

System.Text.Decoder Class

```
[ILAsm]
.class public abstract serializable Decoder extends System.Object

[C#]
public abstract class Decoder
```

Assembly Info:

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

Summary

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

Inherits From: System.Object

Library: BCL

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

Description

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

Example

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

```
[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

Decoder() Constructor

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

Summary

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

Description

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

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

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

Summary

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

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.

Return Value

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

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

Behaviors

As described above.

How and When to Override

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
System.ArgumentNullException	<i>bytes</i> is null.
System.ArgumentOutOfRangeException	<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

Decoder.GetChars(System.Byte[], System.Int32, System.Int32, System.Char[], System.Int32) Method

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

Summary

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

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.

Return Value

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

Description

[*Note:* System.Text.Decoder.GetCharCount can be used to determine the exact number of characters that will be produced for a specified range of bytes. Alternatively, System.Text.Encoding.GetMaxCharCount of the System.Text.Encoding object that produced the current instance can be used to determine the maximum number of characters that might be produced for a specified number of bytes, regardless of the 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
System.ArgumentException	<i>chars</i> does not contain sufficient space to store the decoded characters.
System.ArgumentNullException	<i>bytes</i> is null. -or- <i>chars</i> is null.
System.ArgumentOutOfRangeException	<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