

# 1 System.Text.Decoder Class

```
2 [ILAsm]  
3 .class public abstract serializable Decoder extends System.Object  
4 [C#]  
5 public abstract class Decoder
```

## 6 Assembly Info:

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

## 12 Summary

13 Converts blocks of bytes into blocks of characters, maintaining state across successive  
14 calls for reading from a `System.IO.Stream`.

## 15 Inherits From: System.Object

16

17 **Library:** BCL

18

19 **Thread Safety:** All public static members of this type are safe for multithreaded operations.  
20 No instance members are guaranteed to be thread safe.

21

## 22 Description

23 [*Note:* Following instantiation of a decoder, sequential blocks of bytes are converted into  
24 blocks of characters through calls to the `System.Text.Decoder.GetChars` method. The  
25 decoder maintains state between the conversions, allowing it to correctly decode a  
26 character whose bytes span multiple blocks. This greatly assists decoding streams of  
27 bytes into characters. An instance of a specific implementation of the  
28 `System.Text.Decoder` class is typically obtained through a call to the  
29 `System.Text.Encoding.GetDecoder` method of a `System.Text.Encoding` object.]

30

31

## 32 Example

33 The following example demonstrates using the `System.Text.UTF8Encoding`  
34 implementation of the `System.Text.Decoder` class to convert two byte arrays to a  
35 character array, where one character's bytes span multiple byte arrays. This  
36 demonstrates how to use a `System.Text.Decoder` in streaming-like situations.

37

38 [C#]

```

1
2 using System;
3 using System.Text;
4
5 public class DecoderExample
6 {
7     public static void Main()
8     {
9         // These bytes in UTF-8 correspond to 3 different
10        // Unicode characters - A (U+0041), # (U+0023),
11        // and the biohazard symbol (U+2623). Note the
12        // biohazard symbol requires 3 bytes in UTF-8
13        // (in hex, e2, 98, a3). Decoders store state across
14        // multiple calls to GetChars, handling the case
15        // when one char spans multiple byte arrays.
16
17        byte[] bytes1 = { 0x41, 0x23, 0xe2 };
18        byte[] bytes2 = { 0x98, 0xa3 };
19        char[] chars = new char[3];
20
21        Decoder d = Encoding.UTF8.GetDecoder();
22        int charLen = d.GetChars(bytes1, 0, bytes1.Length,
23                                chars, 0);
24        // charLen is 2.
25
26        charLen += d.GetChars(bytes2, 0, bytes2.Length,
27                                chars, charLen);
28        // charLen is now 3.
29
30        foreach(char c in chars)
31            Console.Write("U+{0:x} ", (ushort)c);
32    }
33 }
34 The output is
35
36 U+41 U+23 U+2623
37

```

38

# 1 Decoder() Constructor

```
2 [ILAsm]  
3 family rtspecialname specialname instance void .ctor()  
4 [C#]  
5 protected Decoder()
```

## 6 Summary

7 Constructs a new instance of the `System.Text.Decoder` class.

## 8 Description

9 This constructor is called only by classes that inherit from the `System.Text.Decoder`  
10 class.

11

# 1 Decoder.GetCharCount(System.Byte[], 2 System.Int32, System.Int32) Method

```
3 [ILAsm]  
4 .method public hidebysig virtual abstract int32 GetCharCount(class  
5 System.Byte[] bytes, int32 index, int32 count)  
  
6 [C#]  
7 public abstract int GetCharCount(byte[] bytes, int index, int count)
```

## 8 Summary

9 Determines the exact number of characters that will be produced by decoding the  
10 specified range of the specified array of bytes.

## 11 Parameters

Parameter	Description
<i>bytes</i>	A System.Byte array to decode.
<i>index</i>	A System.Int32 that specifies the first index in <i>bytes</i> to decode.
<i>count</i>	A System.Int32 that specifies the number elements in <i>bytes</i> to decode.

12

## 13 Return Value

14 A System.Int32 containing the number of characters the next call to  
15 System.Text.Decoder.GetChars will produce if presented with the specified range of  
16 *bytes*.

17

18 [Note: This value takes into account the state in which the current instance was left  
19 following the last call to System.Text.Decoder.GetChars. This contrasts with  
20 System.Text.Encoding.GetChars, which does not maintain state information across  
21 subsequent calls.]

22

23

## 24 Behaviors

25 As described above.

26

## 27 How and When to Override

28 Override this method to return the appropriate value for a particular encoding.

1

## 2 Usage

3 Use this method to determine the appropriate size of a buffer to contain the decoded  
4 values.

5

## 6 Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>bytes</i> is null.
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0.  -or-  <i>count</i> < 0.  -or-  <i>index</i> and <i>count</i> do not specify a valid range in <i>bytes</i> (i.e. ( <i>index</i> + <i>count</i> ) > <i>bytes.Length</i> ).

7

8

# 1 Decoder.GetChars(System.Byte[], 2 System.Int32, System.Int32, System.Char[], 3 System.Int32) Method

```
4 [ILAsm]  
5 .method public hidebysig virtual abstract int32 GetChars(class  
6 System.Byte[] bytes, int32 byteIndex, int32 byteCount, class System.Char[]  
7 chars, int32 charIndex)  
  
8 [C#]  
9 public abstract int GetChars(byte[] bytes, int byteIndex, int byteCount,  
10 char[] chars, int charIndex)
```

## 11 Summary

12 Decodes the specified range of the specified array of bytes into the specified range of  
13 the specified array of characters for a particular encoding.

## 14 Parameters

Parameter	Description
<i>bytes</i>	A System.Byte array to decode.
<i>byteIndex</i>	A System.Int32 that specifies the first index of <i>bytes</i> from which to decode.
<i>byteCount</i>	A System.Int32 that specifies the number elements in <i>bytes</i> to decode.
<i>chars</i>	A System.Char array of characters to decode into.
<i>charIndex</i>	A System.Int32 that specifies the first index of <i>chars</i> to store the decoded bytes.

## 15 16 Return Value

17 A System.Int32 containing the number of characters decoded into *chars* for a particular  
18 encoding.

## 19 Description

20 [Note: System.Text.Decoder.GetCharCount can be used to determine the exact  
21 number of characters that will be produced for a specified range of bytes. Alternatively,  
22 System.Text.Encoding.GetMaxCharCount of the System.Text.Encoding object that  
23 produced the current instance can be used to determine the maximum number of  
24 characters that might be produced for a specified number of bytes, regardless of the  
25 actual byte values.]

1  
2

### 3 Behaviors

4 As described above.

5

### 6 How and When to Override

7 Override this method to decode the values of a `System.Byte` array for a particular  
8 encoding.

9

### 10 Usage

11 Use this method to decode the elements of a byte array for a particular encoding.

12

### 13 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>chars</i> does not contain sufficient space to store the decoded characters.
<b>System.ArgumentNullException</b>	<i>bytes</i> is null. -or- <i>chars</i> is null.
<b>System.ArgumentOutOfRangeException</b>	<i>byteIndex</i> < 0. -or- <i>byteCount</i> < 0. -or- <i>charIndex</i> < 0. -or-

*byteIndex* and *byteCount* do not specify a valid range in *bytes* (i.e. (*byteIndex* + *byteCount*) > *bytes.Length*).

-or-

*charIndex* > *chars.Length*.

1

2