

# 1 System.Version Class

```
2 [ILAsm]
3 .class public sealed serializable Version extends System.Object implements
4 System.ICloneable, System.IComparable, System.IComparable`1<class
5 System.Version>, System.IEquatable`1<class System.Version>
6
7 [C#]
8 public sealed class Version: ICloneable, IComparable,
9 IComparable<Version>, IEquatable<Version>
```

## 9 Assembly Info:

- 10 • *Name:* mscorlib
- 11 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 12 • *Version:* 2.0.x.x
- 13 • *Attributes:*
- 14     o CLSCompliantAttribute(true)

## 15 Implements:

- 16 • **System.ICloneable**
- 17 • **System.IComparable**
- 18 • **System.IComparable<System.Version>**
- 19 • **System.IEquatable<System.Version>**

## 20 Summary

21 Represents the version number of an assembly.

## 22 Inherits From: System.Object

23  
24 **Library:** BCL

25  
26 **Thread Safety:** All public static members of this type are safe for multithreaded operations.  
27 No instance members are guaranteed to be thread safe.

## 28 29 Description

30 *System.Version* numbers for an assembly consist of two to four components: *major*,  
31 *minor*, *build*, and *revision*. Components *major* and *minor* must be defined. *Build* and  
32 *revision* components are optional. Component *revision* can be used if and only if build is  
33 defined. All defined components must be a *System.Int32* greater than or equal to zero.

34  
35 *[Note:* By convention, the components are used as follows:

- 36 • Major: Assemblies with the same name but different major versions are not  
37 interchangeable. This would be appropriate, for example, for a major rewrite of a  
38 product where backwards compatibility cannot be assumed.

- 1 • Minor: If the name and major number on two assemblies are the same, but the  
2 minor number is different, this indicates significant enhancement with the intention  
3 of backwards compatibility. This would be appropriate, for example, on a "point  
4 release" of a product or a fully backward compatible new version of a product.
- 5 • Assemblies with the same name, major, and minor version numbers but different  
6 revisions are intended to be fully interchangeable. This would be appropriate, for  
7 example, to fix a security hole in a previously released assembly.
- 8 • A difference in build number is intended to represent a recompilation of the same  
9 source. This would be appropriate, for example, because of processor, platform, or  
10 compiler changes.

11 ]

12

# 1 Version() Constructor

```
2 [ILAsm]  
3 public rtspecialname specialname instance void .ctor()  
4 [C#]  
5 public Version()
```

## 6 Summary

7 Constructs and initializes a new instance of the `System.Version` class.

## 8 Description

9 `System.Version.Major` and `System.Version.Minor` are set to zero.  
10 `System.Version.Build` and `System.Version.Revision` are unspecified.

11

# 1 Version(System.Int32, System.Int32, 2 System.Int32, System.Int32) Constructor

```
3 [ILAsm]  
4 public rtspecialname specialname instance void .ctor(int32 major, int32  
5 minor, int32 build, int32 revision)  
  
6 [C#]  
7 public Version(int major, int minor, int build, int revision)
```

## 8 Summary

9 Constructs and initializes a new instance of the `System.Version` class with the specified  
10 major, minor, build, and revision numbers.

## 11 Parameters

Parameter	Description
<i>major</i>	A <code>System.Int32</code> specifying the major component.
<i>minor</i>	A <code>System.Int32</code> specifying the minor component.
<i>build</i>	A <code>System.Int32</code> specifying the build component.
<i>revision</i>	A <code>System.Int32</code> specifying the revision component.

## 12 13 Exceptions

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>major</i> , <i>minor</i> , <i>build</i> , or <i>revision</i> is less than zero.

## 14 15 Example

16 The following example sets the version to "6.1.2.4" and writes the result to the console.

```
17 [C#]  
18  
19 using System;  
20  
21 public class Vers {  
22     public static void Main() {
```

```
1
2     Version vers = new Version( 6, 1, 2, 4 );
3     Console.WriteLine( "Version is {0}", vers.ToString() );
4 }
5 }
6
7 The output is
8
9 Version is 6.1.2.4
10
```

# 1 Version(System.Int32, System.Int32, 2 System.Int32) Constructor

```
3 [ILAsm]  
4 public rtspecialname specialname instance void .ctor(int32 major, int32  
5 minor, int32 build)  
  
6 [C#]  
7 public Version(int major, int minor, int build)
```

## 8 Summary

9 Constructs and initializes a new instance of the `System.Version` class using the  
10 specified `major`, `minor`, and `build` values.

## 11 Parameters

Parameter	Description
<i>major</i>	A <code>System.Int32</code> specifying the major component.
<i>minor</i>	A <code>System.Int32</code> specifying the minor component.
<i>build</i>	A <code>System.Int32</code> specifying the build component.

## 12 13 Exceptions

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>major</i> , <i>minor</i> , or <i>build</i> is less than zero.

## 14 15 Example

16 The following example sets the version to "6.1.2" and writes the result to the console.

```
17 [C#]  
18  
19 using System;  
20  
21 public class Vers {  
22     public static void Main() {  
23         Version vers = new Version( 6, 1, 2 );  
24         Console.WriteLine( "Version is {0}", vers.ToString() );  
25     }  
26 }  
27
```

```
1 }  
2  
3 The output is  
4  
5 Version is 6.1.2  
6
```

# 1 Version(System.Int32, System.Int32)

## 2 Constructor

```
3 [ILAsm]  
4 public rtspecialname specialname instance void .ctor(int32 major, int32  
5 minor)  
  
6 [C#]  
7 public Version(int major, int minor)
```

### 8 Summary

9 Constructs and initializes a new instance of the `System.Version` class using the  
10 specified major and minor values.

### 11 Parameters

Parameter	Description
<i>major</i>	A <code>System.Int32</code> specifying the major component.
<i>minor</i>	A <code>System.Int32</code> specifying the minor component.

### 12 13 Exceptions

Exception	Condition
<code>System.ArgumentOutOfRangeException</code>	<i>major</i> or <i>minor</i> is less than zero.

### 14 15 Example

16 The following example sets the version to "6.1" and writes the result to the console.

```
17 [C#]  
18  
19 using System;  
20  
21 public class Vers {  
22     public static void Main() {  
23  
24         Version vers = new Version( 6, 1 );  
25         Console.WriteLine( "Version is {0}", vers.ToString() );  
26     }  
27 }  
28  
29
```

- 1 The output is
- 2
- 3 Version is 6.1
- 4

# 1 Version(System.String) Constructor

```
2 [ILAsm]  
3 public rtspecialname specialname instance void .ctor(string version)  
4 [C#]  
5 public Version(string version)
```

## 6 Summary

7 Constructs and initializes a new instance of the `System.Version` class using the values  
8 represented by the specified `System.String`.

## 9 Parameters

Parameter	Description
<i>version</i>	<p>A <code>System.String</code> that represents 2 to 4 <code>System.Int32</code> integers separated by period characters ('.'). Each component delineated by a period character will be parsed to a <code>System.Int32</code> with <code>System.Int32.Parse(System.String)</code>. The numbers will be processed in the following order: <i>major</i>, <i>minor</i>, <i>build</i>, <i>revision</i>. If the <i>revision</i> or the <i>revision</i> and the <i>build</i> components are not represented by <i>version</i>, their values will be undefined.</p> <p>[<i>Note</i>: The formatting of <i>version</i> must be as follows, with optional components shown in square brackets ('[' and ']'): <i>major.minor[.build[.revision]]</i>, where each component returns a <code>System.Int32</code> with <code>System.Int32.Parse(System.String)</code>.</p> <p>]</p>

10

## 11 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>version</i> has fewer than 2 components or more than 4 components (i.e. fewer than 1 or more than 3 period characters).
<b>System.ArgumentNullException</b>	<i>version</i> is a null reference.
<b>System.ArgumentOutOfRangeException</b>	<i>major</i> , <i>minor</i> , <i>build</i> , or <i>revision</i> is less than zero.

## **System.FormatException**

At least one component of *version* does not parse to a `System.Int32` with `System.Int32.Parse (System.String)`.

1

### 2 **Example**

3 The following example sets the version to "6.1.2.4" and writes the result to the console.

4

5 [C#]

6

7 using System;

8

9 public class Vers {

10 public static void Main() {

11

12 Version vers = new Version( "6.1.2.4" );

13 Console.WriteLine( "Version is {0}", vers.ToString() );

14 }

15 }

16

17 The output is

18

19 Version is 6.1.2.4

20

# 1 Version.Clone() Method

```
2 [ILAsm]  
3 .method public final hidebysig virtual object Clone()  
4 [C#]  
5 public object Clone()
```

## 6 Summary

7 Returns a new `System.Object` with values equal to the property values of the current  
8 instance.

## 9 Return Value

10 A new `System.Object` whose values are equal to the property values of the current  
11 instance.

## 12 Description

13 The `System.Object` returned by this method must be explicitly cast to a  
14 `System.Version` before it can be used as one.

15  
16 [*Note:* This method is implemented to support the `System.ICloneable` interface.]  
17  
18

## 19 Example

20 The following example clones the version number and writes the result to the console.

```
21 [C#]  
22  
23 using System;  
24 class VersionCloneExample {  
25     public static void Main() {  
26         Version vers = new Version("6.1.2.4");  
27         Console.WriteLine("The string representation of the" +  
28             " version is {0}.",  
29             vers.ToString());  
30         Version clone = (Version) vers.Clone();  
31         Console.WriteLine("The original version was" +  
32             " successfully cloned.");  
33         Console.Write("The string representation of the" +  
34             " cloned version is {0}.",  
35             clone.ToString());  
36     }  
37 }  
38
```

39 The output is

40  
41 The string representation of the version is 6.1.2.4.

1  
2  
3  
4  
5  
6  
7  
8

The original version was successfully cloned.

The string representation of the cloned version is 6.1.2.4.

# Version.CompareTo(System.Object) Method

```
[ILAsm]  
.method public final hidebysig virtual int32 CompareTo(object version)  
  
[C#]  
public int CompareTo(object version)
```

## Summary

Returns the sort order of the current instance compared to the specified `System.Object`.

## Parameters

Parameter	Description
<i>version</i>	The <code>System.Object</code> to compare to the current instance.

## Return Value

The return value is a negative number, zero, or a positive number reflecting the sort order of the current instance as compared to *version*. For non-zero return values, the exact value returned by this method is unspecified. The following table defines the return value:

Return Value	Description
A negative number	Current instance < <i>version</i> .
Zero	Current instance == <i>version</i> .
A positive number	Current instance > <i>version</i> , or <i>version</i> is a null reference.

## Description

[Note: The components of `System.Version` in decreasing order of importance are: *major*, *minor*, *build*, and *revision*. An undefined component is assumed to be older than any defined component.

This method is implemented to support the `System.IComparable` interface.

1  
2 ]

### 3 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>version</i> is not a System.Version and is not a null reference

### 4 5 Example

```
6 [C#]
7 using System;
8 class VersionTest {
9     static string Test ( Version v1, Version v2 ) {
10         int i = v1.CompareTo(v2);
11         if ( i < 0 )
12             return "older than";
13         else if ( i == 0 )
14             return "the same as";
15         else
16             return "newer than";
17     }
18     public static void Main() {
19         Version vers1 = new Version( "6.1.2.4" );
20         Version vers2 = new Version( 6, 1 );
21         Version vers3 = new Version( 6, 1, 3 );
22         Console.Write("Version {0} is {1} ",
23             vers1, Test(vers1, vers2));
24         Console.WriteLine("version {0}", vers2);
25         Console.Write("Version {0} is {1} ",
26             vers1, Test(vers1, vers3));
27         Console.WriteLine("version {0}", vers3);
28         Console.Write("Version {0} is {1} ",
29             vers3, Test(vers3, vers3));
30         Console.WriteLine("version {0}", vers3);
31         Console.Write("Version {0} is {1} ",
32             vers2, Test(vers2, vers1));
33         Console.WriteLine("version {0}", vers1);
34     }
35 }
```

36  
37 The output is

```
38
39 Version 6.1.2.4 is newer than version 6.1
40
41
42 Version 6.1.2.4 is older than version 6.1.3
43
44
45 Version 6.1.3 is the same as version 6.1.3
```

1  
2  
3 Version 6.1 is older than version 6.1.2.4  
4  
5

# 1 Version.CompareTo(System.Version) Method

```
2 [ILAsm]  
3 .method public final hidebysig virtual int32 CompareTo(class  
4 System.Version value)  
  
5 [C#]  
6 public int CompareTo(Version value)
```

## 7 Summary

8 Returns the sort order of the current instance compared to the specified  
9 System.Version.

## 10 Parameters

Parameter	Description
<i>value</i>	The System.Version to compare to the current instance.

## 11 Return Value

13 The return value is a negative number, zero, or a positive number reflecting the sort  
14 order of the current instance as compared to *version*. For non-zero return values, the  
15 exact value returned by this method is unspecified. The following table defines the  
16 return value:

Return Value	Description
A negative number	Current instance < <i>value</i> .
Zero	Current instance == <i>value</i> .
A positive number	Current instance > <i>value</i> , or <i>value</i> is a null reference.



## 18 Description

20 [Note: The components of System.Version in decreasing order of importance are:  
21 *major*, *minor*, *build*, and *revision*. An undefined component is assumed to be older than  
22 any defined component.

1  
2  
3  
4  
5  
6  
7  
8

]

[*Note:* This method is implemented to support the  
System.IComparable<System.Version> interface.]

# 1 Version.Equals(System.Object) Method

```
2 [ILAsm]  
3 .method public hidebysig virtual bool Equals(object obj)  
4 [C#]  
5 public override bool Equals(object obj)
```

## 6 Summary

7 Determines whether the current instance and the specified `System.Object` represent the  
8 same type and value.

## 9 Parameters

Parameter	Description
<i>obj</i>	The <code>System.Object</code> to compare to the current instance.

10

## 11 Return Value

12 A `System.Boolean` where true indicates *obj* is the same type as the current instance  
13 and has equal `System.Version.Major`, `System.Version.Minor`,  
14 `System.Version.Build`, and `System.Version.Revision` properties as the current  
15 instance. If *obj* is a null reference or is not an instance of `System.Version`, returns  
16 false.

## 17 Description

18 [Note: This method overrides `System.Object.Equals`.]  
19  
20

## 21 Example

```
22 [C#]  
23 using System;  
24 class VersionEqualsExample {  
25     static void testEquals(Version v1, Version v2) {  
26         Console.WriteLine("It is {0} that version ",  
27             v1.Equals(v2));  
28         Console.WriteLine("{0} is equal to {1}.",  
29             v1, v2);  
30     }  
31     public static void Main() {  
32         Version vers1 = new Version( "6.1.2.4" );  
33         Version vers2 = new Version( 6, 1 );  
34         testEquals( vers1, vers1 );  
35         testEquals( vers1, vers2 );  
}
```

```
1     }
2   }
3
4   The output is
5
6   It is True that version 6.1.2.4 is equal to 6.1.2.4.
7
8
9   It is False that version 6.1.2.4 is equal to 6.1.
10
11
```

# 1 Version.Equals(System.Version) Method

```
2 [ILAsm]  
3 .method public hidebysig virtual bool Equals(class System.Version obj)  
4 [C#]  
5 public override bool Equals(Version obj)
```

## 6 Summary

7 Determines whether the current instance and the specified `System.Version` represent  
8 the same value.

## 9 Parameters

Parameter	Description
<i>obj</i>	The <code>System.Version</code> to compare to the current instance.

10

## 11 Return Value

12 A `System.Boolean` where true indicates *obj* has equal `System.Version.Major`,  
13 `System.Version.Minor`, `System.Version.Build`, and `System.Version.Revision`  
14 properties as the current instance. If *obj* is a null reference, returns false.

## 15 Description

16 [Note: This method is implemented to support the  
17 `System.IEquatable<System.Version>` interface.]  
18  
19

20

# 1 Version.GetHashCode() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual int32 GetHashCode()  
4 [C#]  
5 public override int GetHashCode()
```

## 6 Summary

7 Generates a hash code for the current instance.

## 8 Return Value

9 A `System.Int32` containing the hash code for the current instance.

## 10 Description

11 The algorithm used to generate the hash code is unspecified.

12

13 [*Note:* This method overrides `System.Object.GetHashCode()`.]

14

15

16

# 1 Version.op\_Equality(System.Version, 2 System.Version) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname bool op_Equality(class  
5 System.Version v1, class System.Version v2)  
  
6 [C#]  
7 public static bool operator ==(Version v1, Version v2)
```

## 8 Summary

9 Determines whether two instances of `System.Version` are equal.

## 10 Parameters

Parameter	Description
<code>v1</code>	An instance of the <code>System.Version</code> class.
<code>v2</code>	An instance of the <code>System.Version</code> class.

11

## 12 Return Value

13 A `System.Boolean` where `true` indicates `v1` and `v2` have equal `System.Version.Major`,  
14 `System.Version.Minor`, `System.Version.Build`, and `System.Version.Revision`  
15 properties, or both `v1` and `v2` are `null`; otherwise `false`.

## 16 Description

17 The parts of the version number are compared independently starting with the  
18 `System.Version.Major` property and then the `System.Version.Minor`,  
19 `System.Version.Build`, and `System.Version.Revision` properties, in order. This  
20 method returns as soon as one of the properties is determined not to be equal.

21

# 1 Version.op\_GreaterThan(System.Version, 2 System.Version) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname bool op_GreaterThan(class  
5 System.Version v1, class System.Version v2)  
  
6 [C#]  
7 public static bool operator >(Version v1, Version v2)
```

## 8 Summary

9 Determines whether the first instance of System.Version is greater than the second  
10 instance of System.Version.

## 11 Parameters

Parameter	Description
v1	An instance of the System.Version class.
v2	An instance of the System.Version class.

## 12 13 Return Value

14 A System.Boolean where true indicates v1 is greater than v2; otherwise false. If v1 is  
15 null, false is returned.

## 16 Description

17 The parts of the version number are compared independently starting with the  
18 System.Version.Major property and then the System.Version.Minor,  
19 System.Version.Build, and System.Version.Revision properties, in order. This  
20 method returns as soon as one of the properties is determined not to be equal.

## 21 Exceptions

Exception	Condition
System.ArgumentNullException	v2 is a null reference.

22  
23

# 1 2 Version.op\_GreaterThanOrEqualTo(System.Version, System.Version) Method 3

```
4 [ILAsm]  
5 .method public hidebysig static specialname bool  
6 op_GreaterThanOrEqualTo(class System.Version v1, class System.Version v2)  
7  
8 [C#]  
9 public static bool operator >=(Version v1, Version v2)
```

## 9 Summary

10 Determines whether the first instance of `System.Version` is greater than or equal to the  
11 second instance of `System.Version`.

## 12 Parameters

Parameter	Description
<i>v1</i>	An instance of the <code>System.Version</code> class.
<i>v2</i>	An instance of the <code>System.Version</code> class.

## 13 14 Return Value

15 A `System.Boolean` where `true` indicates *v1* is greater than or equal to *v2*; otherwise  
16 `false`. If *v1* is `null`, `false` is returned.

## 17 Description

18 The parts of the version number are compared independently starting with the  
19 `System.Version.Major` property and then the `System.Version.Minor`,  
20 `System.Version.Build`, and `System.Version.Revision` properties, in order. This  
21 method returns as soon as one of the properties is determined not to be equal.

## 22 Exceptions

Exception	Condition
<code>System.ArgumentNullException</code>	<i>v2</i> is a <code>null</code> reference.

23  
24

# 1 Version.op\_Inequality(System.Version, 2 System.Version) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname bool op_Inequality(class  
5 System.Version v1, class System.Version v2)  
  
6 [C#]  
7 public static bool operator !=(Version v1, Version v2)
```

## 8 Summary

9 Determines whether two instances of System.Version are not equal.

## 10 Parameters

Parameter	Description
v1	An instance of the System.Version class.
v2	An instance of the System.Version class.

11

## 12 Return Value

13 A System.Boolean where true indicates v1 and v2 have at least one unequal property;  
14 otherwise false. If v1 and v2 are both null, returns false; if one is null but not the  
15 other, returns true.

## 16 Description

17 The parts of the version number are compared independently starting with the  
18 System.Version.Major property and then the System.Version.Minor,  
19 System.Version.Build, and System.Version.Revision properties, in order. This  
20 method returns as soon as one of the properties is determined not to be equal.

21

# 1 Version.op\_LessThan(System.Version, 2 System.Version) Method

```
3 [ILAsm]  
4 .method public hidebysig static specialname bool op_LessThan(class  
5 System.Version v1, class System.Version v2)  
  
6 [C#]  
7 public static bool operator <(Version v1, Version v2)
```

## 8 Summary

9 Determines whether the first instance of System.Version is less than the second  
10 instance of System.Version.

## 11 Parameters

Parameter	Description
v1	An instance of the System.Version class.
v2	An instance of the System.Version class.

12

## 13 Return Value

14 A System.Boolean where true indicates v1 is less than v2; otherwise false. If v2 is  
15 null, false is returned.

## 16 Description

17 The parts of the version number are compared independently starting with the  
18 System.Version.Major property and then the System.Version.Minor,  
19 System.Version.Build, and System.Version.Revision properties, in order. This  
20 method returns as soon as one of the properties is determined not to be equal.

## 21 Exceptions

Exception	Condition
System.ArgumentNullException	v1 is a null reference.

22

23

1  
2 **Version.op\_LessThanOrEqualTo(System.Version**  
3 **, System.Version) Method**

```
4 [ILAsm]  
5 .method public hidebysig static specialname bool op_LessThanOrEqualTo(class  
6 System.Version v1, class System.Version v2)  
  
7 [C#]  
8 public static bool operator <=(Version v1, Version v2)
```

9 **Summary**

10 Determines whether the first instance of `System.Version` is less than or equal to the  
11 second instance of `System.Version`.

12 **Parameters**

Parameter	Description
<i>v1</i>	An instance of the <code>System.Version</code> class.
<i>v2</i>	An instance of the <code>System.Version</code> class.

13  
14 **Return Value**

15 A `System.Boolean` where `true` indicates *v1* is less than or equal to *v2*; otherwise `false`.  
16 If *v2* is `null`, `false` is returned.

17 **Description**

18 The parts of the version number are compared independently starting with the  
19 `System.Version.Major` property and then the `System.Version.Minor`,  
20 `System.Version.Build`, and `System.Version.Revision` properties, in order. This  
21 method returns as soon as one of the properties is determined not to be equal.

22 **Exceptions**

Exception	Condition
<b>System.ArgumentNullException</b>	<i>v1</i> is a null reference.

23  
24

# 1 Version.Build Property

```
2 [ILAsm]
3 .property int32 Build { public hidebysig specialname instance int32
4 get_Build() }
5 [C#]
6 public int Build { get; }
```

## 7 Summary

8 Gets the value of the build component of the current instance.

## 9 Property Value

10 A System.Int32 specifying the build component, or -1 if the build component is  
11 undefined.

## 12 Description

13 This property is read-only.

14  
15 *[Note: If the version number is 6.1.2.4, the build component is 2. If the version number*  
16 *is 6.1, the build component is -1, which is considered to be undefined.]*  
17  
18

## 19 Example

```
20 [C#]
21 using System;
22 class VersionBuildExample {
23     public static void Main() {
24         Version vers = new Version("6.1.2.4");
25         Console.Write("The build component of ");
26         Console.WriteLine("version vers = {0}.", vers.Build);
27     }
28 }
29
```

30 The output is

31  
32 The build component of version vers = 2.

33

# 1 Version.Major Property

```
2 [ILAsm]
3 .property int32 Major { public hidebysig specialname instance int32
4 get_Major() }
5 [C#]
6 public int Major { get; }
```

## 7 Summary

8 Gets the value of the major component of the current instance.

## 9 Property Value

10 A System.Int32 specifying the major component.

## 11 Description

12 This property is read-only.

## 13 example

14 If the version number is 6.1, the major version is 6.

15

## 16 Example

```
17 [C#]
18 using System;
19 class VersionMajorExample {
20     public static void Main() {
21         Version vers = new Version("6.1.2.4");
22         Console.Write("The major component ");
23         Console.WriteLine("of version vers = {0}.",
24             vers.Major);
25     }
26 }
27
```

28 The output is

29

30 The major component of version vers = 6.

31

32

# 1 Version.Minor Property

```
2 [ILAsm]
3 .property int32 Minor { public hidebysig specialname instance int32
4 get_Minor() }
5
6 [C#]
7 public int Minor { get; }
```

## 7 Summary

8 Gets the value of the minor component of the current instance.

## 9 Property Value

10 A System.Int32 specifying the minor component.

## 11 Description

12 This property is read-only.

## 13 example

14 If the version number is 6.1, the minor component is 1.

15

## 16 Example

```
17 [C#]
18 using System;
19 class VersionMinorExample {
20     public static void Main() {
21         Version vers = new Version("6.1.2.4");
22         Console.Write("The minor component ");
23         Console.WriteLine("of version vers = {0}.",
24             vers.Minor);
25     }
26 }
27
```

28 The output is

29

30 The minor component of version vers = 1.

31

# 1 Version.Revision Property

```
2 [ILAsm]
3 .property int32 Revision { public hidebysig specialname instance int32
4 get_Revision() }
5 [C#]
6 public int Revision { get; }
```

## 7 Summary

8 Gets the value of the revision component of the current instance.

## 9 Property Value

10 A `System.Int32` specifying the revision component, or -1 if the revision component is  
11 undefined.

## 12 Description

13 This property is read-only.

## 14 example

15 If the version number is 6.1.2.4, the revision component is 4. If the version number is  
16 6.1, the revision component is considered to be undefined.

17

## 18 Example

```
19 [C#]
20 using System;
21 class VersionRevisionExample {
22     public static void Main() {
23         Version vers = new Version("6.1.2.4");
24         Console.Write("The revision component of ");
25         Console.WriteLine("version vers = {0}.",
26             vers.Revision);
27     }
28 }
29
```

30 The output is

31  
32 The revision component of version vers = 4.

33

34