System.Runtime.CompilerServices.MethodImplOptions Enum

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

[C#]
public enum MethodImplOptions
```

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)

Summary

Defines the details of how a method is implemented.

Inherits From: System.Enum

Library: RuntimeInfrastructure

Description

This enumeration is used by `System.Runtime.CompilerServices.MethodImplAttribute`.

MethodImplOptions.ForwardRef Field

```
[ILAsm]
.field public static literal valuetype
System.Runtime.CompilerServices.MethodImplOptions ForwardRef = 16

[C#]
ForwardRef = 16
```

Summary

Specifies that the method is declared, but its implementation is provided elsewhere.

[*Note:* For most languages, it is recommended that the notion of "forward" be attached to methods using language syntax instead of custom attributes.]

MethodImplOptions.InternalCall Field

[ILAsm]
```
.field public static literal valuetype
System.Runtime.CompilerServices.MethodImplOptions InternalCall = 4096
```

[C#]
```
InternalCall = 4096
```

Summary

Specifies an internal call.

[*Note:* An internal call is a call to a method implemented within the system itself, providing additional functionality that regular managed code cannot provide. `System.Object.MemberwiseClone` is an example of an internally called method.]

MethodImplOptions.NoInlining Field

```
[ILAsm]
.field public static literal valuetype
System.Runtime.CompilerServices.MethodImplOptions NoInlining = 8

[C#]
NoInlining = 8
```

Summary

Specifies that the method is not permitted to be inlined.

MethodImplOptions.Synchronized Field

[ILAsm]
```
.field public static literal valuetype
System.Runtime.CompilerServices.MethodImplOptions Synchronized = 32
```

[C#]
```
Synchronized = 32
```

Summary

Specifies the method can be executed by only one thread at a time.

This option specifies that before a thread can execute the target method, the thread is required to acquire a lock on either the current instance or the `System.Type` object for the method's class. If the target method is an instance method, the lock is on the current instance. If the target is a static method, the lock is on the `System.Type` object. Specifying this option causes the target method to behave as though its statements are enclosed by `System.Threading.Monitor.Enter` and `System.Threading.Monitor.Exit` statements locking the previous described object. This option and the `System.Threading.Monitor` methods are functionally equivalent, and both are functionally equivalent to enclosing the target method's code in a C# lock (this) statement.

[*Note:* Because this option holds the lock for the duration of the target method, it should be used only when the entire method must be single threaded. Use the `System.Threading.Monitor` methods (or the C# lock statement) if the object lock can be taken after the method begins, or released before the method ends. Any mechanism that uses locks can cause an application to experience deadlocks and performance degradation; for these reasons, use this option with care.

For most languages, it is recommended that the notion of "synchronized" be attached to methods using language syntax instead of custom attributes.

]

MethodImplOptions.Unmanaged Field

[ILAsm]
```
.field public static literal valuetype
System.Runtime.CompilerServices.MethodImplOptions Unmanaged = 4
```

[C#]
```
Unmanaged = 4
```

Summary

Specifies that the method is implemented in unmanaged code.