

1 System.Security.Permissions.SecurityPermiss 2 ionFlag Enum

```
3 [ILAsm]  
4 .class public sealed serializable SecurityPermissionFlag extends  
5 System.Enum  
  
6 [C#]  
7 public enum SecurityPermissionFlag
```

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 Type Attributes:

- 15 • FlagsAttribute

16 Summary

17 Specifies a set of security permissions applied to a
18 System.Security.Permissions.SecurityPermission instance.

19 Inherits From: System.Enum

20
21 Library: BCL

23 Description

24 This enumeration is used by System.Security.Permissions.SecurityPermission.
25

26 System.Security.Permissions.SecurityPermissionFlag is a bit-field; specify
27 multiple values using the bitwise OR operator.
28

29 For information on security, see Partition II of the CLI Specification.
30

31 [*Note:* Many of these flags are powerful and should only be granted to highly trusted
32 code.]
33
34

35

1 SecurityPermissionFlag.Assertion Field

```
2 [ILAsm]  
3 .field public static literal valuetype  
4 System.Security.Permissions.SecurityPermissionFlag Assertion = 0x1  
  
5 [C#]  
6 Assertion = 0x1
```

7 Summary

8 Specifies the ability to assert that all of the callers of the code granted this permission
9 will pass the check for a specific permission or permission set.

10

11 The ability to assert a specific permission or permission set allows code to ensure that
12 its callers do not fail with a security exception for lack of the specific permission or
13 permission set asserted.

14

15 [*Note:* Asserting a permission is often used when writing library code that accesses
16 protected resources but itself does not expose these resources in any exploitable way to
17 the calling code.]

18

19

20

1 SecurityPermissionFlag.ControlThread Field

```
2 [ILAsm]  
3 .field public static literal valuetype  
4 System.Security.Permissions.SecurityPermissionFlag ControlThread = 0x10  
5 [C#]  
6 ControlThread = 0x10
```

7 Summary

8 Specifies the ability to control thread behavior. The operations protected include
9 System.Threading.Thread.Abort and System.Threading.Thread.ResetAbort.

10

1 SecurityPermissionFlag.Execution Field

```
2 [ILAsm]  
3 .field public static literal valuetype  
4 System.Security.Permissions.SecurityPermissionFlag Execution = 0x8  
5 [C#]  
6 Execution = 0x8
```

7 Summary

8 Specifies permission for the code to run. Without this permission managed code cannot
9 execute.

10

1 SecurityPermissionFlag.NoFlags Field

```
2 [ILAsm]  
3 .field public static literal valuetype  
4 System.Security.Permissions.SecurityPermissionFlag NoFlags = 0x0  
5 [C#]  
6 NoFlags = 0x0
```

7 Summary

8 Specifies that none of the permissions in this enumeration are available.

9

1 SecurityPermissionFlag.SkipVerification Field

```
2 [ILAsm]  
3 .field public static literal valuetype  
4 System.Security.Permissions.SecurityPermissionFlag SkipVerification = 0x4  
5 [C#]  
6 SkipVerification = 0x4
```

7 Summary

8 Specifies the right to skip the verification checks that ensure type safety and metadata
9 correctness in an assembly. If an assembly has been granted this permission it will not
10 fail with a `System.Security.VerificationException` even if the assembly contains
11 unverifiable constructs.

12
13 [*Note:* Code that is unverifiable can execute without causing a
14 `System.Security.VerificationException` if this permission is granted.]
15
16

17

1 SecurityPermissionFlag.UnmanagedCode

2 Field

```
3 [ILAsm]  
4 .field public static literal valuetype  
5 System.Security.Permissions.SecurityPermissionFlag UnmanagedCode = 0x2  
  
6 [C#]  
7 UnmanagedCode = 0x2
```

8 Summary

9 Specifies the ability to call unmanaged code.

10

11 [*Note:* Because unmanaged code potentially allows other permissions to be bypassed,
12 this permission should be used with caution. It is used for applications calling native
13 code using PInvoke.]

14

15

16