

# 1 System.Collections.Generic.Dictionary<TKey, 2 TValue> Class

```
3 [ILAsm]  
4 .class public serializable  
5 System.Collections.Generic.Dictionary`2<TKey,TValue> extends System.Object  
6 implements class System.Collections.Generic.IDictionary`2<!0,!1>,  
7 System.Collections.Generic ICollection`1<valuetype  
8 System.Collections.Generic.KeyValuePair`2<!0,!1>>,  
9 System.Collections.Generic.IEnumerable`1<valuetype  
10 System.Collections.Generic.KeyValuePair`2<!0,!1>>,  
11 System.Collections.IDictionary, System.Collections.ICollection,  
12 System.Collections.IEnumerable  
  
13 [C#]  
14 public class Dictionary<TKey,TValue>: IDictionary<TKey,TValue>,  
15 ICollection<KeyValuePair<TKey,TValue>>,  
16 IEnumerable<KeyValuePair<TKey,TValue>> , IDictionary, ICollection,  
17 IEnumerable
```

## 18 Assembly Info:

- 19 • *Name*: mscorlib
- 20 • *Public Key*: [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 21 • *Version*: 2.0.x.x
- 22 • *Attributes*:
  - 23 ○ CLSCompliantAttribute(true)

## 24 Implements:

- 25 • **System.Collections.ICollection**
- 26 • **System.Collections.IDictionary**
- 27 • **System.Collections.IEnumerable**
- 28 • **System.Collections.Generic.ICollection<KeyValuePair<TKey,TValue>>**
- 29 • **System.Collections.Generic.IDictionary<TKey,TValue>**
- 30 • **System.Collections.Generic.IEnumerable<KeyValuePair<TKey,TValue>>**

## 31 Summary

32 Represents a collection of key/value pairs that are organized based on the key.

## 33 Inherits From: System.Object

34

35 **Library**: BCL

36

37 **Thread Safety**: Static members of this type are thread safe. Any instance members are not  
38 guaranteed to be thread safe. A dictionary can support multiple readers concurrently, as  
39 long as the collection is not modified. Even so, enumerating through a collection is  
40 intrinsically not a thread-safe procedure. To guarantee thread safety during enumeration,

1 you can lock the collection during the entire enumeration. To allow the collection to be  
2 accessed by multiple threads for reading and writing, you must implement your own  
3 synchronization.

4

## 5 **Description**

6 Each element is a key/value pair that can be retrieved as a  
7 `System.Collections.Generic.KeyValuePair<TKey,TValue>` object.

8

9 `System.Collections.Generic.Dictionary<TKey,TValue>` requires an equality  
10 comparer implementation to perform comparisons. If no equality comparer is provided,  
11 the following default equality comparer approach is used: If type `TKey` implements  
12 `System.IEquatable<TKey>`, that implementation is used; otherwise, `TKey`'s  
13 implementations of `System.Object.Equals` and `System.Object.GetHashCode` are used.  
14 In any case, you can specify a  
15 `System.Collections.Generic.IEqualityComparer<TKey>` implementation in a  
16 constructor overload that accepts an equality comparer parameter.

17

18 After its insertion in a dictionary, changes to the value of a key that affect the equality  
19 comparer render the dictionary's behavior unspecified. Every key in a dictionary must be  
20 unique according to the equality comparer. A key cannot be `null`, but a value can be, if  
21 the value type `TValue` is a reference type.

22

23 The capacity of a dictionary is the number of elements that dictionary can hold. As  
24 elements are added to a dictionary, the capacity is automatically increased.

25

26 This type contains a member that is a nested type, called `Enumerator`. Although  
27 `Enumerator` is a member of this type, `Enumerator` is not described here; instead, it is  
28 described in its own entry, `Dictionary<TKey,TValue>.Enumerator`.

29

# 1 Dictionary<TKey,TValue>() Constructor

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

## 6 Summary

7 Initializes a new dictionary that is empty, has the default initial capacity, and uses the  
8 default equality comparer.

9

1  
2 **Dictionary<TKey,TValue> (System.Collections**  
3 **.Generic.IEqualityComparer<TKey>)**  
4 **Constructor**

```
5 [ILAsm]  
6 public rtspecialname specialname instance void .ctor(class  
7 System.Collections.Generic.IEqualityComparer`1<!0> comparer)  
  
8 [C#]  
9 public Dictionary(IEqualityComparer<TKey> comparer)
```

10 **Summary**

11        Initializes a new dictionary that is empty, has the default initial capacity, and uses the  
12        specified equality comparer.

13 **Parameters**

Parameter	Description
<i>comparer</i>	The equality comparer implementation to use when comparing keys.  -or-  null to use the default equality comparer for the type of the key.

14

15

1  
2 **Dictionary<TKey,TValue> (System.Collections**  
3 **.Generic.IDictionary<TKey,TValue>)**  
4 **Constructor**

```
5 [ILAsm]  
6 public rtspecialname specialname instance void .ctor(class  
7 System.Collections.Generic.IDictionary`2<!0,!1> dictionary)  
8  
9 [C#]  
10 public Dictionary(IDictionary<TKey,TValue> dictionary)
```

11 **Summary**

12 Initializes a new dictionary that contains elements copied from the specified dictionary,  
13 has sufficient capacity to accommodate the number of elements copied, and uses the  
14 default equality comparer.

15 **Parameters**

Parameter	Description
<i>dictionary</i>	The dictionary whose elements are to be copied to the new dictionary.

16 **Description**

17 Every key in a dictionary must be unique according to the default equality comparer;  
18 otherwise, a *System.ArgumentException* is thrown; likewise, every key in the source  
19 dictionary must also be unique according to the default equality comparer.

20 **Exceptions**

Exception	Condition
<b>System.ArgumentException</b>	<i>dictionary</i> contains one or more duplicate keys.
<b>System.ArgumentNullException</b>	<i>dictionary</i> is null.

21  
22

# 1 Dictionary<TKey,TValue> (System.Int32)

## 2 Constructor

```
3 [ILAsm]  
4 public rtspecialname specialname instance void .ctor(int32 capacity)  
5 [C#]  
6 public Dictionary(int capacity)
```

### 7 Summary

8 Initializes a new dictionary that is empty, has the specified initial capacity, and uses the  
9 default equality comparer.

### 10 Parameters

Parameter	Description
<i>capacity</i>	The initial number of elements that the dictionary can contain.

### 11 Exceptions

### 12

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>capacity</i> is less than zero.

13  
14

1  
2 **Dictionary<TKey,TValue> (System.Collections**  
3 **.Generic.IDictionary<TKey,TValue> ,**  
4 **System.Collections.Generic.IEqualityComparer**  
5 **r<TKey>) Constructor**

```
6 [ILAsm]  
7 public rtspecialname specialname instance void .ctor(class  
8 System.Collections.Generic.IDictionary`2<!0,!1> dictionary, class  
9 System.Collections.Generic.IEqualityComparer`1<!0> comparer)  
  
10 [C#]  
11 public Dictionary(IDictionary<TKey,TValue> dictionary,  
12 IEqualityComparer<TKey> comparer)
```

13 **Summary**

14 Initializes a new dictionary that contains elements copied from the specified dictionary,  
15 has sufficient capacity to accommodate the number of elements copied, and uses the  
16 specified equality comparer.

17 **Parameters**

Parameter	Description
<i>dictionary</i>	The dictionary whose elements are to be copied to the new dictionary.
<i>comparer</i>	The equality comparer implementation to use when comparing keys.  -or-  null to use the default equality comparer for the type of the key.

18  
19 **Description**

20 Every key in a dictionary must be unique according to the specified; otherwise, a  
21 `System.ArgumentException` is thrown; likewise, every key in the source dictionary must  
22 also be unique according to the specified equality comparer.

23 **Exceptions**

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

<b>System.ArgumentException</b>	<i>dictionary</i> contains one or more duplicate keys.
<b>System.ArgumentNullException</b>	<i>dictionary</i> is null.

1

2

# 1 Dictionary<TKey,TValue> (System.Int32, 2 System.Collections.Generic.IEqualityComparer 3 r<TKey>) Constructor

```
4 [ILAsm]  
5 public rtspecialname specialname instance void .ctor(int32 capacity, class  
6 System.Collections.Generic.IEqualityComparer`1<!0> comparer)  
  
7 [C#]  
8 public Dictionary(int capacity, IEqualityComparer<TKey> comparer)
```

## 9 Summary

10 Initializes a new dictionary that is empty, has the specified initial capacity, and uses the  
11 specified equality comparer.

## 12 Parameters

Parameter	Description
<i>capacity</i>	The initial number of elements that the dictionary can contain.
<i>comparer</i>	The equality comparer implementation to use when comparing keys. -or- null to use the default equality comparer for the type of the key.

## 13 14 Exceptions

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>capacity</i> is less than zero.

15  
16

# 1 Dictionary<TKey,TValue>.Add(TKey, TValue)

## 2 Method

```
3 [ILAsm]  
4 .method public hidebysig void Add(!0 key, !1 value)  
5 [C#]  
6 public void Add(TKey key, TValue value)
```

### 7 Summary

8 Adds an element with the specified key and value to the dictionary.

### 9 Parameters

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

10

### 11 Description

12 You can also use the  
13 `System.Collections.Generic.Dictionary<TKey,TValue>.Item(TKey)` property to add  
14 new elements by setting the value of a key that does not exist in the dictionary.  
15 However, if the specified key already exists in the dictionary, setting the  
16 `System.Collections.Generic.Dictionary<TKey,TValue>.Item(TKey)` property  
17 overwrites the old value. In contrast, the  
18 `System.Collections.Generic.Dictionary<TKey,TValue>.Add(TKey,TValue)` method  
19 does not modify existing elements.

20

21 If `System.Collections.Generic.Dictionary<TKey,TValue>.Count` already equals the  
22 capacity, the capacity of the dictionary is increased.

23

24 A key cannot be `null`, but a value can be, if the value type `TValue` is a reference type.

### 25 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	An element with the same key already exists in the dictionary.
<b>System.ArgumentNullException</b>	<i>key</i> is <code>null</code> .

1

2

# 1 Dictionary<TKey,TValue>.Clear() Method

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

## 6 Summary

7 Removes all elements from the dictionary.

## 8 Description

9 *[Note:* This method is implemented to support the `System.Collections.IDictionary`  
10 interface.  
11

12 ]

13  
14  
15  
16 `System.Collections.Generic ICollection<TKey>.Count` gets set to zero, and  
17 references to other objects from elements of the collection are also released. The  
18 capacity remains unchanged.

19

# 1 Dictionary<TKey,TValue>.ContainsKey(TKey) 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual bool Contains(!0 key)  
5 [C#]  
6 public virtual bool ContainsKey(TKey key)
```

## 7 Summary

8 Determines whether the dictionary contains an element with a specific key.

## 9 Parameters

Parameter	Description
<i>key</i>	The key to locate in the dictionary.

10

## 11 Return Value

12 true, if an element whose key is *key* is found in the dictionary; otherwise, false.

## 13 Description

14 This implementation is close to O(1) in most cases.

## 15 Exceptions

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

16

17

1  
2 **Dictionary<TKey,TValue>.ContainsValue(TValue) Method**  
3

```
4 [ILAsm]  
5 .method public hidebysig bool Contains(!1 value)  
6 [C#]  
7 public bool ContainsValue(TValue value)
```

8 **Summary**

9 Determines whether the dictionary contains an element with a specific value.

10 **Parameters**

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

11  
12 **Return Value**

13 true, if an element whose value is *value* is found in the dictionary; otherwise, false.

14 **Description**

15 This method determines equality using the default equality comparer for the value type  
16 TValue. If TValue implements System.IEquatable<TValue>, that type is used.  
17 Otherwise, System.Object.Equals is used.

18  
19 This method performs a linear search; therefore, the average execution time is  
20 proportional to System.Collections.Generic.Dictionary<TKey,TValue>.Count. That  
21 is, this method is an O(n) operation, where n is  
22 System.Collections.Generic.Dictionary<TKey,TValue>.Count.

23

# 1 Dictionary<TKey,TValue>.GetEnumerator() 2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual class  
5 System.Collections.Generic.Dictionary`2/Enumerator<!0,!1> GetEnumerator()  
  
6 [C#]  
7 public Dictionary<TKey,TValue>.Enumerator GetEnumerator()
```

## 8 Summary

9 Returns an enumerator that can be used to iterate over the dictionary.

## 10 Return Value

11 An enumerator for the dictionary.

## 12 Usage

13 For a detailed description regarding the use of an enumerator, see  
14 System.Collections.Generic.IEnumerator<TKey>.

15

16

# 1 Dictionary<TKey,TValue>.Remove(TKey)

## 2 Method

```
3 [ILAsm]  
4 .method public hidebysig bool Remove(!0 key)  
5 [C#]  
6 public bool Remove(TKey key)
```

### 7 Summary

8 Removes the element with the specified key from the dictionary.

### 9 Parameters

Parameter	Description
<i>key</i>	The key of the element to be removed from the dictionary.

10

### 11 Return Value

12 *true* if the element containing *key* is successfully removed; otherwise, *false*. This  
13 method also returns *false* if *key* was not found in the dictionary.

### 14 Description

15 If the dictionary does not contain an element with the specified key, the dictionary  
16 remains unchanged. No exception is thrown.

17

18 This method shall be implemented with efficiency that is at least an O(1) operation.

### 19 Exceptions

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

20

21

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **Generic.ICollection<System.Collections.Gener**  
4 **ic.KeyValuePair<TKey,TValue>>.Add(System.**  
5 **Collections.Generic.KeyValuePair<TKey,TValu**  
6 **e> ) Method**

```
7 [ILAsm]  
8 .method private hidebysig virtual final void class  
9 System.Collections.Generic.ICollection`1<System.Collections.Generic.KeyVal  
10 uePair`2<!0,!1>>.Add(class  
11 System.Collections.Generic.KeyValuePair`2<!0,!1> keyValuePair)  
  
12 [C#]  
13 void ICollection<KeyValuePair<TKey,TValue>>.Add(KeyValuePair<TKey,TValue>  
14 keyValuePair)
```

15 **Summary**

16 This method is implemented to support the  
17 System.Collections.Generic.ICollection<System.Collections.Generic.KeyValu  
18 ePair<TKey,TValue>> interface.

19

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **Generic.ICollection<System.Collections.Gener**  
4 **ic.KeyValuePair<TKey,TValue>>.Contains(Sy**  
5 **stem.Collections.Generic.KeyValuePair<TKey,**  
6 **TValue> ) Method**

```
7 [ILAsm]  
8 .method private hidebysig virtual final bool  
9 System.Collections.Generic.ICollection`1<System.Collections.Generic.KeyVal  
10 uePair`2<!0,!1>>.Contains(class  
11 System.Collections.Generic.KeyValuePair`2<!0,!1> keyValuePair)  
  
12 [C#]  
13 bool  
14 ICollection<KeyValuePair<TKey,TValue>>.Contains(KeyValuePair<TKey,TValue>  
15 keyValuePair)
```

16 **Summary**

17 This method is implemented to support the  
18 System.Collections.Generic.ICollection<System.Collections.Generic.KeyValu  
19 ePair<TKey,TValue>> interface.

20

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **Generic.ICollection<System.Collections.Gener**  
4 **ic.KeyValuePair<TKey,TValue>>.CopyTo(Syst**  
5 **em.Collections.Generic.KeyValuePair<TKey,T**  
6 **Value>[], System.Int32) Method**

```
7 [ILAsm]  
8 .method private hidebysig virtual final void  
9 System.Collections.Generic.ICollection`1<System.Collections.Generic.KeyVal  
10 uePair`2<!0,!1>>.CopyTo(class  
11 System.Collections.Generic.KeyValuePair`2<!0,!1>[] array, int32 index)  
  
12 [C#]  
13 void  
14 ICollection<KeyValuePair<TKey,TValue>>.CopyTo(KeyValuePair<TKey,TValue>[]  
15 array, int index)
```

16 **Summary**

17 This method is implemented to support the  
18 System.Collections.Generics.ICollection<System.Collections.Generic.KeyValu  
19 ePair<TKey,TValue>> interface.

20

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **Generic.ICollection<System.Collections.Gener**  
4 **ic.KeyValuePair<TKey,TValue>>.Remove(Sys**  
5 **tem.Collections.Generic.KeyValuePair<TKey,T**  
6 **Value> ) Method**

```
7 [ILAsm]  
8 .method private hidebysig virtual final bool  
9 System.Collections.Generic.ICollection`1<System.Collections.Generic.KeyVal  
10 uePair`2<!0,!1>>.Remove(class  
11 System.Collections.Generic.KeyValuePair`2<!0,!1> keyValuePair)  
  
12 [C#]  
13 bool  
14 ICollection<KeyValuePair<TKey,TValue>>.Remove(KeyValuePair<TKey,TValue>  
15 keyValuePair)
```

16 **Summary**

17 This method is implemented to support the  
18 System.Collections.Generic.ICollection<System.Collections.Generic.KeyValu  
19 ePair<TKey,TValue>> interface.

20

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **Generic.IEnumerable<System.Collections.Gen**  
4 **eric.KeyValuePair<TKey,TValue>>.GetEnume**  
5 **rator() Method**

```
6 [ILAsm]  
7 .method private hidebysig virtual abstract class  
8 System.Collections.Generic.IEnumerator<T>  
9 System.Collections.Generic.IEnumerable<System.Collections.Generic.KeyValue  
10 Pair`2<!0,!1>>.GetEnumerator()  
  
11 [C#]  
12 IEnumerator<T> IEnumerable<KeyValuePair<TKey,TValue>>.GetEnumerator()
```

13 **Summary**

14 This method is implemented to support the  
15 System.Collections.Generic.IEnumerable<System.Collections.Generic.KeyValu  
16 ePair<TKey,TValue>> interface.

17

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **ICollection.CopyTo(System.Array,**  
4 **System.Int32) Method**

5 `[ILAsm]`  
6 `.method private hidebysig virtual final void`  
7 `System.Collections.ICollection.CopyTo(class System.Array array, int32`  
8 `index)`

9 `[C#]`  
10 `void ICollection.CopyTo(Array array, int index)`

11 **Summary**

12 This method is implemented to support the System.Collections.ICollection  
13 interface.

14

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IDictionary.Add(System.Object,**  
4 **System.Object) Method**

```
5 [ILAsm]  
6 .method private hidebysig virtual final void  
7 System.Collections.IDictionary.Add(object key, object value)  
  
8 [C#]  
9 void IDictionary.Add(object key, object value)
```

10 **Summary**

11 This method is implemented to support the System.Collections.IDictionary  
12 interface.

13

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IDictionary.Contains(System.Object) Method**

```
4 [ILAsm]  
5 .method private hidebysig virtual final bool  
6 System.Collections.IDictionary.Contains(object key)  
7 [C#]  
8 bool IDictionary.Contains(object key)
```

9 **Summary**

10 This method is implemented to support the System.Collections.IDictionary  
11 interface.

12

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IDictionary.GetEnumerator() Method**

```
4 [ILAsm]  
5 .method private hidebysig virtual final class  
6 System.Collections.IDictionaryEnumerator  
7 System.Collections.IDictionary.GetEnumerator()  
  
8 [C#]  
9 IDictionaryEnumerator IDictionary.GetEnumerator()
```

10 **Summary**

11 This method is implemented to support the System.Collections.IDictionary  
12 interface.

13

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IDictionary.Remove(System.Object) Method**

```
4 [ILAsm]  
5 .method private hidebysig virtual final void  
6 System.Collections.IDictionary.Remove(object key)  
7  
8 [C#]  
9 void IDictionary.Remove(object key)
```

9 **Summary**

10 This method is implemented to support the System.Collections.IDictionary  
11 interface.

12

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IEnumerable.GetEnumerator() Method**

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

10 **Summary**

11 This method is implemented to support the System.Collections.IEnumerable  
12 interface.

13

# 1 Dictionary<TKey,TValue>.TryGetValue(TKey, 2 TValue) Method

```
3 [ILAsm]  
4 .method public hidebysig bool TryGetValue(!0 key, [out] !1 value)  
5 [C#]  
6 public bool TryGetValue(TKey key, out TValue value)
```

## 7 Summary

8 Gets the value associated with the specified key.

## 9 Parameters

Parameter	Description
<i>key</i>	The key of the element to locate in the dictionary.
<i>value</i>	When this method returns, the value associated with the specified key, if the key is found; otherwise, the default value for the type of this parameter.

10

## 11 Return Value

12 true if the dictionary contains an element with the specified key; otherwise, false.

## 13 Description

14 This method combines the functionality of the  
15 System.Collections.Generic.Dictionary<TKey,TValue>.ContainsKey method and  
16 the System.Collections.Generic.Dictionary<TKey,TValue>.Item property.

17  
18 The default value for value types is zero while that for reference types is null.

19  
20 This method is an O(1) operation.

## 21 Exceptions

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

22

23

# 1 Dictionary<TKey,TValue>.Count Property

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

## 7 Summary

8 Gets the number of key/value pairs contained in the dictionary.

## 9 Property Value

10 The number of key/value pairs contained in the dictionary.

## 11 Description

12 This property is read-only.

13

14 Retrieving the value of this property is an O(1) operation.

15

# 1 Dictionary<TKey,TValue>.Item Property

```
2 [ILAsm]  
3 .property !1 Item[!0 key] { public hidebysig specialname !1 get_Item(!0  
4 key) public hidebysig specialname void set_Item(!0 key, !1 value) }  
5 [C#]  
6 public TValue this[TKey key] { get; set; }
```

## 7 Summary

8 Gets or sets the value associated with the specified key.

## 9 Parameters

Parameter	Description
<i>key</i>	The key whose value is to be gotten or set.

10

## 11 Property Value

12 The value associated with the specified key. On a get attempt, if the specified key is not  
13 found, a `System.Collections.Generic.KeyNotFoundException` is thrown. On a set  
14 attempt, if the specified key is not found, a new element using the specified key is  
15 created.

## 16 Description

17 The default value for value types is zero while that for reference types is null.

18

19 You can also use the

20 `System.Collections.Generic.Dictionary<TKey,TValue>.Item(TKey)` property to add  
21 new elements by setting the value of a key that does not exist in the dictionary.

22 However, if the specified key already exists in the dictionary, setting the  
23 `System.Collections.Generic.Dictionary<TKey,TValue>.Item(TKey)` property  
24 overwrites the old value. In contrast, the

25 `System.Collections.Generic.Dictionary<TKey,TValue>.Add(TKey,TValue)` method  
26 does not modify existing elements.

27

28 A key cannot be null, but a value can be, if the value type `TValue` is a reference type.

29

30 Retrieving the value of this property is an O(1) operation; setting the property is also an  
31 O(1) operation.

## 32 Exceptions

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

<b>System.ArgumentNullException</b>	<i>key</i> is null.
<b>System.Collections.Generic. KeyNotFoundException</b>	During a get attempt, <i>key</i> is not found in the dictionary.

1

2

# 1 Dictionary<TKey,TValue>.Keys Property

```
2 [ILAsm]
3 .property class
4 System.Collections.Generic.Dictionary`2/KeyCollection<!0,!1> Keys { public
5 hidebysig specialname instance class
6 System.Collections.Generic.Dictionary`2/KeyCollection<!0,!1> get_Keys() }
7 [C#]
8 public KeyCollection<TKey,TValue> Keys { get; }
```

## 9 Summary

10 Gets a collection that contains the keys in the dictionary.

## 11 Property Value

12 A collection of the keys in the dictionary.

## 13 Description

14 This property is read-only.

15  
16 The order of the keys in the key collection is unspecified, but it is the same order as the  
17 associated values in the value collection returned by the  
18 `System.Collections.Generic.Dictionary<TKey,TValue>.Values` property.

19  
20 If the dictionary is modified, or the value of any key in the dictionary is modified, the  
21 behavior of the key collection is unspecified.

22

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **Generic ICollection<System.Collections.Gener**  
4 **ic.KeyValuePair<TKey,TValue>>.IsReadOnly**  
5 **Property**

```
6 [ILAsm]  
7 .property bool  
8 System.Collections.Generic.ICollection<System.Collections.Generic.KeyValue  
9 Pair`2<!0,!1>>.IsReadOnly { private hidebysig virtual final specialname  
10 bool get_IsReadOnly() }  
  
11 [C#]  
12 bool ICollection<KeyValuePair<TKey,TValue>>.IsReadOnly { get; }
```

13 **Summary**

14 This read-only property is implemented to support the  
15 System.Collections.Generic.ICollection<System.Collections.Generic.KeyValu  
16 ePair<TKey,TValue>> interface.

17

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **Generic.IDictionary<TKey,TValue>.Keys**  
4 **Property**

```
5 [ILAsm]  
6 .property class System.Collections.Generic ICollection`1<!0>  
7 System.Collections.Generic.IDictionary`2<!0,!1>.Keys { private hideby sig  
8 virtual final specialname class  
9 System.Collections.Generic.IDictionary`2<!0,!1>.Keys get_Keys() }  
10 [C#]  
11 ICollection<TKey> IDictionary<TKey,TValue>.Keys { get; }
```

12 **Summary**

13 This read-only property is implemented to support the  
14 System.Collections.Generic.IDictionary<TKey,TValue> interface.

15

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **Generic.IDictionary<TKey,TValue>.Values**  
4 **Property**

```
5 [ILAsm]  
6 .property class System.Collections.Generic ICollection`1<!1>  
7 System.Collections.Generic.IDictionary`2<!0,!1>.Values { private hideby sig  
8 virtual final specialname class  
9 System.Collections.Generic.IDictionary`2<!0,!1>.Values get_Values() }  
10 [C#]  
11 ICollection<TValue> IDictionary<TKey,TValue>.Values { get; }
```

12 **Summary**

13 This read-only property is implemented to support the  
14 System.Collections.Generic.IDictionary<TKey,TValue> interface.

15

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **ICollection.IsSynchronized Property**

```
4 [ILAsm]  
5 .property bool System.Collections.ICollection.IsSynchronized { private  
6 hidebyref virtual final bool get_IsSynchronized() }  
7  
8 [C#]  
9 bool ICollection.IsSynchronized { get; }
```

9 **Summary**

10 This read-only property is implemented to support the  
11 System.Collections.ICollection interface.

12

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **ICollection.SyncRoot Property**

```
4 [ILAsm]  
5 .property object System.Collections.ICollection.SyncRoot { private  
6 hidebyref virtual final object get_SyncRoot() }  
7  
8 [C#]  
9 object ICollection.SyncRoot { get; }
```

9 **Summary**

10 This read-only property is implemented to support the  
11 System.Collections.ICollection interface.

12

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IDictionary.IsFixedSize Property**

```
4 [ILAsm]  
5 .property bool System.Collections.IDictionary.IsFixedSize { private  
6 hidebyref virtual final specialname bool get_IsFixedSize() }  
7  
8 [C#]  
9 bool IDictionary.IsFixedSize { get; }
```

9 **Summary**

10 This read-only property is implemented to support the  
11 System.Collections.IDictionary interface.

12

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IDictionary.IsReadOnly Property**

```
4 [ILAsm]  
5 .property bool System.Collections.IDictionary.IsReadOnly { private  
6 hideby sig virtual final specialname bool get_IsReadOnly() }  
7 [C#]  
8 bool IDictionary.IsReadOnly { get; }
```

9 **Summary**

10 This read-only property is implemented to support the  
11 System.Collections.IDictionary interface.

12

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IDictionary.Item Property**

```
4 [ILAsm]  
5 .property object System.Collections.IDictionary.Item(object key) { private  
6 hidebysig virtual final specialname object get_Item(object key) private  
7 hidebysig virtual final specialname void set_Item(object key, object  
8 value) }  
  
9 [C#]  
10 object IDictionary.this[object key] { get; set; }
```

11 **Summary**

12 This read-only property is implemented to support the  
13 System.Collections.IDictionary interface.

14

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IDictionary.Keys Property**

```
4 [ILAsm]  
5 .property class System.Collections.ICollection  
6 System.Collections.IDictionary.Keys { private hidebysig virtual final  
7 specialname class System.Collections.ICollection get_Keys() }  
  
8 [C#]  
9 ICollection IDictionary.Keys { get; }
```

10 **Summary**

11 This read-only property is implemented to support the  
12 System.Collections.IDictionary interface.

13

1  
2 **Dictionary<TKey,TValue>.System.Collections.**  
3 **IDictionary.Values Property**

```
4 [ILAsm]  
5 .property class System.Collections.ICollection  
6 System.Collections.IDictionary.Values { private hidebyref virtual final  
7 specialname class System.Collections.ICollection get_Values() }  
  
8 [C#]  
9 ICollection IDictionary.Values { get; }
```

10 **Summary**

11 This read-only property is implemented to support the  
12 System.Collections.IDictionary interface.

13

# 1 Dictionary<TKey,TValue>.Values Property

```
2 [ILAsm]
3 .property class
4 System.Collections.Generic.Dictionary`2/ValueCollection<!0,!1> Values {
5 public hidebysig specialname instance class
6 System.Collections.Generic.Dictionary`2/ValueCollection<!0,!1>
7 get_values() }
8
9 [C#]
public ValueCollection<TKey,TValue> Values { get; }
```

## 10 Summary

11 Gets a collection that contains the values in the dictionary.

## 12 Property Value

13 A collection of the values in the dictionary.

## 14 Description

15 This property is read-only.

16  
17 The order of the values in the value collection is unspecified, but it is the same order as  
18 the associated values in the key collection returned by the  
19 `System.Collections.Generic.Dictionary<TKey,TValue>.Keys` property.

20  
21 The returned value collection is not a static copy; instead, it refers back to the values in  
22 the original dictionary. Therefore, changes to the dictionary continue to be reflected in  
23 the value collection.

24