robopowercopy

RoboPowerCopy

A PowerShell "clone" of the famous "ROBOCOPY" tool. In fact I've adapted the funcionality of "robocopy" in PowerShell to provide you and myself a "robust file copy" tool.

Today I want to announce my latest development project.

RoboPowerCopy (v0.1.0)

It’s a PowerShell 2.0 based clone of Microsoft’s famous ***RoboCopy*** tool.

Please see the corresponding post on my blog:

http://ikarstein.wordpress.com/2011/06/16/robopowercopy-a-powershell-based-robocopy-clone-v0-1-0

RoboCopy tool is part of my daily work. At work and at home. – You all now it! I hope so Zwinkerndes Smiley

I spend the spare time of some weeks for this project and now it’s ready to be published as **ALPHA version** (v0.1.0.0). The development is still in progress!! – Hopefully there is somebody out there that will use it – beside me.

I’ve done this using the (also) famous PowerGUI (Community Edition) by Quest.

“But way” you may ask.

RoboCopy is an EXE file and it has closed source. We (the customers) are not able to change it’s behavior. E.g. I’m missing a “verify” feature for years. Some people say: “It’s a copy tool not a backup tool”. Okay. That’s right. But I want to be sure that the copied data is valid and identical to the source data.

Because of that I thought about the possibility to realize a ***RoboCopy*** clone with PowerShell. – So I did it now.

RoboPowerCopy is plain PowerShell. No assemblies or PowerShell extensions are needed. – It uses some inline C# code.

RoboPowerCopy should “understand” at least the commands (options) of ***RoboCopy***. – Currently I’ve implemented a lot of ***RoboCopy***’s abilities. Not all. But some new features on top, e.g. verify and “read-and-check-before-copy”. “Verify” uses .NET hashing algorithms to compare the written data to the source data. If the destination file already exists the “Read-and-check-before-copy” feature reads every portion of data from the destination file before it replaces the data portion on the destination file with the source file’s data. This should reduce the time that is needed to copy large files that have only a few binary changes.

As you may now .NET has no build in capabilities to handle with very long paths with more than 256 characters. I have developed a set of helper classes in C# that enable ***RoboPowerCopy*** to deal with very long paths. - This C# is included in the script an will be compiled at runtime into an assembly in memory.

Here are the abilities:

- Long path support (!!)
- Restartable mode using a temporary "copy header" at the end of the copied file.
- Copy security: SACL, DACL, Owner
- Copy file attributes and timestamps
- Same option set as ***RoboCopy*** (not completely implemented)
- Can be extended because it's open source
- Customizable
- Pure PowerShell with some inline C# code
- Compare file times as UTC
- Modify it on any Windows Operation System >= Win XP / Windows 2003 Server with Microsofts PowerShell development tol *Windows PowerShell Integrated Scripting Environment* that is part of PowerShell.

These are the limitations:

- No multithreading support
- No "Job" files at the moment
- Slower than ***RoboCopy*** (Because ***RoboPowerCopy*** is Powershell and/based on Managed Code but ***RoboCopy*** is native optimized code…)

The following options I have implemented (v0.0.1.0). There are some options that are not available in ***RoboCopy***. These I've marked red.

Option	**Description**	**since Version**
/A	Copy only files that have the Archive attribute set	0.1.0.0
/M	Removes Archive attribute of source file.	0.1.0.0
/A+:[HRAS]	Add this file attributes to destination file	0.1.0.0
/A-:[HRAS]	Remove this file attributes from destination file	0.1.0.0
/E	Copy subfolder, including empty	0.1.0.0
/S	Copy subfolders that are not empty. – "Not empty" means: at least one file is exists and is not excluded by any exclude option	0.1.0.0
/PURGE	Remove files on destination that does not exist on source	0.1.0.0
/MIR	Mirror the directory tree, like "/PURGE" combined with "/E"	0.1.0.0
/COPYALL	Copy all file info: Timestamps, Attributes, Data, Owner, DACL, SACL	0.1.0.0
/COPY:[DATSOU]	Copy selected file info: D=Data, A=Attributes, T=Timestamps, S=DACL, U=SACL, O=Owner	0.1.0.0
/DCOPY:[T]	Copy selected directory info: T=Timestamps	0.1.0.0
/CREATE	Does only create the directory structure. Files will have zero size.	0.1.0.0
/SEC	Copy files with security. Like "/COPY:DATS"	0.1.0.0
/LEV:n	Copy only n directory levels. 1=Copy only the given root directory.	0.1.0.0
/IS	Copy even destination file seems to be the same as the source file.	0.1.0.0
/RCBW	During overwrite this methos reads a file portion of data of the destination file and compares them with the source files portion of data before writing it. This my be useful while copying large files that have only few changes.	0.1.0.0
/VERIFY	Reread written data and compare them to the source data.	0.1.0.0
/SECFIX	Replaces the security info of all destination files even if they are equal to the source.	0.1.0.0
/TIMFIX	Replaces the timestamps of all destination files even if they are equal to the source.	0.1.0.0
/MAX:n	Max file size to copy (in bytes)	0.1.0.0
/MIN:n	Min file size to copy (in bytes)	0.1.0.0
/MAXAGE	Used to specify maximum file age based upon "LastWriteTime". That will exclude files older than n days/date. (If n < 1900 then n = no of days, else n = YYYYMMDD date).	0.1.0.0
/MINAGE	Used to specify minimum file age based upon "LastWriteTime". That will exclude files newer than n days/date. (If n < 1900 then n = no of days, else n = YYYYMMDD date).	0.1.0.0
/MAXLAD		0.1.0.0

	Used to specify maximum file age based upon "LastAccessTime". That will exclude files unused since n days/date. (If n < 1900 then n = no of days, else n = YYYYMMDD date).	
/MINLAD	Used to specify minimum file age based upon "LastAccessTime". That will exclude files used since n days/date. (If n < 1900 then n = no of days, else n = YYYYMMDD date).	0.1.0.0
/XD:<directory> [<directory> […]]	Exclude this directories. Wildcards can be used for name matching.	0.1.0.0
/XF:<file> [<file> […]]	Exclude this files. Wildcards can be used.	0.1.0.0
/XJ	Exclude Junction Points, Symbolic Links (directories), Hard Links (files)	0.1.0.0
/XJD	Exclude Junction Points, Symbolic Links (directories)	0.1.0.0
/XJF	Exclude Hard Links (files)	0.1.0.0
/V	Detailed output.	0.1.0.0
/VV[+][:<file> <filename>]	Verbose output. If no file name is specified the messages will be written to the Powershell host using "write-verbose"	0.1.0.0
/CHUNK:n	Size of the size of a portion of data during copy. In bytes. Default: 10mb.	0.1.0.0
/Z	Restartable mode. A "copy header" will be used at the end of each file during the copy process. The header will be removed after copy is finished.	0.1.0.0
/BYTES	Shows file sizes in bytes	0.1.0.0
/R:n	Maximum retry count if copy fails	0.1.0.0
/W:n	Wait duration before next retry. (In seconds)	0.1.0.0
/256	Disable Long Path Support. This will use standard .NET classes (FileInfo, DirectoryInfo) instead of the special classes for file and directory names longer than 256 characters.	0.1.0.0
/NP	No progress – does not display "%"…	0.1.0.0

Using this option (except the red once) with ***RoboPowerCopy*** and ***RoboCopy*** should have the same result in both tools.

This features of ***RoboCopy*** does not work in ***RoboPowerCopy*** in v0.1.0.0:

- Copy directory attributes
- Set "Encrypted" attribute on destination files
- Set "Compressed" attribute on destination files
- …some more

Once more: It's "ALPHA" in v0.1.0.0! – Therefore I want to ask for your help! – Please help to improve *RoboPowerCopy*. The project needs people how test and develop the script.

Please contact me at the projects Codeplex page if you would to spend your spare time for *RoboPowerCopy*.