System.Threading.Parallel.ParallelFor Class

```
[ILAsm]
.class public sealed serializable ParallelFor extends
System.Threading.Parallel.ParallelLoop<int32>

[C#]
public sealed class ParallelFor: ParallelLoop<int>
```

Assembly Info:

- *Name:* System.Threading.Parallel
- *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

A parallel loop over consecutive integers, starting at 0

Inherits From: System.Threading.Parallel.ParallelLoop<int>

Library: Parallel

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:* ParallelFor provides basic parallelism over an index space known in advance. The index space is 0..(N-1) for some value of N. This is the common case in -for- loops, and one can easily derive more complex arithmetic sequences via linear transformation of the index variable.]

ParallelFor(System.Int32) Constructor

[ILAsm]
```
public rtspecialname specialname instance void .ctor(int32 count)
```

[C#]
```
public ParallelFor(int count)
```

Summary

Constructs a `System.Threading.Parallel.ParallelFor` that will iterate over the integers 0..count-1.

Parameters

Parameter	Description
count	number of loop iterations

Description

The loop starts executing when method `System.Threading.Parallel.ParallelFor.BeginRun` is called.

ParallelFor(System.Int32, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int32 count, int32 numThreads)

[C#]
public ParallelFor(int count, int numThreads)
```

Summary

Constructs a `System.Threading.Parallel.ParallelFor` that will iterate over the integers 0..count-1.

Parameters

Parameter	Description
count	number of loop iterations
numThreads	maximum number of threads to use

Description

The loop starts executing when method `System.Threading.Parallel.ParallelFor.BeginRun` is called.

If numThreads is 0, then up to `System.Threading.Parallel.ParallelEnvironment.MaxThreads` threads are used instead. The value of numThreads includes the thread that created the `System.Threading.Parallel.ParallelFor<T>`, hence using numThreads=1 forces sequential execution.

Exceptions

Exception	Condition
System.ArgumentException	The value for numThreads is negative

ParallelFor.BeginRun(System.Action<T>) Method

```
[ILAsm]
.method public hidebysig override void BeginRun(class System.Action<!0>
action)

[C#]
public override void BeginRun(Action<T> action)
```

Summary

Begin executing iterations.

Parameters

Parameter	Description
action	The `System.Delegate` that processes each work item.

Description

This method is not thread safe. It should be called only once for a given instance of a `System.Threading.Parallel.ParallelFor`.

[*Note:* Implementations, particularly on single-threaded hardware, are free to employ the calling thread to execute all loop iterations.]

Exceptions

Exception	Condition
System.ArgumentNullException	*action* is `null`.

ParallelFor.Cancel() Method

[ILAsm]
```
.method public hidebysig override void Cancel()
```

[C#]
```
public override void Cancel()
```

Summary

Cancel any iterations that have not yet started

Description

This method is safe to call concurrently on the same instance.