System.Version Class

```
[ILAsm]
.class public sealed serializable Version extends System.Object implements
System.ICloneable, System.IComparable, System.IComparable`1<class
System.Version>, System.IEquatable`1<class System.Version>

[C#]
public sealed class Version: ICloneable, IComparable,
IComparable<Version>, IEquatable<Version>
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
    - CLSCompliantAttribute(true)

Implements:

- **System.ICloneable**
- **System.IComparable**
- **System.IComparable<System.Version>**
- **System.IEquatable<System.Version>**

Summary

Represents the version number of an assembly.

Inherits From: System.Object

Library: BCL

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

Description

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

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

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

- Minor: If the name and major number on two assemblies are the same, but the minor number is different, this indicates significant enhancement with the intention of backwards compatibility. This would be appropriate, for example, on a "point release" of a product or a fully backward compatible new version of a product.

- Assemblies with the same name, major, and minor version numbers but different revisions are intended to be fully interchangeable. This would be appropriate, for example, to fix a security hole in a previously released assembly.

- A difference in build number is intended to represent a recompilation of the same source. This would be appropriate, for example, because of processor, platform, or compiler changes.

]

Version() Constructor

[ILAsm]
```
public rtspecialname specialname instance void .ctor()
```

[C#]
```
public Version()
```

Summary

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

Description

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

Version(System.Int32, System.Int32, System.Int32, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int32 major, int32
minor, int32 build, int32 revision)

[C#]
public Version(int major, int minor, int build, int revision)
```

Summary

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

Parameters

Parameter	Description
major	A `System.Int32` specifying the major component.
minor	A `System.Int32` specifying the minor component.
build	A `System.Int32` specifying the build component.
revision	A `System.Int32` specifying the revision component.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*major*, *minor*, *build*, or *revision* is less than zero.

Example

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

[C#]

```
using System;

public class Vers {
  public static void Main() {
    Version vers = new Version( 6, 1, 2, 4 );
    Console.WriteLine( "Version is {0}", vers.ToString() );
  }
}
```

The output is

```
Version is 6.1.2.4
```

Version(System.Int32, System.Int32, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int32 major, int32
minor, int32 build)

[C#]
public Version(int major, int minor, int build)
```

Summary

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

Parameters

Parameter	**Description**
major	A `System.Int32` specifying the major component.
minor	A `System.Int32` specifying the minor component.
build	A `System.Int32` specifying the build component.

Exceptions

Exception	**Condition**
System.ArgumentOutOfRangeException	*major*, *minor*, or *build* is less than zero.

Example

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

[C#]

```
using System;

public class Vers {
  public static void Main() {

    Version vers = new Version( 6, 1, 2 );
    Console.WriteLine( "Version is {0}", vers.ToString() );
  }
}
```

The output is

```
Version is 6.1.2
```

Version(System.Int32, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int32 major, int32 minor)

[C#]
public Version(int major, int minor)
```

Summary

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

Parameters

Parameter	Description
major	A `System.Int32` specifying the major component.
minor	A `System.Int32` specifying the minor component.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*major* or *minor* is less than zero.

Example

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

[C#]

```
using System;

public class Vers {
  public static void Main() {

    Version vers = new Version( 6, 1 );
    Console.WriteLine( "Version is {0}", vers.ToString() );
  }
}
```

The output is

```
Version is 6.1
```

Version(System.String) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(string version)

[C#]
public Version(string version)
```

Summary

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

Parameters

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

Exceptions

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

Example

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

[C#]

```
using System;

public class Vers {
  public static void Main() {

    Version vers = new Version( "6.1.2.4" );
    Console.WriteLine( "Version is {0}", vers.ToString() );
  }
}
```

The output is

```
Version is 6.1.2.4
```

Version.Clone() Method

[ILAsm]
.method public final hidebysig virtual object Clone()

[C#]
public object Clone()

Summary

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

Return Value

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

Description

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

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

Example

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

[C#]

```
using System;
class VersionCloneExample {
  public static void Main() {
    Version vers = new Version("6.1.2.4");
    Console.WriteLine("The string representation of the" +
                      " version is {0}.",
                      vers.ToString());
    Version clone = (Version) vers.Clone();
    Console.WriteLine("The original version was" +
                      " successfully cloned.");
    Console.Write("The string representation of the" +
                  " cloned version is {0}.",
                  clone.ToString());
  }
}
```

The output is

```
The string representation of the version is 6.1.2.4.
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
version	The `System.Object` 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 < *version*.
Zero	Current instance == *version*.
A positive number	Current instance > *version*, or *version* 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.

]

Exceptions

Exception	Condition
System.ArgumentException	*version* is not a `System.Version` and is not a null reference

Example

[C#]

```
using System;
class VersionTest {
   static string Test ( Version v1, Version v2 ) {
      int i = v1.CompareTo(v2);
      if ( i < 0 )
         return "older than";
      else if ( i == 0 )
         return "the same as";
      else
         return "newer than";
   }
   public static void Main() {
      Version vers1 = new Version( "6.1.2.4" );
      Version vers2 = new Version( 6, 1 );
      Version vers3 = new Version( 6, 1, 3 );
      Console.Write("Version {0} is {1} ",
                    vers1, Test(vers1, vers2));
      Console.WriteLine("version {0}", vers2);
      Console.Write("Version {0} is {1} ",
                    vers1, Test(vers1, vers3));
      Console.WriteLine("version {0}", vers3);
      Console.Write("Version {0} is {1} ",
                    vers3, Test(vers3, vers3));
      Console.WriteLine("version {0}", vers3);
      Console.Write("Version {0} is {1} ",
                    vers2, Test(vers2, vers1));
      Console.WriteLine("version {0}", vers1);
   }
}
```

The output is

```
Version 6.1.2.4 is newer than version 6.1


Version 6.1.2.4 is older than version 6.1.3


Version 6.1.3 is the same as version 6.1.3
Version 6.1 is older than version 6.1.2.4
```

Version.CompareTo(System.Version) Method

```
[ILAsm]
.method public final hidebysig virtual int32 CompareTo(class
System.Version value)

[C#]
public int CompareTo(Version value)
```

Summary

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

Parameters

Parameter	Description
value	The `System.Version` 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 < *value*.
Zero	Current instance == *value*.
A positive number	Current instance > *value*, or *value* 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.

]

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

Version.Equals(System.Object) Method

[ILAsm]
```
.method public hidebysig virtual bool Equals(object obj)
```

[C#]
```
public override bool Equals(object obj)
```

Summary

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

Parameters

Parameter	Description
obj	The `System.Object` to compare to the current instance.

Return Value

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

Description

[*Note:* This method overrides `System.Object.Equals`.]

Example

[C#]

```
using System;
class VersionEqualsExample {
   static void testEquals(Version v1, Version v2) {
      Console.Write("It is {0} that version ",
                    v1.Equals(v2));
      Console.WriteLine("{0} is equal to {1}.",
                    v1, v2);
   }
   public static void Main() {
      Version vers1 = new Version( "6.1.2.4" );
      Version vers2 = new Version( 6, 1 );
      testEquals( vers1, vers1 );
      testEquals( vers1, vers2 );
```

```
    }
}
```

The output is

```
It is True that version 6.1.2.4 is equal to 6.1.2.4.


It is False that version 6.1.2.4 is equal to 6.1.
```

Version.Equals(System.Version) Method

```
[ILAsm]
.method public hidebysig virtual bool Equals(class System.Version obj)

[C#]
public override bool Equals(Version obj)
```

Summary

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

Parameters

Parameter	Description
obj	The `System.Version` to compare to the current instance.

Return Value

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

Description

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

Version.GetHashCode() Method

[ILAsm]
```
.method public hidebysig virtual int32 GetHashCode()
```

[C#]
```
public override int GetHashCode()
```

Summary

Generates a hash code for the current instance.

Return Value

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

Description

The algorithm used to generate the hash code is unspecified.

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

Version.op_Equality(System.Version, System.Version) Method

[ILAsm]
```
.method public hidebysig static specialname bool op_Equality(class
System.Version v1, class System.Version v2)
```

[C#]
```
public static bool operator ==(Version v1, Version v2)
```

Summary

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

Parameters

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

Return Value

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

Description

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

Version.op_GreaterThan(System.Version, System.Version) Method

```
[ILAsm]
.method public hidebysig static specialname bool op_GreaterThan(class
System.Version v1, class System.Version v2)

[C#]
public static bool operator >(Version v1, Version v2)
```

Summary

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

Parameters

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

Return Value

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

Description

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

Exceptions

Exception	Condition
System.ArgumentNullException	*v2* is a `null` reference.

Version.op_GreaterThanOrEqual(System.Version, System.Version) Method

```
[ILAsm]
.method public hidebysig static specialname bool
op_GreaterThanOrEqual(class System.Version v1, class System.Version v2)

[C#]
public static bool operator >=(Version v1, Version v2)
```

Summary

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

Parameters

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

Return Value

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

Description

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

Exceptions

Exception	**Condition**
System.ArgumentNullException	*v2* is a `null` reference.

Version.op_Inequality(System.Version, System.Version) Method

```
[ILAsm]
.method public hidebysig static specialname bool op_Inequality(class
System.Version v1, class System.Version v2)

[C#]
public static bool operator !=(Version v1, Version v2)
```

Summary

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

Parameters

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

Return Value

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

Description

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

Version.op_LessThan(System.Version, System.Version) Method

```
[ILAsm]
.method public hidebysig static specialname bool op_LessThan(class
System.Version v1, class System.Version v2)

[C#]
public static bool operator <(Version v1, Version v2)
```

Summary

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

Parameters

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

Return Value

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

Description

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

Exceptions

Exception	Condition
System.ArgumentNullException	*v1* is a `null` reference.

Version.op_LessThanOrEqual(System.Version, System.Version) Method

[ILAsm]
```
.method public hidebysig static specialname bool op_LessThanOrEqual(class
System.Version v1, class System.Version v2)
```

[C#]
```
public static bool operator <=(Version v1, Version v2)
```

Summary

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

Parameters

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

Return Value

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

Description

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

Exceptions

Exception	Condition
System.ArgumentNullException	*v1* is a `null` reference.

Version.Build Property

[ILAsm]
```
.property int32 Build { public hidebysig specialname instance int32
get_Build() }
```

[C#]
```
public int Build { get; }
```

Summary

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

Property Value

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

Description

This property is read-only.

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

Example

[C#]

```
using System;
class VersionBuildExample {
   public static void Main() {
      Version vers = new Version("6.1.2.4");
      Console.Write("The build component of ");
      Console.WriteLine("version vers = {0}.", vers.Build);
   }
}
```

The output is

```
The build component of version vers = 2.
```

Version.Major Property

[ILAsm]
```
.property int32 Major { public hidebysig specialname instance int32
get_Major() }
```

[C#]
```
public int Major { get; }
```

Summary

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

Property Value

A `System.Int32` specifying the major component.

Description

This property is read-only.

example

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

Example

[C#]

```
using System;
class VersionMajorExample {
   public static void Main() {
      Version vers = new Version("6.1.2.4");
      Console.Write("The major component ");
      Console.WriteLine("of version vers = {0}.",
                        vers.Major);
   }
}
```

The output is

```
The major component of version vers = 6.
```

Version.Minor Property

[ILAsm]
```
.property int32 Minor { public hidebysig specialname instance int32
get_Minor() }
```

[C#]
```
public int Minor { get; }
```

Summary

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

Property Value

A `System.Int32` specifying the minor component.

Description

This property is read-only.

example

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

Example

[C#]

```
using System;
class VersionMinorExample {
   public static void Main() {
      Version vers = new Version("6.1.2.4");
      Console.Write("The minor component ");
      Console.WriteLine("of version vers = {0}.",
                        vers.Minor);
   }
}
```

The output is

```
The minor component of version vers = 1.
```

Version.Revision Property

[ILAsm]

```
.property int32 Revision { public hidebysig specialname instance int32
get_Revision() }
```

[C#]

```
public int Revision { get; }
```

Summary

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

Property Value

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

Description

This property is read-only.

example

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

Example

[C#]

```
using System;
class VersionRevisionExample {
   public static void Main() {
      Version vers = new Version("6.1.2.4");
      Console.Write("The revision component of ");
      Console.WriteLine("version vers = {0}.",
                        vers.Revision);
   }
}
```

The output is

```
The revision component of version vers = 4.
```