System.Collections.Generic.IEnumerator<T> Interface

```
[ILAsm]
.class interface public abstract IEnumerator`1<T> implements
System.IDisposable, System.Collections.IEnumerator

[C#]
public interface IEnumerator<T>: IDisposable, IEnumerator
```

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)

Implements:

- **System.IDisposable**
- **System.Collections.IEnumerator**

Summary

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

Library: BCL

Description

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

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

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

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

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

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

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

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

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

IEnumerator<T>.Current Property

```
[ILAsm]
.property !0 Current { public hidebysig virtual abstract specialname !0
get_Current() }

[C#]
T Current { get; }
```

Summary

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

Property Value

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

Description

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

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

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

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

This property is read-only.

Exceptions

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

1
2