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:* 1.0.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** implementation of the **System.Text.Encoder** class to convert one character array to two byte arrays.

[C#]

```
using System;
using System.Text;

public class EncoderExample
{

   public static void Main()
   {

      string str = "Encoder";
      char[] cAry = str.ToCharArray();
      UTF8Encoding utf = new UTF8Encoding();

      Encoder e = utf.GetEncoder();
      int count1 =
         e.GetByteCount(cAry,0,cAry.Length-4,false);
      int count2 =
         e.GetByteCount(cAry,cAry.Length-4,4,true);
      byte[] bytes1 = new byte[count1];
      byte[] bytes2 = new byte[count2];

      e.GetBytes(cAry,0,cAry.Length-4,bytes1,0,false);
      e.GetBytes(cAry,cAry.Length-4,4,bytes2,0,true);

      Console.Write("Bytes1: ");
      foreach (byte b in bytes1)
         Console.Write(" '{0}' ", b);
      Console.WriteLine();

      Console.Write("Bytes2: ");
      foreach (byte b in bytes2)
         Console.Write(" '{0}' ", b);
      Console.WriteLine();

   }

}
```

The output is

```
Bytes1: '69' '110' '99'


Bytes2: '111' '100' '101' '114'
```

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(class System.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
chars	A **System.Char** array of characters to encode.
index	A **System.Int32** that specifies the first index of *chars* to encode.
count	A **System.Int32** that specifies the number of elements in *chars* to encode.
flush	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**.

Return Value

A **System.Int32** containing the number of bytes required to encode the range in *chars* from *index* to *index* + *count* 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

As described above.

How and When to Override

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

Usage

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

Exceptions

Exception	Condition
System.ArgumentNullException	*chars* is **null**.
System.ArgumentOutOfRangeException	Return value is greater than **System.Int32.MaxValue**. -or- *index* < 0. -or- *count* < 0. -or- *index* and *count* do not specify a valid range in *chars* (i.e. (*index* + *count*) > *chars*.Length).

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

[ILASM]

```
.method public hidebysig virtual abstract int32
GetBytes(class System.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
chars	A **System.Char** array of characters to encode.
charIndex	A **System.Int32** that specifies the first index of *chars* to encode.
charCount	A **System.Int32** that specifies the number of elements in *chars* to encode.
bytes	A **System.Byte** array to encode into.
byteIndex	A **System.Int32** that specifies the first index of *bytes* to encode into.
flush	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.

Description

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

Behaviors

As described above.

How and When to Override

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

Usage

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

Exceptions

Exception	Condition
System.ArgumentException	*bytes* does not contain sufficient space to store the encoded characters.
System.ArgumentNullException	*chars* is **null**. -or- *bytes* is **null**.
System.ArgumentOutOfRangeException	*charIndex* < 0. -or- *charCount* < 0. -or- *byteIndex* < 0. -or- (*chars*.Length - *charIndex*) < *charCount*. -or-

	byteIndex > *bytes*.Length.