

1 System.Threading.WaitHandle Class

```
2 [ILAsm]  
3 .class public abstract WaitHandle extends System.MarshalByRefObject  
4 implements System.IDisposable  
  
5 [C#]  
6 public abstract class WaitHandle: MarshalByRefObject, IDisposable
```

7 Assembly Info:

- 8 • *Name:* mscorlib
- 9 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- 10 • *Version:* 2.0.x.x
- 11 • *Attributes:*
 - 12 ○ CLSCompliantAttribute(true)

13 Implements:

- 14 • **System.IDisposable**

15 Summary

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

18 Inherits From: System.MarshalByRefObject

19
20 **Library:** BCL

21
22 **Thread Safety:** All public static members of this type are safe for multithreaded operations.
23 No instance members are guaranteed to be thread safe.

24 25 Description

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

31
32 The static methods of this class are used to block a `System.Threading.Thread` until one
33 or more synchronization objects receive a signal.

34
35]

36

1 WaitHandle() Constructor

```
2 [ILAsm]  
3 public rtspecialname specialname instance void .ctor()  
4 [C#]  
5 public WaitHandle()
```

6 Summary

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

8

1 WaitHandle.Close() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual void Close()  
4 [C#]  
5 public virtual void Close()
```

6 Summary

7 Releases all resources held by the current instance.

8 Description

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

11 Behaviors

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

15

16 Default

17 As described above.

18

19 How and When to Override

20 Override this property to release resources allocated in subclasses.

21

22 Usage

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

25

26

1 WaitHandle.Dispose(System.Boolean)

2 Method

```
3 [ILAsm]  
4 .method family hidebysig virtual void Dispose(bool explicitDisposing)  
5 [C#]  
6 protected virtual void Dispose(bool explicitDisposing)
```

7 Summary

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

10 Parameters

Parameter	Description
<i>explicitDisposing</i>	true to release both managed and unmanaged resources; false to release only unmanaged resources.

11

12 Behaviors

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

17

18 How and When to Override

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

23

24 Usage

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

1

2

1 WaitHandle.Finalize() Method

```
2 [ILAsm]  
3 .method family hidebysig virtual void Finalize()  
4 [C#]  
5 ~WaitHandle()
```

6 Summary

7 Releases the resources held by the current instance.

8 Description

9 [Note: Application code does not call this method; it is automatically invoked during
10 garbage collection unless finalization by the garbage collector has been disabled. For
11 more information, see `System.GC.SuppressFinalize`, and `System.Object.Finalize`.
12
13 This method overrides `System.Object.Finalize`.
14
15]

16

1 WaitHandle.System.IDisposable.Dispose() 2 Method

```
3 [ILAsm]  
4 .method private final hidebysig virtual void System.IDisposable.Dispose()  
5 [C#]  
6 void IDisposable.Dispose()
```

7 Summary

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

10

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
<i>waitHandles</i>	A <code>System.Threading.WaitHandle</code> 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	<i>waitHandles</i> is null or one or more elements in the <i>waitHandles</i> array is null.
System.DuplicateWaitObjectException	<i>waitHandles</i> contains elements that are duplicates.
System.NotSupportedException	The number of objects in <i>waitHandles</i> is greater than the system permits.

19

20

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

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
<i>waitHandles</i>	A <code>System.Threading.WaitHandle</code> 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	<i>waitHandles</i> is null or one or more elements in the <i>waitHandles</i> array is null.
System.DuplicateWaitObjectException	<i>waitHandles</i> contains elements that are duplicates.
System.NotSupportedException	The number of objects in <i>waitHandles</i> is greater than the system permits.

1

2

1 WaitHandle.WaitOne() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual bool WaitOne()  
4 [C#]  
5 public virtual bool WaitOne()
```

6 Summary

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

8 Return Value

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

10 Behaviors

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

13

14 How and When to Override

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

17

18 Usage

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

22

23 Exceptions

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

24

25