Søren Madsen
Part of my descriptive guides collection, Version 1.0

Simple Backup

Microsoft Batch based backup

Table of contents

Intro

This document describes the Microsoft Batch based script Simple Backup.

1.1 Disclaimer

The script has only been sporadically tested on:

- Microsoft Windows 10 version 2004 64Bit installation (Regional settings: Danish).
- Microsoft Windows 10 version 20H2 64Bit installation (Regional settings: Danish).

In other words it has NOT been tested on a full blown Continuous Integration server with 100% code coverage. The script should not be considered production ready software and it was developed for my personal setup on my own PC. Some of the files contain functions that are untested and only stored not to forget the implementation idea. However the functions that are actually used in the backup process have been tested rather extensively by manual execution of the script during my own backups.

Feel free to use this script/software at your own risk.

1.2 Basic requirements

The script should be able to archive individual folders and not just entire partitions.

The script should be able to calculate checksums for the files created and store these in a folder with the archive file/files.

The script should be able to log to a file, enabling the user to read the program feedback and store this file in a folder with the archive file/files.

The script should support subversion versioning tools to ease the backup of svn repositories as an included backup functionality in the standard backup performed by the script.

The code should be generalized and wrapped in easy to use functions.

1.3 Limitations and shortcomings

Generally speaking the naming convention of files and folders could be improved a bit. The code base has been developed using an adhoc approach leaving some process improvements to be desired :-). That is a luxury one has when coding own projects for fun, but a better process would minimize a lot of “rushed code errors” and thereby minimize the time spent debugging the code afterwards.

The code base contains untested functions that are kept to ensure the implementation idea is not forgotten if they could be used at a later point in time. These functions might be a bit “messy” but they are a result of implementation research and could come in handy in the future. These functions are NOT used in the backup process. An improved naming convention would probably identify these functions explicitly.

The function Backup.cmd->evaluation is error prone. Because of the way SETLOCAL encapsulates variable values in a locked local scope it has been problematic to pass the environment vari-

able ERRORLEVEL to another function when DELAYEDEXPANSION is on. That results in an evaluation function that sometimes work as expected (without delayedexpansion enabled) and other times fail to report the ERRORLEVEL value. Microsoft Batch does not seem to be developed for this kind of programming style and returning and parsing values to and from functions have been difficult to achieve.
This generally means that encapsulation in this project is minimized and some functions are rather large. It would have been nice to be able to break the bigger functions down to smaller functions and thereby generalizing more of the code. This minimizes maintenance by only having a single implementation of each functional requirement.
The project does not handle thread synchronization problems. So running multiple scripts at the same time is out of the scope of this implementation.

1.4 Structure

The script consists of several cmd files, and ini-files. I tried to “mimic” the function-calling syntax in regular general-purpose procedural programming languages to avoid repetitive code and to wrap general functionality into usable functions. The primary benefit of this is single point of failure in functions instead of sequential coding, which would generate a code base with a lot of repeating code patterns resulting in changes that require a lot of code editing.

The structure of the script:

```
|-- .\SimpleBackup
    |-- Backup.cmd
    |-- fileSystem.cmd
    |-- logging.cmd
    |-- Multiple_Backups.cmd
    |-- SvnRepoFunctions.cmd
    |-- utillity_functions.cmd
    |-- Settings.ini
    |-- HowTo_Description.pdf
    |-- .\FullBackup
        |-- BackupFolders.cmd
        |-- Backup.txt
        |-- BackupSettings.ini
    |-- .\AnotherBackupConfigurationFolder
        |-- BackupFolders.cmd
        |-- Backup.txt
        |-- BackupSettings.ini
```

Script base folder: SimpleBackup

This folder is the root folder of the entire script.

File: backup.cmd

This file contains the main backup script and performs all the essential functions. This includes precondition checks, the function requested, user feedback and error-handling.

File: fileSystems.cmd

This file handles file system operations. Move folder, copy folder, URL-verification, path cleanup functions, file deletion, creation etc. Not everything is implemented since functions were implemented as I needed them during development.

File: logging.cmd

This file handles logging. It wraps the echo command and it supports echoing to graphical user interface and to file.

File: Multiple_Backups.cmd

This file makes it possible to utilize more the one application function. Each backup configuration can only handle a single application function, which is defined in the

associated ini-file (BackupSettings.ini). By combining more backup configurations in this file more application functions can be performed.

One of my use cases is:

A backup of my Raspberry Pi Os operating systems installation including svn commit of

the image checksum file.

Then

A full export of my svn repositories and a full backup of my important folders.

Using this file makes that procedure possible with a single mouse click.

File: SvnRepoFunctions.cmd

This file handles all the subversion functionality. Svn update, svn status, svn add, svn commit, svn dump file export. To make this functionality work Subversion command-line tools must be installed on the system.

File: utility_functions.cmd

This file is sort of a toolbox file with no predetermined function purpose. All functions that do not fit in either of the other files are regarded as being general purpose utility functions.

File: Settings.ini

This file sets up the general script parameters. Paths to executables, code page definition to support specific cmd.exe regional setups. Default setup is utf-8.

File: Howto_Description.pdf

This file is this file :-). Documentation.

Folder: FullBackup (Configuration folder)

This folder is an example of a backup configuration. It will backup the folders defined in Backup.txt

File: Backup.txt

This file defines which files and folders will be backed up. All sub folders and files in a given folder will be backed up.

File: BackupFolders.cmd

This file reads the ini-files to setup internal variables used by backup.cmd. Furthermore it is used to perform post backup procedures if the user requires it. An example of this is my implementation of 'svn committing' the checksum file of my raspberry pi image backup.

File: BackupSettings.ini

This file defines the actual backup application function used in a backup configuration and various optional settings. Examples: Password, archive split file functionality, enable/disable file logging etc.

1.5 Installation tips

The script requires the following software to work:

- Microsoft Windows (Tested on Windows 10 Home version 2004).
- Microsoft Windows (Tested on Windows 10 Home version 20H2).
- An archive program – Default is 7zip.
  Changing to another archive program will probably require some script up dates.
- Subversion command line tools: svn.exe and svnadmin.exe.
  Tested using Visual SVNServer and the provided command line tools.
  Alternatively tortoiseSvn also provides the command line tools.
- Checksum calculations are done using certutil.
  7zip also provides the ability to calculate checksum if an alternative is wanted.

If all the software is available on the system you will have to:

1. Copying the SimpleBackup files to the PC to a folder the user has sufficient access to.
2. Configure the .\SimpleBackup\Settings.ini

FullBackup example:

3. Configure the file .\SimpleBackup\FullBackup\BackupSettings.ini
4. Configure the file .\SimpleBackup\FullBackup\Backup.txt to define which folders to backup.
5. Run the file .\SimpleBackup\FullBackup\BackupFolders.cmd

1.6 File verification

This project supports 3 types of file verification.

1.6.1 7zip archive integrity verification

This is the verification method used by application function varAppFunctionIntegrityCheck. It is used to verify the validity and integrity of the archive file parsed to 7zip. It tests if the file "works" or if it is corrupted. If it tests OK the file will be able to be opened and extracted correctly. If it is cor-

rupted there is no guaranty that the file is usable. This test does NOT ensure that the archive file/ files are unchanged.

1.6.2 Checksum verification using certutil

This is the verification method used by application function varAppFunctionVerifyChecksum. It is used to test if the archive file has changed since creation by comparing a newly calculated checksum to the checksum stored in the checksum file when the archive file was created.

If the 2 checksums are equal the file remains intact and unchanged.

If the 2 checksum are not equal the file has changed since it was created. This can be due to corruption or just plain editing by opening the file in the archive manager program and adding/deleting or changing files.

Simple Backup supports the following checksum algorithms using certutil v10.0.19041.1 on Microsoft Windows 10 20H2: (MD2 | MD4 | MD5 | SHA1 | SHA256 | SHA384 | SHA512).

The default value is set to SHA512. Choosing a lower bit length increases the probability of a checksum collision. A checksum collision is a scenario in which the chosen checksum algorithm generates the same checksum value for 2 different files. Choosing SHA512 ensures the lowest probability for a checksum collision. Choosing MD2 however increases the probability of a checksum collision when compared to the other higher bit length algorithms.

1.6.3 Svnadmin status - VarCheckWorkingCopyChanges

The third and last technique used to check things is using the fact that the project files are stored in a svn repository. By using svnadmin status command it is possible to obtain a high degree of certainty that the project files are in a consistent state and will perform the functionality as expected. This does not verify the generated backup files but checks the executable script files themselves. The function CheckWorkingCopyForChanges in the file svnRepoFunctions.cmd is the implementation of this functionality and it supports quite a bit of flexibility through the parameters the user can provide it. Further details can be found in CheckWorkingCopyForChanges in the file svnRepoFunctions.cmd.

In my setup all the cmd files found in the root folder of SimpleBackup may not have changes in them compared to the file version found in the svn repository.

However I have accepted changes to the file Multiple_Jobs.cmd when comparing it to the svn repository version to make it easier to enable/disable the Raspberry Pi image backup configuration without having to do a svn commit each time I enable or disable the mentioned backup configuration.

1.7 Function and settings description

This chapter describes all the entries in the configuration ini-file BackupSettings.ini which is the main user entry point to the script. Settings.ini from the base folder should be self explanatory.
All the entries in the ini-file must be defined or the script will exit with an error message. No spaces are allowed in the variable name and the equal operator must be directly aligned without spaces between the variable name and the value.
The paths do however support spaces in folder names!

The example below works in most cases.
C:\folder with spaces\folder1
If Svn functions are called with a path like the above example problems might occur. If a space occurs in a folder name before the root folder of the svn repository, the svn.exe and svnadmin.exe will fail and the script will exit. Please use symbolic linking to solve this (mklink.exe). If a space occurs in a folder name INSIDE the svn repository working copy it works fine and svn.exe and svnadmin.exe will handle it without problems.

The variables are defined in the format
variableName=Value

###################### General settings #

varBackupLocation
File system path: The backup destination on the user file system.

varFileList
File system path: Path to the file defining the folders to be backed up.

###################### Application functions #

varAppFunctionBackupFiles
Application function: This option enables Folder backup. This enables the backup function which backs up the folders from Backup.txt. (YES | NO)

varAppFunctionIntegrityCheck
Application function: This option enables Integrity test. It tests an existing archive file for archive file integrity. (YES | NO)

varAppFunctionUpdateArchive
Application function: This option enables the archive update function. It updates an existing archive file with the folders from Backup.txt. (YES | NO)

varAppFunctionExtractFilestoFolder
Application function: This option enables the extract to folder function. It extracts an existing archive file to a destination folder WITHOUT preserving folder structure. (YES | NO)

varAppFunctionExtractFilesWithFullFilePath
Application function: This option enables the extract to folder with full path function. It extracts an existing archive file to a destination folder and preserves the original folder structure. (YES | NO)

varAppFunctionVerifyChecksum
Application function: this option enables the function that verifies the SHA512 checksum/ checksums stored in a file in the backup destination directory. This works for both single file archives and split file archives.

###################### Existing archive - Integrity test, update, extraction #

varExistingArchivePath
File system path: Path to the backup folder that contains the archive to use. This is required by all function except varAppFunctionBackupFiles.

varExistingArchiveFileName=2020-11-09_10-34-backup.zip
File system path: Name of the archive file to use. This is required by all function except varAppFunctionBackupFiles.

VarExistingChecksumFile
File system path: Name of the SHA512 checksum file to use. This is required by the function varAppFunctionVerifyChecksum.

varExtractionLocation
File system path: Target location for the extracted files and folders. Value can be either DEFAULT_LOCATION or a path to a file system destination. DEFAULT_LOCATION == varExistingArchivePath\ExtractedArchiveContent

###################### Advanced settings #

varPassword
Application option: Enable the password function of the archive program. (YES | NO)

varSecretPassword
Application option: Password, If a password is defined in this option the script will automatically add this to the archive if varPassword=YES. That way there is no need to input a password when the script is running. The password is visible in clear text in the ini-file though. Default value: NO

varSplitArchiveFile
Application option: This option enables archive split file functionality. It will split the archive into chunks. (YES | NO)

varEnableFileLogging
Application option: This option enables or disables file logging. (YES | NO).
The script generates a backup log and a svn dump file export log in either of the relevant destination folders.

varMoveFolders=NO
varMoveFoldersBack=NO
Application options: These options enable or disable the inbuilt move-folder functionality. If certain folders should not be part of the backup archive but resides inside the file system that is going to be backed up this functionality can temporarily move those folders and move them back after backup has finished. Currently only 2 folders can be defined and if more folders are required the script requires modification.
varSrcPathFolder01=D:\test\Source\TestFolder\
varSrcPathFolder02=D:\test\Source\TestFolder2\
varDstPathFolder01=D:\test\Destination\TestFolder\
varDstPathFolder02=D:\test\Destination\TestFolder2\
File system paths: Source and destination paths to the folders that should be moved.

To add more folders: Update backup.cmd->:MoveMultipleFolders and :MoveMultiple-FoldersBack. Possibly also the relevant backup.cmd→PreconditionalCheck function.

varOverWriteFiles
Application option: This option is used by the extract files application function. It defines what to do when file collisions occur.
Values: OVERWRITE_EXISTING_FILES, SKIP__EXISTING_FILES, AUTO_RENAME_EXTRACTING_FILE, AUTO_RENAME_EXISTING_FILE

varUpdateMode=DEFAULT_FUNCTIONALITY
Application option: # UNTESTED: 7zip supports update flags that define how 7zip handles file updates in varying scenarios.
Values: DEFAULT_FUNCTIONALITY (Same as using no flags), https://sevenzip.osdn.jp/chm/cmdline/switches/update.htm
If a none default value is used the value in the ini-file is directly copied to the 7z.exe statement. No intermediate parsing. This has not been tested successfully but 7zip supports it. So if the user wants to experiment the ground work has been made.

varZipUtcMode
Application option: Enable UTC mode for zip archives in backup and update mode. This is highly recommended if using UTC based file systems like NTFS and the file is being sent to another time zone. (YES | NO).

VarChecksumVerificationDuringBackup
Include SHA512 checksum verification after generating the backup archive. (YES | NO)
This is not the same as the application function varAppFunctionVerifyChecksum. Instead this option performs a SHA512 checksum verification on the backup archive generated by varAppFunctionVerifyChecksum.

varIntegrityTestDuringBackup
Application option: Include integrity-test after generating the backup archive. (YES | NO)
This is not the same as the application function varAppFunctionIntegrityCheck. Instead this option performs an integrity check on the backup archive generated by varAppFunction-BackupFiles.

Application option:
Enable checksum verification during the backup process. This ensures that the calculated checksum is verified before the archive file is stored in persistent storage. Incorrectchecksum generation might occur if the logging.cmd file is in an inconsistent state when storing the value. This is caused by the way the echo command is used because trailing white space might affect the calculated value.
varChecksumVerificationDuringBackup=YES

Application option: SHA checksum bitlength
Choose which checksum algorithm to use when generating archive file checksum values. Default value is SHA512. Choosing a lower bit length increases the probability of a checksum collision (different files having the same checksum value), but it decreases the checksum generation time.
varChecksumBitlength=SHA512

###################### Subversion: Options, paths and settings #

This script can handle subversion repositories. The script is hard-coded to handle 2 repositories and svn checkouts. Other user requirements will demand script update.

varCheckWorkingCopyChanges
Application option: This function is used to verify the svn state of the working copy before performing critical functions like calculating checksums etc. (YES | NO)

varSimpleBackupCheckoutPath
File system path: Path to the svn working copy of SimpleBackup

varExportSvn
Application option: Repository export: Export svn repository. (YES | NO)

varRepositoryLocation=C:\tmp\repository
File system path: Repository export: Svn repository base folder location

varRepositoryDumpLocation
File system path: Repository dump location: Where to store the repository dump files

varSvnRepo1
File system path: This is the name of the repository to export. This folder name is appended to varRepositoryLocation in the script.

Repository export: Repository export: Svn repository 2:
varSvnRepo2
File system path: This is the name of the repository to export. This folder name is appended to varRepositoryLocation in the script.

###################### 7zip specific settings #

varFormat
Application option: Backup format. This variable enables the usage of multiple format. Currently zip and 7z is the only supported formats

varCompressionLvl
Application option: Compressionslvl 0 (copy mode), 1 (fast), 3, 5 (Normal), 7, 9 (Ultra). Copy mode: -mx0

varThreadAffinity
Application option: Number of used threads (-mmtN). 10 threads: -mmt10

varSolidMode
Application option: Solid Mode. This is a 7zip specific option. Default value is -ms=on. Solid archives cannot be updated. (YES | NO)

varSplitVolumesize
Application option: VolumeSize, Chosen size options for this script: 1MB, 2MB, 5MB, 10MB, 100MB, 1GB, 2GB, 5GB, 10GB, 100GB. 1MB = -v1m, 2MB = -v2m, 5MB = -v5m,

10MB = -v10m, 100MB = -v100m, 1GB = -v1g, 2MB = -v2g, 5GB = -v5g, 10GB = -v10g, 100GB = -v100g

1.8 Configured example of an ini-file

```
#All variables must be initialized or the script will not work. All varItem=VALUE must be set.
###################### General settings #
# Target location for the backup
varBackupLocation=E:\Backup\FullBackup

# List file
varFileList=Backup.txt

###################### Application functions #
# Backup files into archive-file. (YES | NO)
varAppFunctionBackupFiles=YES

# Test archive integrity. (YES | NO)
varAppFunctionIntegrityCheck=NO

# Backup files into archive-file. (YES | NO)
varAppFunctionUpdateArchive=NO

# Backup files into archive-file. (YES | NO)
varAppFunctionExtractFilestoFolder=NO

# Backup files into archive-file. (YES | NO)
varAppFunctionExtractFilesWithFullFilePath=NO

# Verify the SHA512 checksum/checksums for an existing archive/Split file archive. (YES | NO)
varAppFunctionVerifyChecksum=NO

###################### Existing archive - Integrity test, update, extraction #
#  Path to folder that contains the archive to use.
varExistingArchivePath=E:\Backup\FullBackup\2020-01-01_10-10

# Name of the file.
varExistingArchiveFileName=2020-01-01_10-10-backup.7z

# Name of the file.
```

varExistingChecksumFile=2020-01-01_10-10-Checksum-SHA512.txt

Target location for the extracted files and folders. DEFAULT_LOCATION == varExistingArchivePath\ExtractedArchiveContent

Value can be either DEFAULT_LOCATION or a path to a file-system destination.

varExtractionLocation=C:\test\Extract

########## Advanced settings #

Set password (YES | NO)

varPassword=YES

That way there is no need to input a password when the script is running. The password is visible in clear text in this file though.

Default value: NO

varSecretPassword=MyPassWord

Create volumes, split archive into chunks (YES | NO) - 7Zip documentation: -v{Size}[b|k|m|g] : Create volumes

varSplitArchiveFile=NO

Enable logging

varEnableFileLogging=YES

Settings for moving folders before backup and moving them back afterwards.

To add more folders: Update backup.cmd->:MoveMultipleFolders and :MoveMultipleFoldersBack. Possibly also backup.cmd->PreconditionalChecks

varMoveFolders=NO

varMoveFoldersBack=NO

varSrcPathFolder01=D:\test\Source\TestFolder\

varSrcPathFolder02=D:\test\Source\TestFolder2\

varDstPathFolder01=D:\test\Destination\TestFolder\

varDstPathFolder02=D:\test\Destination\TestFolder2\

Overwrite existing files. This is used by extract functions.

Values: OVERWRITE_EXISTING_FILES, SKIP__EXISTING_FILES, AUTO_RENAME_EXTRACTING_FILE, AUTO_RENAME_EXISTING_FILE

varOverWriteFiles=SKIP__EXISTING_FILES

UNTESTED: 7zip supports update flags that define how 7zip handles file updates in varying scenarios.

Values: DEFAULT_FUNCTIONALITY (Same as using no flags), https://sevenzip.osdn.jp/chm/cmdline/switches/update.htm

If a none default value is used the value in the ini-file is directly copied to the 7z.exe statement. No intermediate parsing.

varUpdateMode=DEFAULT_FUNCTIONALITY

Enable UTC mode for zip archives in backup and update mode. This is highly recommended if using UTC based file systems like NTFS

and the file is being sent to another time zone. (YES | NO)

varZipUtcMode=NO

Include integrity-test after generating the backup archive. (YES | NO)

varIntegrityTestDuringBackup=NO

Enable checksum verification during the backup process.

This ensures that the calculated checksum is verified before the archive file is stored

in persistent storage. Incorrect checksumgeneration might occur if the logging.cmd file is in an inconsistent state

when storing the value. This is caused by the way the echo command is used because trailing whitespace might affect the calculated value.

Values: (YES | NO)

varChecksumVerificationDuringBackup=NO

SHA checksum bitlength

Values supported by certutil v10.0.19041.1 on Microsoft Windows 10 20H2: (MD2 | MD4 | MD5 | SHA1 | SHA256 | SHA384 | SHA512)

Default value is SHA512. Choosing a lower bitlength increases the probability of a checksum collision (different files having the same checksum value),

but it decreases the checksum generation time.

varChecksumBitlength=SHA512

###################### Subversion: Options, paths and settings #

This script can handle subversion repositories. Use the options and settings in this section to configure this functionality.

The script is hard-coded to handle 2 repositories and svn checkouts. Other user requirements will demand script update.

This function is used to verify the svn state of the working copy before performing critical functions like calculating checksums etc.

varCheckWorkingCopyChanges=YES

Simple Backup

```
# Path to the svn working copy of SimpleBackup
varSimpleBackupCheckoutPath=C:\workingcopyPath

# Repository export: Export svn repository (YES | NO)
varExportSvn=YES

# Repository export: Svn repository location
varRepositoryLocation=D:\Repositories

# Repository dump location: Where to store the repository dump files
varRepositoryDumpLocation=D:\SVN_DUMP

# Repository export: Svn repository 1:
varSvnRepo1=repoName1

# Repository export: Repository export: Svn repository 2:
varSvnRepo2=repoName2

####################### 7zip specific settings #
# varFormat=zip
varFormat=7z

# Compressionslvl 0 (copy mode), 1 (fast), 3, 5 (Normal), 7, 9 (Ultra)
varCompressionLvl=-mx5

# Number of used threads (-mmtN)
varThreadAffinity=-mmt10

# Solid Mode. This is a 7zip specific option. Default value is -ms=on.
# Solid archives cannot be updated. (YES | NO)
varSolidMode=NO
# VolumeSize, Chosen size options for this script: 1MB, 2MB, 5MB, 10MB, 100MB, 1GB, 2GB, 5GB, 10GB, 100GB
# 1MB = -v1m, 2MB = -v2m, 5MB = -v5m, 10MB = -v10m, 100MB = -v100m, 1GB = -v1g, 2MB = -v2g, 5GB = -v5g, 10GB = -v10g, 100GB = -v100g
varSplitVolumesize=-v5g
```