_path' }]
[[,]SECONDARY_REPLICA_COPY_ONLY]
[[,]SINGLERESULTSET]
[[,]SQLCOMPRESSION]
[[,]THREADCOUNT={ n }]
[[,]THREADPRIORITY={ 0|1|2|3|4|5|6 }]
[[,]USEQUEUEDCOPY]
[[,]USESIMPLECOPY]
[[,]VERIFY]
[[,]VERIFYINTERVAL={ n }]
]

```

Backing up a transaction log to a single file or split into multiple files

```

BACKUP LOG { database_name }
TO { DISK } = { 'physical_backup_device_name'| '<AUTO>' }[,...n]
[ WITH

```

```

[[,]{CHECK_PREFERRED_AG_REPLICA}]
[[,]{CHECKSUM|NO_CHECKSUM}]
[[,]COMPRESSION={0|1|2|3|4}]
[[,]{CONTINUE_AFTER_ERROR|STOP_ON_ERROR}]
[[,]COPY_ONLY]
[[,]COPYTO={ 'target_folder_name' }[,...n]
[[,]COPYTO_HOSTED]
[[,]DESCRIPTION={ 'text' }]
[[,]DISCONNECT_EXISTING]
[[,]DISKRETRYCOUNT={ n }]
[[,]DISKRETRYINTERVAL={ seconds }]
[[,]ERASEFILES|ERASEFILES_ATSTART|ERASEFILES_PRIMARY={ days|hours{h}|except latest{b} }]
[[,]ERASEFILES_REMOTE|ERASEFILES_SECONDARY={ days|hours{h}|except latest{b} }]
[[,]FILECOUNT={ n }]
[[,]FILEOPTIONS={ 1|2|3|4|5|6|7 }]
[[,]INIT]
[[,]KEYSIZE={ 128|256 }]
[[,]LOG_ONERROR]
[[,]LOG_ONERRORONLY]
[[,]LOGTO={ 'target_folder_name'| 'file_name' }]
[[,]MAILTO={ 'recipients' }]
[[,]MAILTO_NOLOG]
[[,]MAILTO_ONERROR={ 'recipients' }]
[[,]MAILTO_ONERRORONLY={ 'recipients' }]
[[,]MAXDATABLOCK={ 65536|131072|...|2097152 }]
[[,]MAXTRANSFERSIZE={ 65536|131072|...|1048576 }]
[[,]MIRRORFILE={ 'physical_backup_device_name' }[,...n]
[[,]NAME={ 'backup_set_name' }]
[[,]NO_TRUNCATE]
[[,]NOCOMPRESSWRITE|NOWRITE]
[[,]NOLOG]
[[,]{NORECOVERY|STANDBY='standby_file_name' }]
[[,]PASSWORD={ 'password'| 'FILE:file_path' }]
[[,]SINGLERESULTSET]
[[,]SQLCOMPRESSION]
[[,]THREADCOUNT={ n }]
[[,]THREADPRIORITY={ 0|1|2|3|4|5|6 }]
[[,]USESIMPLECOPY]
[[,]VERIFY]
[[,]VERIFYINTERVAL={ n }]
]

```

Backing up multiple transaction logs to single files or split into multiple files

```

BACKUP LOGS [EXCLUDE] { [ list_of_databases ] }
TO { DISK } = { 'physical_backup_device_name'| '<AUTO>' }[,...n]
[ WITH

```

```

[[,]{CHECK_PREFERRED_AG_REPLICA}]
[[,]{CHECKSUM|NO_CHECKSUM}]
[[,]COMPRESSION={0|1|2|3|4}]
[[,]{CONTINUE_AFTER_ERROR|STOP_ON_ERROR}]
[[,]COPY_ONLY]
[[,]COPYTO={'target_folder_name'}[,...n]
[[,]COPYTO_HOSTED]
[[,]DESCRIPTION={'text'}]
[[,]DISCONNECT_EXISTING]
[[,]DISKRETRYCOUNT={n}]
[[,]DISKRETRYINTERVAL={seconds}]
[[,]ERASEFILES|ERASEFILES_ATSTART|ERASEFILES_PRIMARY={days|hours{h}|except latest{b}}]
[[,]ERASEFILES_REMOTE|ERASEFILES_SECONDARY={days|hours{h}|except latest{b}}]
[[,]FILECOUNT={n}]
[[,]FILEOPTIONS={1|2|3|4|5|6|7}]
[[,]FULLIFREQUIRED]
[[,]INIT]
[[,]KEYSIZE={128|256}]
[[,]LOG_ONERROR]
[[,]LOG_ONERRORONLY]
[[,]LOGTO={'target_folder_name'|'file_name'}]
[[,]MAILTO={'recipients'}]
[[,]MAILTO_NOLOG]
[[,]MAILTO_ONERROR={'recipients'}]
[[,]MAILTO_ONERRORONLY={'recipients'}]
[[,]MAXDATABLOCK={65536|131072|...|2097152}]
[[,]MAXTRANSFERSIZE={65536|131072|...|1048576}]
[[,]NAME={'backup_set_name'}]
[[,]NO_TRUNCATE]
[[,]NOCOMPRESSWRITE|NOWRITE]
[[,]NOLOG]
[[,]{NORECOVERY|STANDBY='standby_file_name'}]
[[,]PASSWORD={'password'|'FILE:file_path'}]
[[,]SINGLERESULTSET]
[[,]SECONDARY_REPLICA_COPY_ONLY]
[[,]SQLCOMPRESSION]
[[,]THREADCOUNT={n}]
[[,]THREADPRIORITY={0|1|2|3|4|5|6}]
[[,]USESIMPLECOPY]
[[,]VERIFY]
[[,]VERIFYINTERVAL={n}]
]

```

Arguments

DATABASE argument

Specifies a backup of a single database. The database name must be enclosed in square brackets if it includes reserved words or spaces; if the database name does not include reserved words or spaces, square brackets are optional. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\Full\pubs.sqb'"
```

FILE = 'logical_file_name' | FILEGROUP = 'logical_filegroup_name' [,...n]

Specifies the files or filegroups that are to be backed up. There is no limit on the number of files or filegroups that can be included in a BACKUP command, provided they all belong to the same database. Identify files or filegroups using logical names. For example:

```
"BACKUP DATABASE [Database1] FILE = 'SalesF1', FILE= 'SalesF2' TO DISK = 'C:\Backups\SalesFiles\<AUTO>' "
```

READ_WRITE_FILEGROUPS [, FILEGROUP = 'logical_filegroup_name']

Specifies a partial backup of all read/write files in the database, together with any specified read-only files or filegroups. For example:

```
"BACKUP DATABASE [Database1] READ_WRITE_FILEGROUPS [, FILEGROUP = SalesArchive] TO DISK = 'C:\Backups\SalesFiles\<AUTO>' "
```

For more information, refer to your SQL Server documentation.

DATABASES argument

Specifies multiple full or differential database backups.

'[list_of_databases]'

Is a comma-separated list of the names of the databases that are to be backed up. For example:

```
"BACKUP DATABASES [Database1, Database2, Database3] TO DISK = 'C:\Backups\<AUTO>' "
```

The list must be enclosed in square brackets [], and the list must **not** contain any other square brackets. You can use a single wildcard character. For example:

```
"BACKUP DATABASES [*] TO DISK = 'C:\Backups\<AUTO>' "
```

This backs up all databases that are currently online and operational, including read-only databases. Databases that are unrecovered or unavailable (for example, offline) are not backed up.

As an alternative to using a list of databases, you can specify `BACKUP ALL DATABASES`. This is equivalent to `BACKUP DATABASES [*]`

To back up all *user* databases on an instance (excluding *master*, *model*, and *msdb*), use `BACKUP USER DATABASES`.

To back up all *system* databases on an instance (only *master*, *model*, and *msdb*), use `BACKUP SYSTEM DATABASES`.

EXCLUDE

Specifies that all online databases **except** those listed (or specified by `ALL`, `USER`, or `SYSTEM`) are to be backed up. For example:

```
"BACKUP DATABASES EXCLUDE [Database1, Database2, Database3] TO DISK = 'C:\Backups\<AUTO>' "
```

This backs up all databases that are currently online and operational, except for *Database1*, *Database2*, and *Database3*. You cannot use wildcard characters in exclusion lists.

You can also combine the `EXCLUDE` keyword with the `ALL`, `SYSTEM`, or `USER` keywords. For example:

```
"BACKUP USER DATABASES EXCLUDE [Database1, Database2, Database3] TO DISK = 'C:\Backups\<AUTO>' "
```

This backs up all databases that are currently online and operational, but excludes the system databases (master, model, and msdb) as well as excluding *Database1*, *Database2*, and *Database3*.

TO { DISK } = { 'physical_backup_device_name' | '<AUTO>' } [,...n]

Creates the backups on the specified disk. You are recommended to include the `<AUTO>` parameter to ensure that each database is backed up to a unique file name. For example:

```
"BACKUP DATABASES [Database1, Database2, Database3] TO DISK = '<AUTO>' "
```

Alternatively, you could set up your own unique path using tags, such as the `<DATETIME>` tag. For more details, see the [TO DISK argument](#) below.

LOG argument

Specifies a log backup from a single database. The database must be using the `FULL` or `BULK-LOGGED` recovery model. The database name must be enclosed in square brackets if it includes reserved words or spaces; if the database name does not include reserved words or spaces, square brackets are optional. For example:

```
"BACKUP LOG [pubs] TO DISK = 'C:\Backups\Logs\pubs.sqb' "
```

For more information, refer to your SQL Server documentation.

LOGS argument

Specifies multiple transaction log backups.

'[list_of_databases]'

Is a comma-separated list of the names of the databases for which the transaction logs are to be backed up. The databases must use the FULL or BULK-LOGGED recovery model. For example:

```
"BACKUP LOGS [Database1, Database2, Database3] TO DISK = 'C:\Backups\Logs\<AUTO>' "
```

The list must be enclosed in square brackets [], and the list must not contain any other square brackets. Alternatively, a single wildcard character can be used. For example:

```
"BACKUP LOGS [*] TO DISK = 'C:\Backups\Logs\<AUTO>' "
```

This backs up transaction logs for all databases that are currently online, operational, and using the FULL or BULK-LOGGED recovery models.

EXCLUDE

Specifies that transaction logs of all online databases **except** those listed are to be backed up. For example:

```
"BACKUP LOGS EXCLUDE [Database1, Database2, Database3] TO DISK = 'C:\Backups\Logs\<AUTO>' "
```

This backs up the transaction logs of all databases that are currently online and operational, except for *Database1*, *Database2*, and *Database3*.

TO { DISK } = { '*physical_backup_device_name*' | '<AUTO>' } [,...n]

Creates the backups on the specified disk. **Must** include the <AUTO> parameter to ensure that each transaction log is backed up to a unique file name. For details, see the [TO DISK argument](#) below. For example:

```
"BACKUP LOGS [*] TO DISK = 'C:\Backups\Logs\<AUTO>' "
```

TO DISK argument

You can specify multiple DISK values, up to a maximum of 32. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\File1.sqb', DISK = 'C:\Backups\File2.sqb' "
```

TO DISK = '<AUTO>'

If you specify <AUTO>, SQL Backup Pro generates the backup file path and file name using the backup file locations specified in the File management options. If no backup file locations have been set up, SQL Backup Pro uses the SQL Server instance's default backup folder, and the default format for file names. For details, see [File management options](#).

TO DISK = '*path*\<AUTO>'

If you specify a path and <AUTO>, SQL Backup Pro uses the specified path, and generates the file name using the File management options. If no backup file locations have been set up, SQL Backup Pro uses the default format for file names.

You can use tags in the path. For example, if you are backing up multiple databases and you want the backup files for each database to be created in a separate folder, you can specify the <DATABASE> tag. The following example creates the backups in a different folder for each database name:

```
"BACKUP DATABASES [pubs, northwind, AdventureWorks] TO DISK = 'C:\Backups\<DATABASE>\<AUTO>' "
```

You can specify more than one tag in the path. In the following example, backups are grouped by database name, date, and type of backup:

```
"BACKUP DATABASES [pubs, northwind, AdventureWorks] TO DISK = 'C:\Backups\<DATABASE>\<DATETIME>  
yyyyymmdd>\<TYPE>\<AUTO>' "
```

For details of the tags you can use, see [File location tags](#).

TO DISK = '*file_name*'

If you specify a file name with the TO DISK argument, SQL Backup Pro uses the File management options to generate the backup file path, and uses the specified name for the file. If no backup file locations have been set up, SQL Backup Pro uses the SQL Server instance's default backup folder. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'pubs_20120229.sqb' "
```

TO DISK = '<AUTO>.sqb'

If you specify <AUTO> with a file extension, SQL Backup Pro uses the File management options to generate the backup file path and file name, but uses the specified extension. If no backup file locations have been set up, SQL Backup Pro uses the SQL Server instance's default backup folder, and the default format for file names.

WITH options

CHECK_PREFERRED_AG_REPLICA

Specifies whether a backup checks whether the replica is the backup preference for the availability group.

CHECKSUM|NO_CHECKSUM

Specifies whether a backup checksum should be created. If **CHECKSUM** is included, any page checksums are validated and a backup checksum is generated and stored on the backup media for use on restore. If an invalid page checksum is encountered, the backup will fail. If **NO_CHECKSUM** is specified, no checksums are validated or generated. This is the default behavior in a **BACKUP** command. For more information, refer to your SQL Server documentation.

COMPRESSION

Specifies the compression level. The default value is 1. If you do not want to compress the backups, specify 0. For more information, see [Compression levels](#).

To use native SQL Server compression (available for SQL Server 2008 Enterprise Edition, SQL Server 2008 R2 Standard Edition and SQL Server 2012), use the **SQLCOMPRESSION** keyword. You cannot include both **COMPRESSION** and **SQLCOMPRESSION** in a command.

CONTINUE_AFTER_ERROR | STOP_ON_ERROR

If **CHECKSUM** is included in the **BACKUP** command, **CONTINUE_ON_ERROR** specifies that the backup process should continue if an invalid page checksum or torn page is encountered. **STOP_ON_ERROR** specifies that the backup process should stop if an invalid page checksum or torn page is encountered. If neither option is included, **STOP_ON_ERROR** is the default behavior.

For more information, refer to your SQL Server documentation.

COPY_ONLY

Specifies a copy-only backup. A copy-only full backup is independent of the sequence of backups and does not affect the differential base LSN. A copy-only full backup cannot be used as a base for restoring a differential backup.

A copy-only log backup is independent of the sequence of regular log backups, and does not affect the log archive point. Taking a copy-only log backup will not affect log shipping operations.

This option cannot be used with the **DIFFERENTIAL** option. For more information, refer to your SQL Server documentation.

COPYTO

Specifies that a copy of the backup files is to be created in the specified folder when the backup process completes. For example:


```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH COPYTO = '\\BACKUPSERVER001\Folder1' "
```

You can use [tags](#) in the path. To create a copy of the backup in more than one folder, use multiple **COPYTO** arguments. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH COPYTO = '\\BACKUPSERVER001\<DATABASE>', COPYTO = '\\BACKUPSERVER002\<DATABASE>' "
```

You must ensure that you have sufficient rights to the specified folders.

COPYTO_HOSTED

 COPYTO_HOSTED is only available in SQL Backup Pro 7.3 and later.

Specifies that a copy of the backup files is to be uploaded to the SQL Backup Pro Hosted Storage account linked to the SQL Server. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH COPYTO_HOSTED"
```

May be used in addition to the COPYTO option. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH COPYTO_HOSTED, COPYTO =  
'\\BACKUPSERVER001\<DATABASE>' "
```

The SQL Server must be linked to a SQL Backup Pro Hosted Storage account in order to use this option. To create an account, go to <http://sqlbackup.red-gate.com>. To link a SQL Server to an account, from the **Tools** menu select **Server Options**, then select the **Hosted Storage Settings** tab and enter your account details.

SQL Backup Pro will start uploading the backup once it has been written to disk. If you have split a backup into multiple files (using multiple TO DISK arguments or the FILECOUNT option), all the files will be written to disk before SQL Backup Pro starts uploading them to hosted storage. If you are backing up multiple databases or transaction logs, the upload process will start once the first backup has been written to disk.

If the upload fails, SQL Backup Pro will try uploading again at regular intervals for 24 hours. You can view the progress of the upload from the In Progress tab in the SQL Backup Pro GUI. For details, see [The In Progress tab](#).

For more information, see [Backing up to hosted storage](#).

DESCRIPTION

Specifies the text describing the backup set. The description is limited to 255 characters. For more information, refer to your SQL Server documentation.

DIFFERENTIAL

Limits the backup to the portions of the database or file that have changed since the last full backup. For more information, refer to your SQL Server documentation.

DISCONNECT_EXISTING

Kills any existing connections to the database before starting a transaction log backup. DISCONNECT_EXISTING is only valid when you are backing up the tail of the transaction log (with NORECOVERY or STANDBY).

DISKRETRYCOUNT

In combination with DISKRETRYINTERVAL, this argument controls network resilience behavior.

DISKRETRYCOUNT specifies the maximum number of times to retry a failed data-transfer operation (writing or copying a backup file). If you omit this keyword, the default value of 10 is used; specify a value of 0 to prevent any retries following a failed data-transfer operation. If you specify a value for DISKRETRYCOUNT, you should also specify a value for DISKRETRYINTERVAL.

If you have also specified the COPYTO keyword, you can prevent retries for the copying operation only, by specifying USESIMPLECOPY.

DISKRETRYINTERVAL

In combination with DISKRETRYCOUNT, this argument controls network resilience behavior.

DISKRETRYINTERVAL specifies the time interval between retries, in seconds, following a failed data-transfer operation (writing or copying a backup file). If you omit this keyword, the default value of 30 seconds is used. If you specify a value for DISKRETRYINTERVAL, you should also specify a value for DISKRETRYCOUNT.

If you have also specified the COPYTO keyword, you can prevent retries for the copying operation only, by specifying USESIMPLECOPY.

ERASEFILES

Manages deletion of existing SQL Backup backups from the primary backup folder (specified using DISK and MIRRORFILE). If multiple DISK locations are specified, the setting is applied to each folder.

The backup files are deleted only if the backup process completes successfully. **Note:** To delete backup files before the start of the backup process, use ERASEFILES_ATSTART.

You can choose to delete SQL Backup files based on:

- Age: files older than the specified number of days or hours are deleted. Specify a number for days, or type *h* after the number for hours. For example, `ERASEFILES = 24` deletes files that are more than 24 days old; `ERASEFILES = 24h` deletes files that are more than 24 hours old. Note that a day is calculated as a period of 24 hours, and takes no account of calendar date.
- Number of backups to keep: only the latest 'x' backups will be kept. To specify the number of backups to be kept, type *b* after the number. For example, `ERASEFILES = 5b` ensures the latest 5 backups are kept; older backups are deleted.



If you include both `ERASEFILES` and `COPYTO` in a `BACKUP DATABASE` command, the full or differential backup is written and old backups are deleted from the `DISK` location before the new backup is copied to the `COPYTO` location.

If you include both `ERASEFILES` and `COPYTO` in a `BACKUP LOG` command, the transaction log backup is written and added to the log copy queue for copying to the `COPYTO` location before the old backups are deleted from the `DISK` location. SQL Backup does not wait for the transaction log backup to be copied successfully before deleting old backups from the `DISK` location.

Files are deleted only if the following details match the details of the database being backed up:

- The name of the SQL Server, instance (if applicable), and database recorded in the file header.
- The backup type (full, differential, transaction log).
- The backup password. If the `PASSWORD` option is not specified (because the backup is not encrypted), any existing encrypted backups in the `DISK` location will not be identified by `ERASEFILES` because the file header cannot be read. This may result in backups older than the specified age or in excess of the specified number being retained.



If SQL Backup Pro cannot list the contents of the folder that contains the files to be deleted, it cannot delete the files. Ensure the SQL Backup Agent service startup account (or, if you are using the command line, the user account from which you are running `SQLBackupC.exe`) has permissions to list the folder contents.

Examples

The following example creates a full backup in `C:\Backups\pubs`, then deletes any full backups of the `pubs` database older than 5 days from that folder.

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<DATABASE>\<AUTO>.sqb' WITH ERASEFILES = 5"
```

The following example creates a full backup in `C:\Backups\pubs`, then deletes all full backups of the `pubs` database other than the latest 7 from that folder.

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<DATABASE>\<AUTO>.sqb' WITH ERASEFILES = 7b"
```

The following example creates a full backup in a remote folder, `\\BACKUPSERVER001\pubs`, then deletes all full backups of the `pubs` database over 12 hours old from that folder.

```
"BACKUP DATABASE [pubs] TO DISK = '\\BACKUPSERVER001\<DATABASE>\<AUTO>.sqb' WITH ERASEFILES = 12h"
```

You can use `ERASEFILES_REMOTE` to manage deletion of files from a remote location specified using the `COPYTO` option.



If the `DISK` location is a *remote* location and `ERASEFILES_REMOTE` is also included in the command, the `ERASEFILES_REMOTE` setting will override the `ERASEFILES` setting.

If a *local* `COPYTO` location is specified, the `ERASEFILES` setting will only apply to backups in the `COPYTO` location if `ERASEFILES_REMOTE` is also included in the command.

To prevent deletion of files that have their archive attribute set, use `FILEOPTIONS`.

To overwrite existing backups of the same name, use the `INIT` option.

ERASEFILES_ATSTART

Manages deletion of existing SQL Backup backups from the primary backup folder (specified using `DISK` and `MIRRORFILE`) *before* the backup process starts. If multiple `DISK` locations are specified, the setting is applied to each folder before the backup process starts. To delete backup files only if the backup process completes successfully, use `ERASEFILES` or `ERASEFILES_PRIMARY`.

You can choose to delete SQL Backup files based on:

- Age: files older than the specified number of days or hours are deleted. Specify a number for days, or type *h* after the number for hours. For example, `ERASEFILES_ATSTART = 24` deletes files that are more than 24 days old; `ERASEFILES_ATSTART = 24h` deletes files that are more than 24 hours old. Note that a day is calculated as a period of 24 hours, and takes no account of calendar date.
- Number of backups to keep: only the latest 'x' backups will be kept. To specify the number of backups to be kept, type *b* after the number. For example, `ERASEFILES_ATSTART = 5b` ensures the latest 5 backups are kept; older backups are deleted.

Files are deleted only if the following details match the details of the database being backed up:

- The name of the SQL Server, instance (if applicable), and database recorded in the file header.
- The backup type (full, differential, transaction log).
- The backup password. If the `PASSWORD` option is not specified (because the backup is not encrypted), any existing encrypted backups in the `DISK` location will not be identified by `ERASEFILES_ATSTART` because the file header cannot be read. This may result in backups older than the specified age or in excess of the specified number being retained.



If SQL Backup cannot list the contents of the folder that contains the files to be deleted, it cannot delete the files. Ensure the SQL Backup Agent service startup account (or, if you are using the command line, the user account from which you are running *SQLBackupC.exe*) has permissions to list the folder contents.

To prevent deletion of files that have their archive attribute set, use `FILEOPTIONS`.

To overwrite existing backups of the same name, use the `INIT` option.

You can use `ERASEFILES_REMOTE` to manage deletion of files in a remote location.



There are known issues when `ERASEFILES_ATSTART` and `ERASEFILES_REMOTE` are used in the same `BACKUP` command. You are recommended to test the command to ensure it results in the desired behavior before using it in a production environment.

ERASEFILES_PRIMARY



`ERASEFILES_PRIMARY` is only available in SQL Backup Pro 7.4 and later.



To manage deletion of backup files, use either:

- `ERASEFILES_PRIMARY` and `ERASEFILES_SECONDARY`, or
- `ERASEFILES`, `ERASEFILES_ATSTART`, `ERASEFILES_REMOTE` and `FILEOPTIONS`

Do not use `ERASEFILES_PRIMARY` in the same command as `ERASEFILES`, `ERASEFILES_ATSTART`, `ERASEFILES_REMOTE` or `FILEOPTIONS`.

Manages deletion of existing SQL Backup backups from the primary backup folder (specified using `DISK` and `MIRRORFILE`). If multiple `DISK` locations are specified, the setting is applied to each folder. The backup files are deleted only if the backup process completes successfully.

You can choose to delete SQL Backup files based on:

- Age: files older than the specified number of days or hours are deleted. Specify a number for days, or type *h* after the number for hours. For example, `ERASEFILES_PRIMARY = 24` deletes files that are more than 24 days old; `ERASEFILES_PRIMARY = 24h` deletes files that are more than 24 hours old. Note that a day is calculated as a period of 24 hours, and takes no account of calendar date.
- Number of backups to keep: only the latest 'x' backups will be kept. To specify the number of backups to be kept, type *b* after the number. For example, `ERASEFILES_PRIMARY = 5b` ensures the latest 5 backups are kept; older backups are deleted.

Files are deleted only if the following details match the details of the database being backed up:

- The name of the SQL Server, instance (if applicable), and database recorded in the file header.
- The backup type (full, differential, transaction log).
- The backup password. If the `PASSWORD` option is not specified (because the backup is not encrypted), any existing encrypted backups in the `DISK` location will not be identified by `ERASEFILES_PRIMARY` because the file header cannot be read. This may result in backups older than the specified age or in excess of the specified number being retained.



If SQL Backup cannot list the contents of the folder that contains the files to be deleted, it cannot delete the files. Ensure the SQL Backup Agent service startup account (or, if you are using the command line, the user account from which you are running *SQLBackupC.exe*) has permissions to list the folder contents.

Example

The following example creates a full backup in `C:\Backups\pubs`, then deletes all full backups of the *pubs* database older than 5 days from that folder.

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<DATABASE>\<AUTO>.sqb' WITH ERASEFILES_PRIMARY = 5"
```

It is not possible to overwrite existing files of the same name, or to delete old backups before the backup has been written, with `ERASEFILES_PRIMARY`.

ERASEFILES_REMOTE

Manages deletion of existing SQL Backup backups from *remote* locations, whether the primary backup folder (specified using *DISK* and *MIRRORFILE*) or the secondary backup folder (specified using *COPYTO*). The backup files are deleted only if the backup process completes successfully.

You can choose to delete SQL Backup files based on:

- Age: files older than the specified number of days or hours are deleted. Specify a number for days, or type *h* after the number for hours. For example, `ERASEFILES_REMOTE = 24` deletes files that are more than 24 days old; `ERASEFILES_REMOTE = 24h` deletes files that are more than 24 hours old. Note that a day is calculated as a period of 24 hours, and takes no account of calendar date.
- Number of backups to keep: only the latest 'x' backups will be kept. To specify the number of backups to be kept, type *b* after the number. For example, `ERASEFILES_REMOTE = 5b` ensures the latest 5 backups are kept; older backups are deleted.

Files are deleted only if the following details match the details of the database being backed up:

- The name of the SQL Server, instance (if applicable), and database recorded in the file header.
- The backup type (full, differential, transaction log).
- The backup password. If the `PASSWORD` option is not specified (because the backup is not encrypted), any existing encrypted backups in the *DISK* location will not be identified by `ERASEFILES_REMOTE` because the file header cannot be read. This may result in backups older than the specified age or in excess of the specified number being retained.



If SQL Backup cannot list the contents of the folder that contains the files to be deleted, it cannot delete the files. Ensure that SQL Backup Agent service startup account (or, if you are using the command line, the user account from which you are running *SQLBackupC.exe*) has permissions to list the folder contents.

Examples

The following example creates a full backup in *C:\Backups\pubs*, copies the backup to a remote folder, *\\BACKUPSERVER001\pubs*, then deletes all full backups of the *pubs* database older than 5 days from the remote folder.

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<DATABASE>\<AUTO>.sqb' WITH COPYTO =  
'\\BACKUPSERVER001\<DATABASE>', ERASEFILES_REMOTE = 5"
```

The following example creates a full backup in *C:\Backups\pubs*, deletes all but the last 5 full backups of the *pubs* database from that folder, then copies the backup to a remote folder, *\\BACKUPSERVER001\pubs*, and deletes all full backups of the *pubs* database older than 10 days from the remote folder.

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<DATABASE>\<AUTO>.sqb' WITH COPYTO =  
'\\BACKUPSERVER001\<DATABASE>', ERASEFILES = 5b, ERASEFILES_REMOTE = 10"
```

The following example creates a full backup in *\\BACKUPSERVER001\pubs*, deletes all full backups of the *pubs* database older than 30 days from that folder, then copies the backup to a remote folder, *\\BACKUPSERVER002\pubs*, and deletes any full backups of the *pubs* database older than 30 days from the that folder.

```
"BACKUP DATABASE [pubs] TO DISK = '\\BACKUPSERVER001\<DATABASE>\<AUTO>.sqb' WITH COPYTO =  
'\\BACKUPSERVER002\<DATABASE>', ERASEFILES_REMOTE = 30"
```

To apply different settings to remote *DISK* and *COPYTO* locations, use `ERASEFILES_PRIMARY` and `ERASEFILES_SECONDARY`.

To overwrite existing files of the same name, use `FILEOPTIONS`.

ERASEFILES_SECONDARY



`ERASEFILES_SECONDARY` is only available in SQL Backup Pro 7.4 and later.



To manage deletion of backup files, use either:

- `ERASEFILES_PRIMARY` and `ERASEFILES_SECONDARY`, or
- `ERASEFILES`, `ERASEFILES_ATSTART`, `ERASEFILES_REMOTE` and `FILEOPTIONS`

Do not use `ERASEFILES_SECONDARY` in the same command as `ERASEFILES`, `ERASEFILES_ATSTART`, `ERASEFILES_REMOTE` or `FILEOPTIONS`.

Manages deletion of existing SQL Backup backups from the secondary backup folder (specified using *COPYTO*). The backup files are deleted only if the backup process completes successfully.

You can choose to delete SQL Backup files based on:

- Age: files older than the specified number of days or hours are deleted. Specify a number for days, or type *h* after the number for hours. For example, `ERASEFILES_SECONDARY = 24` deletes files that are more than 24 days old; `ERASEFILES_SECONDARY = 24h` deletes files that are more than 24 hours old. Note that a day is calculated as a period of 24 hours, and takes no account of calendar date.
- Number of backups to keep: only the latest 'x' backups will be kept. To specify the number of backups to be kept, type *b* after the number. For example, `ERASEFILES_SECONDARY = 5b` ensures the latest 5 backups are kept; older backups are deleted.

Files are deleted only if the following details match the details of the database being backed up:

- The name of the SQL Server, instance (if applicable), and database recorded in the file header.
- The backup type (full, differential, transaction log).
- The backup password. If the `PASSWORD` option is not specified (because the backup is not encrypted), any existing encrypted backups in the `DISK` location will not be identified by `ERASEFILES_SECONDARY` because the file header cannot be read. This may result in backups older than the specified age or in excess of the specified number being retained.



If SQL Backup cannot list the contents of the folder that contains the files to be deleted, it cannot delete the files. Ensure the SQL Backup Agent service startup account (or, if you are using the command line, the user account from which you are running *SQLBackupC.exe*) has permissions to list the folder contents.

Examples

The following example creates a full backup in `C:\Backups\pubs`, then copies the backup to `\\BACKUPSERVER001\pubs` and deletes all full backups of the *pubs* database older than 5 days from that folder.

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<DATABASE>\<AUTO>.sqb' WITH COPYTO =
'\\BACKUPSERVER001\<DATABASE>', ERASEFILES_SECONDARY = 5"
```

The following example creates a full backup in `C:\Backups\pubs` and deletes all but the latest 7 full backups of the *pubs* database from that folder, then copies the backup to `E:\Archive\pubs` and deletes all full backups of the *pubs* database older than 60 days from that folder.

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<DATABASE>\<AUTO>.sqb' WITH COPYTO = 'E:\Archive\<DATABASE>',
ERASEFILES_PRIMARY = 7b, ERASEFILES_SECONDARY = 60"
```

It is not possible to overwrite existing files of the same name with `ERASEFILES_SECONDARY`.

FILECOUNT

Specifies the number of backup files to be generated if you are splitting the backup across a number of files, where *n* is an integer between 2 and 32 inclusive. The name of the file specified in the `TO DISK` argument is used as the base name for the generated files. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\Pubs\FULL_20120229.sqb' WITH FILECOUNT = 4"
```

will generate four backup files:

`C:\Backups\Pubs\FULL_20120229_01.sqb`

`C:\Backups\Pubs\FULL_20120229_02.sqb`

`C:\Backups\Pubs\FULL_20120229_03.sqb`

`C:\Backups\Pubs\FULL_20120229_04.sqb`

Can be used together with the `<AUTO>` keyword (see the [TO DISK argument](#)). For example:

```
"BACKUP DATABASE [pubs] TO DISK = '<AUTO>' WITH FILECOUNT = 4"
```

In this example, SQL Backup generates the additional files using the `<AUTO>` naming convention, appending `_02`, `_03`, `_04` and so on to the root name.



If you are using the `FILECOUNT` keyword to create multiple files:

- You cannot use `TO DISK` multiple times.
- You cannot use the `THREADCOUNT` option.

FILEOPTIONS



FILEOPTIONS = 1 has been deprecated. Use ERASEFILES_REMOTE or ERASEFILES_SECONDARY.

Specifies whether old backup files are to be deleted or overwritten in the primary backup folder and any COPYTO folders. Specify the sum of the values that correspond to the options you require:

1	Delete old backup files in the secondary backup folders (specified using COPYTO) if they are older than the number of days or hours specified in ERASEFILES or ERASEFILES_ATSTART. Backups are deleted from the secondary backup folder after the backup is copied, regardless of whether ERASEFILES or ERASEFILES_ATSTART is used.
2	Delete old backup files in the primary backup folder (specified using DISK) if they are older than the number of days or hours specified in ERASEFILES, ERASEFILES_ATSTART, or ERASEFILES_REMOTE unless they have the ARCHIVE flag set.
4	Overwrite existing files of the same name in the COPYTO folder. If combined with option 1 or ERASEFILES_REMOTE, existing files are deleted before files of the same name are overwritten. This can result in fewer backups remaining in the folder than expected.

Valid values are 1 to 7.

If you specify option 1 or 2 you must also set the age of the files to delete using ERASEFILES, ERASEFILES_ATSTART, or ERASEFILES_REMOTE. For example, to delete old backup files in the COPYTO folder that are older than 5 days:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH COPYTO = '\\BACKUPSERVER001\<DATABASE>' , ERASEFILES = 5, FILEOPTIONS = 1"
```

To overwrite any existing files in the COPYTO folder and also delete old backup files in the COPYTO folder that are older than 5 days (values 1 + 4):

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH COPYTO = '\\BACKUPSERVER001\<DATABASE>' , ERASEFILES = 5, FILEOPTIONS = 5"
```

FULLIFREQUIRED



FULLIFREQUIRED is only available in SQL Backup Pro 7.1 and later.

This option is only available when taking differential or transaction log backups of multiple databases (i.e. when using BACKUP DATABASES ... WITH DIFFERENTIAL or BACKUP LOGS). This option cannot be used when backing up a single database or transaction log.

When using this option, the TO DISK clause must include either the <AUTO> tag or both the <DATABASE> and <TYPE> tags.

This option specifies that a full backup should be created if required to take a differential or transaction log backup. A full backup is created for any databases:

- for which a full backup has not previously been taken, or
- that have been moved from simple recovery model to full or bulk-logged and for which a full backup has not since been taken (transaction log backups only).

If a full backup is created, warning 474 is returned and details of the full backup are included in the log.

Any other options specified in the command will be applied to the full backup, with the exception of the ERASEFILES options, FILEOPTIONS and INIT.



This option is not available if SQL Server does not return SQL error 3035 or 4214 when a full backup is required. This is the case when performing transaction log backups on most SQL Server 2000 or 2005 service packs, and when performing differential backups on most SQL Server 2005 service packs. When using this option on SQL Server 2000 or 2005, warning 483 is returned to inform you that a full backup may not have been taken.

For more information see [Backing up all databases on an instance](#).

INIT

Specifies that files with the same name in the primary backup folder should be overwritten.

Because SQL Backup Pro supports a maximum of one backup set in a media set, specifying INIT also overwrites any media-set data (media header) associated with the existing backup files. This is equivalent to using the FORMAT argument in SQL Server.

KEYSIZE

Specifies the size of the encryption key to use; you must also use the `PASSWORD` keyword to specify a password for the encrypted backups. You can specify a 128-bit key or a 256-bit key. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH PASSWORD = 'MyPassword', KEYSIZE = 128"
```

If you include `PASSWORD` and do not specify the key size, SQL Backup Pro uses a 256-bit key.

LOG_ONERROR

Specifies that a log file should only be created if SQL Backup Pro encounters an error during the backup process, or the backup completes successfully but with warnings. Use this option if you want to restrict the number of log files created by your backup processes, but maintain log information whenever warnings or errors occur. This argument controls the creation of log files on disk only; emailed log files are not affected. (See the [MAILTO options](#) below for details on emailing log files.)

LOG_ONERRORONLY

Specifies that a log file should only be created if SQL Backup Pro encounters an error during the backup process. Use this option if you want to restrict the number of log files created by your backup processes, but maintain log information whenever errors occur. This argument controls the creation of log files on disk only; emailed log files are not affected. (See the [MAILTO options](#) below for details on emailing log files.)

LOGTO

Specifies that a copy of the log file is to be saved.

By default, the primary log file is created in the folder `%ProgramData%\Red Gate\SQL Backup\Log` (Windows Vista, Windows 2008 and later) or `%ALLUSERSPROFILE%\Application Data\Red Gate\SQL Backup\Log` (Windows XP and Windows 2003); you can change this location in your [file management options](#).

To create a copy with the same name as the primary log file, specify the folder. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH LOGTO = 'C:\Logs'"
```

To create a copy with a different name from the primary log file, specify the folder and file name. For example:

```
"BACKUP DATABASE pubs TO DISK = 'C:\Backups\<AUTO>' WITH LOGTO = 'C:\Logs\SQLBSecondaryLog.txt'"
```

To copy the log file to more than one location, use multiple `LOGTO` commands.

MAILTO

Specifies that the outcome of the backup operation is emailed to one or more users; the email includes the contents of the log file. SQL Backup Pro uses the settings specified in your [email settings](#) to send the email. To specify multiple recipients, separate the email addresses with a semi-colon (;). For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH MAILTO = 'dba01@myco.com;dba02@myco.com'"
```

If you have not defined [email settings](#), the email will not be sent and a warning will be reported.

MAILTO_NOLOG

Specifies that SQL Backup Pro should not include the contents of the log file in the email. An email will still be sent to notify the specified recipients of success and/or failure, depending on which `MAILTO` parameter has been specified.

MAILTO_ONERROR

Specifies that the outcome of the backup operation is emailed to one or more users if SQL Backup Pro encounters an error during the backup process or if the backup completes successfully but with warnings. The email includes the contents of the log file. SQL Backup Pro uses the settings specified in your [email settings](#) to send the email. The email includes the contents of the log file. To specify multiple recipients, separate the email addresses with a semi-colon (;). For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH MAILTO_ONERROR = 'dba01@myco.com;dba02@myco.com'"
```

If you have not defined [email settings](#), the email will not be sent and a warning will be reported.

MAILTO_ONERRORONLY

Specifies that the outcome of the backup operation is emailed to one or more users if SQL Backup Pro encounters an error during the backup process. SQL Backup Pro uses the settings specified in your [email settings](#) to send the email. To specify multiple recipients, separate the email addresses with a semi-colon (;). For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH MAILTO_ONERRORONLY = 'dba01@myco.com;dba02@myco.com' "
```

If you have not defined [email settings](#), the email will not be sent and a warning will be reported.

MAXDATABLOCK

Specifies the maximum size of data blocks to be used when SQL Backup Pro stores backup data. Valid values are integers in multiples of 65536, up to a maximum value of 2097152. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH MAXDATABLOCK = 655360 "
```

If not specified, SQL Backup Pro uses the value defined in the following DWORD registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Red Gate\SQL Backup\BackupSettingsGlobal\<instance name>MAXDATABLOCK

MAXTRANSFERSIZE

Specifies the maximum size of each block of memory (in bytes) to be used when SQL Backup Pro stores backup data. You may want to specify this argument if a SQL Server reports that it has insufficient memory to service requests from SQL Backup Pro.

Valid values are integers in multiples of 65536, up to a maximum value of 1048576.

For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH MAXTRANSFERSIZE = 262144 "
```

If not specified, defaults to 1048576. However, if you have created the following DWORD registry key, SQL Backup Pro uses the defined value as the default value:

HKEY_LOCAL_MACHINE\SOFTWARE\Red Gate\SQL Backup\BackupSettingsGlobal\<instance name>MAXTRANSFERSIZE

MIRRORFILE

Indicates that you want to create a duplicate backup file (in addition to the backup file named in the `TO DISK` argument). Specify the value as described for the `TO DISK` argument. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH MIRRORFILE = 'Z:\AlternativeLocation\MirrorBackup.sqb' "
```

A single backup is created, which is then written in parallel to the disk location and mirror locations. All the files will therefore finish being written at the same time.

`MIRRORFILE` is useful if you want to write backups to locations on similar devices. If you want to write to devices with different characteristics such as processor speed, it is better to use `COPYTO` to prevent unnecessary load on your SQL Server.

You cannot use `MIRRORFILE` if you are backing up multiple databases, if you specify multiple `DISK` options, or if you use the `FILECOUNT` option.

To create more than one duplicate backup file, use multiple `MIRRORFILE` arguments (up to a maximum of 32 files).

Before the backup process starts, the locations specified in the `MIRRORFILE` arguments are verified; if a location does not exist, or the file cannot be created at that location, then the backup will not start.

During the backup process, if any of the files cannot be written a warning is raised. However, the backup process continues as long as at least one specified backup file can be written. If none of the files can be written, an error is raised and the backup process is stopped.

NAME

Specifies the name of the backup set. The name is limited to 128 characters. If this option is not included, the name is left blank. For more information, refer to your SQL Server documentation.

NOCOMPRESSWRITE

Determines the maximum backup process throughput of your SQL Server. When you specify this argument, SQL Backup Pro simulates a backup process without compression; no backup files are created.

You can compare the results obtained using `NOCOMPRESSWRITE` and `NOWRITE` to deduce the effects of compression on the backup throughput. For details, see [Optimizing backup speed](#).

NOLOG

Prevents a log file from being created for the backup process, even if errors or warnings are generated. You may want to use this option if you are concerned about generating a large number of log files, and are certain that you will not need to review the details of errors or warnings (for example, because it's possible to run the process again without needing to know why it failed). This argument controls the creation of log files on disk only; emailed log files are not affected. (See the [MAILTO options](#) above for details on emailing log files.)

NORECOVERY

Backs up the tail of the transaction log and leaves the database in an unrecovered state. Incomplete transactions are not rolled back. The database is not usable, but differential and transaction log backups can be restored to it. For more information, refer to your SQL Server documentation.

NO_TRUNCATE

Specifies that the transaction log should not be truncated after a transaction log backup. For more information, refer to your SQL Server documentation.

NOWRITE

Determines the maximum backup process throughput of your SQL Server using compression. When you specify this argument, SQL Backup Pro simulates a backup process using the specified compression level; no backup files are created.

You can compare the results obtained using `NOCOMPRESSWRITE` and `NOWRITE` to deduce the effects of compression on the backup throughput. For details, see [Optimizing backup speed](#).

PASSWORD

Specifies the password to be used with encrypted backup files. You must supply the same password when you restore the backup. You cannot include square brackets in a password.

You can specify the size of the encryption key using the `KEYSIZE` keyword.



In standard Transact-SQL syntax, the `PASSWORD` argument attaches a password to the backup file, but does not encrypt the file contents.

If you subsequently [convert](#) the SQL Backup file to a Microsoft Tape Format (MTF) file, you must supply the password; the MTF file will not be protected by the password.



Storing your password in a plain text file (only available in SQL Backup 7.5 and later)

If you don't want the password to be stored in your SQL Agent jobs, you can use a password stored in a plain text file instead. This means access to the password can be restricted using Windows file permissions.

To do this, specify the file path and name after the `PASSWORD` keyword instead of the password itself.

Example

```
PASSWORD = 'FILE:C:\mypasswords\password.txt'
```

SQL Backup will read only the first line of text in the file (up to the first line return), and ignore everything after.

SECONDARY_REPLICA_COPY_ONLY

Automatically applies `COPY_ONLY` for full backups (if required by replica)

SINGLERESULTSET

Specifies that the results returned by the `BACKUP` command should be limited to just one result set. This may be useful if you want to manipulate results using a Transact-SQL script. Such scripts can only manipulate results when a single result set is returned. The `BACKUP` command will return two result sets by default in most cases, unless you specify the `SINGLERESULTSET` keyword.

SQLCOMPRESSION

Specifies that the backup should be compressed using SQL Server's native compression. This feature is available for SQL Server 2008 Enterprise Edition, SQL Server 2008 R2 Standard Edition and SQL Server 2012.

To use SQL Backup Pro's compression algorithms, use the `COMPRESSION` keyword. Do not specify both `SQLCOMPRESSION` and `COMPRESSION`.

STANDBY

Backs up the tail of the transaction log and leaves the database in a read-only and `STANDBY` state. The `STANDBY` clause writes standby data (performing rollback, but with the option of further restores).

'undo_file_name'

Is a standby file, whose location is stored in the log of the database. This file holds the rolled back changes, which must be reversed if `RESTORE LOG` operations are to be subsequently applied.

When used with the `BACKUP LOGS` command (backing up multiple transaction logs) 'undo_file_name' must include the <DATABASE> tag.

When used with the `BACKUP LOG` command (backing up a single transaction log) 'undo_file_name' can include tags, but these are not required.

THREADCOUNT

Specifies the number of threads to be used to create the backup, where n is an integer between 2 and 32 inclusive. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH THREADCOUNT = 4"
```

This example creates a single backup file using four threads.



If you are using the `THREADCOUNT` keyword to use multiple threads:

- you cannot use `TO DISK` multiple times
- you cannot use the `FILECOUNT` keyword to create multiple files

THREADPRIORITY

Sets the SQL Backup Pro thread priority when the backup or restore process is run. Valid values are **0** to **6**, and correspond to the following priorities:

0	Idle
1	Very low
2	Low
3	Normal
4	High
5	Very high
6	Time critical

If this value is not specified, normal priority (3) is used.

USEQUEUEDCOPY



`USEQUEUEDCOPY` is only available in SQL Backup 7.5 and later

Allows you to put a non transaction log backup with a `COPYTO` into the log copy queue. This means that in the event of long network outages, SQL Backup will keep trying to copy the file for 24 hours.

USESIMPLECOPY

Prevents retries occurring during copy operations (specified using the `COPYTO` keyword). With `USESIMPLECOPY` specified, copying behavior is the same as in SQL Backup version 5.

VERIFY

Checks that the backup set is complete and can be read, but does not verify the structure of the data inside the backup. If `WITH CHECKSUM` is included in the `BACKUP` command, both the backup checksum and any page checksums are also validated.

The process will continue if any errors (such as invalid checksums) are encountered. The results of the check (including any errors encountered) are displayed as text output when the backup completes. You can also check the results later by viewing the [Activity History](#) in the graphical user interface.

This is equivalent to running `RESTORE VERIFYONLY` on a backup file.

VERIFYINTERVAL

Specifies how long (in seconds) after the start of the backup the `VERIFY` process should begin. For example:

```
"BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\<AUTO>' WITH VERIFY, VERIFYINTERVAL = 60"
```

This example creates a backup of the `pubs` database and starts the `VERIFY` process (equivalent to `RESTORE VERIFYONLY`) 60 seconds after the backup has begun. This is useful if you are encountering problems with the `VERIFY` process starting before the backup has completed, causing an error.

Deprecated WITH options

The following `WITH` options have been deprecated:

- `THREADS` (replaced by `FILECOUNT`)
- `ERASEFILEOPTIONS` (deprecated in version 4)
- `FILEOPTIONS` (value 1 only)

Examples

Back up a database to a single file

This example creates a full backup of the `pubs` database in a single file.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb' "
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'' " '
```

Create a mirrored backup

This example simultaneously creates a full backup of the `pubs` database and two duplicate backup files.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb' WITH  
MIRRORFILE = 'E:\Backups\pubs_01_alt1.sqb' MIRRORFILE = 'F:\Backups\pubs_01_alt2.sqb' "
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'' WITH MIRRORFILE =  
''E:\Backups\pubs_01_alt1.sqb'' MIRRORFILE = ''F:\Backups\pubs_01_alt2.sqb'' " '
```

Split the backup file

This example creates a full backup of the `pubs` database and splits the backup across two files.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb', DISK = 'C:  
\Backups\pubs_02.sqb' "
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'', DISK = ''C:  
\Backups\pubs_02.sqb'' " '
```

Upload a copy of the backup file to hosted storage

This example creates a full backup of the *pubs* database and then uploads the file to the Hosted Storage account linked to the SQL Server instance (SQL Backup Pro 7.3 and later).

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb' WITH COPYTO_HOSTED"
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'' WITH COPYTO_HOSTED" ' '
```

Create copies of the backup file in another disk location

This example creates a full backup of the *pubs* database and then copies the backup file to two remote folders on completion of the backup process.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb' WITH COPYTO = '\\BACKUPSERVER001\share', COPYTO = '\\BACKUPSERVER002\share' "
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'' WITH COPYTO = '\\BACKUPSERVER001\share'', COPYTO = '\\BACKUPSERVER002\share'' " ' '
```

Specify the compression level for a backup

This example creates a full backup of the *pubs* database using compression level 4.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb' WITH COMPRESSION = 4"
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'' WITH COMPRESSION = 4" ' '
```

Encrypt the backup file

This example creates a full backup of the *pubs* database and encrypts the backup file with password *MyPassword* and a 256-bit key.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb' WITH PASSWORD = 'MyPassword', KEYSIZE = 256"
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'' WITH PASSWORD = 'MyPassword'', KEYSIZE = 256" ' '
```

Encrypt the backup file using a password stored in a text file



Using a password in a text file is only available in SQL Backup 7.5 and later

This example creates a full backup of the *pubs* database and encrypts the backup file using a password stored in the text file *password.txt* and a 256-bit key.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb' WITH PASSWORD = 'FILE:C:\temp\password.txt', KEYSIZE = 256"
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'' WITH PASSWORD = 'FILE:C:\temp\password.txt', KEYSIZE = 256" ' '
```

Check the backup file

This example creates a full backup of the *pubs* database, testing any page checksums in the process. A backup checksum is generated and tested, and the backup is checked to ensure it is complete and readable.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb' WITH  
CHECKSUM, VERIFY"
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'' WITH CHECKSUM,  
VERIFY" '
```

Send email notification of the backup process

This example creates a full backup of the *pubs* database and sends an email of the completion log to two users upon completion of the backup process.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [pubs] TO DISK = 'C:\Backups\pubs_01.sqb' WITH MAILTO =  
'dba01@myco.com;dba02@myco.com' "
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [pubs] TO DISK = ''C:\Backups\pubs_01.sqb'' WITH MAILTO =  
'dba01@myco.com;dba02@myco.com'' " '
```

Back up multiple databases with exclusions

This example creates a full backup of all the databases except *master* and *model*.

```
-SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASES EXCLUDE [master, model] TO DISK = 'C:\Backups\<AUTO>' "  
"
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASES EXCLUDE [master, model] TO DISK = ''C:\Backups\<AUTO>'' " '
```

Back up multiple databases to split files

This example creates full backups of the *northwind* and *pubs* databases, splitting each into two files.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASES [northwind, pubs] TO DISK = 'C:\Backups\<AUTO>' WITH  
FILECOUNT = 2"
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASES [northwind, pubs] TO DISK = ''C:\Backups\<AUTO>'' WITH  
FILECOUNT = 2" '
```

Back up multiple databases to split files on different disks

This example creates full backups of the *northwind* and *pubs* databases, splitting both backups to separate files on different disks.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASES [northwind, pubs] TO DISK = 'C:\Backups\<AUTO>_01', TO  
DISK = 'D:\Backups\<AUTO>_02' "
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASES [northwind, pubs] TO DISK = ''C:\Backups\<AUTO>_01'', TO DISK  
= ''D:\Backups\<AUTO>_02'' " '
```

Back up transaction logs for multiple databases

This example creates transaction log backups for databases *northwind* and *pubs* in the default location.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP LOGS [northwind, pubs] TO DISK = '<AUTO>' "
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP LOGS [northwind, pubs] TO DISK = ''<AUTO>'' " ' '
```

Create differential backups for multiple databases using threads

This example creates differential backups for databases *northwind* and *pubs* using two threads.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASES [northwind, pubs] TO DISK = 'C:\Backups\<AUTO>' WITH  
THREADCOUNT = 2, DIFFERENTIAL "
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASES [northwind, pubs] TO DISK = ''C:\Backups\<AUTO>'' WITH  
THREADCOUNT = 2, DIFFERENTIAL " ' '
```

Create a partial filegroup backup

This example backs up the read/write files in the *FileGroupTest* database using the standard Transact-SQL argument `READ_WRITE_FILEGROUPS`. For detailed information about this argument, refer to your [SQL Server documentation](#) about the `BACKUP` command.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [FileGroupTest] READ_WRITE_FILEGROUPS TO DISK = 'C:  
\Backups\<AUTO>' "
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [FileGroupTest] READ_WRITE_FILEGROUPS TO DISK = ''C:  
\Backups\<AUTO>'' " ' '
```

Create a partial differential filegroup backup

This example creates a differential backup of the read/write files in the *FileGroupTest* database using the standard Transact-SQL argument `READ_WRITE_FILEGROUPS`. For detailed information about this argument, refer to your [SQL Server documentation](#) about the `BACKUP` command.

```
SQLBackupC.exe -I {instance name} -SQL "BACKUP DATABASE [FileGroupTest] READ_WRITE_FILEGROUPS TO DISK = 'C:  
\Backups\<AUTO>' WITH DIFFERENTIAL"
```

```
EXECUTE master..sqlbackup '-SQL "BACKUP DATABASE [FileGroupTest] READ_WRITE_FILEGROUPS TO DISK = ''C:  
\Backups\<AUTO>'' WITH DIFFERENTIAL" ' '
```