

1 System.Environment Class

```
2 [ILAsm]  
3 .class public sealed Environment extends System.Object  
  
4 [C#]  
5 public sealed class Environment
```

6 Assembly Info:

- 7 • *Name:* mscorlib
- 8 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 9 • *Version:* 2.0.x.x
- 10 • *Attributes:*
 - 11 ○ CLSCompliantAttribute(true)

12 Summary

13 Provides the current settings for, and information about, the execution environment.

14 Inherits From: System.Object

15

16 **Library:** BCL

17

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

20

21 Description

22 [*Note:* Use this class to retrieve the following information:

- 23 • Command line arguments
- 24 • Exit codes
- 25 • Environment variable settings
- 26 • Contents of the call stack
- 27 • Time since last system boot
- 28 • Version of the execution engine

29]

30

1 **Environment.Exit(System.Int32) Method**

```
2 [ILAsm]
3 .method public hidebysig static void Exit(int32 exitCode)
4 [C#]
5 public static void Exit(int exitCode)
```

6 **Summary**

7 Terminates the current process and sets the process exit code to the specified value.

8 **Parameters**

Parameter	Description
<i>exitCode</i>	A <code>System.Int32</code> value that is provided to the operating system.

9
10 **Description**

11 This method causes an executing program to halt.

12 **Exceptions**

Exception	Condition
System.Security.SecurityException	The immediate caller does not have the required permission.

13
14 **Permissions**

Permission	Description
System.Security.Permissions.SecurityPermission	Requires unmanaged code permission. See <code>System.Security.Permissions.SecurityPermissionFlag.UnmanagedCode</code> .

15
16

1 Environment.GetCommandLineArgs() Method

```
2 [ILAsm]  
3 .method public hidebysig static string[] GetCommandLineArgs()  
4 [C#]  
5 public static string[] GetCommandLineArgs()
```

6 Summary

7 Returns the arguments specified on the command line.

8 Return Value

9 Returns a `System.String` array. Each `System.String` in the array contains a single
10 command line argument.

11 Description

12 The first element in the array contains the filename of the executing program. If the
13 filename is not available, the first element is equal to `System.String.Empty`. The
14 remaining elements contain any additional tokens entered on the command line.

15
16 [*Note:* The program filename can, but is not required to, include path information.

17
18 To obtain the command line as a single `System.String`, use the
19 `System.Environment.CommandLine` property.

20
21]

22

1
2 **Environment.GetEnvironmentVariable(System**
3 **.String) Method**

```
4 [ILAsm]  
5 .method public hidebysig static string GetEnvironmentVariable(string  
6 variable)  
7 [C#]  
8 public static string GetEnvironmentVariable(string variable)
```

9 **Summary**

10 Returns the value of the specified environment variable.

11 **Parameters**

Parameter	Description
<i>variable</i>	A System.String containing the name of an environment variable.

12
13 **Return Value**

14 A System.String containing the current setting of *variable*, or null.

15 **Description**

16 If *variable* contains a valid name for an environment variable, and if the caller has
17 sufficient permissions, this method returns the current setting for *variable*. Environment
18 variable names are case-insensitive.

19
20 If *variable* specifies an invalid name or the system does not support environment
21 variables, this method returns null.

22
23 [Note: To obtain names and settings for all environment variables, use the
24 System.Environment.GetEnvironmentVariables method.]
25
26

27 **Exceptions**

Exception	Condition
System.ArgumentNullException	<i>variable</i> is a null reference.
System.Security.SecurityException	The caller does not have the required permission.

1

2 Permissions

Permission	Description
System.Security.Permissions.EnvironmentPermission	Requires permission to read environment variables. See System.Security.Permissions.EnvironmentPermissionAccess.Read.

3

4

1 Environment.GetEnvironmentVariables() 2 Method

```
3 [ILAsm]  
4 .method public hidebysig static class System.Collections.IDictionary  
5 GetEnvironmentVariables()  
  
6 [C#]  
7 public static IDictionary GetEnvironmentVariables()
```

8 Summary

9 Returns all environment variables and their current settings.

10 Return Value

11 A System.Collections.IDictionary object containing environment variable names
12 and settings, or null if the system does not support environment variables.

13 Description

14 The names and settings for the environment variables are stored in the returned
15 System.Collections.IDictionary object as keys and values, respectively.
16

17 [Note: To obtain the setting of a single environment variable, use the
18 System.Environment.GetEnvironmentVariable method.]
19
20

21 Exceptions

Exception	Condition
System.Security.SecurityException	The caller does not have the required permission.

23 Example

24 The following example prints the names and values of all environment variables defined
25 in the environment.

```
26 [C#]  
27  
28 using System;  
29 using System.Collections;  
30  
31 class EnvTest:Object {  
32     public static void Main() {  
33         Console.WriteLine("Environment Variables");
```

```
1     IDictionary envvars =
2         Environment.GetEnvironmentVariables();
3     IDictionaryEnumerator varEnumerator =
4         envvars.GetEnumerator();
5     while(varEnumerator.MoveNext() != false) {
6         Console.WriteLine("{0}={1}",
7             varEnumerator.Key,
8             varEnumerator.Value);
9     }
10 }
11 }
```

13 The output will vary depending on your system.

14 Permissions

Permission	Description
System.Security.Permissions.EnvironmentPermission	Requires permission to read environment variables. See System.Security.Permissions.EnvironmentPermissionAccess.Read.

15

16

1 Environment.CommandLine Property

```
2 [ILAsm]  
3 .property string CommandLine { public hidebysig static specialname string  
4 get_CommandLine() }  
5 [C#]  
6 public static string CommandLine { get; }
```

7 Summary

8 Gets the information entered on the command line when the current process was
9 started.

10 Property Value

11 A `System.String` containing the command line arguments.

12 Description

13 This property is read-only.

14
15 This property provides access to the program name and any arguments specified on the
16 command line when the current process was started.

17
18 If the environment does not support a program name, as can be the case with compact
19 devices, then the program name is equal to `System.String.Empty`.

20
21 The format of the information returned by this property is implementation-specific.

22
23 [*Note:* The program name can, but is not required to, include path information.

24
25 Use the `System.Environment.GetCommandLineArgs` method to retrieve the command
26 line information parsed and stored in an array of strings.

27
28]

29

1 Environment.ExitCode Property

```
2 [ILAsm]  
3 .property int32 ExitCode { public hidebysig static specialname int32  
4 get_ExitCode() public hidebysig static specialname void set_ExitCode(int32  
5 value) }  
  
6 [C#]  
7 public static int ExitCode { get; set; }
```

8 Summary

9 Gets or sets the exit code of a process.

10 Property Value

11 A System.Int32 value returned by a process. The default value is zero.

12 Description

13 When a process exits, if the process does not return a value, the value of
14 System.Environment.ExitCode is returned. If the value of this property is not set by an
15 application, zero is returned.

16
17 On operating systems that do not support process exit codes, CLI implementations are
18 required to fully support getting and setting values for this property.

19

1 Environment.HasShutdownStarted Property

```
2 [ILAsm]  
3 .property bool HasShutdownStarted { public hidebysig static specialname  
4 instance bool get_HasShutdownStarted() }  
  
5 [C#]  
6 public static bool HasShutdownStarted { get; }
```

7 Summary

8 Gets a value indicating whether an application has started to shut down.

9 Property Value

10 A System.Boolean where true indicates the shutdown process has started; otherwise
11 false.

12 Description

13 This property is read-only.

14
15 [*Note:* This property is for use inside the finalizer of an application. If the shutdown
16 process has started, static members should not be accessed; they might have been
17 cleaned up by the garbage collector. If the member has been cleaned up, any access
18 attempt will cause an exception to be thrown.

19
20 System.Console.Out is a special case that is always available after the shutdown
21 process has started.

22
23]

24

1 Environment.NewLine Property

```
2 [ILAsm]
3 .property string NewLine { public hidebysig static specialname string
4 get_NewLine() }
5 [C#]
6 public static string NewLine { get; }
```

7 Summary

8 Gets the newline string for the current platform.

9 Property Value

10 A System.String containing the characters required to write a newline.

11 Description

12 This property is read-only.

13
14 [*Note:* This property is intended for platform-independent formatting of multi-line
15 strings. This value is automatically appended to text when using WriteLine methods,
16 such as System.Console.WriteLine.]
17
18

19 Example

20 The following example demonstrates using the System.Environment.NewLine property.
21 The string returned by System.Environment.NewLine is inserted between "Hello" and
22 "World", causing a line break between the words in the output.

```
23 [C#]
24
25 using System;
26 class TestClass {
27     public static void Main() {
28         Console.WriteLine("Hello,{0}World",
29                             Environment.NewLine);
30     }
31 }
```

32 The output is

```
33
34 Hello,
35
36 World
37
```

38

39

1 Environment.StackTrace Property

```
2 [ILAsm]  
3 .property string StackTrace { public hidebysig static specialname string  
4 get_StackTrace() }  
  
5 [C#]  
6 public static string StackTrace { get; }
```

7 Summary

8 Returns a string representation of the state of the call stack.

9 Property Value

10 A System.String containing a description of the methods currently in the call stack.
11 This value can be System.String.Empty.

12 Description

13 This property is read-only.

14
15 [Note: An example of how the System.String returned by this property might be
16 formatted follows, where one line of information is provided for each method on the call
17 stack:

18
19 at *FullClassName.MethodName(MethodParms)* in *FileName*:line *LineNumber*

20
21 *FullClassName*, *MethodName*, *MethodParms*, *FileName*, and *LineNumber* are defined as
22 follows:

Item	Description
<i>FullClassName</i>	The fully qualified name of the class.
<i>MethodName</i>	The name of the method.
<i>MethodParms</i>	The list of parameter type/name pairs. Each pair is separated by a comma (,). This information is omitted if <i>MethodName</i> takes zero parameters.
<i>FileName</i>	The name of the source file where the <i>MethodName</i> method is declared. This information is omitted if debug symbols are not available.
<i>LineNumber</i>	The number of the line in <i>FileName</i> that contains the source code from <i>MethodName</i> for the instruction that is on the call stack. This information is omitted if debug symbols are not available.

1
2 The literal "at" is preceded by a single space.
3
4 The literals "in" and ":line" are omitted if debug symbols are not available.
5
6 The method calls are described in reverse chronological order (the most recent method
7 call is described first).
8
9 System.Environment.StackTrace might not report as many method calls as expected,
10 due to code transformations that occur during optimization.
11
12]

13 Example

14 The following example gets the System.Environment.StackTrace property from within
15 a series of nested calls.

```
16 [C#]  
17  
18 using System;  
19 public class TestCallStack {  
20     public void MyMethod1 () {  
21         MyMethod2();  
22     }  
23     public void MyMethod2 () {  
24         MyMethod3();  
25     }  
26     public void MyMethod3 () {  
27         Console.WriteLine("TestCallStack: {0}",  
28             Environment.StackTrace);  
29     }  
30     public static void Main() {  
31         TestCallStack t = new TestCallStack();  
32         t.MyMethod1();  
33     }  
34 }
```

35 Without debug symbols the output is

```
36 TestCallStack: at System.Environment.GetStackTrace(Exception e)  
37  
38  
39  
40 at System.Environment.GetStackTrace(Exception e)  
41  
42  
43 at System.Environment.get_StackTrace()  
44  
45  
46 at TestCallStack.Main()  
47  
48
```

49 With debug symbols the output is

```
1
2 TestCallStack: at System.Environment.GetStackTrace(Exception e)
3
4
5 at System.Environment.GetStackTrace(Exception e)
6
7
8 at System.Environment.get_StackTrace()
9
10
11 at TestCallStack.MyMethod3() in c:\ECMAExamples\envstack.cs:line 10
12
13
14 at TestCallStack.MyMethod2() in c:\ECMAExamples\envstack.cs:line 8
15
16
17 at TestCallStack.MyMethod1() in c:\ECMAExamples\envstack.cs:line 5
18
19
20 at TestCallStack.Main() in c:\ECMAExamples\envstack.cs:line 15
21
22
```

1 Environment.TickCount Property

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

7 Summary

8 Gets the number of milliseconds elapsed since the system was started.

9 Property Value

10 A `System.Int32` value containing the amount of time in milliseconds that has passed
11 since the last time the computer was started.

12 Description

13 This property is read-only.

14
15 The resolution of the `System.Environment.TickCount` property cannot be less than 500
16 milliseconds.

17
18 The value of this property is derived from the system timer.

19
20 The `System.Environment.TickCount` property handles an overflow condition by
21 resetting its value to zero. The minimum value returned by
22 `System.Environment.TickCount` is 0.

23
24 [*Note:* `System.Environment.TickCount` is measured in milliseconds, not in "ticks".

25
26 The `System.Environment.TickCount` reaches its maximum value after approximately
27 24.8 days of continuous up time.

28
29 For applications that require a finer granularity or a larger maximum time than
30 `System.Environment.TickCount` supports, see `System.DateTime.Now`.

31
32]

33

1 Environment.Version Property

```
2 [ILAsm]  
3 .property class System.Version Version { public hidebysig static  
4 specialname class System.Version get_Version() }  
  
5 [C#]  
6 public static Version Version { get; }
```

7 Summary

8 Gets the current version of the execution engine.

9 Property Value

10 A `System.Version` object that contains the major, minor, build, and revision numbers of
11 the execution engine.

12 Description

13 This property is read-only.

14