|  |  |
| --- | --- |
| From: | 京奇電腦 |
| To: |  |
| Date: | 2016-06-26 |
| Subject: | 版本編號 |

說明軟體或文件使用的版本編號.

# 目錄

內容

[目錄 1](#_Toc455955798)

[京奇電腦版本編號 1](#_Toc455955799)

[Zib版本編號 1](#_Toc455955800)

[微軟慣用的版本編號 2](#_Toc455955801)

[Assembly Version 4](#_Toc455955802)

[Assembly File Version 4](#_Toc455955803)

[AssemblyInformationalVersion 5](#_Toc455955804)

# 京奇電腦版本編號

## Zib版本編號

2016-07起適用:

英文加數字, 不受8個字長度限制, 區分大小寫:

格式: ZLibVVMMNN

ZLib: 固定為ZLib.

VV: Target .NET version, 20=.net 2.0, 45=.net 4.5.

Year: 原始建立時的西元年4碼. Original Creation Year.

MM: Major Version. 00-99.

NN: Minor Version. 00-99.

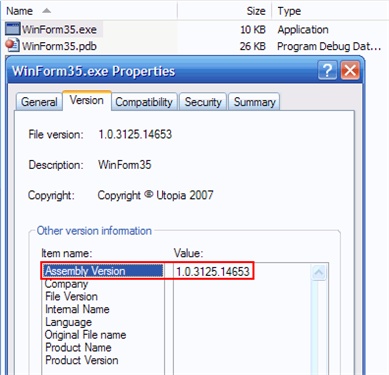
|  |  |  |  |
| --- | --- | --- | --- |
| Filename | Date | Target | Description |
| ZLib2020160000 | 2016-07-10 | .Net 2.0 | 2016年度第一版本0.0 for .Net 2.0.  剛建立的版本為0.x.x.x, 對外Release時須調整為1.x.x.x |

# 微軟慣用的版本編號

微軟慣用的版本編號分為四組數字，例如: 1.0.2553.14653，這四個數字依序是: **Major.Minor.Build.Revision**

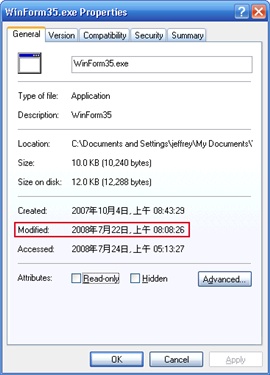
* **Major Version**: 主版號，大多在大規模的功能、架構變革時才會更動
* **Minor Version**: 次版號，用於小規模的功能、架構修正。一般而言，這兩個版號變更意味某些方法參數、型別的變動，有可能導致元件的不相容。
* **Build**: 組建，一般用來區別程式是在哪一天組建(Build)的。在軟體工程中，有所謂的 daily build法，透過每天重新編譯並重新進行測試，確保每天在進行的程式碼修改沒把整個軟體搞爛。而軟體要正式發行時，也會從諸多的Build中挑出一個問題最少，品質最好的先選作Release Candidate。
* **Revision**: 修訂，一般保留給為了修復特定錯誤的後續組建，有時也稱作Emergency Bug Fix。(常用於Quick Fix Engineering, QFE, Hotfix的版次標示)

想知道.NET程式/元件的版號，我們可以用Windows檔案總管來檢視。EXE、DLL檔案的內容視窗，第二個Tab裡就可以看到版本資訊，還包含發行公司等資訊，且不限.NET編譯的。(這招在調查來路不明檔案時，格外有用。)



了解了版號的意義，另一個有趣的問題是: 當我們設成"1.0.\*"時，Visual Studio會如何自動跳號決定Build及Revision?

**答案是由Build的日期與時間決定，Build = 以2000/01/01起算的天數，Revision = 以凌晨00:00:00起算的秒數除以2。**



用以上的WinForm35.exe來驗算一下，程式的編譯日期時間是2008/07/22 08:08:26，而版號為1.0.3125.14653。所以我們叫出Mini C# Lab，寫個兩行Code:

DateTime d = new DateTime(2000, 1, 1);  
Console.WriteLine(d.AddDays(**3125**).AddSeconds(**14653** \* 2).ToString("yyyy/MM/dd HH:mm:ss"));

計算結果正是: 2008/07/22 08:08:26!

AssemblyFileVersion可以避免變更AssemblyVersion衍生的重新編譯需求，在實務上也很常被拿來做成版本識別的依據。

AssemblyFileVersion因為被參考的 DLL 其實會被綁定某個 AssemblyVersion, 所以若因為小修正而改變了 AssemblyVersion, 會造成其他參考他的 DLL 必須跟著重新編譯, 所以我們大都是用 AssemblyFileVersion 來做版本判別.

只有當 interface 改變, 才異動 AssemblyVersion, 否則若只是修正 bug 或是加強內部演算法, interface 不變的狀況下, 都只改變 AssemblyFileVersion.

## Assembly Version

Where other assemblies that reference your assembly will look. If this number changes, other assemblies have to update their references to your assembly!

其他的assemblies參考引用時依據的版本編號. 這個版本編號變更以後, 參考引用的assemblies就必須跟著變更參考到正確的版本編號.

Format: marjor.minor

Code: [assembly: AssemblyVersion("1.0")]

## Assembly File Version

Used for deployment. You can increase this number for every deployment. It is used by setup programs. Use it to mark assemblies that have the same AssemblyVersion, but are generated from different builds.

使用在部屬軟體用途. 每次部屬軟體時可增加編號. 適用於安裝作業的程式. 用來標示(不同編譯編號產生的相同AssemblyVersion).

In Windows, it can be viewed in the file properties.

Windows作業系統中, 這個編號可以從檔案屬性中檢視.

If possible, let it be generated by MSBuild. The AssemblyFileVersion is optional. If not given, the AssemblyVersion is used.

建議由MSBuild自動產生.

這個編號可省略, 如果沒有這個編號, 則會以AssemblyVersion替代.

I use the format: major.minor.revision.build, where I use revision for development stage (Alpha, Beta, RC and RTM), service packs and hot fixes. This would result in:

Format: major.minor.revision.build

Code: [assembly: AssemblyFileVersion("1.0.3100.1242")]

## AssemblyInformationalVersion

The Product version of the assembly. This is the version you would use when talking to customers or for display on your website. This version can be a string, like '1.0 Release Candidate'. Unfortunately, when you use a string, it will generate a false warning -- already reported to Microsoft (fixed in VS2010). The AssemblyInformationalVersion is optional. If not given, the AssemblyVersion is used.

產品版本編號. 與客戶溝通使用的版本編號. 給客戶檢視使用.

這個編號可省略, 如果沒有這個編號, 則會以AssemblyVersion替代.

VS2010才可使用這個編號.

I use the format: major.minor [revision as string]. This would result in:

Format: major.minor [revision as string]

Code: [assembly: AssemblyInformationalVersion("1.0 RC1")]