System.Threading.ThreadAbortException Class

```
[ILAsm]
.class public sealed serializable ThreadAbortException extends
System.SystemException

[C#]
public sealed class ThreadAbortException: SystemException
```

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

Thrown by the system when a call is made to `System.Threading.Thread.Abort`.

Inherits From: System.SystemException

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

Instances of this exception type can only be created by the system.

When a call is made to `System.Threading.Thread.Abort` to terminate a thread, the system throws a `System.Threading.ThreadAbortException` in the target thread. `System.Threading.ThreadAbortException` is a special exception that can be caught by application code, but is rethrown at the end of the catch block unless `System.Threading.Thread.ResetAbort` is called. When the `ThreadAbortException` exception is raised, the system executes any `finally` blocks for the target thread. The finally blocks are executed even if `System.Threading.Thread.ResetAbort` is called. If the abort is successful, the target thread is left in the `System.Threading.ThreadState.Stopped` and `System.Threading.ThreadState.Aborted` states.

Example

The following example demonstrates aborting a thread. The thread that receives the `System.Threading.ThreadAbortException` uses the

`System.Threading.Thread.ResetAbort` method to cancel the abort request and continue executing.

[C#]

```
using System;
using System.Threading;
using System.Security.Permissions;

public class ThreadWork {
  public static void DoWork() {
    try {
      for (int i=0; i<100; i++) {
        Console.WriteLine("Thread - working.");
        Thread.Sleep(100);
      }
    }
    catch (ThreadAbortException e) {
      Console.WriteLine("Thread - caught ThreadAbortException - resetting.");
      Thread.ResetAbort();
    }
    Console.WriteLine("Thread - still alive and working.");
    Thread.Sleep(1000);
    Console.WriteLine("Thread - finished working.");
  }
}

class ThreadAbortTest{
  public static void Main() {
    ThreadStart myThreadDelegate = new ThreadStart(ThreadWork.DoWork);
    Thread myThread = new Thread(myThreadDelegate);
    myThread.Start();
    Thread.Sleep(100);
    Console.WriteLine("Main - aborting my thread.");
    myThread.Abort();
    myThread.Join();
    Console.WriteLine("Main ending.");
  }
}
```

The output is

```
Thread - working.


Main - aborting my thread.


Thread - caught ThreadAbortException - resetting.


Thread - still alive and working.
Thread - finished working.


Main ending.


```

ThreadAbortException.ExceptionState Property

[ILAsm]
```
.property object ExceptionState { public hidebysig specialname instance
object get_ExceptionState() }
```

[C#]
```
public object ExceptionState { get; }
```

Summary

Gets an object that contains application-specific information related to the thread abort.

Property Value

A `System.Object`.

Description

This property is read-only.

The object returned by this property is specified via the *stateInfo* parameter of `System.Threading.Thread.Abort`. This property returns `null` if no object was specified, or the `System.Threading.Thread.Abort` method with no parameters was called. The exact content and usage of this object is application-defined; it is typically used to convey information that is meaningful to the thread being aborted.