

System.Collections.Generic.IDictionary<TKey, TValue> Interface

```
[ILAsm]
.class interface public abstract IDictionary`2<TKey,TValue> implements
System.Collections.Generic ICollection`1<valuetype
System.Collections.Generic.KeyValuePair`2<!0,!1>>,
System.Collections.Generic.IEnumerable`1<valuetype
System.Collections.Generic.KeyValuePair`2<!0,!1>>

[C#]
public interface IDictionary<TKey,TValue>:
ICollection<KeyValuePair<TKey,TValue>>,
IEnumerable<KeyValuePair<TKey,TValue>>
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
 - CLSCompliantAttribute(true)

Type Attributes:

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

Implements:

- System.Collections.Generic.ICollection<KeyValuePair<TKey,TValue>>
- System.Collections.Generic.IEnumerable<KeyValuePair<TKey,TValue>>

Summary

Represents a generic collection of key/value pairs.

Library: BCL

Description

This interface class is the base interface for generic collections of key/value pairs. The implementing class must have a method for comparing keys.

Each element is a key/value pair stored in a key value pair object.

Each pair must have a non-null key unique according to the comparison method of the class implementing this interface. The value can be null and need not be unique. The System.Collections.Generic.IDictionary<TKey,TValue> interface allows the

1 contained keys and values to be enumerated, but it does not imply any particular sort
2 order.

3
4 Some implementations of this interface might permit null keys, and some might not. A
5 dictionary implementation that prohibits null keys shall throw
6 `System.ArgumentNullException` whenever a method or indexer is called with a null
7 key.

8

1 IDictionary<TKey,TValue>.Add(TKey, TValue) Method

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

7 Summary

8 Adds an entry with the provided key and value to the current instance.

9 Parameters

Parameter	Description
<i>key</i>	The TKey to use as the key of the entry to add.
<i>value</i>	The TValue to use as the value of the entry to add.

11 Description

12 You can also use the
13 `System.Collections.Generic.IDictionary<TKey,TValue>.Item(TKey)` property to
14 add 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.IDictionary<TKey,TValue>.Item(TKey)` property
17 overwrites the old value. In contrast, the
18 `System.Collections.Generic.IDictionary<TKey,TValue>.Add(TKey,TValue)`
19 method does not modify existing elements.

20
21 Implementations can vary in how they determine equality of objects.

22 Exceptions

Exception	Condition
System.ArgumentException	An entry with the same key already exists in the current instance.
System.NotSupportedException	The current instance is read-only.

I Dictionary<TKey,TValue>.ContainsKey(TKey)) Method

```
[ILAsm]  
.method public hidebysig virtual abstract bool ContainsKey(!0 key)  
  
[C#]  
bool ContainsKey(TKey key)
```

Summary

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

Parameters

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

Return Value

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

Description

Implementations can vary in how they determine equality of objects.

I Dictionary<TKey,TValue>.Remove(TKey)

Method

```
[ILAsm]  
.method public hidebysig virtual abstract bool Remove(!0 key)  
  
[C#]  
bool Remove(TKey key)
```

Summary

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

Parameters

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

Return Value

true if the element is successfully removed; otherwise, false. *[Note: This method also returns false if key was not found.]*

]

Description

Implementations can vary in how they determine equality of objects.

Exceptions

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

1 IDictionary<TKey,TValue>.Item Property

```
2 [ILAsm]
3 .property !1 Item[!0 key] { public hidebysig virtual abstract specialname
4 !1 get_Item(!0 key) public hidebysig virtual abstract specialname void
5 set_Item(!0 key, !1 value) }

6 [C#]
7 TValue this[TKey key] { get; set; }
```

8 Summary

9 Gets or sets the element in the current instance that is associated with the specified
10 key.

11 Parameters

Parameter	Description
<i>key</i>	The key of the element to get or set.

13 Property Value

14 The value associated with the given key.

15 Description

16 This property provides the ability to access a specific element in the collection.

17 You can also use the

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

20 However, if the specified key already exists in the dictionary, setting the
21 `System.Collections.Generic.IDictionary<TKey,TValue>.Item(TKey)` property
22 overwrites the old value. In contrast, the
23 `System.Collections.Generic.IDictionary<TKey,TValue>.Add(TKey,TValue)`
24 method does not modify existing elements.

25 Implementations can vary in how they determine equality of objects.

28 Exceptions

Exception	Condition
System.ArgumentException	The property is read but <i>key</i> is not found.

System.NotSupportedException

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

1

2

I Dictionary<TKey,TValue>.Keys Property

```
[ILAsm]
.property class System.Collections.Generic ICollection`1<!0> Keys { public
hidebysig virtual abstract specialname class
System.Collections.Generic ICollection`1<!0> get_Keys() }

[C#]
ICollection<TKey> Keys { get; }
```

Summary

Gets a collection containing the keys of the current instance.

Property Value

A collection containing the keys of the current instance.

Description

This property is read-only.

The order of the keys in the returned `System.Collections.Generic.ICollection<TKey>` is unspecified, but it is guaranteed to be the same order as the corresponding values in the collection returned by the `System.Collections.Generic.IDictionary<TKey,TValue>.Values` property.

1 I Dictionary<TKey,TValue>.Values Property

```
2    [ILAsm]
3    .property class System.Collections.Generic ICollection`1<!!> Values {
4    public hidebysig virtual abstract specialname class
5    System.Collections.Generic ICollection`1<!!> get_Values() }

6    [C#]
7    ICollection<TValue> Values { get; }
```

8 Summary

9 Gets a collection containing the values in the current instance.

10 Property Value

11 A collection containing the values in the current instance.

12 Description

13 This property is read-only.

14 The order of the values in the returned
15 System.Collections.Generic.ICollection<TKey> is unspecified, but it is guaranteed
16 to be the same order as the corresponding keys in the collection returned by the
17 System.Collections.Generic.IDictionary<TKey,TValue>.Keys property.
18