

1 System.Collections.Generic.IEnumerator<T>

2 Interface

```
3 [ILAsm]
4 .class interface public abstract IEnumerator`1<T> implements
5 System.IDisposable, System.Collections.IEnumerator
6 [C#]
7 public interface IEnumerator<T>: IDisposable, IEnumerator
```

8 Assembly Info:

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

14 Implements:

- 15 • **System.IDisposable**
- 16 • **System.Collections.IEnumerator**

17 Summary

18 Implemented by generic classes that support a simple iteration over a collection.

19 **Library**: BCL

21 Description

22 Enumerators can be used to read the data in the collection, but they cannot be used to
23 modify the underlying collection.

24
25 Initially, the enumerator is positioned before the first element in the collection. At this
26 position, calling `System.Collections.Generic.IEnumerator<T>.Current` is unspecified.
27 Therefore, you must call `System.Collections.IEnumerator.MoveNext` to advance the
28 enumerator to the first element of the collection before reading the value of
29 `System.Collections.Generic.IEnumerator<T>.Current`.

30
31 `System.Collections.Generic.IEnumerator<T>.Current` returns the same object until
32 `System.Collections.IEnumerator.MoveNext` is called.
33 `System.Collections.IEnumerator.MoveNext` sets
34 `System.Collections.Generic.IEnumerator<T>.Current` to the next element.

35
36 If `System.Collections.IEnumerator.MoveNext` passes the end of the collection, the
37 enumerator is positioned after the last element in the collection and
38 `System.Collections.IEnumerator.MoveNext` returns false. When the enumerator is at
39 this position, subsequent calls to `System.Collections.IEnumerator.MoveNext` also

1 return false. If the last call to `System.Collections.IEnumerator.MoveNext` returned
2 false, calling `System.Collections.Generic.IEnumerator<T>.Current` is unspecified.
3 You cannot set `System.Collections.Generic.IEnumerator<T>.Current` to the first
4 element of the collection again; you must create a new enumerator instance instead.
5

6 An enumerator remains valid as long as the collection remains unchanged and the
7 enumerator is not disposed. If changes are made to the collection, such as adding,
8 modifying, or deleting elements, the enumerator is irrecoverably invalidated and its
9 behavior is unspecified.
10

11 The enumerator does not have exclusive access to the collection; therefore,
12 enumerating through a collection is intrinsically not a thread-safe procedure. To
13 guarantee thread safety during enumeration, you can lock the collection during the
14 entire enumeration. To allow the collection to be accessed by multiple threads for
15 reading and writing, you must implement your own synchronization.
16

17 Default implementations of collections in `System.Collections.Generic` are not
18 synchronized.
19

20 [Note: Implementing this interface requires implementing the non-generic interface
21 `System.Collections.IEnumerator`. The methods `MoveNext`, `Reset` and `Dispose` do not
22 depend on the type parameter `T`, and appear only on the non-generic interface
23 `System.Collections.IEnumerator`. The property `Current` appears on both interfaces,
24 but with different return types. Implementations should provide the non-generic
25 `Current` property as an explicit interface member implementation. This allows any
26 consumer of the non-generic interface to consume the generic interface.]

27

1 IEnumarator<T>.Current Property

```
2 [ILAsm]  
3 .property !0 Current { public hidebysig virtual abstract specialname !0  
4 get_Current() }  
5 [C#]  
6 T Current { get; }
```

7 Summary

8 Gets the element in the collection over which the current instance is positioned.

9 Property Value

10 The element in the collection over which the current instance is positioned.

11 Description

12 `System.Collections.Generic.IEnumarator<T>.Current` is unspecified after any of the
13 following conditions:

- 14 • The enumerator is positioned before the first element in the collection, immediately
15 after the enumerator is created. `System.Collections.IEnumarator.MoveNext` must
16 be called to advance the enumerator to the first element of the collection before
17 reading the value of `System.Collections.Generic.IEnumarator<T>.Current`.
- 18 • The last call to `System.Collections.IEnumarator.MoveNext` returned `false`, which
19 indicates the end of the collection.
- 20 • The enumerator is invalidated due to changes made in the collection, such as adding,
21 repositioning, or deleting elements.
- 22 • If it has been disposed.

23 If `System.Collections.Generic.IEnumarator<T>.Current` is accessed when its value is
24 unspecified, an exception of unspecified type can be, but need not be, thrown.

25
26 `System.Collections.Generic.IEnumarator<T>.Current` returns the same object until
27 `System.Collections.IEnumarator.MoveNext` is called.
28 `System.Collections.IEnumarator.MoveNext` sets
29 `System.Collections.Generic.IEnumarator<T>.Current` to the next element.

30
31 This property is read-only.

32 Exceptions

Exception	Condition
An unspecified exception type	If <code>System.Collections.IEnumerator.MoveNext</code> is not called before the first call to <code>System.Collections.Generic.IEnumerator<T>.Current</code> . -or- If the previous call to <code>System.Collections.IEnumerator.MoveNext</code> returned <code>false</code> , indicating the end of the collection.

1

2