System.Threading.EventWaitHandle Class

[ILAsm]
```
.class public beforefieldinit EventWaitHandle extends
System.Threading.WaitHandle
```

[C#]
```
public class EventWaitHandle: System.Threading.WaitHandle
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 4.0.0.0
- *Attributes:*
    - CLSCompliantAttribute(true)

Summary

Represents a thread synchronization event.

Inherits From: System.Threading.WaitHandle

Library: BCL

Description

The `System.Threading.EventWaitHandle` class allows threads to communicate with each other by signaling. Typically, one or more threads block on an `System.Threading.EventWaitHandle` until an unblocked thread calls the `System.Threading.EventWaitHandle.Set` method, releasing one or more of the blocked threads. A thread can signal an `System.Threading.EventWaitHandle` and then block on it, as an atomic operation, by calling the `staticSystem.Threading.WaitHandle.SignalAndWait` method.

The behavior of an `System.Threading.EventWaitHandle` that has been signaled depends on its reset mode. An `System.Threading.EventWaitHandle` created with the `System.Threading.EventResetMode.AutoReset` flag resets automatically when signaled, after releasing a single waiting thread. An `System.Threading.EventWaitHandle` created with the `System.Threading.EventResetMode.ManualReset` flag remains signaled until its `System.Threading.EventWaitHandle.Reset` method is called.

Automatic reset events provide exclusive access to a resource. If an automatic reset event is signaled when no threads are waiting, it remains signaled until a thread attempts to wait on it. The event releases the thread and immediately resets, blocking subsequent threads.

Manual reset events are like gates. When the event is not signaled, threads that wait on it will block. When the event is signaled, all waiting threads are released, and the event remains signaled (that is, subsequent waits do not block) until its `System.Threading.EventWaitHandle.Reset` method is called. Manual reset events are useful when one thread must complete an activity before other threads can proceed.

`System.Threading.EventWaitHandle` objects can be used with the `staticSystem.Threading.WaitHandle.WaitAll` and `System.Threading.WaitHandle.WaitAny` methods.

EventWaitHandle(System.Boolean, System.Threading.EventResetMode) Constructor

```
[ILAsm]
.method public hidebysig specialname rtspecialname instance void
.ctor(bool initialState, valuetype System.Threading.EventResetMode mode)
cil managed

[C#]
public EventWaitHandle (bool initialState, System.Threading.EventResetMode
mode)
```

Summary

Initializes a new instance of the `System.Threading.EventWaitHandle` class, specifying whether the wait handle is initially signaled, and whether it resets automatically or manually.

Parameters

Parameter	Description
initialState	`true` to set the initial state to signaled; `false` to set it to nonsignaled.
mode	One of the `System.Threading.EventResetMode` values that determines whether the event resets automatically or manually.

Description

If the initial state of the event is nonsignaled, threads that wait on the event will block. If the initial state is signaled, and the `System.Threading.EventResetMode.ManualReset` flag is specified for *mode*, threads that wait on the event will not block. If the initial state is signaled, and *mode* is `System.Threading.EventResetMode.AutoReset`, the first thread that waits on the event will be released immediately, after which the event will reset, and subsequent threads will block.

EventWaitHandle.Reset() Method

```
[ILAsm]
.method public hidebysig instance bool Reset() cil managed

[C#]
public bool Reset ()
```

Summary

Sets the state of the event to nonsignaled, causing threads to block.

Return Value

`true` if the operation succeeds; otherwise, `false`.

EventWaitHandle.Set() Method

```
[ILAsm]
.method public hidebysig instance bool Set() cil managed

[C#]
public bool Set ()
```

Summary

Sets the state of the event to signaled, allowing one or more waiting threads to proceed.

Return Value

`true` if the operation succeeds; otherwise, `false`.

Description

For an `System.Threading.EventWaitHandle` with `System.Threading.EventResetMode.AutoReset` (including `System.Threading.AutoResetEvent`), the `System.Threading.EventWaitHandle.Set` method releases a single thread. If there are no waiting threads, the wait handle remains signaled until a thread attempts to wait on it, or until its `System.Threading.EventWaitHandle.Reset` method is called.

[*Note:* There is no guarantee that every call to the `System.Threading.EventWaitHandle.Set` method will release a thread from an `System.Threading.EventWaitHandle` whose reset mode is `System.Threading.EventResetMode.AutoReset`. If two calls are too close together, so that the second call occurs before a thread has been released, only one thread is released. It is as if the second call did not happen. Also, if `System.Threading.EventWaitHandle.Set` is called when there are no threads waiting and the `System.Threading.EventWaitHandle` is already signaled, the call has no effect.

]

For an `System.Threading.EventWaitHandle` with `System.Threading.EventResetMode.ManualReset` (including `System.Threading.ManualResetEvent`), calling the `System.Threading.EventWaitHandle.Set` method leaves the wait handle in a signaled state until its `System.Threading.EventWaitHandle.Reset` method is called.