

1 System.Security.Permissions.EnvironmentPer 2 mission Class

```
3 [ILAsm]  
4 .class public sealed serializable EnvironmentPermission extends  
5 System.Security.CodeAccessPermission  
  
6 [C#]  
7 public sealed class EnvironmentPermission: CodeAccessPermission
```

8 Assembly Info:

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

14 Implements:

- 15 • **System.Security.IPermission**

16 Summary

17 Controls access to environment variables.

18 Inherits From: System.Security.CodeAccessPermission

19
20 **Library:** BCL

21
22 **Thread Safety:** All public static members of this type are safe for multithreaded operations.
23 No instance members are guaranteed to be thread safe.

24 25 Description

26 [Note: System.Security.Permissions.EnvironmentPermission objects describe
27 protected operations on environment variables. This permission distinguishes between
28 the following types of access provided by
29 System.Security.Permissions.EnvironmentPermissionAccess:

- 30 • **Read:** Read values from environment variables.
- 31 • **Write:** Write values to environment variables. Also allows for creating and deleting
32 values.
- 33 • **NoAccess:** No access to environment variables.

- 1 • `AllAccess`: Full access to environment variables. Identical to specifying `Read` and
2 `Write` access.

3 These access levels are independent, meaning that rights to one do not imply rights to
4 another. For example, `Write` permission does not imply permission to `Read`.
5 `System.Security.Permissions.EnvironmentPermissionAccess` values can be combined
6 using a bitwise OR operator.

7
8 The `System.Environment` class is used to access environment variables, subject to the
9 permissions defined by `System.Security.Permissions.EnvironmentPermission`.
10 Environment variables are case-insensitive.

11
12]

13
14 The XML encoding of a `System.Security.Permissions.EnvironmentPermission` instance
15 is defined below in EBNF format. The following conventions are used:

- 16 • All non-literals in the grammar below are shown in normal type.
17 • All literals are in bold font.

18 The following meta-language symbols are used:

- 19 • '*' represents a meta-language symbol suffixing an expression that can appear zero
20 or more times.
21 • '?' represents a meta-language symbol suffixing an expression that can appear zero
22 or one time.
23 • '+' represents a meta-language symbol suffixing an expression that can appear one
24 or more times.
25 • '(,)' is used to group literals, non-literals or a mixture of literals and non-literals.
26 • '|' denotes an exclusive disjunction between two expressions.
27 • '::=' denotes a production rule where a left hand non-literal is replaced by a right
28 hand expression containing literals, non-literals or both.

29 `BuildVersion` refers to the build version of the shipping CLI. This is specified as a dotted
30 build number such as '2412.0'.

31
32 `ECMAPubKeyToken`::= `b77a5c561934e089`

33
34 `EnvironmentVariable` refers to the name of a single environment variable, such as 'PROMPT'.

35
36 The XML encoding of an `EnvironmentPermission` instance is as follows:
37

```

1 EnvironmentPermissionXML ::=
2
3
4 <IPermission
5
6
7 class="
8
9
10 System.Security.Permissions.EnvironmentPermission,
11
12
13 mscorlib,
14
15
16 Version=1.0.BuildVersion,
17
18
19 Culture=neutral,
20
21
22 PublicKeyToken=ECMAPubKeyToken"
23
24
25 version="1"
26
27
28 (
29
30
31 Unrestricted="true"
32
33
34 )
35
36
37 |
38
39
40 (
41
42
43 (Read=" EnvironmentVariable (; EnvironmentVariable)*" )?
44
45
46 (Write="EnvironmentVariable (; EnvironmentVariable)* " )?
47
48
49 )
50

```

1
2 />
3
4

1
2 **EnvironmentPermission(System.Security.Permissions.PermissionState) Constructor**
3

```
4 [ILAsm]  
5 public rtspecialname specialname instance void .ctor(valuetype  
6 System.Security.Permissions.PermissionState state)  
  
7 [C#]  
8 public EnvironmentPermission(PermissionState state)
```

9 **Summary**

10 Constructs and initializes a new instance of the
11 System.Security.Permissions.EnvironmentPermission class with the specified
12 System.Security.Permissions.PermissionState value.

13 **Parameters**

Parameter	Description
<i>state</i>	A System.Security.Permissions.PermissionState value.

14
15 **Description**

16 [Note: The instance returned by this constructor has either fully restricted
17 (System.Security.Permissions.PermissionState.None) or unrestricted
18 (System.Security.Permissions.PermissionState.Unrestricted) access to all
19 environment variables.
20
21]

22 **Exceptions**

Exception	Condition
System.ArgumentException	<i>state</i> is not a valid System.Security.Permissions.PermissionState value.

23
24

EnvironmentPermission(System.Security.Permissions.EnvironmentPermissionAccess, System.String) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(valuetype
System.Security.Permissions.EnvironmentPermissionAccess flag, string
pathList)

[C#]
public EnvironmentPermission(EnvironmentPermissionAccess flag, string
pathList)
```

Summary

Constructs a new instance of the `System.Security.Permissions.EnvironmentPermission` class with the specified access to the specified environment variables.

Parameters

Parameter	Description
<i>flag</i>	One of values defined by <code>System.Security.Permissions.EnvironmentPermissionAccess</code> .
<i>pathList</i>	A <code>System.String</code> containing one or more case-insensitive environment variable names separated by <code>System.IO.Path.PathSeparator</code> .

Description

The specified access is applied to all environment variables in *pathList*.

Exceptions

Exception	Condition
System.ArgumentNullException	<i>pathList</i> is null.
System.ArgumentException	<i>flag</i> specifies a value not defined in <code>System.Security.Permissions.EnvironmentPermissionAccess</code> .

21

1 EnvironmentPermission.Copy() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual class System.Security.IPermission Copy()  
4 [C#]  
5 public override IPermission Copy()
```

6 Summary

7 Returns a new `System.Security.Permissions.EnvironmentPermission` object
8 containing the same values as the current instance.

9 Return Value

10 A new `System.Security.Permissions.EnvironmentPermission` containing the same
11 values as the current instance.

12 Description

13 [*Note:* The object returned by this method represents the same level of access to the
14 same environment variables as the current instance.]

15

16

17

18 This method overrides `System.Security.CodeAccessPermission.Copy` and is
19 implemented to support the `System.Security.IPermission` interface.

20

1
2 **EnvironmentPermission.FromXml(System.Security.SecurityElement) Method**
3

```
4 [ILAsm]  
5 .method public hidebysig virtual void FromXml(class  
6 System.Security.SecurityElement esd)  
  
7 [C#]  
8 public override void FromXml(SecurityElement esd)
```

9 **Summary**

10 Reconstructs the state of a `System.Security.Permissions.EnvironmentPermission`
11 object using the specified XML encoding.

12 **Parameters**

Parameter	Description
<i>esd</i>	A <code>System.Security.SecurityElement</code> instance containing the XML encoding to use to reconstruct the state of a <code>System.Security.Permissions.EnvironmentPermission</code> object.

13
14 **Description**

15 The state of the current instance is changed to the state encoded in *esd*.

16
17 [Note: For the XML encoding for this class, see the
18 `System.Security.Permissions.EnvironmentPermission` class page.

19
20 This method overrides `System.Security.CodeAccessPermission.FromXml`.

21
22]

23 **Exceptions**

Exception	Condition
System.ArgumentNullException	<i>esd</i> is null.
System.ArgumentException	<i>esd</i> does not contain the encoding for a <code>System.Security.Permissions.EnvironmentPermission</code> instance.

	The version number of <i>esd</i> is not valid.
--	--

1

2

EnvironmentPermission.Intersect(System.Security.IPermission) Method

```
[ILAsm]
.method public hidebysig virtual class System.Security.IPermission
Intersect(class System.Security.IPermission target)

[C#]
public override IPermission Intersect(IPermission target)
```

Summary

Returns a new System.Security.Permissions.EnvironmentPermission object that is the intersection of the current instance and the specified object.

Parameters

Parameter	Description
<i>target</i>	A System.Security.Permissions.EnvironmentPermission instance to intersect with the current instance.

Return Value

A new System.Security.Permissions.EnvironmentPermission instance that represents the intersection of the current instance and *target*. If the intersection is empty or *target* is null, returns null. If the current instance is unrestricted, returns a copy of *target*. If *target* is unrestricted, returns a copy of the current instance.

Description

[Note: The intersection of two permissions is a permission that secures the resources and operations secured by both permissions. Specifically, it represents the minimum permission such that any demand that passes both permissions will also pass their intersection.

This method overrides System.Security.CodeAccessPermission.Intersect and is implemented to support the System.Security.IPermission interface.

]

Exceptions

Exception	Condition
-----------	-----------

System.ArgumentException

target is not null and is not of type
System.Security.Permissions.EnvironmentPermission.

1

2

EnvironmentPermission.IsSubsetOf(System.Security.IPermission) Method

```
[ILAsm]
.method public hidebysig virtual bool IsSubsetOf(class
System.Security.IPermission target)

[C#]
public override bool IsSubsetOf(IPermission target)
```

Summary

Determines whether the current instance is a subset of the specified object.

Parameters

Parameter	Description
<i>target</i>	A System.Security.Permissions.EnvironmentPermission instance that is to be tested for the subset relationship.

Return Value

true if the current instance is a subset of *target*; otherwise, false. If the current instance is unrestricted, and *target* is not, returns false. If *target* is unrestricted, returns true. If *target* is null and no environment variables are secured by the current instance, returns true. If *target* is null, and the current instance secures one or more environment variables, returns false.

Description

[Note: The current instance is a subset of *target* if the current instance specifies a set of accesses to resources that is wholly contained by *target*. For example, a permission that represents read access to a file is a subset of a permission that represents read and write access to the file.

If this method returns true, the current instance describes a level of access to a set of environment variables that is also described by *target*.

This method overrides System.Security.CodeAccessPermission.IsSubsetOf and is implemented to support the System.Security.IPermission interface.

]

Exceptions

Exception	Condition
System.ArgumentException	<i>target</i> is not null and is not of type System.Security.Permissions.EnvironmentPermission.

1

2

1 EnvironmentPermission.ToXml() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual class System.Security.SecurityElement  
4 ToXml()  
5 [C#]  
6 public override SecurityElement ToXml()
```

7 Summary

8 Returns the XML encoding of the current instance.

9 Return Value

10 A System.Security.SecurityElement containing the XML encoding of the state of the
11 current instance.

12 Description

13 [*Note:* For the XML encoding for this class, see the
14 System.Security.Permissions.EnvironmentPermission class page.

15
16 This method overrides System.Security.CodeAccessPermission.ToXml.

17
18]

19

EnvironmentPermission.Union(System.Security.IPermission) Method

```
[ILAsm]
.method public hidebysig virtual class System.Security.IPermission
Union(class System.Security.IPermission other)

[C#]
public override IPermission Union(IPermission other)
```

Summary

Returns a new `System.Security.Permissions.EnvironmentPermission` that is the union of the current instance and the specified object.

Parameters

Parameter	Description
<i>other</i>	A <code>System.Security.Permissions.EnvironmentPermission</code> instance to combine with the current instance.

Return Value

A new `System.Security.Permissions.EnvironmentPermission` instance that represents the union of the current instance and *other*. If the current instance or *other* is unrestricted, returns a `System.Security.Permissions.EnvironmentPermission` instance that is unrestricted. If *other* is null, returns a copy of the current instance via the `System.Security.IPermission.Copy` method. If the current instance and *other* do not specify any environment variables, returns null.

Description

[*Note:* The result of a call to `System.Security.Permissions.EnvironmentPermission.Union` is a permission that represents the access to environment variables represented by the current instance as well as the access to environment variables represented by *other*. Any demand that passes either the current instance or *other* passes their union.

This method overrides `System.Security.CodeAccessPermission.Union` and is implemented to support the `System.Security.IPermission` interface.

]

Exceptions

Exception	Condition
System.ArgumentException	<i>other</i> is not null and is not of type System.Security.Permissions.EnvironmentPermission.

1
2