

# System.Collections.IList Interface

```
[ILASM]
.class interface public abstract IList implements
System.Collections.ICollection,
System.Collections.IEnumerable

[C#]
public interface IList: ICollection, IEnumerable
```

## Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- *Version:* 1.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

## Type Attributes:

- DefaultMemberAttribute("Item") [*Note:* This attribute requires the RuntimeInfrastructure library.]

## Implements:

- **System.Collections.ICollection**
- **System.Collections.IEnumerable**

## Summary

Implemented by classes that support a collection of objects that can be individually indexed.

**Library:** BCL

## Description

[*Note:* **System.Collections.IList** implementations fall into three categories: read-only, fixed-size, variable-size. A read-only list cannot be modified. A fixed-size list allows the modification of existing elements, but does not allow the addition or removal of elements. A variable-size list allows the modification, addition, and removal of elements.]

# 1 IList.Add(System.Object) Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract int32 Add(object  
4 value)  
5 [C#]  
6 int Add(object value)
```

## 7 Summary

8 Adds an item to the current instance.

## 9 Parameters

10  
11

Parameter	Description
<i>value</i>	The <b>System.Object</b> to add to the current instance.

12

## 13 Return Value

14

15 A **System.Int32** containing the index of the current instance into  
16 which the new element was inserted.

## 17 Behaviors

18 As described above.

## 19 Usage

20 Use the **System.Collections.IList.Add** method to add another  
21 element to the current instance. The index into which that element is  
22 added is implementation-dependent.

## 23 Exceptions

24  
25

Exception	Condition
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

26  
27  
28

# 1 IList.Clear() Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract void Clear()  
4 [C#]  
5 void Clear()
```

## 6 Summary

7 Removes all items from the current instance.

## 8 Behaviors

9 As described above.

## 10 How and When to Override

11 Implementations of this method can vary in how a call to this method  
12 affects the capacity of a list. Typically, the count is set to zero. The  
13 capacity can be set to zero, some default, or remain unchanged.

## 14 Usage

15 Use this method to delete all values from the current instance.

## 16 Exceptions

17  
18

Exception	Condition
System.NotSupportedException	The current instance is read-only.

19  
20  
21

# 1 IList.Contains(System.Object) Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract bool  
4 Contains(object value)  
5 [C#]  
6 bool Contains(object value)
```

## 7 Summary

8 Determines whether the current instance contains a specific value.

## 9 Parameters

10  
11

Parameter	Description
<i>value</i>	The <b>System.Object</b> to locate in the current instance.

12  
13  
14

## 13 Return Value

15 **true** if the **System.Object** is found in the current instance; otherwise,  
16 **false**.

## 17 Behaviors

18 As described above.

## 19 Usage

20 Use the **System.Collections.IList.Contains** method to determine if a  
21 particular **System.Object** is an element of the current instance.

22

# 1 IList.IndexOf(System.Object) Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract int32  
4 IndexOf(object value)  
5  
6 [C#]  
7 int IndexOf(object value)
```

## 7 Summary

8 Determines the index of a specific item in the current instance.

## 9 Parameters

10  
11

Parameter	Description
<i>value</i>	The <b>System.Object</b> to locate in the current instance.

12

## 13 Return Value

14

15 The index of *value* if found in the current instance; otherwise, -1.

## 16 Behaviors

17 As described above.

## 18 How and When to Override

19 The default implementations of this method use  
20 **System.Object.Equals** to search for value in the current instance.

## 21 Usage

22 Use **System.Collections.IList.IndexOf** to determine if a  
23 **System.Object** is contained in the current instance and, if it is  
24 contained, its index in the current instance.

25

# 1 IList.Insert(System.Int32, System.Object) 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract void Insert(int32  
5 index, object value)  
  
6 [C#]  
7 void Insert(int index, object value)
```

## 8 Summary

9 Inserts an item to the current instance at the specified position.

## 10 Parameters

11  
12

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index at which <i>value</i> is inserted.
<i>value</i>	The <b>System.Object</b> to insert into the current instance.

13  
14

## 15 Behaviors

16 If *index* equals the number of items in the **System.Collections.IList**,  
17 then *value* is required to be appended to the end of the current  
18 instance.

## 19 Usage

20 Use **System.Collections.IList.Insert** to place a new element into a  
21 specific position in the current instance.

## 22 Exceptions

23  
24

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> is not a valid index in the current instance (i.e. is greater than the number of elements in the current instance).
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

1  
2  
3

# 1 IList.Remove(System.Object) Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract void  
4 Remove(object value)  
5  
6 [C#]  
7 void Remove(object value)
```

## 7 Summary

8 Removes the first occurrence of a specified **System.Object** from the  
9 current instance.

## 10 Parameters

Parameter	Description
<i>value</i>	The <b>System.Object</b> to remove from the current instance.

13  
14

## 15 Behaviors

16 As described above.

17  
18 In addition, if *value* is **null** or is not found in the current instance, it is  
19 required that no exception be thrown and the current instance remain  
20 unchanged.

## 21 How and When to Override

22 The default implementations of this method use  
23 **System.Object.Equals** to search for value in the current instance.

## 24 Usage

25 Use **System.Collections.IList.Remove** to delete a specified  
26 **System.Object** from the current instance.

## 27 Exceptions

28  
29

Exception	Condition
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

1  
2  
3

# 1 IList.RemoveAt(System.Int32) Method

```
2 [ILASM]  
3 .method public hidebysig virtual abstract void  
4 RemoveAt(int32 index)  
5  
6 [C#]  
7 void RemoveAt(int index)
```

## 7 Summary

8 Removes the item at the specified index of the current instance.

## 9 Parameters

10  
11

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index of the item to remove.

12  
13

## 14 Behaviors

15 As described above.

## 16 Usage

17 Use **System.Collections.IList.RemoveAt** to delete a specified  
18 **System.Object** from the current instance.

## 19 Exceptions

20  
21

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> is not a valid index in current instance.
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

22  
23  
24

# 1 IList.IsFixedSize Property

```
2 [ILASM]
3 .property bool IsFixedSize { public hidebysig virtual
4 abstract specialname bool get_IsFixedSize() }
5
6 [C#]
7 bool IsFixedSize { get; }
```

## 7 Summary

8 Gets a **System.Boolean** value indicating whether the current instance  
9 has a fixed size.

## 10 Property Value

11

12 **true** if the current instance has a fixed size; otherwise, **false**.

## 13 Description

14 This property is read-only.

15

16 [*Note:* A collection with a fixed size does not allow the addition or  
17 removal of elements, but it allows the modification of existing  
18 elements.]

## 19 Behaviors

20 Any method that adds or removes an element of a collection is  
21 required to check the value of this property for the particular collection  
22 before adding or removing elements. If the value of this property is  
23 **false**, any attempt to add or remove an element of the current  
24 instance is required to throw a **System.NotSupportedException**.

## 25 Default

26 The default of this property is **false**.

## 27 How and When to Override

28 Override this property, setting the value to **true**, in order to prevent  
29 the addition or removal of elements in the current instance.

## 30 Usage

31 Use **System.Collections.IList.IsFixedSize** to secure the current  
32 instance from modification from methods, such as  
33 **System.Collections.IList.Add** and

1      **System.Collections.IList.Remove**, which add or remove elements  
2      from a list.

3

# 1 IList.IsReadOnly Property

```
2 [ILASM]
3 .property bool IsReadOnly { public hidebysig virtual
4 abstract specialname bool get_IsReadOnly() }
5
6 [C#]
7 bool IsReadOnly { get; }
```

## 7 Summary

8 Gets a value indicating whether the current instance is read-only.

## 9 Property Value

10

11 **true** if the current instance is read-only; otherwise, **false**.

## 12 Description

13 This property is read-only.

14

15 [*Note:* A collection that is read-only does not allow the modification,  
16 addition, or removal of elements.]

## 17 Behaviors

18 Any method that modifies, adds, or removes an element of a collection  
19 is required to check the value of this property for the particular  
20 collection before executing. If the value of this property is **false**, any  
21 attempt to modify, add, or remove an element of the current instance  
22 is required to throw a **System.NotSupportedException**.

## 23 Default

24 The default of this property is **false**.

## 25 How and When to Override

26 Override this property, setting the value to **true**, in order to prevent  
27 the modification, addition, or removal of elements in the current  
28 instance.

## 29 Usage

30 Use **System.Collections.IList.IsReadOnly** to secure the current  
31 instance from modification from methods, such as  
32 **System.Collections.IList.Add** and  
33 **System.Collections.IList.Remove**, which modify, add, or remove  
34 elements from a list.



# 1 IList.Item Property

```
2 [ILASM]
3 .property object Item[int32 index] { public hidebysig
4 virtual abstract specialname object get_Item(int32 index)
5 public hidebysig virtual abstract specialname void
6 set_Item(int32 index, object value) }
7
8 [C#]
9 object this[int index] { get; set; }
```

## 9 Summary

10 Gets or sets the element at the specified index in the current instance.

## 11 Parameters

12  
13

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index of the element to get or set.

14  
15  
16

## 15 Property Value

17 The element at the specified index in the current instance.

## 18 Behaviors

19 As described above.

## 20 Usage

21 Use this property for subscript indexing for the current instance in the  
22 following form: `myCollection[index]`.

## 23 Exceptions

24  
25

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> is not a valid index in the current instance.
<b>System.NotSupportedException</b>	The property is being set and the current instance is read-only.

26  
27