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#]

```
using System;
using System.Text;

public class DecoderExample
{
   public static void Main()
   {
      // These bytes in UTF-8 correspond to 3 different
      // Unicode characters - A (U+0041), # (U+0023),
      // and the biohazard symbol (U+2623). Note the
      // biohazard symbol requires 3 bytes in UTF-8
      // (in hex, e2, 98, a3). Decoders store state across
      // multiple calls to GetChars, handling the case
      // when one char spans multiple byte arrays.

      byte[] bytes1 = { 0x41, 0x23, 0xe2 };
      byte[] bytes2 = { 0x98, 0xa3 };
      char[] chars = new char[3];

      Decoder d = Encoding.UTF8.GetDecoder();
      int charLen = d.GetChars(bytes1, 0, bytes1.Length,
                               chars, 0);
      // charLen is 2.

      charLen += d.GetChars(bytes2, 0, bytes2.Length,
                            chars, charLen);
      // charLen is now 3.

      foreach(char c in chars)
         Console.Write("U+{0:x} ", (ushort)c);
   }
}
```

The output is

```
U+41 U+23 U+2623
```

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
bytes	A `System.Byte` array to decode.
index	A `System.Int32` that specifies the first index in *bytes* to decode.
count	A `System.Int32` that specifies the number elements in *bytes* 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.

Usage

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

Exceptions

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

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
bytes	A `System.Byte` array to decode.
byteIndex	A `System.Int32` that specifies the first index of *bytes* from which to decode.
byteCount	A `System.Int32` that specifies the number elements in *bytes* to decode.
chars	A `System.Char` array of characters to decode into.
charIndex	A `System.Int32` that specifies the first index of *chars* 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.]

Behaviors

As described above.

How and When to Override

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

Usage

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

Exceptions

Exception	Condition
System.ArgumentException	*chars* does not contain sufficient space to store the decoded characters.
System.ArgumentNullException	*bytes* is `null`. -or- *chars* is `null`.
System.ArgumentOutOfRangeException	*byteIndex* < 0. -or- *byteCount* < 0. -or- *charIndex* < 0. -or- *byteIndex* and *byteCount* do not specify a valid range in *bytes* (i.e. (*byteIndex* + *byteCount*) > *bytes*.Length). -or-

	charIndex > *chars*.Length.