

1 System.Text.Encoder Class

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

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 characters into blocks of bytes.

14 Inherits From: System.Object

15

16 **Library:** BCL

17

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

20

21 Description

22 [Note: Following instantiation of a `System.Text.Encoder`, sequential blocks of
23 characters are converted into blocks of bytes through calls to the
24 `System.Text.Encoder.GetBytes` method. The encoder maintains state between the
25 conversions, allowing it to correctly encode character sequences that span adjacent
26 blocks. An instance of a specific implementation of the `System.Text.Encoder` class is
27 typically obtained through a call to the `System.Text.Encoding.GetEncoder`.]
28
29

30 Example

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

```
33 [C#]  
34  
35 using System;  
36 using System.Text;  
37  
38 public class EncoderExample  
39 {
```

```

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

1 Encoder() Constructor

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

6 Summary

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

8 Description

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

11

1 Encoder.GetByteCount(System.Char[], 2 System.Int32, System.Int32, 3 System.Boolean) Method

```
4 [ILAsm]  
5 .method public hidebysig virtual abstract int32 GetByteCount(char[] chars,  
6 int32 index, int32 count, bool flush)  
  
7 [C#]  
8 public abstract int GetByteCount(char[] chars, int index, int count, bool  
9 flush)
```

10 Summary

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

13 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 true to flush the internal state of the current instance following a conversion; otherwise, specify false.

14 15 Return Value

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

18
19 [Note: This value takes into account the state in which the current instance was left
20 following the last call to System.Text.Encoder.GetBytes.]
21
22

23 Description

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

25 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
<code>System.ArgumentNullException</code>	<i>chars</i> is null.
<code>System.ArgumentOutOfRangeException</code>	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

1 Encoder.GetBytes(System.Char[], 2 System.Int32, System.Int32, System.Byte[], 3 System.Int32, System.Boolean) Method

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

11 Summary

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

14 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.]

15 16 Return Value

17 A System.Int32 containing the number of bytes encoded into *bytes* for a particular
18 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-

byteIndex < 0.

-or-

(chars.Length - charIndex) < charCount.

-or-

byteIndex > *bytes.Length.*

1

2