System.Threading.WaitHandle Class

```
[ILAsm]
.class public abstract WaitHandle extends System.MarshalByRefObject
implements System.IDisposable

[C#]
public abstract class WaitHandle: MarshalByRefObject, IDisposable
```

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)

Implements:

- **System.IDisposable**

Summary

Encapsulates operating-system specific objects that wait for exclusive access to shared resources.

Inherits From: System.MarshalByRefObject

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

[*Note:* This class is typically used as a base class for synchronization objects. Classes derived from `System.Threading.WaitHandle` define a signaling mechanism to indicate taking or releasing exclusive access to a shared resource, but use the inherited `System.Threading.WaitHandle` methods to block while waiting for access to shared resources.

The static methods of this class are used to block a `System.Threading.Thread` until one or more synchronization objects receive a signal.

]

WaitHandle() Constructor

[ILAsm]
```
public rtspecialname specialname instance void .ctor()
```

[C#]
```
public WaitHandle()
```

Summary

Constructs and initializes a new instance of the `System.Threading.WaitHandle` class.

WaitHandle.Close() Method

```
[ILAsm]
.method public hidebysig virtual void Close()

[C#]
public virtual void Close()
```

Summary

Releases all resources held by the current instance.

Description

This method is the public version of the `System.IDisposable.Dispose` method implemented to support the `System.IDisposable` interface.

Behaviors

This method releases any unmanaged resources held by the current instance. This method can, but is not required to, suppress finalization during garbage collection by calling the `System.GC.SuppressFinalize` method.

Default

As described above.

How and When to Override

Override this property to release resources allocated in subclasses.

Usage

Use this method to release all resources held by an instance of `WaitHandle`. Once this method is called, references to the current instance cause undefined behavior.

WaitHandle.Dispose(System.Boolean) Method

```
[ILAsm]
.method family hidebysig virtual void Dispose(bool
explicitDisposing)

[C#]
protected virtual void Dispose(bool explicitDisposing)
```

Summary

Releases the unmanaged resources used by the `System.Threading.WaitHandle` and optionally releases the managed resources.

Parameters

Parameter	Description
explicitDisposing	`true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

Behaviors

This method releases all unmanaged resources held by the current instance. When the *explicitDisposing* parameter is `true`, this method releases all resources held by any managed objects referenced by the current instance. This method invokes the `Dispose()` method of each referenced object.

How and When to Override

Override this method to dispose of resources allocated by types derived from `System.Threading.WaitHandle`. When overriding `Dispose`(`System.Boolean`), be careful not to reference objects that have been previously disposed in an earlier call to `Dispose` or `Close`. `Dispose` can be called multiple times by other objects.

Usage

This method is called by the public `System.Threading.WaitHandle.Dispose` method and the `System.Object.Finalize` method. `Dispose()` invokes this method with the *explicitDisposing* parameter set to `true`. `System.Object.Finalize` invokes `Dispose` with *explicitDisposing* set to `false`.

WaitHandle.Finalize() Method

[ILAsm]
```
.method family hidebysig virtual void Finalize()
```

[C#]
```
~WaitHandle()
```

Summary

Releases the resources held by the current instance.

Description

[*Note:* Application code does not call this method; it is automatically invoked during garbage collection unless finalization by the garbage collector has been disabled. For more information, see `System.GC.SuppressFinalize`, and `System.Object.Finalize`.

This method overrides `System.Object.Finalize`.

]

WaitHandle.System.IDisposable.Dispose() Method

[ILAsm]
```
.method private final hidebysig virtual void
System.IDisposable.Dispose()
```

[C#]
```
void IDisposable.Dispose()
```

Summary

Implemented to support the `System.IDisposable` interface. [Note: For more information, see `System.IDisposable.Dispose`.]

WaitHandle.WaitAll(System.Threading.WaitHandle[]) Method

```
[ILAsm]
.method public hidebysig static bool WaitAll(class
System.Threading.WaitHandle[] waitHandles)

[C#]
public static bool WaitAll(WaitHandle[] waitHandles)
```

Summary

Waits for all of the elements in the specified array to receive a signal.

Parameters

Parameter	**Description**
waitHandles	A `System.Threading.WaitHandle` array containing the objects for which the current instance will wait. This array cannot contain multiple references to the same object (duplicates).

Return Value

Returns `true` when every element in *waitHandles* has received a signal. If the current thread receives a request to abort before the signals are received, this method returns `false`.

The maximum number of objects specified in the *waitHandles* array is system defined.

Exceptions

Exception	**Condition**
System.ArgumentNullException	*waitHandles* is `null` or one or more elements in the *waitHandles* array is `null`.
System.DuplicateWaitObjectException	*waitHandles* contains elements that are duplicates.
System.NotSupportedException	The number of objects in *waitHandles* is greater than the system permits.

WaitHandle.WaitAny(System.Threading.WaitHandle[]) Method

```
[ILAsm]
.method public hidebysig static int32 WaitAny(class
System.Threading.WaitHandle[] waitHandles)

[C#]
public static int WaitAny(WaitHandle[] waitHandles)
```

Summary

Waits for any of the elements in the specified array to receive a signal.

Parameters

Parameter	Description
waitHandles	A `System.Threading.WaitHandle` array containing the objects for which the current instance will wait. This array cannot contain multiple references to the same object (duplicates).

Return Value

Returns a `System.Int32` set to the index of the element in *waitHandles* that received a signal.

The maximum number of objects specified in the *waitHandles* array is system defined.

Exceptions

Exception	Condition
System.ArgumentNullException	*waitHandles* is `null` or one or more elements in the *waitHandles* array is `null`.
System.DuplicateWaitObjectException	*waitHandles* contains elements that are duplicates.
System.NotSupportedException	The number of objects in *waitHandles* is greater than the system permits.

WaitHandle.WaitOne() Method

```
[ILAsm]
.method public hidebysig virtual bool WaitOne()

[C#]
public virtual bool WaitOne()
```

Summary

Blocks the current thread until the current instance receives a signal.

Return Value

Returns `true` when the current instance receives a signal.

Behaviors

The caller of this method blocks indefinitely until a signal is received by the current instance.

How and When to Override

Override this method to customize the behavior of types derived from `System.Threading.WaitHandle`.

Usage

Use this method to block until a `WaitHandle` receives a signal from another thread, such as is generated when an asynchronous operation completes. For more information, see the `System.IAsyncResult` interface.

Exceptions

Exception	Condition
System.ObjectDisposedException	The current instance has already been disposed.