System.Threading.Interlocked Class

```
[ILAsm]
.class public sealed Interlocked extends System.Object

[C#]
public sealed class Interlocked
```

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

The System.Threading.Interlocked class provides atomic operations for variables that are shared by multiple threads.

Inherits From: System.Object

Library: BCL

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

Description

The System.Threading.Interlocked methods protect against errors that can occur when the scheduler switches contexts while a thread is updating a variable that can be accessed by other threads. The members of this class do not throw exceptions.

[*Note:* The System.Threading.Interlocked.Increment method and its counterpart, System.Threading.Interlocked.Decrement, increment or decrement a variable and store the resulting value, as an atomic operation.

The System.Threading.Interlocked.Exchange method atomically exchanges the values of the specified variables. The System.Threading.Interlocked.CompareExchange method provides an atomic operation that compares two values and stores a third value in one of the variables, based on the outcome of the comparison.

]

Interlocked.CompareExchange(System.Int32 &, System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static int32 CompareExchange(int32& location1,
int32 value, int32 comparand)

[C#]
public static int CompareExchange(ref int location1, int value, int
comparand)
```

Summary

Compares two `System.Int32` values for equality and stores a specified value if they are equal.

Parameters

Parameter	Description
location1	A `System.Int32` reference whose value is updated with *value* if the original value of *location1* is equal to *comparand*.
value	A `System.Int32` whose value will replace the value of *location1* if *location1* and *comparand* are equal.
comparand	A `System.Int32` to be compared to *location1.*

Return Value

The original value of *location1*.

Description

The compare and store operations are performed as an atomic operation.

The following member must be implemented if the ExtendedNumerics library is present in the implementation.

Interlocked.CompareExchange(System.Single &, System.Single, System.Single) Method

```
[ILAsm]
.method public hidebysig static float32 CompareExchange(float32&
location1, float32 value, float32 comparand)

[C#]
public static float CompareExchange(ref float location1, float value,
float comparand)
```

Summary

Compares two System.Single values for equality and stores a specified value if they are equal.

Parameters

Parameter	Description
location1	A System.Single whose value is updated with *value* if its original value is equal to *comparand*.
value	The System.Single value that will replace value of *location1* if *location1* and *comparand* are equal.
comparand	A System.Single to be compared to *location1.*

Return Value

A System.Single containing the original value of *location1*.

Description

The compare and store operations are performed as an atomic operation.

Exceptions

Exception	Condition
System.ArgumentNullException	The address of *location1* is `null`.

1
2

3

Interlocked.CompareExchange(System.Object &, System.Object, System.Object) Method

```
[ILAsm]
.method public hidebysig static object CompareExchange(object& location1,
object value, object comparand)

[C#]
public static object CompareExchange(ref object location1, object value,
object comparand)
```

Summary

Compares two `System.Object` variables for equality and stores a specified object if they are equal.

Parameters

Parameter	Description
location1	A `System.Object` reference that is set to *value* if the object to which it refers is equal to *comparand*.
value	The reference that will replace the value of *location1* if *location1* and *comparand* are equal.
comparand	An object to be compared to that referred to by *location1.*

Return Value

A `System.Object` containing the original value of *location1*.

Description

The compare and store operations are performed as an atomic operation.

Exceptions

Exception	Condition
System.ArgumentNullException	The address of *location1* is `null`.

1
2

3

Interlocked.Decrement(System.Int32&) Method

```
[ILAsm]
.method public hidebysig static int32 Decrement(int32& location)

[C#]
public static int Decrement(ref int location)
```

Summary

Decrements the specified variable and stores the result as an atomic operation.

Parameters

Parameter	Description
location	A `System.Int32` containing the variable whose value is to be decremented.

Return Value

A `System.Int32` containing the decremented value.

Description

This method handles an overflow condition by wrapping: if *location* = `System.Int32.MinValue`, *location* - `1` = `System.Int32.MaxValue`. No exception is thrown.

Interlocked.Decrement(System.Int64&) Method

```
[ILAsm]
.method public hidebysig static int64 Decrement(int64& location)

[C#]
public static long Decrement(ref long location)
```

Summary

Decrements the specified variable and stores the result as an atomic operation.

Parameters

Parameter	Description
location	A `System.Int64` containing the variable whose value is to be decremented.

Return Value

A `System.Int64` containing the decremented value.

Description

This method handles an overflow condition by wrapping: if *location* = `System.Int64.MinValue`, *location* - `1` = `System.Int64.MaxValue`. No exception is thrown.

The 64-bit versions of `System.Threading.Interlocked.Increment` and `System.Threading.Interlocked.Decrement` are truly atomic only on systems where a `System.IntPtr` is 64-bits long. On other systems, these methods are atomic with respect to each other, but not with respect to other means of accessing the data.

Interlocked.Exchange(System.Int32&, System.Int32) Method

```
[ILAsm]
.method public hidebysig static int32 Exchange(int32& location1, int32
value)

[C#]
public static int Exchange(ref int location1, int value)
```

Summary

Sets a `System.Int32` variable to a specified value as an atomic operation and returns the original value.

Parameters

Parameter	Description
location1	A `System.Int32` variable to set to the supplied value as an atomic operation.
value	The `System.Int32` value to which *location1* is set.

Return Value

A `System.Int32` containing the value of *location1* before the exchange.

The following member must be implemented if the ExtendedNumerics library is present in the implementation.

Interlocked.Exchange(System.Single&, System.Single) Method

```
[ILAsm]
.method public hidebysig static float32 Exchange(float32& location1,
float32 value)

[C#]
public static float Exchange(ref float location1, float value)
```

Summary

Sets a `System.Single` variable to a specified value as an atomic operation and returns the original value.

Parameters

Parameter	Description
location1	A `System.Single` variable to set to the supplied value as an atomic operation.
value	The `System.Single` value to which *location1* is set.

Return Value

A `System.Single` containing the value of *location1* before the exchange.

Interlocked.Exchange(System.Object&, System.Object) Method

```
[ILAsm]
.method public hidebysig static object Exchange(object& location1, object
value)

[C#]
public static object Exchange(ref object location1, object value)
```

Summary

Sets a `System.Object` reference to refer to a specified object as an atomic operation and returns a reference to the original object.

Parameters

Parameter	Description
location1	The variable to set.
value	The reference to which *location1* is set.

Return Value

The original value of *location1*.

Exceptions

Exception	Condition
System.ArgumentNullException	The address of *location1* is `null`.

Interlocked.Increment(System.Int32&) Method

```
[ILAsm]
.method public hidebysig static int32 Increment(int32& location)

[C#]
public static int Increment(ref int location)
```

Summary

Increments the specified variable and stores the result as an atomic operation.

Parameters

Parameter	Description
location	A `System.Int32` containing the variable whose value is to be incremented.

Return Value

A `System.Int32` containing the incremented value.

Description

This method handles an overflow condition by wrapping: if *location* = `System.Int32.MaxValue`, *location* + `1` = `System.Int32.MinValue`. No exception is thrown.

Interlocked.Increment(System.Int64&) Method

```
[ILAsm]
.method public hidebysig static int64 Increment(int64& location)

[C#]
public static long Increment(ref long location)
```

Summary

Increments the specified variable and stores the result as an atomic operation.

Parameters

Parameter	Description
location	A `System.Int64` containing the variable whose value is to be incremented.

Return Value

A `System.Int64` containing the incremented value.

Description

This method handles an overflow condition by wrapping: if *location* = `System.Int64.MaxValue`, *location* + `1` = `System.Int64.MinValue`. No exception is thrown.

The 64-bit versions of `System.Threading.Interlocked.Increment` and `System.Threading.Interlocked.Decrement` are truly atomic only on systems where a `System.IntPtr` is 64-bits long. On other systems, these methods are atomic with respect to each other, but not with respect to other means of accessing the data.