

# System.Threading.Parallel.ParallelForEach<T> Class

```
[ILAsm]
.class public sealed serializable ParallelForEach<T> extends
System.Threading.Parallel.ParallelLoop<!0>

[C#]
public sealed class ParallelForEach<T>: ParallelLoop<T>
```

## 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 a collection containing types of T.

## Inherits From: System.Threading.Parallel.ParallelLoop<T>

**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

A `System.Threading.Parallel.ParallelForEach<T>` iterates over an enumerable collection. Method `System.Threading.Parallel.ParallelForEach<T>.BeginRun` activates processing of the iterations, using a callback provided. The collection shall not change while the `System.Threading.Parallel.ParallelForEach<T>` is active, otherwise the behavior is undefined. Inherited method `System.Threading.Parallel.ParallelLoop<T>.EndRun` blocks until all iterations are finished. Inherited method `System.Threading.Parallel.ParallelLoop<T>.Run` is shorthand for `System.Threading.Parallel.ParallelForEach<T>.BeginRun` and `System.Threading.Parallel.ParallelLoop<T>.EndRun`.

[*Note:* `System.Threading.Parallel.ParallelForEach<T>` is generally none-scalable in terms of parallelism, because the enumerator is inherently sequential. If the collection allows random access, consider using class `System.Threading.Parallel.ParallelFor` instead.]

# ParallelForEach<T> (System.I Enumerable<T> ) Constructor

```
[ILAsm]  
public rtspecialname specialname instance void .ctor(class  
System.Collections.Generic.IEnumerable<T> collection)  
  
[C#]  
public ParallelForEach(IEnumerable<T> collection)
```

## Summary

Constructs a System.Threading.Parallel.ParallelForEach<T> for iterating over a collection.

## Parameters

Parameter	Description
<i>collection</i>	collection of values over which to iterate

## Description

The loop does not start executing until at least method System.Threading.Parallel.ParallelForEach<T>.BeginRun is called and possibly not until method System.Threading.Parallel.ParallelLoop<T>.EndRun is called.

# ParallelForEach<T> (System.I Enumerable<T> , System.Int32) Constructor

```
[ILAsm]  
public rtspecialname specialname instance void .ctor(class  
System.Collections.Generic.IEnumerable<T> collection, int32 numThreads)  
  
[C#]  
public ParallelForEach(IEnumerable<T> collection, int numThreads)
```

## Summary

Constructs a `System.Threading.Parallel.ParallelForEach<T>` for iterating over a collection.

## Parameters

Parameter	Description
<i>collection</i>	collection of values over which to iterate
<i>numThreads</i>	maximum number of threads to use

## Description

The loop does not start executing until at least method `System.Threading.Parallel.ParallelForEach<T>.BeginRun` is called and possibly not until method `System.Threading.Parallel.ParallelLoop<T>.EndRun` is called.

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

## Exceptions

Exception	Condition
<b>System.ArgumentException</b>	The value for <code>numThreads</code> is negative

# ParallelForEach<T>.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
<i>action</i>	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.ParallelWhile<T>.

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

## Exceptions

Exception	Condition
System.ArgumentNullException	<i>action</i> is null.

## 1 ParallelForEach<T>.Cancel() Method

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

### 6 Summary

7 Cancel any iterations that have not yet started

### 8 Description

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

10  
11 Does not cancel any future iterations that might be added.