

# System.Collections.Hashtable Class

```
[ILASM]
.class public serializable Hashtable extends System.Object
implements System.ICloneable,
System.Collections.ICollection,
System.Collections.IDictionary,
System.Collections.IEnumerable

[C#]
public class Hashtable: ICloneable, ICollection,
IDictionary, 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.IDictionary**
- **System.Collections.ICollection**
- **System.Collections.IEnumerable**
- **System.ICloneable**

## Summary

Represents a hash table.

## Inherits From: System.Object

**Library:** BCL

**Thread Safety:** This class is safe for multiple readers and a single writer.

## Description

A **System.Collections.Hashtable** represents a dictionary with a constant lookup time that contains entries of associated keys and values. The type of each entry in a **System.Collections.Hashtable** is

1 **System.Collections.DictionaryEntry**. A statement that exposes  
2 each element in the collection is required to iterate over this type.  
3 [Note: See example.]  
4

5 Objects used as keys in a **System.Collections.Hashtable** is required  
6 to either implement both **System.Object.GetHashCode** and  
7 **System.Object.Equals** or neither. Furthermore, for a particular key,  
8 these methods are required to produce the same results when called  
9 with the same parameters while that key exists in a particular  
10 **System.Collections.Hashtable**. Keys cannot be mutated while they  
11 are used in the table.  
12

13 Every key in a **System.Collections.Hashtable** is required to be  
14 unique compared to every other key in the table. An object that  
15 implements **System.Collections.IComparer** can determine whether  
16 two keys are unequal. The default comparer for a key is the key's  
17 implementation of **System.Object.Equals**.  
18

19 Each value in a **System.Collections.Hashtable** is required to provide  
20 its own hash function, which can be accessed by calling  
21 **System.Collections.Hashtable.GetHash**. Alternatively, if an object  
22 that implements **System.Collections.IHashCodeProvider** is passed  
23 to a **System.Collections.Hashtable** constructor, the custom hash  
24 function provided by that object is used for every value in the table.  
25

26 [Note: The default capacity (i.e. the default number of entries that can  
27 be contained) of a **System.Collections.Hashtable** is zero.  
28

29 When an entry is added to the **System.Collections.Hashtable**, the  
30 entry is placed into a bucket based on the hash code obtained from  
31 the **System.Collections.IHashCodeProvider** implementation of the  
32 table, or the **System.Object.GetHashCode** if no specific  
33 **System.Collections.IHashCodeProvider** was provided. Subsequent  
34 lookups of the key use the hash code of the key to search in only one  
35 particular bucket, substantially reducing the number of key  
36 comparisons required to find an entry.  
37

38 As entries are added to a **System.Collections.Hashtable**, and the  
39 maximum capacity of the table is reached, the number of buckets in  
40 the table is automatically increased to the smallest prime number that  
41 is larger than twice the current number of buckets.  
42

43 A **System.Collections.Hashtable** can safely support one writer and  
44 multiple readers concurrently. To support multiple writers, all  
45 operations are required to be done through the wrapper returned by  
46 the **System.Collections.Hashtable.Synchronized** method.]

## 47 Example

48

49 The following example shows how to iterate over the elements of a  
50 **System.Collections.Hashtable**.

```
1
2     [C#]
3
4
5     foreach (DictionaryEntry myEntry in myHashtable)
6
7
```

# 1 Hashtable() Constructor

```
2 [ILASM]  
3 public rtspecialname specialname instance void .ctor()  
4 [C#]  
5 public Hashtable()
```

## 6 Summary

7 Constructs and initializes a new instance of the  
8 **System.Collections.Hashtable** class.

## 9 Description

10 The new instance is initialized with the default capacity,  
11 **System.Collections.IHashCodeProvider**, and  
12 **System.Collections.IComparer**.

13

# 1 Hashtable(System.Int32) Constructor

```
2 [ILASM]  
3 public rtspecialname specialname instance void .ctor(int32  
4 capacity)  
5  
6 [C#]  
7 public Hashtable(int capacity)
```

## 7 Summary

8 Constructs and initializes a new instance of the  
9 **System.Collections.Hashtable** class with the specified initial  
10 capacity.

## 11 Parameters

12  
13

Parameter	Description
<i>capacity</i>	A <b>System.Int32</b> that specifies the number of entries that the new <b>System.Collections.Hashtable</b> instance can initially contain.

14  
15

## 15 Description

16 The new instance is initialized with the default  
17 **System.Collections.IHashCodeProvider** and  
18 **System.Collections.IComparer**.

## 19 Exceptions

20  
21

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>capacity</i> < 0.

22  
23  
24

# 1 Hashtable(System.Collections.IHashCode 2 Provider, System.Collections.IComparer) 3 Constructor

```
4 [ILASM]  
5 public rtspecialname specialname instance void .ctor(class  
6 System.Collections.IHashCodeProvider hcp, class  
7 System.Collections.IComparer comparer)  
8  
9 [C#]  
10 public Hashtable(IHashCodeProvider hcp, IComparer comparer)
```

## 10 Summary

11 Constructs and initializes a new instance of the  
12 **System.Collections.Hashtable** class with the specified  
13 **System.Collections.IHashCodeProvider** and the specified  
14 **System.Collections.IComparer**.

## 15 Parameters

Parameter	Description
<i>hcp</i>	The <b>System.Collections.IHashCodeProvider</b> that supplies the hash codes for all keys in the <b>System.Collections.Hashtable</b> ; or, <b>null</b> to use the default hash code provider.
<i>comparer</i>	The <b>System.Collections.IComparer</b> to use to determine whether two keys are equal; or, <b>null</b> to use the default comparer.

## 18 19 Description

20 The new instance is initialized with the default capacity.

21

1 **Hashtable(System.Int32,**  
2 **System.Collections.IHashCodeProvider,**  
3 **System.Collections.IComparer)**  
4 **Constructor**

```
5 [ILASM]  
6 public rtspecialname specialname instance void .ctor(int32  
7 capacity, class System.Collections.IHashCodeProvider hcp,  
8 class System.Collections.IComparer comparer)
```

```
9 [C#]  
10 public Hashtable(int capacity, IHashCodeProvider hcp,  
11 IComparer comparer)
```

12 **Summary**

13 Constructs and initializes a new instance of the  
14 **System.Collections.Hashtable** class with the specified initial  
15 capacity, the specified **System.Collections.IHashCodeProvider**,  
16 and the specified **System.Collections.IComparer**.

17 **Parameters**

18  
19

Parameter	Description
<i>capacity</i>	A <b>System.Int32</b> that specifies the number of entries that the new <b>System.Collections.Hashtable</b> instance can initially contain.
<i>hcp</i>	The <b>System.Collections.IHashCodeProvider</b> that supplies the hash codes for all keys in the <b>System.Collections.Hashtable</b> ; or, <b>null</b> to use the default hash code provider.
<i>comparer</i>	The <b>System.Collections.IComparer</b> to use to determine whether two keys are equal, or <b>null</b> to use the default comparer.

20  
21  
22

# 1 Hashtable(System.Collections.IDictionary 2 ) Constructor

```
3 [ILASM]  
4 public rtspecialname specialname instance void .ctor(class  
5 System.Collections.IDictionary d)  
6  
7 [C#]  
8 public Hashtable(IDictionary d)
```

## 8 Summary

9 Constructs and initializes a new instance of the  
10 **System.Collections.Hashtable** class using the values of the  
11 specified **System.Collections.IDictionary**.

## 12 Parameters

13  
14

Parameter	Description
<i>d</i>	The <b>System.Collections.IDictionary</b> used to initialize the elements of the new instance.

15  
16

## 16 Description

17 The initial capacity of the new instance is set to the number of entries  
18 in *d*. The new instance is initialized with the default  
19 **System.Collections.IHashCodeProvider** and  
20 **System.Collections.IComparer**.

## 21 Exceptions

22  
23

Exception	Condition
<b>System.ArgumentNullException</b>	<i>d</i> is <b>null</b> .

24  
25  
26

1 **Hashtable(System.Collections.IDictionary,**  
2 **System.Collections.IHashCodeProvider,**  
3 **System.Collections.IComparer)**  
4 **Constructor**

```
5 [ILASM]  
6 public rtsspecialname specialname instance void .ctor(class  
7 System.Collections.IDictionary d, class  
8 System.Collections.IHashCodeProvider hcp, class  
9 System.Collections.IComparer comparer)  
10  
11 [C#]  
12 public Hashtable(IDictionary d, IHashCodeProvider hcp,  
13 IComparer comparer)
```

13 **Summary**

14 Constructs and initializes a new instance of the  
15 **System.Collections.Hashtable** class using the values of the  
16 specified **System.Collections.IDictionary**, the specified  
17 **System.Collections.IHashCodeProvider**, and the specified  
18 **System.Collections.IComparer**.

19 **Parameters**

Parameter	Description
<i>d</i>	The <b>System.Collections.IDictionary</b> used to initialize the elements of the new instance.
<i>hcp</i>	The <b>System.Collections.IHashCodeProvider</b> that supplies the hash codes for all keys in the new instance; or, <b>null</b> to use the default hash code provider.
<i>comparer</i>	The <b>System.Collections.IComparer</b> to use to determine whether two keys are equal in the new instance, or <b>null</b> to use the default comparer.

22  
23 **Description**

24 The initial capacity of the new instance is set to the number of entries  
25 in *d*.

26 **Exceptions**

Exception	Condition
-----------	-----------

1  
2  
3

<b>System.ArgumentNullException</b>	<i>d</i> is <b>null</b> .
-------------------------------------	---------------------------

# 1 Hashtable.Add(System.Object, 2 System.Object) Method

```
3 [ILASM]  
4 .method public hidebysig virtual void Add(object key,  
5 object value)  
  
6 [C#]  
7 public virtual void Add(object key, object value)
```

## 8 Summary

9 Adds an entry with the specified key and value into the current  
10 instance.

## 11 Parameters

12  
13

Parameter	Description
<i>key</i>	The key of the entry to add.
<i>value</i>	The value of the entry to add.

14  
15  
16  
17

## Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>key</i> is <b>null</b> .
<b>System.ArgumentException</b>	An entry with the same key already exists in the current instance.
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

18  
19  
20

# 1 Hashtable.Clear() Method

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

## 6 Summary

7 Removes all entries from the current instance.

## 8 Description

9 [Note: This method is implemented to support the  
10 **System.Collections.IDictionary** interface.]

## 11 Behaviors

12 As described above.

## 13 Default

14 The value of each key and value in the current instance is set to **null**.  
15 The **System.Collections.Hashtable.Count** property of the current  
16 instance is set to zero. The capacity of the current instance remains  
17 unchanged.

18  
19 If the current instance is empty, it remains unchanged and no  
20 exception is thrown.

## 21 Exceptions

22  
23

Exception	Condition
<b>System.NotSupportedException</b>	The current instance is read-only.

24  
25  
26

# 1 Hashtable.Clone() Method

```
2 [ILASM]  
3 .method public hidebysig virtual object Clone()  
4 [C#]  
5 public virtual object Clone()
```

## 6 Summary

7 Creates a **System.Object** that is a copy of the current instance.

## 8 Return Value

9

10 A **System.Object** that is a copy of the current instance.

## 11 Description

12 [*Note:* This method is implemented to support the  
13 **System.ICloneable** interface.]

## 14 Behaviors

15 As described above.

## 16 Default

17 This method creates a new **System.Collections.Hashtable** instance  
18 is initialized with the same count,  
19 **System.Collections.IHashCodeProvider** implementation, and  
20 **System.Collections.IComparer** implementation as the current  
21 instance. The references to the objects contained by the current  
22 instance are copied to the new instance.

23

# 1 Hashtable.Contains(System.Object) 2 Method

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

## 7 Summary

8 Determines whether the current instance contains the specified key.

## 9 Parameters

10  
11

Parameter	Description
key	The key to locate in the current instance.

12  
13  
14

## 13 Return Value

15 **true** if the current instance contains *key*; otherwise, **false**.

## 16 Description

17 [Note: This method is implemented to support the  
18 **System.Collections.IDictionary** interface.]

## 19 Behaviors

20 As described above.

## 21 Default

22 This method is equivalent to  
23 **System.Collections.Hashtable.ContainsKey**.

24  
25  
26

[Note: For the default implementation, this method has a constant (O(1)) lookup time.]

## 27 Exceptions

28  
29

Exception	Condition
<b>System.ArgumentNullException</b>	key is <b>null</b> .

1  
2  
3

# 1 Hashtable.ContainsKey(System.Object) 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual bool ContainsKey(object  
5 key)  
6 [C#]  
7 public virtual bool ContainsKey(object key)
```

## 8 Summary

9 Determines whether the current instance contains an entry with the  
10 specified key.

## 11 Parameters

12  
13

Parameter	Description
key	The key of the entry to locate in the current instance.

14  
15  
16

## 15 Return Value

17 **true** if the current instance contains an entry with *key*; otherwise,  
18 **false**.

## 19 Description

## 20 Behaviors

21 As described above.

## 22 Default

23 This method uses **System.Collections.Hashtable.KeyEquals** to  
24 compare *key* to the keys in the current instance.

25  
26  
27

[*Note:* For the default implementation, this method has a constant  
(O(1)) lookup time.]

## 28 Exceptions

29  
30

Exception	Condition
<b>System.ArgumentNullException</b>	<i>key</i> is <b>null</b> .

1  
2  
3

# 1 Hashtable.ContainsValue(System.Object)

## 2 Method

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

### 8 Summary

9 Determines whether the current instance contains an entry with the  
10 specified value.

### 11 Parameters

12  
13

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

14  
15  
16

### 15 Return Value

17 **true** if the current instance contains an entry with *value*; otherwise,  
18 **false**.

### 19 Description

20 [Note: This method is implemented to support the  
21 **System.Collections.IDictionary** interface.]

### 22 Behaviors

23 As described above.

### 24 Default

25 This method is equivalent to  
26 **System.Collections.Hashtable.ContainsKey**.

27  
28 [Note: For the default implementation, this method has a constant  
29 (O(1)) lookup time.]

30

# 1 Hashtable.CopyTo(System.Array, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual void CopyTo(class  
5 System.Array array, int32 arrayIndex)  
  
6 [C#]  
7 public virtual void CopyTo(Array array, int arrayIndex)
```

## 8 Summary

9 Copies the entries of the current instance to a one-dimensional  
10 **System.Array** starting at the specified index.

## 11 Parameters

12  
13

Parameter	Description
<i>array</i>	The one-dimensional, zero-indexed <b>System.Array</b> that is the destination of the objects copied from the current instance.
<i>arrayIndex</i>	A <b>System.Int32</b> that specifies the zero-based index in <i>array</i> at which copying begins. This value is between 0 and <i>array.Count</i> minus the <b>System.Collections.Hashtable.Count</b> of the current index, inclusive.

14  
15

## 16 Behaviors

17 As described above.

## 18 Default

19 The **System.Collections.DictionaryEntry** elements in the current  
20 instance are copied to the **System.Array** in the same order in which  
21 they are contained the current instance. If  
22 **System.Collections.DictionaryEntry** is not assignment-compatible  
23 with the type of *array*, a **System.InvalidCastException** is thrown. If  
24 an exception is thrown while copying, the state of the current instance  
25 is undefined.

## 26 Exceptions

27  
28

Exception	Condition
<b>System.ArgumentNullException</b>	<i>array</i> is <b>null</b> .
<b>System.ArgumentOutOfRangeException</b>	<i>arrayIndex</i> < 0.

1  
2  
3

<b>System.ArgumentException</b>	<i>array</i> has more than one dimension.  <i>arrayIndex</i> > <i>array.Count</i> - The <b>System.Collections.Hashtable.Count</b> of the current instance.
<b>System.InvalidCastException</b>	The type of the current instance is not assignment-compatible with the type of <i>array</i> .

# 1 Hashtable.GetEnumerator() Method

```
2 [ILASM]  
3 .method public hidebysig virtual class  
4 System.Collections.IDictionaryEnumerator GetEnumerator()  
5 [C#]  
6 public virtual IDictionaryEnumerator GetEnumerator()
```

## 7 Summary

8 Returns a **System.Collections.IDictionaryEnumerator** for the  
9 current instance.

## 10 Return Value

11

12 A **System.Collections.IDictionaryEnumerator** for the current  
13 instance.

## 14 Description

15 If the elements of the current instance are modified while an  
16 enumeration is in progress, a call to  
17 **System.Collections.IEnumerator.MoveNext** or  
18 **System.Collections.IEnumerator.Current** throws  
19 **System.InvalidOperationException**.

20  
21 [*Note:* For detailed information regarding the use of an enumerator,  
22 see **System.Collections.IEnumerator**.

23  
24 This property is implemented to support the  
25 **System.Collections.IList** interface.]

## 26 Behaviors

27 As described above.

28

# 1 Hashtable.GetHashCode(System.Object)

## 2 Method

```
3 [ILASM]  
4 .method family hidebysig virtual int32 GetHashCode(object key)  
5 [C#]  
6 protected virtual int GetHashCode(object key)
```

### 7 Summary

8 Generates a hash code for the specified key in the current instance.

### 9 Parameters

10  
11

Parameter	Description
key	The <b>System.Object</b> whose hash code is to be generated.

12  
13  
14

### Return Value

15 A **System.Int32** containing the hash code for *key*.

### 16 Description

17 This method is accessible only through this class or a derived class.

### 18 Behaviors

19 As described above.

### 20 Default

21 If the current instance was instantiated with a specific  
22 **System.Collections.IHashCodeProvider** implementation, this  
23 method uses that hash code provider; otherwise, it uses the  
24 **System.Object.GetHashCode** implementation of *key*.

### 25 Exceptions

26  
27

Exception	Condition
<b>System.ArgumentNullException</b>	<i>key</i> is <b>null</b> .

28  
29  
30

# 1 Hashtable.KeyEquals(System.Object, 2 System.Object) Method

```
3 [ILASM]  
4 .method family hidebysig virtual bool KeyEquals(object  
5 item, object key)  
  
6 [C#]  
7 protected virtual bool KeyEquals(object item, object key)
```

## 8 Summary

9 Determines whether the specified **System.Object** and the specified  
10 key in the current instance represent the same value.

## 11 Parameters

12  
13

Parameter	Description
<i>item</i>	The <b>System.Object</b> to compare with <i>key</i> .
<i>key</i>	The key in the current instance to compare with <i>item</i> .

14  
15  
16

## 15 Return Value

17 **true** if *item* and *key* represent the same value; otherwise, **false**.

## 18 Description

19 This method is accessible only through this class or a derived class.

## 20 Behaviors

21 As described above.

## 22 Default

23 If the current instance was initialized with a specified  
24 **System.Collections.IComparer** implementation, this method uses  
25 that implementation to perform the comparison; otherwise, the  
26 **System.Object.Equals** implementation of *item* is used.

## 27 Exceptions

28  
29

Exception	Condition
-----------	-----------

1  
2  
3

**System.ArgumentNullException**

*item* is **null**.

-or-

*key* is **null**.

# 1 Hashtable.Remove(System.Object)

## 2 Method

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

### 7 Summary

8 Removes the entry with the specified key from the current instance.

### 9 Parameters

10  
11

Parameter	Description
key	The key of the entry to remove.

12

### 13 Description

14 [Note: This method is implemented to support the  
15 **System.Collections.IDictionary** interface.]

### 16 Behaviors

17 As described above.

### 18 Default

19 This method uses the **System.Object.Equals** implementation of *key*  
20 to locate it in the current instance. If *key* is found in the current  
21 instance, the values of both *key* and its associated value are set to  
22 **null**. If *key* is not found in the current instance, no exception is thrown  
23 and the current instance remains unchanged.

### 24 Exceptions

25  
26

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

27

28

29

# 1 Hashtable.Synchronized(System.Collections. 2 ns.Hashtable) Method

```
3 [ILASM]  
4 .method public hidebysig static class  
5 System.Collections.Hashtable Synchronized(class  
6 System.Collections.Hashtable table)  
  
7 [C#]  
8 public static Hashtable Synchronized(Hashtable table)
```

## 9 Summary

10 Returns a synchronized (thread-safe) wrapper for the specified  
11 **System.Collections.Hashtable**.

## 12 Parameters

13  
14

Parameter	Description
<i>table</i>	The <b>System.Collections.Hashtable</b> to synchronize.

15  
16  
17

## 16 Return Value

18 A synchronized (thread-safe) wrapper for *table*.

## 19 Description

20 This method returns a new **System.Collections.Hashtable** instance  
21 that contains values equal to the values of *table*, and provides  
22 synchronized access to those values.

23  
24  
25  
26  
27

If more than one thread is to write to a  
**System.Collections.Hashtable** concurrently, all write operations are  
required to be done through this wrapper.

28 [Note: A **System.Collections.Hashtable** can safely support one  
29 writer and multiple readers concurrently.]

## 30 Exceptions

31  
32

Exception	Condition
<b>System.ArgumentNullException</b>	<i>table</i> is <b>null</b> .

33  
34  
35

# 1 Hashtable.System.Collections.IEnumerabl 2 e.GetEnumerator() Method

```
3 [ILASM]  
4 .method private final hidebysig virtual class  
5 System.Collections.IEnumerator  
6 System.Collections.IEnumerable.GetEnumerator()  
7  
8 [C#]  
9 IEnumerator IEnumerable.GetEnumerator()
```

## 9 Summary

10 Implemented to support the **System.Collections.IEnumerable**  
11 interface. [Note: For more information, see  
12 **System.Collections.IEnumerable.GetEnumerator.**]

13

# 1 Hashtable.Count Property

```
2 [ILASM]
3 .property int32 ICollection.Count { public hideby sig
4 virtual abstract specialname int32 get_ICollection.Count()
5 }
6
6 [C#]
7 int ICollection.Count { get; }
```

## 8 Summary

9 Implemented to support the **System.Collections.ICollection**  
10 interface. [Note: For more information, see  
11 **System.Collections.ICollection.Count.**]

12

# 1 Hashtable.Count Property

```
2 [ILASM]  
3 .property int32 Count { public hidebysig virtual  
4 specialname int32 get_Count() }  
5 [C#]  
6 public virtual int Count { get; }
```

## 7 Summary

8 Gets the number of key-and-value pairs contained in the current  
9 instance.

## 10 Property Value

11

12 A **System.Int32** that specifies the number of key-and-value pairs  
13 contained in the current instance.

## 14 Description

15 This property is read-only.

## 16 Behaviors

17 As described above.

18

# 1 Hashtable.IsFixedSize Property

```
2 [ILASM]
3 .property bool IDictionary.IsFixedSize { public hideby sig
4 virtual abstract specialname bool
5 get_IDictionary.IsFixedSize() }
6
7 [C#]
8 bool IDictionary.IsFixedSize { get; }
```

## 8 Summary

9 Implemented to support the **System.Collections.IDictionary**  
10 interface. [Note: For more information, see  
11 **System.Collections.IDictionary.IsFixedSize.**]

12

# 1 Hashtable.IsFixedSize Property

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

## 7 Summary

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

## 10 Property Value

11

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

## 13 Description

14 This property is a read-only.

15

16 [*Note:* Elements can be modified in, but not added to or removed from  
17 a **System.Collections.Hashtable** with a fixed size.]

## 18 Behaviors

19 As described above.

## 20 Default

21 The default value of this property is **false**.

## 22 How and When to Override

23 Override this property, setting it to **true**, to prevent addition or  
24 removal of entries in the current instance.

25

# 1 Hashtable.IsReadOnly Property

```
2 [ILASM]
3 .property bool IDictionary.IsReadOnly { public hideby sig
4 virtual abstract specialname bool
5 get_IDictionary.IsReadOnly() }
6
7 [C#]
8 bool IDictionary.IsReadOnly { get; }
```

## 8 Summary

9 Implemented to support the **System.Collections.IDictionary**  
10 interface. [Note: For more information, see  
11 **System.Collections.IDictionary.IsReadOnly.**]

12

# 1 Hashtable.IsReadOnly Property

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

## 7 Summary

8 Gets a **System.Boolean** value indicating whether the current instance  
9 is read-only.

## 10 Property Value

11

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

## 13 Description

14 This property is read-only.

15

16 [*Note:* Elements cannot be modified in, added to, or removed from a  
17 **System.Collections.Hashtable** that is read-only.]

## 18 Behaviors

19 As described above.

## 20 Default

21 The default value of this property is **false**.

## 22 How and When to Override

23 Override this property, setting it to **true**, in order to prevent the  
24 addition, removal, or modification of entries in the current instance.

25

# 1 Hashtable.IsSynchronized Property

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

## 8 Summary

9 Implemented to support the **System.Collections.ICollection**  
10 interface. [Note: For more information, see  
11 **System.Collections.ICollection.IsSynchronized.**]

12

# 1 Hashtable.IsSynchronized Property

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

## 7 Summary

8 Gets a **System.Boolean** value indicating whether access to the  
9 current instance is synchronized (thread-safe).

## 10 Property Value

11

12 **true** if access to the current instance is synchronized (thread-safe);  
13 otherwise, **false**.

## 14 Description

15 This property is read-only.

16

17 [*Note:* This property is implemented to support the  
18 **System.Collections.ICollection** interface.

19

20 For more information regarding synchronization of access to a  
21 **System.Collections.Hashtable**, see  
22 **System.Collections.Hashtable.Synchronized**.]

## 23 Behaviors

24 As described above.

## 25 Default

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

## 27 How and When to Override

28 Override this property, setting it to **true**, if thread-safety can be  
29 guaranteed for the current instance. In order to obtain this safety, use  
30 **System.Collections.Hashtable.SyncRoot** or  
31 **System.Collections.Hashtable.Synchronized**.

32

# 1 Hashtable.Item Property

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

## 9 Summary

10 Gets or sets the value in the current instance that is associated with  
11 the specified key.

## 12 Parameters

13  
14

Parameter	Description
key	The key whose value to get or set.

15  
16  
17

## 16 Property Value

18 The value in the current instance that is associated with *key*. If *key* is  
19 not contained in the current instance, attempting to get it returns **null**,  
20 and attempting to set it creates a new entry using *key*.

## 21 Description

22 [*Note:* This property provides the ability to access a specific element in  
23 the current instance using the following notation:  
24 myCollection[key].]

## 25 Behaviors

26 As described above.

## 27 Default

28 If this property is being set and *key* is already contained in the current  
29 instance, the value associated with the old key is replaced.

## 30 Exceptions

31  
32

Exception	Condition
-----------	-----------

<b>System.ArgumentNullException</b>	<i>key</i> is <b>null</b> .
	The property is being set and the current instance is read-only.
<b>System.NotSupportedException</b>	The property is being set, <i>key</i> is not contained in the current instance, and the current instance has a fixed size.

1  
2  
3

# 1 Hashtable.Keys Property

```
2 [ILASM]
3 .property class System.Collections.ICollection
4 IDictionary.Keys { public hidebysig virtual abstract
5 specialname class System.Collections.ICollection
6 get_IDictionary.Keys() }
7
8 [C#]
9 ICollection IDictionary.Keys { get; }
```

## 9 Summary

10 Implemented to support the **System.Collections.IDictionary**  
11 interface. [Note: For more information, see  
12 **System.Collections.IDictionary.Keys**.]

13

# 1 Hashtable.Keys Property

```
2 [ILASM]
3 .property class System.Collections.ICollection Keys {
4 public hidebysig virtual specialname class
5 System.Collections.ICollection get_Keys() }
6
7 [C#]
8 public virtual ICollection Keys { get; }
```

## 8 Summary

9 Gets a **System.Collections.ICollection** containing the keys of the  
10 current instance.

## 11 Property Value

12

13 A **System.Collections.ICollection** containing the keys of the current  
14 instance.

## 15 Description

16 This property is read-only.

## 17 Behaviors

18 As described above.

## 19 Default

20 The order of the keys in the **System.Collections.ICollection** is  
21 unspecified, but it is the same order as the associated values in the  
22 **System.Collections.ICollection** returned by the  
23 **System.Collections.Hashtable.Values** method.

24

25 The returned **System.Collections.ICollection** is a reference to the  
26 current instance, not a static copy. Therefore, changes to the current  
27 instance continue to be reflected in the  
28 **System.Collections.ICollection**.

29

# 1 Hashtable.SyncRoot Property

```
2 [ILASM]
3 .property object ICollection.SyncRoot { public hidebysig
4 virtual abstract specialname object
5 get_ICollection.SyncRoot() }
6
7 [C#]
8 object ICollection.SyncRoot { get; }
```

## 8 Summary

9 Implemented to support the **System.Collections.ICollection**  
10 interface. [Note: For more information, see  
11 **System.Collections.ICollection.SyncRoot.**]

12

# 1 Hashtable.SyncRoot Property

```
2 [ILASM]
3 .property object SyncRoot { public hidebysig virtual
4 specialname object get_SyncRoot() }
5
6 [C#]
7 public virtual object SyncRoot { get; }
```

## 7 Summary

8 Gets a **System.Object** that can be used to synchronize access to the  
9 current instance.

## 10 Property Value

11

12 A **System.Object** that can be used to synchronize access to the  
13 current instance.

## 14 Description

15 This property is read-only.

16

17 A thread is required to perform synchronized operations only on the  
18 **System.Collections.Hashtable.SyncRoot** of a  
19 **System.Collections.Hashtable**, not directly on the table itself. This  
20 maintains proper synchronization with any other threads concurrently  
21 modifying the table.

22

23 [*Note:* This property is implemented to support the  
24 **System.Collections.ICollection** interface.]

## 25 Behaviors

26 As described above.

## 27 Default

28 This method returns a reference to the current instance.

## 29 How and When to Override

30 Override this property to return an object on which to lock when  
31 implementing a collection that wraps another collection (using a subset  
32 of it, for example). This is useful when providing synchronized access  
33 through two or more wrapper collections to the same underlying  
34 collection. Typically, this property returns a reference to the current  
35 instance.

## 36 Usage

1 Use this property to obtain a **System.Object** that can be used to  
2 synchronize access to the current instance.

3

# 1 Hashtable.Values Property

```
2 [ILASM]
3 .property class System.Collections.ICollection
4 IDictionary.Values { public hidebysig virtual abstract
5 specialname class System.Collections.ICollection
6 get_IDictionary.Values() }
7
8 [C#]
9 ICollection IDictionary.Values { get; }
```

## 9 Summary

10 Implemented to support the **System.Collections.IDictionary**  
11 interface. [Note: For more information, see  
12 **System.Collections.IDictionary.Values.**]

13

# 1 Hashtable.Values Property

```
2 [ILASM]
3 .property class System.Collections.ICollection Values {
4 public hidebysig virtual specialname class
5 System.Collections.ICollection get_Values() }
6
7 [C#]
8 public virtual ICollection Values { get; }
```

## 8 Summary

9 Gets a **System.Collections.ICollection** containing the values of the  
10 current instance.

## 11 Property Value

12

13 A **System.Collections.ICollection** containing the values of the  
14 current instance.

## 15 Description

16 This property is read-only.

## 17 Behaviors

18 As described above.

## 19 Default

20 The order of the values in the **System.Collections.ICollection** is  
21 unspecified, but it is the same order as the associated keys in the  
22 **System.Collections.ICollection** returned by the  
23 **System.Collections.Hashtable.Keys** method.

24

25 The returned **System.Collections.ICollection** is a reference to the  
26 current instance, not a static copy. Therefore, changes to the current  
27 instance continue to be reflected in the  
28 **System.Collections.ICollection**.

29