Why is Windows Compressed Folders (Zip folders) support stuck at the turn of the century?

devblogs.microsoft.com/oldnewthing/20180515-00

May 15, 2018

Raymond Chen

Every so often, a customer will ask whether Windows Compressed Folders (Zip folders) supports something fancy like AES encryption, and we have to shake our head and apologize. "Sorry, no."

Why this sad state of affairs?

The compression and decompression code for Zip folders was licensed from a third party. This happened during the development of Windows XP. This means that the feature set of Zip folders was locked to whatever features were hip and cool as of around the year 2000.

Since its release in Windows XP, Zip folders has not been actively developed. The reason is the usual: Because adding features requires engineering resources, and engineering resources are limited. Furthermore, since the compression and decompression code weren't written by anybody from Microsoft, there is no expertise in the code base, which means that debugging and making changes is a very difficult undertaking. (Sound familiar?)

AES encryption was added to the ZIP file format in 2003, which is after active development ceased (and after the ink on the license agreement had dried), so the Zip folders code in Windows XP doesn't have support for it.

One thing that did make it out of the minus 100 points deficit (which really is more like *minus 1000 points* because there is nobody around who understands the code) was adding Unicode file name support in Windows 7.

Bonus chatter: On of the terms of the license is that the compression and decompression code for Zip folders should be tied to UI actions and not be programmatically drivable. The main product for the company that provided the compression and decompression code is the compression and decompression code itself. If Windows allowed programs to compress and decompression files by driving the shell namespace directly, then that company would have given away their entire business!

This is why Zip folders may work really well when manipulated in the user interface, but they aren't very helpful when you try to use them programmatically. They don't tell you when a Copy operation is done. They display password prompts for password-protected ZIP files, even if you said not to display UI. Various annoyances to make it impractical to use the Zip folders compression and decompression engine programmatically.

I recall one time a customer was setting up a system that received ZIP files from clients and uncompressed them on the server. They were planning on programmatically driving the Zip folders shell extension to accomplish this. Since it involved Zip folders, the question was sent to me for my thoughts. The first thing I thought was, "Well, I can DoS your server by sending it a password-protected ZIP file."

If your mission is manipulate ZIP files programmatically, you should use something designed and supported for programmatic manipulation of ZIP files, something like, say, the `ZipFile` class.

Thanks to PowerShell, you can do this from script:

```
[Reflection.Assembly]::LoadWithPartialName(
    "System.IO.Compression.FileSystem") | Out-Null
[System.IO.Compression.ZipFile]::CreateFromDirectory(
    $directory, $zipfile,
    [System.IO.Compression.CompressionLevel]::Optimal, $false)
```

Raymond Chen

Follow