

System.Text.Encoder Class

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

[C#]
public abstract class Encoder
```

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 characters into blocks of bytes.

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 `System.Text.Encoder`, sequential blocks of characters are converted into blocks of bytes through calls to the `System.Text.Encoder.GetBytes` method. The encoder maintains state between the conversions, allowing it to correctly encode character sequences that span adjacent blocks. An instance of a specific implementation of the `System.Text.Encoder` class is typically obtained through a call to the `System.Text.Encoding.GetEncoder`.]

Example

The following example demonstrates using the `System.Text.UTF8Encoding` class to convert one character array to two byte arrays.

```
[C#]

using System;
using System.Text;

public class EncoderExample
{
```

```

1
2 public static void Main()
3 {
4
5     string str = "Encoder";
6     char[] cAry = str.ToCharArray();
7     UTF8Encoding utf = new UTF8Encoding();
8
9     Encoder e = utf.GetEncoder();
10    int count1 =
11        e.GetByteCount(cAry,0,cAry.Length-4,false);
12    int count2 =
13        e.GetByteCount(cAry,cAry.Length-4,4,true);
14    byte[] bytes1 = new byte[count1];
15    byte[] bytes2 = new byte[count2];
16
17    e.GetBytes(cAry,0,cAry.Length-4,bytes1,0,false);
18    e.GetBytes(cAry,cAry.Length-4,4,bytes2,0,true);
19
20    Console.Write("Bytes1: ");
21    foreach (byte b in bytes1)
22        Console.Write(" '{0}' ", b);
23    Console.WriteLine();
24
25    Console.Write("Bytes2: ");
26    foreach (byte b in bytes2)
27        Console.Write(" '{0}' ", b);
28    Console.WriteLine();
29
30 }
31
32 }
33 The output is
34
35 Bytes1: '69' '110' '99'
36
37
38 Bytes2: '111' '100' '101' '114'
39
40

```

Encoder() Constructor

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

Summary

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

Description

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

Encoder.GetByteCount(System.Char[], System.Int32, System.Int32, System.Boolean) Method

```
[ILAsm]  
.method public hidebysig virtual abstract int32 GetByteCount(char[] chars,  
int32 index, int32 count, bool flush)
```

```
[C#]  
public abstract int GetByteCount(char[] chars, int index, int count, bool  
flush)
```

Summary

Determines the exact number of bytes required to encode the specified range in the specified array of characters.

Parameters

Parameter	Description
<i>chars</i>	A System.Char array of characters to encode.
<i>index</i>	A System.Int32 that specifies the first index of <i>chars</i> to encode.
<i>count</i>	A System.Int32 that specifies the number of elements in <i>chars</i> to encode.
<i>flush</i>	A System.Boolean value that determines whether the current instance flushes its internal state following a conversion. Specify <code>true</code> to flush the internal state of the current instance following a conversion; otherwise, specify <code>false</code> .

Return Value

A System.Int32 containing the number of bytes required to encode the range in *chars* from *index* to *index* + *count* -1 for a particular encoding.

[Note: This value takes into account the state in which the current instance was left following the last call to System.Text.Encoder.GetBytes.]

Description

The state of the current instance is not affected by a call to this method.

Behaviors

1 As described above.

2

3 How and When to Override

4 Override this method to retrieve the exact number of bytes required to encode a
5 specified range of an array of `System.Char` objects for a particular encoding.

6

7 Usage

8 Use this method to determine the exact number of bytes required to encode the
9 specified range of an array of `System.Char` objects for a particular encoding.

10

11 Exceptions

Exception	Condition
System.ArgumentNullException	<i>chars</i> is null.
System.ArgumentOutOfRangeException	Return value is greater than <code>System.Int32.MaxValue</code> .
	-or-
	<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>chars</i> (i.e. (<i>index</i> + <i>count</i>) > <i>chars.Length</i>).

12

13

Encoder.GetBytes(System.Char[], System.Int32, System.Int32, System.Byte[], System.Int32, System.Boolean) Method

```
[ILAsm]  
.method public hidebysig virtual abstract int32 GetBytes(char[] chars,  
int32 charIndex, int32 charCount, class System.Byte[] bytes, int32  
byteIndex, bool flush)  
  
[C#]  
public abstract int GetBytes(char[] chars, int charIndex, int charCount,  
byte[] bytes, int byteIndex, bool flush)
```

Summary

Encodes the specified range of the specified array of characters into the specified range of the specified array of bytes.

Parameters

Parameter	Description
<i>chars</i>	A System.Char array of characters to encode.
<i>charIndex</i>	A System.Int32 that specifies the first index of <i>chars</i> to encode.
<i>charCount</i>	A System.Int32 that specifies the number of elements in <i>chars</i> to encode.
<i>bytes</i>	A System.Byte array to encode into.
<i>byteIndex</i>	A System.Int32 that specifies the first index of <i>bytes</i> to encode into.
<i>flush</i>	A System.Boolean value. Specify true to flush the internal state of the current instance following a conversion; otherwise, specify false. [Note: To ensure correct termination of a sequence of blocks of encoded bytes, it is recommended that the last call to System.Text.Encoder.GetBytes specify true.]

Return Value

A System.Int32 containing the number of bytes encoded into *bytes* for a particular encoding.

1 Description

2 The encoding takes into account the state in which the current instance was left
3 following the last call to this method if *flush* was specified as `true` for that call.

4 Behaviors

5 As described above.

6

7 How and When to Override

8 Override this method to encode the values of an array of `System.Char` objects as an
9 array of `System.Byte` objects for a particular encoding.

10

11 Usage

12 Use this method to encode the values of an array of `System.Char` objects as an array of
13 `System.Byte` objects for a particular encoding.

14

15 Exceptions

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

	<i>byteIndex</i> < 0. -or- (<i>chars.Length</i> - <i>charIndex</i>) < <i>charCount</i> . -or- <i>byteIndex</i> > <i>bytes.Length</i> .
--	--

1

2