

1 System.Security.Permissions.SecurityPermiss 2 ionAttribute Class

```
3 [ILAsm]  
4 .class public sealed serializable SecurityPermissionAttribute extends  
5 System.Security.Permissions.CodeAccessSecurityAttribute  
  
6 [C#]  
7 public sealed class SecurityPermissionAttribute:  
8 CodeAccessSecurityAttribute
```

9 Assembly Info:

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

15 Type Attributes:

- 16 • AttributeUsageAttribute(AttributeTargets.Assembly | AttributeTargets.Class |
17 AttributeTargets.Struct | AttributeTargets.Constructor | AttributeTargets.Method,
18 AllowMultiple=true, Inherited=false)

19 Summary

20 Used to apply a security action and a set of security permissions to program code.

21 **Inherits From:** System.Security.Permissions.CodeAccessSecurityAttribute

22 **Library:** BCL

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

25 Description

26 [Note: The security permissions are defined in the
27 System.Security.Permissions.SecurityPermissionFlag enumeration and are
28 specified using the
29 System.Security.Permissions.SecurityPermissionAttribute.Flags property.

30 The security information declared by a security attribute is stored in the metadata of the
31 attribute target, and is accessed by the system at run-time. Security attributes are used
32 for declarative security only. For imperative security, use the corresponding permission
33 class, System.Security.Permissions.SecurityPermission.
34
35
36
37
38

1 The allowable `System.Security.Permissions.SecurityPermissionAttribute` targets
2 are determined by the `System.Security.Permissions.SecurityAction` passed to the
3 constructor.
4
5]

6 **Example**

7 In the following example, the attribute target is an assembly. The attribute declares that
8 the ability to assert permissions on behalf of callers is the minimum permission required
9 for the assembly to execute.
10
11 [assembly:SecurityPermissionAttribute(SecurityAction.RequestMinimum,
12 Assertion=true)]

13

1
2 **SecurityPermissionAttribute(System.Security.Permissions.SecurityAction) Constructor**
3

```
4 [ILAsm]  
5 public rtspecialname specialname instance void .ctor(valuetype  
6 System.Security.Permissions.SecurityAction action)  
7  
8 [C#]  
9 public SecurityPermissionAttribute(SecurityAction action)
```

9 **Summary**

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

13 **Parameters**

Parameter	Description
<i>action</i>	A System.Security.Permissions.SecurityAction value.

14
15 **Exceptions**

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

16
17

1 2 SecurityPermissionAttribute.CreatePermissio 3 n() Method

```
4 [ILAsm]  
5 .method public hidebysig virtual class System.Security.IPermission  
6 CreatePermission()  
7 [C#]  
8 public override IPermission CreatePermission()
```

9 Summary

10 Returns a new System.Security.Permissions.SecurityPermission object that
11 contains the security information of the current instance.

12 Return Value

13 A new System.Security.Permissions.SecurityPermission object with the security
14 information of the current instance.

15 Description

16 [Note: Applications typically do not call this method; it is intended for use by the
17 system.

18
19 The security information declared by a security attribute is stored in the metadata of the
20 attribute target, and is accessed by the system at run-time. The system uses the object
21 returned by this method to convert the security information of the current instance into
22 the form stored in metadata.

23
24 This method overrides
25 System.Security.Permissions.SecurityAttribute.CreatePermission.

26
27]

28

1 SecurityPermissionAttribute.Flags Property

```
2 [ILAsm]  
3 .property valuetype System.Security.Permissions.SecurityPermissionFlag  
4 Flags { public hidebysig specialname instance valuetype  
5 System.Security.Permissions.SecurityPermissionFlag get_Flags() public  
6 hidebysig specialname instance void set_Flags(valuetype  
7 System.Security.Permissions.SecurityPermissionFlag value) }  
  
8 [C#]  
9 public SecurityPermissionFlag Flags { get; set; }
```

10 Summary

11 Gets or sets values that define the permissions declared by the current instance.

12 Property Value

13 One or more System.Security.Permissions.SecurityPermissionFlag values. To
14 specify multiple values in a set operation, use the bitwise OR operator.

15