This has been moved from Codeplex (http://ps2exe.codeplex.com)...

## PS2EXE - v0.5

**Release Notes:**

- v0.5
- v0.4
- v0.3

### Original description of version 0.1 with updates - especially in the "Usage" section

In the last days I created the tool “PS2EXE”. It is able to “convert” PowerShell scripts to “standalone” EXE files.

But: It does **not** convert the PowerShell script to an other language! It encapsulates the script with a lightweight PowerShell host written in C# and compiles the dynamically generated C# source code in memory to an EXE file. The resulting EXE is an .NET assembly that contains the source script encoded in Base64. The EXE includes all stuff that is needed to execute an PowerShell through the .NET object model. It is based on classes in the namespace System.Management.Automation that reperents the PowerShell engine. – Therefore the EXE file is **not** a real “standalone” EXE file. It needs PowerShell to be installed!!! And – of course – it needs .NET Framework v2.0. Furthermore “script execution” have to be allowed (see cmdlet: set-executionpolicy). – The resulting EXE is “MSIL” and is able to execute as x64 or x86.

The tool “PS2EXE” itself is a PowerShell script! – It does the in-memory compilation and generates the EXE file. It uses the CSharpCodeProvider class of namespace Microsoft.CSharp.

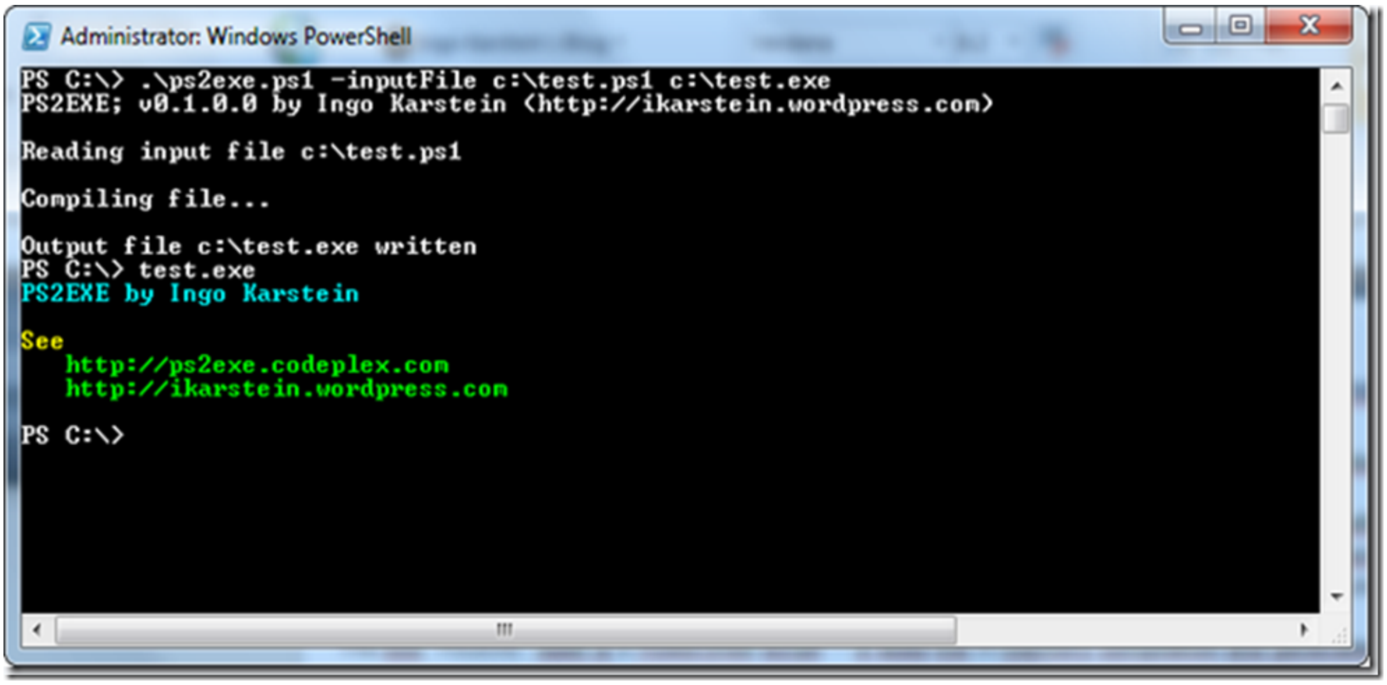
The script is really simple. I contains a multiline string that represents the PowerShell host I’ve written. This is much more interesting than the PS2EXE.ps1 script itself. – Have a look into it!

Usage:

Call  the script with this parameters:

| | |
|---|---|
| -inputFile | PowerShell script file |
| -outputFile | file name (with path) for the destination EXE file |
| -debug | (switch) generate debug info in the destination EXE file. The dynamically generated .CS file will stored beside the output EXE file. Furthermore a .PDB file will be generated for the EXE file |
| -verbose | (switch) shows also verbose informations – if any. |
| -x86 | (switch) compile EXE to run as 32 bit application |
| -x64 | (switch) compile EXE to run as 64 bit application |
| -runtime20 | (switch) force running the EXE in PowerShell 2.0 using .NET 2.0 |
| -runtime30 | (switch) force running the EXE in PowerShell 3.0 using .NET 4.0 |
| -lcid | specify language ID for threads |
| -sta | run PowerShell environment in Single Thread Apartment mode |
| -mta | run PowerShell environment in Multithread Apartment mode |
| -noconsole | compile PS script as Windows application |

Sample:



```
PS C:\> .\ps2exe.ps1 -inputFile c:\test.ps1 c:\test.exe
PS2EXE; v0.1.0.0 by Ingo Karstein (http://ikarstein.wordpress.com)

Reading input file c:\test.ps1

Compiling file...

Output file c:\test.exe written
PS C:\> test.exe
PS2EXE by Ingo Karstein

See
   http://ps2exe.codeplex.com
   http://ikarstein.wordpress.com

PS C:\>
```

This creates "test.exe" out of the PowerShell source file "test.ps1"

Limitations: It's not easy to create a fully functional PowerShell host such as "Console host" (powershell.exe) or "ISE" (powershell_ise.exe). So there may be functionality that does not work properly.

The generated EXE can also be calls using command line options. There are 4 options that are used by the PowerShell host:

| | |
|---|---|
| -debug | Forces the EXE to be debugged. It calls "System.Diagnostics.Debugger.Break()". |
| -extract:"Filename" | Extracts the PowerShell script inside the EXE and saves it as "Filename". The script will not be executed. |
| -wait | At the end of the script execution it writes "Press a key..." and waits for a key to be pressed. |
| -end | All following options will be passed to the script inside the EXE. All preceding options are used by the EXE and will not be passed to the script. |

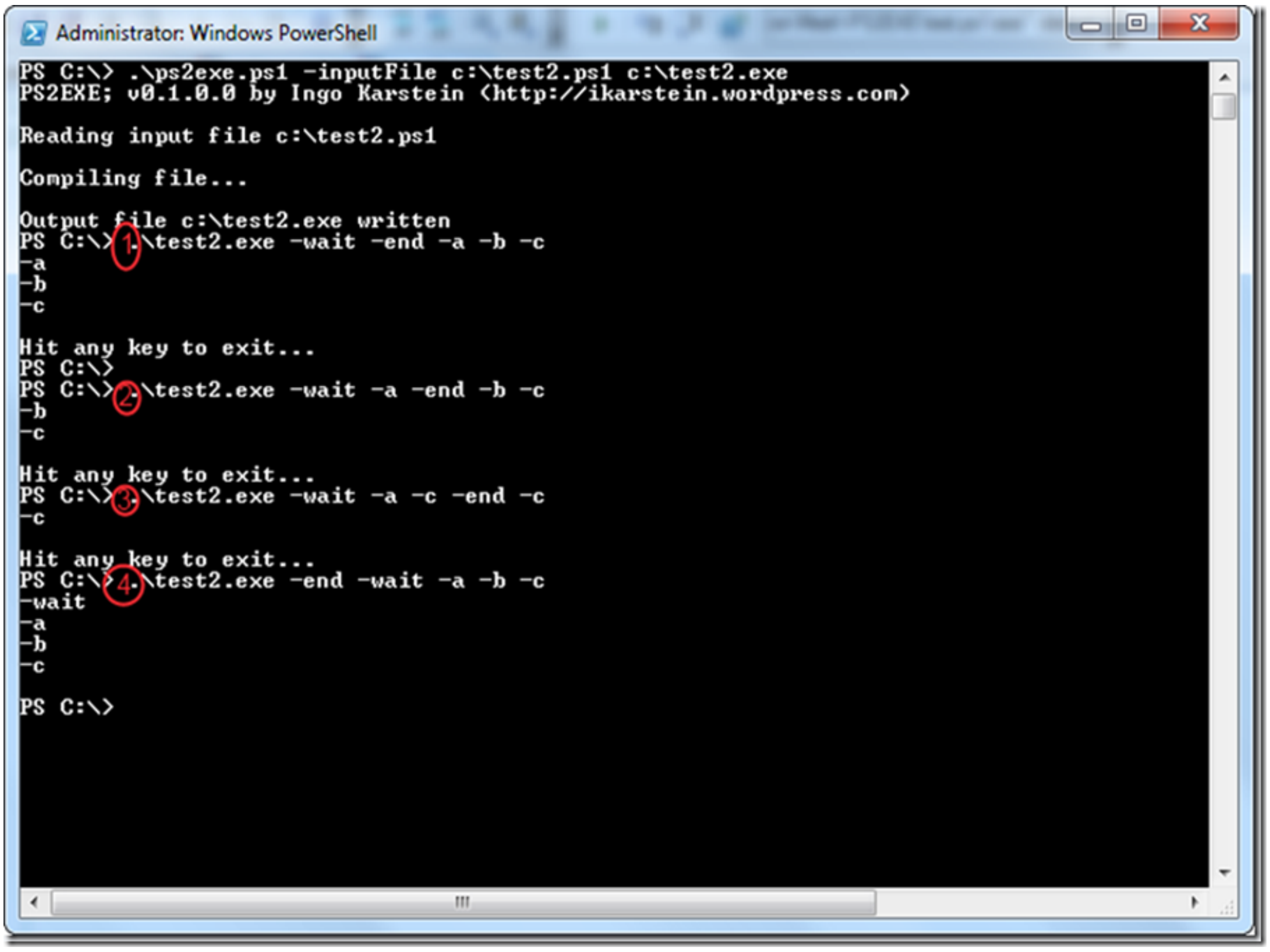
Sample:

I create a script file containing this line of code:

PowerShell

```
$args | Write-Host
```

I save it as "c:\test2.ps1" and convert it as EXE by using PS2EXE:



```
PS C:\> .\ps2exe.ps1 -inputFile c:\test2.ps1 c:\test2.exe
PS2EXE; v0.1.0.0 by Ingo Karstein (http://ikarstein.wordpress.com)

Reading input file c:\test2.ps1

Compiling file...

Output file c:\test2.exe written
PS C:\> 1.\test2.exe -wait -end -a -b -c
-a
-b
-c

Hit any key to exit...
PS C:\>
PS C:\> 2.\test2.exe -wait -a -end -b -c
-b
-c

Hit any key to exit...
PS C:\> 3.\test2.exe -wait -a -c -end -c
-c

Hit any key to exit...
PS C:\> 4.\test2.exe -end -wait -a -b -c
-wait
-a
-b
-c

PS C:\>
```

Sample 1.: "-wait" forces the "Hit any key..." message. All options following "-end" will be passed to the script.

Sample 2., 3. : The script will not get options preceding to "-end".

Sample 4: "-wait" follows to "-end" and so it's passed to the script. No message "Hit any key...".

So. That's it for the moment. Please feel free to modify the script and let me know.

Possible tasks:

- Sign the script inside the EXE with code signature
- Sign the EXE with code signature
- Compress the script inside the EXE
- Improve the PSHost implementation inside the EXE.

Have fun!

**New in v0.5.0.0 (14/11/2013):**

- support for PowerShell 4.0
- VS project updated to VS2013 and .NET 4.0

**New in v0.4.0.0 (03/09/2013):**

- parameter *-sta* for running PowerShell in Single Thread Apartment mode
- parameter *-mta* for running PowerShell in Multi Thread Apartment mode
- parameter *-noconsole* to compile the resulting exe file als "Windows Application". the only implemented GUI interaction is for "get-credential"!!
- **Blog article coming soon...**

**New in v0.3.0.0 (03/08/2013):**

- Support for PowerShell 3.0 and 2.0
- Bug fixes (especially "Compilation fails - Assembly not referenced")
- **see Blog Article: http://blog.karstein-consulting.com/2013/03/08/update-of-ps2exe-version-0-3-0-0-now-supports-powershell-3-0-and-2-0/**

**Here are the previous articles on my blog:**

- [v0.2.0.0] https://blog.karstein-consulting.com/2012/04/30/ps2exe-v0-2-0-0-improvements-platform-switch-x64-or-x86-or-anycpu-new-exe-config-file-for-supported-runtime/
- [original] http://blog.karstein-consulting.com/2011/06/21/ps2exe-tool-for-converting-powershell-scripts-to-standalone-exe-files/