Rugland Packer for (.Net) eXecutables 1.3

Compress .NET executables and pack multiple assemblies into a single executable bundle. It requires no additional external assembly or library to decompress so you get really good compression with only about a 7KB overhead. It's developed in C#

Compress .Net Executables.

- Compress your .Net assembly for faster download.
- High ZIP compression with no unpack library overhead.
- Bundle multiple assemblies into a single distribution.
- Works for Windows Forms and Console Applications.
- Create a toolkit application that is a composite of multiple assemblies accessible via a menu.
- Persist or explicitly define your assembly info from the original assembly or from a source file.
- Persist or add a new icon to your packed executable.

Notes on assembly load process

The order of the locations that .Net will look for assemblies that are referenced by your application is important and may affect your application at runtime.

The order is:

1. GAC
2. File System
3. Compressed Resource

This means that
A. If your application references an assembly that can be found in the GAC then your application will favor this at runtime.

B. Else if your application references an assembly that can be found in the file system (in the same directory as your application) then your application will favor this at runtime.

C. Failing both A and B your application will attempt to load the referenced assembly from the compressed resource inside the application executable file.

Unless your application loads the class in an assembly from an additional AppDomain in which case the assembly must be in the GAC or the file system.

It is not possible to load a class from the executing assembly from an additional AppDomain when the executing assembly has been compressed.

See the AppDomain Tests section of the Demos for examples of how this works.

Usage

RPX <exec file> <additional assemblies> [±C] [/P] [/T tools csv] [/O output] [/D] [/I icon] [/A AssemblyInfo.cs] [/H <password>] [/B] [/Q] [/V ConsoleVerbosity] [±W] /?? /?D <path>

- **±Console**

  Output as a console or forms application

- **/PassArgs**

  Pass command line arguments

- **/Tools**

  A CSV of tools in the format "[Tool Name]:[Tool Path],.."

- **/Output**

Explicit full path of the executable to create

- **/Decorate**

  Decorate the console output of the resulting assembly

- **/Icon**

  Path of the icon to use on the resulting assembly

- **/AssemblyInfo**

  Path of a AssemblyInfo.cs code file to use

- **/Hide**

  Hide and protect compressed assemblies

- **/Build**

  Build mode

- **/Quiet**

  Quiet mode

- **/Verbose**

  Message verbosity level

- **±Warning**

  Warnings are errors

- **/??**

  Show application usage and about screen

- **/?D**

  Write this applications documentation to a text file

Usage In Detail

<exec file>
The full or relative path to a executable file to be compressed.
Unless the /Toolkit argument is defined then it will be the location of the resulting toolkit.

<additional assemblies>
List of additional assemblies to be bundled into the resulting executable.
See additional notes on assembly load order by using the /?? switch

[±C] ±Console
Output as a console or forms application, if not defined then the value is derived from assembly supplied in the <exec file> argument.

[/P] /PassArgs
Pass command line arguments

If this switch if defined then any command line arguments supplied when executing the resulting assembly will be passed on.

[/T tools csv] /Tools
A CSV of tools in the format "[Tool Name]:[Tool Path],.."

Use this argument to create composite toolkit from multiple assemblies.

The supplied value should be a comma separated value (CSV) string in the format "[Tool Name]:[Tool Path],[Tool Name2]:[Tool Path2],.."
Where [Tool Name] is the name of the menu item within the composite toolkit assembly and [Tool Path] is the full path to the tool assembly.

[/O output] /Output
Explicit full path of the executable to create

Use this argument if you what to compile to a specific location. If this argument is not supplied then the default <exec file> argument will be used

[/D] /Decorate
Decorate the console output of the resulting assembly

This comes at a small cost to the assembly code size.

[/I icon] /Icon
Path of the icon to use on the resulting assembly

The assembly icon cannot be compressed so be careful and monitor its size.

[/A AssemblyInfo.cs] /AssemblyInfo
Path of a AssemblyInfo.cs code file to use

Use this argument to use the assembly info from a source file.

If the /Toolkit argument is defined then you must supply this argument.

If neither the /Toolkit or this option is defined then an attempt will be made to retrieve the info from the assembly at the location specified in the <exec file> argument via reflection.

[/H <password>] /Hide
Hide and protect compressed assemblies

This helps protect your assemblies from unwanted unpacking or inspection by a third party.

If the optional password is supplied then the resource will be encrypted and only accessible by supplying the password as the fist argument when running the application.

If this switch is not supplied then applications like WindowsExplore and WinRar will register the compressed assembly as a archive file and may give options to the user for unpacking it.

This comes at a small cost to the final size of the assembly but also has a cost in start up time proportional to the size of the internal compressed resource.

[/B] /Build
Build mode generates errors compatible with MS VisualStudio

[/Q] /Quiet
Quiet mode

Don't display messages about every action

[/V ConsoleVerbosity] /Verbose
Message verbosity level
Possible values are : **Silent**, **Minimal**, **Quiet**, **Normal**, **Verbose**, **Debug**

Use this argument to specify the level of detail RPX outputs whilst it is running.

[±W] ±Warning
When +w warning will be reported as errors.
When -w warnings will not be reported at all.

This switch is to be used in conjunction with the /build switch.

Uses Ionic.Zip for re-compression.
http://www.codeplex.com/DotNetZip

Inspired by UPX (the Ultimate Packer for eXecutables)
Markus F.X.J. Oberhumer, László Molnár & John F. Reiser
http://upx.sourceforge.net/
Last edited Jan 18 at 1:08 AM by RugCode, version 10