System.Security.Permissions.SecurityPermissionFlag Enum

```
[ILAsm]
.class public sealed serializable SecurityPermissionFlag extends
System.Enum

[C#]
public enum SecurityPermissionFlag
```

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)

Type Attributes:

- FlagsAttribute

Summary

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

Inherits From: System.Enum

Library: BCL

Description

This enumeration is used by `System.Security.Permissions.SecurityPermission`.

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

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

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

SecurityPermissionFlag.Assertion Field

```
[ILAsm]
.field public static literal valuetype
System.Security.Permissions.SecurityPermissionFlag Assertion = 0x1

[C#]
Assertion = 0x1
```

Summary

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

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

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

SecurityPermissionFlag.ControlThread Field

```
[ILAsm]
.field public static literal valuetype
System.Security.Permissions.SecurityPermissionFlag ControlThread = 0x10

[C#]
ControlThread = 0x10
```

Summary

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

SecurityPermissionFlag.Execution Field

```
[ILAsm]
.field public static literal valuetype
System.Security.Permissions.SecurityPermissionFlag Execution = 0x8

[C#]
Execution = 0x8
```

Summary

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

SecurityPermissionFlag.NoFlags Field

```
[ILAsm]
.field public static literal valuetype
System.Security.Permissions.SecurityPermissionFlag NoFlags = 0x0

[C#]
NoFlags = 0x0
```

Summary

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

SecurityPermissionFlag.SkipVerification Field

```
[ILAsm]
.field public static literal valuetype
System.Security.Permissions.SecurityPermissionFlag SkipVerification = 0x4

[C#]
SkipVerification = 0x4
```

Summary

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

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

SecurityPermissionFlag.UnmanagedCode Field

```
[ILAsm]
.field public static literal valuetype
System.Security.Permissions.SecurityPermissionFlag UnmanagedCode = 0x2

[C#]
UnmanagedCode = 0x2
```

Summary

Specifies the ability to call unmanaged code.

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