The RESTORE command

Use the RESTORE command with the SQL Backup Pro -SQL parameter to restore a SQL Backup backup using the command line or extended stored procedure.

- Syntax provides the grammar for the RESTORE command.
- Arguments describes the arguments for the RESTORE command.
- WITH options describes the options that can be used in the RESTORE command.
- Examples provides examples of using the command line and extended stored procedure to restore 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 only available with SQL Server 2005 and later:

- CHECKSUM / NO_CHECKSUM
- CONTINUE_AFTER_ERROR / STOP_ON_ERROR
- PARTIAL

The KEEP_CDC argument is only available with SQL Server 2008 and later.

Restore an entire database

```
RESTORE DATABASE { database_name }
[ FROM
{
        {DISK} = { 'physical_backup_device_name' | 'file_search_pattern' } [ ,...n ]
        [
           [ LATEST_FULL | LATEST_DIFF | LATEST_ALL ]
           |
           SOURCE = 'source_database_name' { LATEST_FULL | LATEST_DIFF | LATEST_ALL }
        ]
   |
        {BACKUPHISTORY} [ = 'history_database_name' ]
        { LATEST_FULL | LATEST_DIFF | LATEST_ALL }
   }
]
[ WITH
   [ [ , ] CHECKDB = { [NO_INFOMSGS] , [ALL_ERRORMSGS] , [TABLOCK] , [PHYSICAL_ONLY] , [DATA_PURITY] ,
[EXTENDED_LOGICAL_CHECKS] , [VERBOSE] }
   [ [ , ] { CHECKSUM | NO_CHECKSUM } ]
   [ [ , ] { CONTINUE_AFTER_ERROR | STOP_ON_ERROR } ]
   [ [ , ] DISCONNECT_EXISTING ]
   [ [ , ] DISKRETRYCOUNT = { n } ]
   [ [ , ] DISKRETRYINTERVAL = { n } ]
   [ [ , ] DROPDB ]
   [ [ , ] DROPDBIFSUCCESSFUL ]
   [ [ , ] ERASEFILES | ERASEFILES_PRIMARY = { days | hours{h} | except latest{b} } ]
   [ [ , ] ERASEFILES_REMOTE | ERASEFILES_SECONDARY = { days | hours{h} | except latest{b} } ]
   [ [ , ] FILEOPTIONS = { 1 | 2 | 3 } ]
   [ [ , ] KEEP_CDC ]
   [ [ , ] KEEP_REPLICATION ]
   [ [ , ] LOG_ONERROR ]
   [ [ , ] LOG_ONERRORONLY ]
   [ [ , ] LOGTO = { 'target_folder_name' | 'file_name' } ] [ ,...n ]
   [ [ , ] MAILTO = { 'recipients' } ]
   [ [ , ] MAILTO_NOLOG ]
   [ [ , ] MAILTO_ONERROR = { 'recipients' } ]
   [ [ , ] MAILTO_ONERRORONLY = { 'recipients' } ]
   [ [ , ] MAXTRANSFERSIZE = { 65536 | 131072 | ... | 1048576 } ]
   [ [ , ] MOVE DATAFILES TO { 'operating_system_folder' } ]
   [ [ , ] MOVE FILESTREAMS TO { 'operating_system_folder' } ]
   [ [ , ] MOVE FULLTEXTCATALOGS TO { 'operating_system_folder' } ]
   [ [ , ] MOVE LOGFILES TO { 'operating_system_folder' } ]
   [ [ , ] MOVE 'logical_file_name' TO 'operating_system_file_name' ] [ ,...n ]
   [ [ , ] MOVETO = { 'target_folder_name' } ]
   [ [ , ] NOLOG ]
   [ [ , ] { NORECOVERY | RECOVERY | STANDBY = 'standby_file_name' } ]
   [ [ , ] ORPHAN_CHECK ]
   [ [ , ] PASSWORD = { 'password' | 'FILE:file_path'}  ]
   [ [ , ] REPLACE ]
   [ [ , ] RESTRICTED_USER ]
   [ [ , ] SINGLERESULTSET ]
   [ [ , ] THREADPRIORITY = { 0 | 1 | 2 | 3 | 4 | 5 | 6 } ]
]
```

Restore part of a database

```
RESTORE DATABASE { database_name }
   [ FILE = { 'logical_file_name' } | FILEGROUP = { 'logical_filegroup_name' } | PAGE = { 'file:page' } ] [ ,...n ]
FROM DISK = { 'physical_backup_device_name' | 'file_search_pattern' } [ ,...n ]
[ WITH
   PARTIAL
   [ [ , ] CHECKDB = { [NO_INFOMSGS] , [ALL_ERRORMSGS] , [TABLOCK] , [PHYSICAL_ONLY] , [DATA_PURITY] ,
[EXTENDED_LOGICAL_CHECKS] , [VERBOSE] } ]
   [ [ , ] { CHECKSUM | NO_CHECKSUM } ]
   [ [ , ] { CONTINUE_AFTER_ERROR | STOP_ON_ERROR } ]
   [ [ , ] DISCONNECT_EXISTING ]
   [ [ , ] DISKRETRYCOUNT = { n } ]
   [ [ , ] DISKRETRYINTERVAL = { n } ]
   [ [ , ] DROPDB ]
   [ [ , ] DROPDBIFSUCCESSFUL ]
   [ [ , ] ERASEFILES | ERASEFILES_PRIMARY= { days | hours{h} | except latest{b} } ]
   [ [ , ] ERASEFILES_REMOTE | ERASEFILES_SECONDARY= { days | hours{h} | except latest{b} } ]
   [ [ , ] FILEOPTIONS = { 1 | 2 | 3 } ]
   [ [ , ] LOG_ONERROR ]
   [ [ , ] LOG_ONERRORONLY ]
   [ [ , ] LOGTO = { 'target_folder_name' | 'file_name' } ] [ ,...n ]
   [ [ , ] MAILTO = { 'recipients' } ]
   [ [ , ] MAILTO_NOLOG ]
   [ [ , ] MAILTO_ONERROR = { 'recipients' } ]
   [ [ , ] MAILTO_ONERRORONLY = { 'recipients' } ]
   [ [ , ] MOVE DATAFILES TO { 'operating_system_folder' } ]
   [ [ , ] MOVE FILESTREAMS TO { 'operating_system_folder' } ]
   [ [ , ] MOVE FULLTEXTCATALOGS TO { 'operating_system_folder' } ]
   [ [ , ] MOVE LOGFILES TO { 'operating_system_folder' } ]
   [ [ , ] MOVE 'logical_file_name' TO 'operating_system_file_name' ] [ ,...n ]
   [ [ , ] MOVETO = { 'target_folder_name' } ]
   [ [ , ] NOLOG ]
   [ [ , ] NORECOVERY ]
   [ [ , ] ORPHAN_CHECK ]
   [ [ , ] PASSWORD = { 'password' | 'FILE:file_path'}  ]
   [ [ , ] REPLACE ]
   [ [ , ] RESTRICTED_USER ]
   [ [ , ] SINGLERESULTSET ]
   [ [ , ] THREADPRIORITY = { 0 | 1 | 2 | 3 | 4 | 5 | 6 } ]
]
```

Restore a transaction log

```
RESTORE LOG { database_name }
   [ FILE = { 'logical_file_name' } | FILEGROUP = { 'logical_filegroup_name' } | PAGE = { 'file:page' } ] [ ,...n ]
[ FROM { DISK } = { 'physical_backup_device_name' | 'file_search_pattern' } ] [ ,...n ]
[ WITH
   [ [ , ] CHECKDB = { [NO_INFOMSGS] , [ALL_ERRORMSGS] , [TABLOCK] , [PHYSICAL_ONLY] , [DATA_PURITY] ,
[EXTENDED_LOGICAL_CHECKS] , [VERBOSE] } ]
   [ [ , ] { CHECKSUM | NO_CHECKSUM } ]
   [ [ , ] { CONTINUE_AFTER_ERROR | STOP_ON_ERROR } ]
   [ [ , ] DELAY = {seconds} ]
   [ [ , ] DISCONNECT_EXISTING ]
   [ [ , ] DISKRETRYCOUNT = { n } ]
   [ [ , ] DISKRETRYINTERVAL = { n } ]
   [ [ , ] DROPDB ]
   [ [ , ] DROPDBIFSUCCESSFUL ]
   [ [ , ] ERASEFILES | ERASEFILES_PRIMARY = { days | hours{h} | except latest{b} } ]
   [ [ , ] ERASEFILES_REMOTE | ERASEFILES_SECONDARY = { days | hours{h} | except latest{b} } ]
   [ [ , ] FILEOPTIONS = { 1 | 2 | 3 } ]
   [ [ , ] KEEP_CDC ]
   [ [ , ] KEEP_REPLICATION ]
   [ [ , ] LOG_ONERROR ]
   [ [ , ] LOG_ONERRORONLY ]
   [ [ , ] LOGTO = { 'target_folder_name' | 'file_name' } ] [ ,...n ]
   [ [ , ] MAILTO = { 'recipients' } ]
   [ [ , ] MAILTO_NOLOG ]
   [ [ , ] MAILTO_ONERROR = { 'recipients' } ]
   [ [ , ] MAILTO_ONERRORONLY = { 'recipients' } ]
   [ [ , ] MAXTRANSFERSIZE = { 65536 | 131072 | ... | 1048576 } ]
   [ [ , ] MOVE 'logical_file_name' TO 'operating_system_file_name' ] [ ,...n ]
   [ [ , ] MOVETO = { 'target_folder_name' } ]
   [ [ , ] NOLOG ]
   [ [ , ] { NORECOVERY | RECOVERY | STANDBY = 'standby_file_name' } ]
   [ [ , ] ORPHAN_CHECK ]
   [ [ , ] PASSWORD = { 'password' | 'FILE:file_path'}  ]
   [ [ , ] REPLACE ]
   [ [ , ] RESTRICTED_USER ]
   [ [ , ] SINGLERESULTSET ]
   [ [ , ] { STOPAT = { 'date_time' | @date_time_var } ]
    | STOPATMARK = { 'mark_name' | 'lsn:lsn_number' }
        [ AFTER 'datetime' ]
    | STOPBEFOREMARK = { 'mark_name' | 'lsn:lsn_number' }
        [ AFTER 'datetime']
   } ]
   [ [ , ] THREADPRIORITY = { 0 | 1 | 2 | 3 | 4 | 5 | 6 } ]
]
```

Restore file list

```
RESTORE FILELISTONLY
[ FROM { DISK } = { 'physical_backup_device_name' } ]
[ WITH
   [ [ , ] { CHECKSUM | NO_CHECKSUM } ]
   [ [ , ] MAXTRANSFERSIZE = { 65536 | 131072 | ... | 1048576 } ]
   [ [ , ] PASSWORD = { 'password' | 'FILE:file_path'}  ]
   [ [ , ] SINGLERESULTSET ]
]
```

You cannot use wildcards in the FROM DISK argument with RESTORE FILELISTONLY.

Restore header

```
RESTORE HEADERONLY
[ FROM { DISK } = { 'physical_backup_device_name' } ]
[ WITH
   [ [ , ] { CHECKSUM | NO_CHECKSUM } ]
   [ [ , ] MAXTRANSFERSIZE = { 65536 | 131072 | ... | 1048576 } ]
   [ [ , ] PASSWORD = { 'password' | 'FILE:file_path'}  ]
   [ [ , ] SINGLERESULTSET ]
]
```

We recommend you use the SQL Backup Pro command RESTORE SQBHEADERONLY to retrieve the header information for SQL Backup backup files, because it is much quicker than using the native command RESTORE HEADERONLY. For details, see The RESTORE SQBHEADERONLY command.

You cannot use wildcards in the FROM DISK argument with RESTORE HEADERONLY.

Verify backup set

RESTORE VERIFYONLY
[FROM { DISK } = { '*physical_backup_device_name*' }] [,...n]
[WITH
  [[,] { CHECKSUM | NO_CHECKSUM }]
  [[,] MAILTO = { '*recipients*' }]
  [[,] MAILTO_NOLOG]
  [[,] MAILTO_ONERROR = { '*recipients*' }]
  [[,] MAILTO_ONERRORONLY = { '*recipients*' }]
  [[,] MAXTRANSFERSIZE = { 65536 | 131072 | ... | 1048576 }]
  [[,] PASSWORD = { 'password' | 'FILE:*file_path*'}]
  [[,] SINGLERESULTSET]
]

You cannot use wildcards in the FROM DISK argument with RESTORE VERIFYONLY.

Arguments

DATABASE argument

It is only possible to restore one database at a time. 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:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb'"
```

[FILE = '*logical_file_name*' | FILEGROUP = '*logical_filegroup_name*' | PAGE = '*file:page*'] [,...n]

Specifies the files, filegroups or pages of the database that are to be restored. There is no limit on the number of files, filegroups or pages that can be restored, provided thay all belong to the same database. Identify files or filegroups using logical names. For example:

```
"RESTORE DATABASE pubs FILE = 'SalesF1', FILE = 'SalesF2' FROM DISK = 'C:\Backups\pubs\salesFiles.sqb'"
```

If the database is using the simple recovery model, the specified files or filegroups must be read-only unless `WITH PARTIAL` is specified.

Individual pages of read/write filegroups can be restored, provided the database is using the full or bulk-logged recovery model. Identify the pages to be restored in the format `PAGE = 'fileID:pageID'`. To restore multiple pages, use the format `PAGE = 'fileID:pageID, fileID:pageID'`.

For more information, refer to your SQL Server documentation.

LOG argument

To restore transaction log backups, the database must be in an unrecovered or standby state (see `NORECOVERY` and `STANDBY` below). 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:

```
"RESTORE LOG [pubs] FROM DISK = 'C:\Backups\pubs\LOG__20120229_111500.sqb'"
```

To restore log backups using successive restore commands, include `WITH NORECOVERY` in each `RESTORE LOG` command. For example:

```
"RESTORE LOG [pubs] FROM DISK = 'C:\Backups\pubs\LOG_20120229_111500.sqb' WITH NORECOVERY"
"RESTORE LOG [pubs] FROM DISK = 'C:\Backups\pubs\LOG_20120229_113000.sqb' WITH NORECOVERY"
```

Restore the last log `WITH RECOVERY` to recover the database to a usable state. For information on restoring multiple log backups in one `RESTORE` command, see the `FROM DISK` argument below.

FILE = '*logical_file_name*' | FILEGROUP = '*logical_filegroup_name*' | PAGE = '*file:page*'] [,...n]

Specifies the files, filegroups or pages of the database that are to be restored. There is no limit on the number of files, filegroups or pages that can be restored, provided thay all belong to the same database. Identify files or filegroups using logical names. For example:

```
"RESTORE LOG pubs FILE = 'SalesF1', FILE = SalesF2' FROM DISK = 'C:\Backups\pubs\SalesFiles.sqb'"
```

If the database is using the simple recovery model, the specified files or filegroups must be read-only unless `WITH PARTIAL` is specified.

Individual pages of read/write filegroups can be restored, provided the database is using the full or bulk-logged recovery model. Identify the pages to be restored in the format `PAGE = 'fileID:pageID'`. To restore multiple pages, use the format `PAGE = 'fileID:pageID, fileID:pageID'`.

For more information, refer to your SQL Server documentation.

FROM DISK argument

You can specify up to 32 DISK values. This is useful when you have split a backup across multiple files. You can also enter a 'file search pattern' by specifying wildcard characters in the physical backup device name. All files that match the wildcard characters must belong to the same backup set. For example, instead of:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs_01.sqb', DISK = 'C:\Backups\pubs_02.sqb', DISK = 'C:
\Backups\pubs_03.sqb', DISK = 'C:\Backups\pubs_04.sqb', DISK = 'C:\Backups\pubs_05.sqb'"
```

you can enter:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs_*.sqb'"
```

You can also use the file search pattern to restore multiple transaction log backups. The database that the logs are restored to must be in an unrecovered state, that is, it must have been restored using the `NORECOVERY` or `STANDBY` option. For example:

```
"RESTORE LOG [pubs] FROM DISK = 'C:\Backups\Pubs\Logs*.*'"
```

SQL Backup Pro ensures that the files are restored in the correct sequence. To specify a number of folders, use the `DISK` command repeatedly. For example:

```
"RESTORE LOG [pubs] FROM DISK = 'C:\Backups\Pubs\Logs*.*', DISK = 'E:\OtherBackups\Pubs\Logs*.*'"
```

The backup files that match the wildcard characters must belong to the same backup set. Any encrypted files must use the same password.

You cannot use wildcard characters in the `FROM DISK` argument for `RESTORE FILELISTONLY`, `RESTORE HEADERONLY`, and `RESTORE VERIFYONLY` commands.

LATEST_FULL

You can use the optional `LATEST_FULL` keyword to select the most recent full backup of the destination database that matches the `DISK` values you specify. The `DISK` values you specify must contain the '*' wildcard. For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs*.sqb' LATEST_FULL"
```

will find backup files of the destination database in the *C:\Backups* directory with a file name matching *pubs*.sqb*, and will then restore the latest full backup. You can specify multiple disk values; this is useful if you have split a backup across multiple files. The files that match the file search pattern must belong to the same backup set.

Note that you cannot use `LATEST_FULL` when restoring a file or filegroup backup.

LATEST_DIFF

You can use the optional `LATEST_DIFF` keyword to select the most recent differential backup of the destination database that matches the disk values you specify. You can specify multiple locations; this is useful if you have split the backup across multiple files. The `DISK` values you specify must contain the '*' wildcard. For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\sales*.sqb', DISK = 'D:\Backups\sales*.sqb' LATEST_DIFF"
```

will find all differential backup files of the destination database in the *C:\Backups* and *D:\Backups* directories with a file name matching *sales*.sqb*, and will then restore the latest differential backup. The files that match the file search pattern must belong to the same backup set.

Note that if you specify `LATEST_DIFF`, you must restore to a database that has already had the most recent full backup applied. You cannot use `LATEST_DIFF` when restoring a file or filegroup backup.

LATEST_ALL

If you specify the `LATEST_ALL` keyword, the most recent full backup of the destination database will be restored, followed by the most recent differential backup (if one exists), and then finally by the most recent transaction log backups (if any exist). The `DISK` values you specify must contain the '*' wildcard. For example:

```
"RESTORE DATABASE pubs FROM DISK = 'C:\Backups\pubs*.sqb' LATEST_ALL"
```

will find backup files of the destination database in the *C:\Backups* directory with a file name matching *pubs*.sqb*, and will then restore the latest full backup, the latest differential backup (if applicable) and the latest log backups (if applicable). All the files that match the file search pattern must belong to the same backup set.

You can specify multiple disk values. This is useful if you store different backup types in separate locations or if you have split backups across multiple files. For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\FULL_pubs*.sqb', DISK = 'D:\Backups\DIFF_pubs*.sqb', DISK = 'E:
\Backups\LOG_pubs*.sqb' LATEST_ALL"
```


You cannot use `LATEST_ALL` when restoring a file or filegroup backup.

SOURCE

You can restore the latest backup files taken from a database other than the destination database by including `SOURCE = 'source_database_name'`. For example:

```
"RESTORE DATABASE [Sales_Test] FROM DISK = 'D:\backups*.sqb' SOURCE = 'Sales_Prod' LATEST_ALL"
```

will restore the latest full backup of the *Sales_Prod* database (followed by subsequent differential and transaction log backups, if applicable), to the *Sales_Test* database.

Note that you can only use `SOURCE` if `LATEST_FULL`, `LATEST_DIFF` or `LATEST_ALL` is included in the `RESTORE` statement. You cannot use `SOURCE` when restoring a file or filegroup backup.

FROM BACKUPHISTORY argument

Use `FROM BACKUPHISTORY` when you want to restore the latest full, latest differential, or all the latest backups (including transaction log backups) for a particular database, without having to list the individual backup file locations. SQL Backup Pro searches its own backup history to determine which backup files to restore. You must also specify the `LATEST_FULL`, `LATEST_DIFF`, or `LATEST_ALL` keyword. For example:

```
"RESTORE DATABASE [sales] FROM BACKUPHISTORY LATEST_FULL WITH RECOVERY, REPLACE"
```

will search the backup history for the 'sales' database, and will then restore the latest full backup over the current *sales* database.

To search the backup history of a different database, you can specify this as part of the `BACKUPHISTORY` parameter. For example:

```
"RESTORE DATABASE [sales_dev] FROM BACKUPHISTORY = 'sales' LATEST_FULL WITH RECOVERY, REPLACE"
```

will search the backup history for the 'sales' database, and will then restore the latest full backup to the *sales_dev* database.

WITH options

CHECKDB

Runs a database integrity check (DBCC CHECKDB) on the database once the restore is complete. This checks the logical and physical integrity of all the objects in the specified database. `CHECKDB` cannot be used in conjunction with `NORECOVERY`. For more information, refer to your SQL Server documentation.

NO_INFOMSGS	Suppresses informational messages. Informational messages are only returned if other `CHECKDB` options are included in the command.
ALL_ERRORMSGS	Displays all reported errors and includes them in the log file.
TABLOCK	Obtains a lock on the database while the integrity check is performed. This includes a temporary exclusive lock which will prevent concurrent access to the database.
PHYSICAL_ONLY	Limits the database integrity check to the physical structure of the database. The contents of the check depends on the version of SQL Server you are using; refer to your SQL Server documentation for more information. This option cannot be included with `EXTENDED_LOGICAL_CHECKS` or `DATA_PURITY`.
DATA_PURITY	Checks database columns for invalid or out-of-range values. Column values are checked by default in databases created in SQL Server 2005 or later. This option cannot be included with `PHYSICAL_ONLY`.
EXTENDED_LOGICAL_CHECKS	Only available with SQL Server 2008 or later. Performs logical checks on an indexed view, XML indexes, and spatial indexes, where present. Running this check can have a considerable effect on performance. This option cannot be included with `PHYSICAL_ONLY`.
VERBOSE	Displays all reported error and informational messages and includes them in the log file.

CHECKSUM|NO_CHECKSUM

By default, if the backup process included `WITH CHECKSUM` the backup checksum and any page checksums are validated on restore. If the backup does not include a backup checksum, any page checksums will not be validated. Specify `NO_CHECKSUM` to disable default validation of checksums. If you specify `CHECKSUM`, the backup checksum and any page checksums will be validated as by default, but if the backup does not include a backup checksum, an error is returned. For more information, refer to your SQL Server documentation.

CONTINUE_AFTER_ERROR|STOP_ON_ERROR

`CONTINUE_AFTER_ERROR` specifies that the `RESTORE` process should continue after an error is encountered, restoring what it can. This is the default behavior for `RESTORE VERIFYONLY` (see VERIFY in The BACKUP command). The `RESTORE VERIFYONLY` process then reports all errors it has encountered.

`STOP_ON_ERROR` specifies that the `RESTORE` process should stop if an error is encountered. This is the default behavior for `RESTORE`.

For more information, refer to your SQL Server documentation.

DELAY

> DELAY is only available in SQL Backup Pro 7.4 and later.

Specifies a minimum age, in seconds, for transaction log backups. Only backups older than the specified age will be restored. This is useful if you are log shipping to maintain a standby database and want to delay restores to that database, for example, to help protect against corrupt or erroneous data.

The following example restores all transaction log backups in *C:\Backups\pubs\LOG* that are at least 30 minutes old to the *pubs* database. The backups are restored in the correct order.

```
"RESTORE LOG [pubs] FROM DISK = 'C:\Backups\pubs\LOG\*.sqb' WITH DELAY = 1800 "
```

DISCONNECT_EXISTING

Kills any existing connections to the database before starting the restore. Restoring to an existing database will fail if there are any connections to the database.

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 (reading or moving a backup file). If you omit this keyword, the default value of 10 is used. If you specify a value for `DISKRETRYCOUNT`, you should also specify a value for `DISKRETRYINTERVAL`.

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 (reading or moving 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.

DROPDB

Drops the database after the restore process (and database integrity check if used in conjunction with CHECKDB). The restored database is removed from the SQL Server instance regardless of whether any errors or warnings were returned.

DROPDBIFSUCCESSFUL

Drops the database if the restore completed successfully. When used in conjunction with CHECKDB, drops the database if the restore completed successfully and the database integrity check completed without errors or warnings.

ERASEFILES

Specifies the number of existing SQL Backup backups to be deleted from the MOVETO folder. This is useful for managing the number of backups in the MOVETO folder when log shipping. **Note:** You must also include FILEOPTIONS.

You can choose to delete SQL Backup backups 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.

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 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 restores a transaction log backup to the *pubs* database and leaves it in a non-operational state, then moves the backup to *C:\processed_logs* and deletes all transaction log backups of *pubs* other than the latest 10 from that folder:

```
"RESTORE LOG [pubs] FROM DISK = 'C:\shipped_logs\pubs\LOG_20120229_151009.sqb' WITH MOVETO = 'C:
\processed_logs', ERASEFILES = 10b, FILEOPTIONS = 1, NORECOVERY"
```


If ERASEFILES and ERASEFILES_REMOTE are included in the same command, the ERASEFILES_REMOTE setting overrides the ERASEFILES setting for *remote* MOVETO locations.

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_REMOTE and FILEOPTIONS

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

Manages deletion of existing SQL Backup backups from the DISK location. 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 restores a full backup to the *pubs* database, then deletes all full backups of the *pubs* database older than 5 days from *C:\Backups\pubs*.

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20130501_213000.sqb' WITH ERASEFILES_PRIMARY = 5"
```

To delete existing backups from the `MOVETO` location, use `ERASEFILES_SECONDARY`.

ERASEFILES_REMOTE

Manages deletion of existing SQL Backup backups from *remote* `MOVETO` folders. This is useful for managing the number of files in the `MOVETO` folder when log shipping.

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 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 restores a transaction log backup to the *pubs* database and leaves it in a non-operational state, then moves the backup to *\\Server01\processed_logs* and deletes all backups older than 6 hours from that folder:

```
"RESTORE LOG [pubs] FROM DISK = 'C:\shipped_logs\pubs\LOG_20120229_151009.sqb' WITH MOVETO =
'\\Server01\processed_logs', ERASEFILES_REMOTE = 6h, NORECOVERY"
```

To delete files from local `MOVETO` folders, use `ERASEFILES` and `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_REMOTE and FILEOPTIONS

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

Manages deletion of existing SQL Backup backups from the MOVETO folder.

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 restores a transaction log backup to the *pubs* database and leaves it in a non-operational state, then moves the backup to *C:\processed_logs* and deletes all transaction log backups of *pubs* over 12 hours old from that folder.

```
"RESTORE LOG [pubs] FROM DISK = 'C:\shipped_logs\pubs\LOG_20120229_151009.sqb' WITH MOVETO = 'C:
\processed_logs', ERASEFILES_SECONDARY = 12h, NORECOVERY"
```

The following example restores a full backup to the *pubs* database and deletes all full backups of *pubs* other than the latest 5 from *C:\Backups\pubs*, then moves the backup to *E:\Archive\pubs* and deletes all full backups of the *pubs* database older than 60 days from that folder.

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20130501_213000.sqb' WITH MOVETO = 'E:
\Archive\<DATABASE>', ERASEEFILES_PRIMARY = 5b, ERASEFILES_SECONDARY = 60"
```

FILEOPTIONS

Use in conjunction with ERASEFILES. Specifies whether backup files are to be deleted from the MOVETO folder. Specify the sum of the values that correspond to the options you require:

1	Delete backup files in the MOVETO folder if they are older than the number of days or hours specified in ERASEFILES.
2	Do not delete backup files in the MOVETO folder that are older than the number of days or hours specified in ERASEFILES if they have the ARCHIVE flag set.

Valid values are **1**, **2**, and **3**.

You must also set the age of the files to delete using ERASEFILES. For example, to delete backup files in the MOVETO folder that are older than 5 days:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH MOVETO = 'C:
\Backups\Archive\pubs', ERASEFILES = 5, FILEOPTIONS = 1"
```

To delete any existing files in the MOVETO folder that are older than 5 days and do not have the ARCHIVE flag set, (values 1 + 2):

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH MOVETO = 'C:
\Backups\Archive\pubs', ERASEFILES = 5, FILEOPTIONS = 3"
```

KEEP_CDC

Specifies that Change Data Capture settings are to be retained when a database or log is restored to another server.

This option cannot be included with `NORECOVERY`. Refer to your SQL Server documentation for more information.

KEEP_REPLICATION

This option is for use when log shipping is used in conjunction with replication. Specifies that replication settings are to be retained when a database or log is restored to a standby server.

This option cannot be included with `NORECOVERY`. Refer to your SQL Server documentation for more information.

LOG_ONERROR

Specifies that a log file should only be created if SQL Backup Pro encounters an error during the restore process, or the restore completes successfully but with warnings. Use this option if you want to restrict the number of log files created by your restore 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 restore process. Use this option if you want to restrict the number of log files created by your restore 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:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' 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:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH LOGTO = 'C:\Logs', LOGTO = 'C:
\Logs\SQBSecondaryLog.txt'"
```

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

MAILTO

Specifies that the outcome of the restore 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:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' 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 that the outcome of the restore operation is emailed to one or more users if SQL Backup Pro encounters an error during the restore process or the restore process 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. To specify multiple recipients, separate the email addresses with a semi-colon (**;**). For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' 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 that the outcome of the restore operation is emailed to one or more users if SQL Backup Pro encounters an error during the restore process. 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:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' 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.

MAXTRANSFERSIZE

Specifies the maximum size of each block of memory to be used when SQL Backup Pro restores 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:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' 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

MOVE DATAFILES TO

Specifies the data files should be restored to the specified location using the operating system file names defined in the backup file. The specified location must exist before the restore, otherwise the restore will fail. If this option is not included, the data files will be restored to the default locations.

This option can be used in conjunction with `MOVE 'logical_file_name' TO 'operating_system_file_name'` to move one data file to a new location and all other data files to a different location. For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH MOVE DATAFILES TO 'C:
\Test\NewDataLocation', MOVE 'TestDB_Primary2' TO 'E:\Test\NewDataLocation\AnotherFolder\TestDBPrimary2.mdf'"
```

In this example, *TestDB_Primary2* could have been restored to a new file name as well as to a different folder.

This option can also be used in conjunction with `LATEST_FULL` and `LATEST_ALL` (see `FROM DISK` argument).

MOVE FILESTREAMS TO

Only available with SQL Server 2008 and later. Specifies that filestreams should be restored to the specified location. The specified location must exist before the restore, otherwise the restore will fail. If the database contains multiple filestreams, each filestream will be restored to a separate subfolder. If this argument is not included, the filestreams will be restored to the default locations.

If the database includes multiple filestreams, this option can be used in conjunction with `MOVE 'logical_name' TO 'operating_system_file_name'` to specify one location for a particular filestream data container and another location for all other filestreams. For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH MOVE FILESTREAMS TO 'C:
\Test\NewFilestreamLocation', MOVE 'Test_Data' TO 'C:\Test\NewFilestreamLocation\AnotherFolder\Test_FSData'"
```

This option can be used in conjunction with `LATEST_FULL` and `LATEST_ALL` (see `FROM DISK` argument).

MOVE FULLTEXTCATALOGS TO

Only available with SQL Server 2005. Specifies that full text catalogs should be restored to the specified location. If the database contains multiple full text catalogs, each full text catalog will be restored to a separate subfolder. If this option is not included the full text catalogs will be restored to the default locations.

If the database includes multiple full text catalogs, this option can be used in conjunction with `MOVE 'logical_name' TO 'operating_system_file_name'` to specify one location for a particular full text catalog and another location for all other full text catalogs. For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH MOVE FULLTEXTCATALOGS TO 'C:
\Test\NewFullTextLocation', MOVE 'sysft_MoveFilesCatalog' TO 'C:
\Test\NewFullTextLocation\AnotherFolder\NewMoveFilesCatalog'"
```

This option can be used in conjunction with `LATEST_FULL` and `LATEST_ALL` (see `FROM DISK` argument).

MOVE LOGFILES TO

Specifies the log files should be restored to a new location with the operating system file names specified in the backup file. The specified location must exist before the restore process, otherwise the restore will fail. If this option is not included, the log files will be restored to the default locations.

This option can be used in conjunction with `MOVE 'logical_file_name' TO 'operating_system_file_name'` to move one log file to a new location and all other log files to a different location. For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH MOVE LOGFILES TO 'C:
\Test\NewLogLocation', MOVE 'TestDB_Log2' TO 'E:\Test\NewLogLocation\AnotherFolder\TestDBLog2.ldf'"
```

In this example, *TestDB_Log2* could have been restored to a new file name as well as to a different folder.

This option can also be used in conjunction with `LATEST_FULL` and `LATEST_ALL` (see `FROM DISK` argument).

MOVE '*logical_file_name*' TO '*operating_system_file_name*'

Specifies that the data file, log file, full text catalog (SQL Server 2005 only) or filestream data (SQL Server 2008 or later) identified by the logical file name should be restored to the physical location specified. The location must exist before the `RESTORE` command is executed. This option can also be used to rename the physical files. For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH MOVE 'pubs_data' TO 'F:
\Pubs02\Data\pubs_data02'"
```

MOVETO

Specifies that the backup files should be moved to another folder when the restore process completes. If the folder you specify does not exist, it will be created.

You must ensure that you have permission to delete files from the original folder, and to write to the MOVETO folder.

You can also use tags with the `MOVETO` argument, for example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH MOVETO = 'C:
\Backups\Archive\<INSTANCE>\<DATABASE>\'"
```

NOLOG

Prevents a log file from being created for the restore 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

Specifies that incomplete transactions are not to be rolled back on restore. The database cannot be used but differential backups and transaction log backups can be restored. For more information, refer to your SQL Server documentation.

ORPHAN_CHECK

Specifies that once the restore has completed, the database should be checked for orphaned users. Database user names are considered to be orphaned if they do not have a corresponding login defined on the SQL Server instance. Orphaned users are often created when you restore a database backup to a different SQL Server instance.

Note that `ORPHAN_CHECK` will only detect database users that are based on SQL Server logins; orphaned users based on Windows principals are not detected.

If orphaned users are detected, warning 472 is generated and each orphaned user is listed in the SQL Backup log file along with the associated SID.

PARTIAL

Specifies a partial restore of a database. The primary filegroup is restored, together with any specified secondary filegroups. See the DATABASE argument above for details on how to specify particular filegroups in a restore operation.

For more information, refer to your SQL Server documentation.

PASSWORD

Specifies the password to be used with encrypted backup files.

You cannot use the encrypted form of the password. This is to prevent unauthorized users from restoring backups if they have access to the encrypted password from the backup script.

SQL Backup version 3 allowed the use of encrypted passwords; these will no longer work. You must specify the password in unencrypted form:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs\FULL_20120229.sqb' WITH PASSWORD = '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.

RECOVERY

Specifies that incomplete transactions are to be rolled back. Recovery is completed and the database is in a usable state. Further differential backups and transaction log backups cannot be restored.

If no recovery completion state is specified, `WITH RECOVERY` is the default behavior. For more information, refer to your SQL Server documentation.

REPLACE

Specifies that the database should be restored, even if another database of that name already exists. The existing database will be deleted. `REPLACE` is required to prevent a database of a different name being overwritten by accident.

`REPLACE` is not required to overwrite a database which matches the name recorded in the backup.

For more information, refer to your SQL Server documentation.

RESTRICTED_USER

Specifies that access to the restored database is to be limited to members of the db_owner, dbcreator or sysadmin roles. Return the database to multi-user or single-user mode using your SQL Server application. For more information, refer to your SQL Server documentation.

SINGLERESULTSET

Specifies that the results returned by the RESTORE 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 `RESTORE` command will return two result sets by default in most cases, unless you specify the `SINGLERESULTSET` keyword.

STANDBY

Specifies a standby file that allows the recovery effects to be undone. The `STANDBY` option is allowed for offline restore (including partial restore). The option is disallowed for online restore.

Refer to your SQL Server documentation for more information about the `STANDBY` argument.

'standby_file_name'

Is a standby file used to keep a "copy-on-write" pre-image for pages modified during the undo pass of a `RESTORE WITH STANDBY`.

When used with the `RESTORE DATABASE` or `RESTORE LOG` command, '**standby_file_name**' can include tags, but these are not required.

STOPAT

Specifies a point in time to which a transaction log backup should be restored. The database will be recovered up to the last transaction commit that occurred at or before the specified time. When restoring to a point in time, include this option in each `RESTORE LOG` statement. For example:

```
"RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs_full.sqb' WITH NORECOVERY'"
"RESTORE LOG [pubs] FROM DISK = 'C:\Backups\pubs_log_20120601093000.sqb' WITH NORECOVERY, STOPAT = '2012-06-
01T09:40:30'"
"RESTORE LOG [pubs] FROM DISK = 'C:\Backups\pubs_log_20120601094500.sqb' WITH RECOVERY, STOPAT = '2012-06-01T09:
40:30'"
```

For more information, refer to your SQL Server documentation.

STOPATMARK

Specifies the point to which a transaction log backup should be restored, using either the log sequence number or a marked transaction. The database will be recovered up to and including the log record that contains the specified LSN or the marked transaction.

`AFTER` can be used when specifying a marked transaction and is useful when the mark name is not unique. The database is recovered as far as the first marked transaction to have occurred on or after the specified time.

For more information, refer to your SQL Server documentation.

STOPBEFOREMARK

Specifies the point to which a transaction log backup should be restored, using either the log sequence number or a marked transaction. The database will be recovered up to but excluding the log record that contains the specified LSN or the marked transaction.

`AFTER` can be used when specifying a marked transaction and is useful when the mark name is not unique. The database is recovered up to the first marked transaction to have occurred on or after the specified time.

For more information, refer to your SQL Server documentation.

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 is used.

Examples

Restore a database from a single file

This example restores a full backup of the pubs database from a single file.

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

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs] FROM DISK = ''C:\Backups\pubs_01.sqb'' WITH REPLACE" '
```

Restore a database from multiple (split) backup files

This example restores a full backup of the *pubs* database from two files.

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

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs] FROM DISK = ''C:\Backups\pubs_01.sqb'', DISK = ''C:
\Backups\pubs_02.sqb'' WITH REPLACE" '
```

Restore a database to a new name and move the database files

This example restores a full backup of the pubs database and restores it to a new database called *pubs02*. It also renames the database data and log files and moves them to a new location.

```
SQLBackupC.exe -I {instance name} -SQL "RESTORE DATABASE [pubs02] FROM DISK = 'C:\Backups\pubs_01.sqb' WITH
MOVE 'pubs' TO 'E:\Data\pubs02.mdf', MOVE 'pubs_log' TO 'E:\Data\pubs02.ldf' "
```

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs02] FROM DISK = ''C:\Backups\pubs_01.sqb'' WITH MOVE
''pubs'' TO ''E:\Data\pubs02.mdf'', MOVE ''pubs_log'' TO ''E:\Data\pubs02.ldf'' " '
```

Restore a database from the latest full backup and move the database files

This example restores the most recent full backup of the *pubs* database that matches the file name pattern and moves the database files to different locations. The specified locations must already exist.

```
SQLBackupC.exe -I {instance name} -SQL "RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs*.sqb' LATEST_FULL
WITH MOVE DATAFILES TO 'E:\Data', MOVE LOGFILES TO 'E:\Logs', REPLACE"
```

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs] FROM DISK = ''C:\Backups\pubs*.sqb'' LATEST_FULL WITH
MOVE DATAFILES TO ''E:\Data'', MOVE LOGFILES TO ''E:\Logs'', REPLACE" '
```

Restore a database from the latest full backup on different disks

This example searches multiple disk locations for full backups of the *pubs* database that match the file name pattern and restores the latest full backup.

```
SQLBackupC.exe -I {instance name} -SQL "RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups*pubs*.sqb', DISK = 'D:
\Backups*pubs*.sqb', DISK = 'E:\Backups*pubs*.sqb' LATEST_FULL WITH REPLACE"
```

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs] FROM DISK = ''C:\Backups*pubs*.sqb'', DISK = ''D:
\Backups*pubs*.sqb'', DISK = ''E:\Backups*pubs*.sqb'' LATEST_FULL WITH REPLACE
```

Restore a database from the latest backup set on different disks

This example searches multiple disk locations for full, differential and transaction log backups of the *pubs* database that match the file name pattern, and restores the most recent full backup, followed by the most recent differential backup and the most recent transaction log backups.

```
SQLBackupC.exe -I {instance name} -SQL "RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups*pubs*.sqb', DISK = 'D:
\Backups*pubs*.sqb', DISK = 'E:\Backups*pubs*.sqb' LATEST_ALL WITH REPLACE"
```

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs] FROM DISK = ''C:\Backups*pubs*.sqb'', FROM DISK = ''D:
\Backups*pubs*.sqb'', DISK = ''E:\Backups*pubs*.sqb'' LATEST_ALL WITH REPLACE" '
```

Restore to a new database from the latest backup set and check for orphaned users

This example restores the most recent full backup of the *pubs* database, followed by the most recent differential backup and the most recent transaction log backups available from C:\Backups to a new database called pubs02 and checks for orphaned users.

```
SQLBackupC.exe -I {instance name} -SQL "RESTORE DATABASE [pubs02] FROM DISK = 'C:\Backups\pubs*.sqb' SOURCE =
'pubs' LATEST_ALL WITH ORPHAN_CHECK"
```

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs02] FROM DISK = ''C:\Backups\pubs*.sqb'' SOURCE =
''pubs'' LATEST_ALL WITH ORPHAN_CHECK" '
```

Restore a database from an encrypted backup file

This example restores an encrypted backup of the *pubs* database, specifying the password *MyPassword*.

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

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs] FROM DISK = ''C:\Backups\pubs_01.sqb'' WITH PASSWORD =
''MyPassword'' "
```

Restore a database from an encrypted backup file using a password stored in a text file

This example restores an encrypted backup of the *pubs* database using the password stored in the *password.txt* file.

```
SQLBackupC.exe -I {instance name} -SQL "RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs_01.sqb' WITH
PASSWORD = 'FILE:C:\mypasswords\password.txt' "
```

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs] FROM DISK = ''C:\Backups\pubs_01.sqb'' WITH PASSWORD =
'FILE:C:\mypasswords\password.txt' "
```

Restore a database in NORECOVERY mode

This example restores a full backup of the *pubs* database, specifying that the database is to be left in an unrecovered state so that differential and transaction log backups can be restored to it.

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

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs] FROM DISK = ''C:\Backups\pubs_01.sqb'' WITH
NORECOVERY"
```

Restore a database in READ-ONLY mode

This example restores a full backup of the *pubs* database, specifying that the database should be left in an unrecovered, read-only state so that its data can be viewed and differential and transaction log backups can be restored to it. The location of the standby file is specified.

```
SQLBackupC.exe -I {instance name} -SQL "RESTORE DATABASE [pubs] FROM DISK = 'C:\Backups\pubs_01.sqb' WITH
STANDBY = 'C:\Standby\pubs_log.DAT' "
```

```
EXECUTE master..sqlbackup '-SQL "RESTORE DATABASE [pubs] FROM DISK = ''C:\Backups\pubs_01.sqb'' WITH STANDBY =
''C:\Standby\pubs_log.DAT'' " '
```