

1 System.Collections.IList Interface

```
2 [ILAsm]  
3 .class interface public abstract IList implements  
4 System.Collections.ICollection, System.Collections.IEnumerable  
  
5 [C#]  
6 public interface IList: ICollection, IEnumerable
```

7 Assembly Info:

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

13 Type Attributes:

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

16 Implements:

- 17 • **System.Collections.ICollection**
- 18 • **System.Collections.IEnumerable**

19 Summary

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

22 **Library:** BCL

23

24 Description

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

30

31

32

1 IList.Add(System.Object) Method

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

6 Summary

7 Adds an item to the current instance.

8 Parameters

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

9 Return Value

11 A `System.Int32` containing the index of the current instance into which the new element
12 was inserted.

13 Behaviors

14 As described above.

16 Usage

17 Use the `System.Collections.IList.Add` method to add another element to the current
18 instance. The index into which that element is added is implementation-dependent.

20 Exceptions

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

21

22

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

11 How and When to Override

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

15

16 Usage

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

18

19 Exceptions

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

20

21

1 IList.Contains(System.Object) Method

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

6 Summary

7 Determines whether the current instance contains a specific value.

8 Parameters

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

9 Return Value

11 true if the System.Object is found in the current instance; otherwise, false.

12 Behaviors

13 As described above.

15 Usage

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

18
19

1 IList.IndexOf(System.Object) Method

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

6 Summary

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

8 Parameters

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

9 Return Value

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

12 Behaviors

13 As described above.

14

15 How and When to Override

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

18

19 Usage

20 Use `System.Collections.IList.IndexOf` to determine if a `System.Object` is contained
21 in the current instance and, if it is contained, its index in the current instance.

22

23

1 IList.Insert(System.Int32, System.Object)

2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual abstract void Insert(int32 index, object  
5 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

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

11

12 Behaviors

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

15

16 Usage

17 Use `System.Collections.IList.Insert` to place a new element into a specific position
18 in the current instance.

19

20 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<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).

System.NotSupportedException

The current instance is read-only or has a fixed size.

1

2

1 IList.Remove(System.Object) Method

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

6 Summary

7 Removes the first occurrence of a specified `System.Object` from the current instance.

8 Parameters

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

9

10 Behaviors

11 As described above.

12

13 In addition, if *value* is `null` or is not found in the current instance, it is required that no
14 exception be thrown and the current instance remain unchanged.

15 How and When to Override

16 The default implementations of this method use `System.Object.Equals` to search for
17 value in the current instance.

18

19 Usage

20 Use `System.Collections.IList.Remove` to delete a specified `System.Object` from the
21 current instance.

22

23 Exceptions

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

1

2

1 IList.RemoveAt(System.Int32) Method

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

6 Summary

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

8 Parameters

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

9

10 Behaviors

11 As described above.

12

13 Usage

14 Use System.Collections.IList.RemoveAt to delete a specified System.Object from
15 the current instance.

16

17 Exceptions

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

18

19

1 IList.IsFixedSize Property

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

7 Summary

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

9 Property Value

10 `true` if the current instance has a fixed size; otherwise, `false`.

11 Description

12 This property is read-only.

13
14 [*Note:* A collection with a fixed size does not allow the addition or removal of elements,
15 but it allows the modification of existing elements.]
16
17

18 Behaviors

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

23

24 Default

25 The default of this property is `false`.

26

27 How and When to Override

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

30

31 Usage

1 Use `System.Collections.IList.IsFixedSize` to secure the current instance from
2 modification from methods, such as `System.Collections.IList.Add` and
3 `System.Collections.IList.Remove`, which add or remove elements from a list.

4

5

1 IList.IsReadOnly Property

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

7 Summary

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

9 Property Value

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

11 Description

12 This property is read-only.

13
14 [*Note:* A collection that is read-only does not allow the modification, addition, or
15 removal of elements.]
16
17

18 Behaviors

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

23

24 Default

25 The default of this property is false.

26

27 How and When to Override

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

30

31 Usage

1 Use `System.Collections.IList.IsReadOnly` to secure the current instance from
2 modification from methods, such as `System.Collections.IList.Add` and
3 `System.Collections.IList.Remove`, which modify, add, or remove elements from a
4 list.

5

6

1 IList.Item Property

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

8 Summary

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

10 Parameters

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

11 12 Property Value

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

14 Behaviors

15 As described above.

17 Usage

18 Use this property for subscript indexing for the current instance in the following form:
19 myCollection[index].

21 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>index</i> is not a valid index in the current instance.

System.NotSupportedException

The property is being set and the current instance is read-only.

1

2