

# 1 System.Text.Encoding Class

2  
3

```
4 [ILASM]  
5 .class public abstract serializable Encoding extends  
6 System.Object  
  
7 [C#]  
8 public abstract class Encoding
```

## 9 Assembly Info:

- 10 • *Name:* mscorlib
- 11 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 12 • *Version:* 1.0.x.x
- 13 • *Attributes:*
  - 14 ○ CLSCompliantAttribute(true)

## 15 Summary

16

17 Represents a character encoding.

## 18 Inherits From: System.Object

19

20 **Library:** BCL

21

22 **Thread Safety:** This type is safe for multithreaded operations.

23

## 24 Description

25

26 Characters are abstract entities that can be represented using many  
27 different character schemes or codepages. For example, Unicode UTF-  
28 16 encoding represents, or encodes, characters as sequences of 16-bit  
29 integers while Unicode UTF-8 represents the same characters as  
30 sequences of 8-bit bytes.

31

31 The BCL includes the following types derived from

32

### **System.Text.Encoding:**

33

- **System.Text.ASCIIEncoding** - encodes Unicode characters as 7-bit ASCII characters. This encoding only supports code points between U+0000 and U+007F inclusive.

34

35

36

- **System.Text.UnicodeEncoding** - encodes each Unicode character as two consecutive bytes. Both little-endian and big-endian byte orders are supported.

37

38

- **System.Text.UTF8Encoding** - encodes Unicode characters using the UTF-8 (UCS Transformation Format, 8-bit form) encoding. This encoding supports all Unicode character values.

An application can use the properties of this class such as **System.Text.Encoding.ASCII**, **System.Text.Encoding.Default**, **System.Text.Encoding.Unicode**, and **System.Text.Encoding.UTF8** to obtain encodings. Applications can initialize new instances of **System.Text.Encoding** objects through the **System.Text.ASCIIEncoding**, **System.Text.UnicodeEncoding**, and **System.Text.UTF8Encoding** classes.

Through an encoding, the **System.Text.Encoding.GetBytes** method is used to convert arrays of Unicode characters to arrays of bytes, and the **System.Text.Encoding.GetChars** method is used to convert arrays of bytes to arrays of Unicode characters. The **System.Text.Encoding.GetBytes** and **System.Text.Encoding.GetChars** methods maintain no state between conversions. When the data to be converted is only available in sequential blocks (such as data read from a stream) or when the amount of data is so large that it needs to be divided into smaller blocks, an application can choose to use a **System.Text.Decoder** or a **System.Text.Encoder** to perform the conversion. Decoders and encoders allow sequential blocks of data to be converted and they maintain the state required to support conversions of data that spans adjacent blocks. Decoders and encoders are obtained using the **System.Text.Encoding.GetDecoder** and **System.Text.Encoding.GetEncoder** methods.

The core **System.Text.Encoding.GetBytes** and **System.Text.Encoding.GetChars** methods require the caller to provide the destination buffer and ensure that the buffer is large enough to hold the entire result of the conversion. When using these methods, either directly on a **System.Text.Encoding** object or on an associated **System.Text.Decoder** or **System.Text.Encoder**, an application can use one of two methods to allocate destination buffers.

1. The **System.Text.Encoding.GetByteCount** and **System.Text.Encoding.GetCharCount** methods can be used to compute the exact size of the result of a particular conversion, and an appropriately sized buffer for that conversion can then be allocated.
2. The **System.Text.Encoding.GetMaxByteCount** and **System.Text.Encoding.GetMaxCharCount** methods can be used to compute the maximum possible size of a conversion of a given number of characters or bytes, regardless of the actual

1 character or byte values, and a buffer of that size can then be  
2 reused for multiple conversions.

3 The first method generally uses less memory, whereas the second  
4 method generally executes faster.

5

# 1 Encoding() Constructor

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

## 6 Summary

7 Constructs a new instance of the **System.Text.Encoding** class.

8

# 1 Encoding.Convert(System.Text.Encoding, 2 System.Text.Encoding, System.Byte[]) 3 Method

```
4 [ILASM]  
5 .method public hidebysig static class System.Byte[]  
6 Convert(class System.Text.Encoding srcEncoding, class  
7 System.Text.Encoding dstEncoding, class System.Byte[]  
8 bytes)  
9  
10 [C#]  
11 public static byte[] Convert(Encoding srcEncoding, Encoding  
dstEncoding, byte[] bytes)
```

## 12 Summary

13 Converts the specified **System.Byte** array from one specified  
14 encoding to another specified encoding.

## 15 Parameters

16  
17

Parameter	Description
<i>srcEncoding</i>	The <b>System.Text.Encoding</b> that <i>bytes</i> is in.
<i>dstEncoding</i>	The <b>System.Text.Encoding</b> desired for the returned <b>System.Byte</b> array.
<i>bytes</i>	The <b>System.Byte</b> array containing the values to convert.

18  
19  
20

## Return Value

21 A **System.Byte** array containing the result of the conversion.

## 22 Exceptions

23  
24

Exception	Condition
<b>System.ArgumentNullException</b>	<i>srcEncoding</i> , <i>dstEncoding</i> or <i>bytes</i> is <b>null</b> .

25  
26  
27

# Encoding.Convert(System.Text.Encoding, System.Text.Encoding, System.Byte[], System.Int32, System.Int32) Method

```
[ILASM]
.method public hidebysig static class System.Byte[]
Convert(class System.Text.Encoding srcEncoding, class
System.Text.Encoding dstEncoding, class System.Byte[]
bytes, int32 index, int32 count)

[C#]
public static byte[] Convert(Encoding srcEncoding, Encoding
dstEncoding, byte[] bytes, int index, int count)
```

## Summary

Converts the specified range of the specified **System.Byte** array from one specified encoding to another specified encoding.

## Parameters

Parameter	Description
<i>srcEncoding</i>	The <b>System.Text.Encoding</b> that <i>bytes</i> is in.
<i>dstEncoding</i>	The <b>System.Text.Encoding</b> desired for the returned <b>System.Byte</b> array.
<i>bytes</i>	The <b>System.Byte</b> array containing the values to convert.
<i>index</i>	A <b>System.Int32</b> containing the first index of <i>bytes</i> from which to convert.
<i>count</i>	A <b>System.Int32</b> containing the number of bytes to convert.

## Return Value

A **System.Byte** array containing the result of the conversion.

## Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>srcEncoding</i> , <i>dstEncoding</i> , or <i>bytes</i> is <b>null</b> .
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> and <i>count</i> do not denote a valid range in <i>bytes</i> .

1  
2  
3

# 1 Encoding.Equals(System.Object) Method

```
2 [ILASM]  
3 .method public hidebysig virtual bool Equals(object value)  
4 [C#]  
5 public override bool Equals(object value)
```

## 6 Summary

7 Determines whether the current instance and the specified  
8 **System.Object** represent the same type and value.

## 9 Parameters

10  
11

Parameter	Description
<i>value</i>	The <b>System.Object</b> to compare to the current instance.

12  
13  
14

## Return Value

15 **true** if *obj* represents the same type and value as the current  
16 instance. If *obj* is a null reference or is not an instance of  
17 **System.Text.Encoding**, returns **false**.

## 18 Description

19 [Note: This method overrides **System.Object.Equals**.]  
20

# 1 Encoding.GetByteCount(System.Char[], 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract int32  
5 GetByteCount(class System.Char[] chars, int32 index, int32  
6 count)  
  
7 [C#]  
8 public abstract int GetByteCount(char[] chars, int index,  
9 int count)
```

## 10 Summary

11 Returns the number of bytes required to encode the specified range of  
12 characters in the specified Unicode character array.

## 13 Parameters

Parameter	Description
<i>chars</i>	The <b>System.Char</b> array to encode.
<i>index</i>	A <b>System.Int32</b> containing the first index of <i>chars</i> to encode.
<i>count</i>	A <b>System.Int32</b> containing the number of characters to encode.

## 16 Return Value

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

## 21 Behaviors

22 As described above.

## 23 How and When to Override

24 This method is overridden by types derived from  
25 **System.Text.Encoding** to return the appropriate number of bytes for  
26 the particular encoding.

## 27 Usage

28 **System.Text.Encoding.GetByteCount** can be used to determine the  
29 exact the number of bytes that will be produced from encoding a given  
30 range of characters. An appropriately sized buffer for that conversion  
31 can then be allocated.  
32

1 Alternatively, **System.Text.Encoding.GetMaxByteCount** can be  
2 used to determine the maximum number of bytes that will be  
3 produced from converting a given number of characters, regardless of  
4 the actual character values. A buffer of that size can then be reused  
5 for multiple conversions.  
6

7 **System.Text.Encoding.GetByteCount** generally uses less memory  
8 and **System.Text.Encoding.GetMaxByteCount** generally executes  
9 faster.

## 10 Exceptions

11  
12

Exception	Condition
<b>System.ArgumentNullException</b>	<i>chars</i> is <b>null</b> .
	The number of bytes required to encode the specified elements in <i>chars</i> is greater than <b>System.Int32.MaxValue</b> .
	-or-
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> or <i>count</i> is less than zero.
	-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> ).

13  
14  
15

# 1 Encoding.GetByteCount(System.String)

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 GetByteCount(string  
5 s)  
6  
7 [C#]  
8 public virtual int GetByteCount(string s)
```

### 8 Summary

9 Returns the number of bytes required to encode the specified  
10 **System.String**.

### 11 Parameters

12  
13

Parameter	Description
s	The <b>System.String</b> to decode.

14  
15  
16

### 15 Return Value

17 A **System.Int32** containing the number of bytes needed to encode s.

### 18 Behaviors

19 As described above.

### 20 How and When to Override

21 This method is overridden by types derived from  
22 **System.Text.Encoding** to return the appropriate number of bytes for  
23 the particular encoding.

### 24 Usage

25 **System.Text.Encoding.GetByteCount** can be used to determine the  
26 exact number of bytes that will be produced from encoding the given  
27 **System.String**. An appropriately sized buffer for that conversion can  
28 then be allocated.

29  
30 Alternatively, **System.Text.Encoding.GetMaxByteCount** can be  
31 used to determine the maximum number of bytes that will be  
32 produced from converting a given number of characters, regardless of  
33 the actual character values. A buffer of that size can then be reused  
34 for multiple conversions.

1  
2  
3  
4

**System.Text.Encoding.GetByteCount** generally uses less memory and **System.Text.Encoding.GetMaxByteCount** generally executes faster.

5  
6  
7

#### Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	s is <b>null</b> .

8  
9  
10

# 1 Encoding.GetByteCount(System.Char[])

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 GetByteCount(class  
5 System.Char[] chars)  
  
6 [C#]  
7 public virtual int GetByteCount(char[] chars)
```

### 8 Summary

9 Returns the number of bytes required to encode the specified  
10 **System.Char** array.

### 11 Parameters

12  
13

Parameter	Description
<i>chars</i>	The <b>System.Char</b> array to encode.

14  
15  
16

### 15 Return Value

17 A **System.Int32** containing the number of bytes needed to encode  
18 *chars*.

### 19 Behaviors

20 As described above.

### 21 How and When to Override

22 This method is overridden by types derived from  
23 **System.Text.Encoding** to return the appropriate number of bytes for  
24 the particular encoding.

### 25 Usage

26 **System.Text.Encoding.GetByteCount** can be used to determine the  
27 exact number of bytes that will be produced from encoding the given  
28 array of characters. An appropriately sized buffer for that conversion  
29 can then be allocated.

30  
31 Alternatively, **System.Text.Encoding.GetMaxByteCount** can be  
32 used to determine the maximum number of bytes that will be  
33 produced from converting a given number of characters, regardless of  
34 the actual character values. A buffer of that size can then be reused

1 for multiple conversions.

2

3 **System.Text.Encoding.GetByteCount** generally uses less memory  
4 and **System.Text.Encoding.GetMaxByteCount** generally executes  
5 faster.

6 **Exceptions**

7

8

Exception	Condition
System.ArgumentNullException	<i>chars</i> is <b>null</b> .

9

10

11

# 1 Encoding.GetBytes(System.String, 2 System.Int32, System.Int32, 3 System.Byte[], System.Int32) Method

```
4 [ILASM]  
5 .method public hidebysig virtual int32 GetBytes(string s,  
6 int32 charIndex, int32 charCount, class System.Byte[]  
7 bytes, int32 byteIndex)
```

```
8 [C#]  
9 public virtual int GetBytes(string s, int charIndex, int  
10 charCount, byte[] bytes, int byteIndex)
```

## 11 Summary

12 Encodes the specified range of the specified **System.String** into the  
13 specified range of the specified **System.Byte** array.

## 14 Parameters

15  
16

Parameter	Description
<i>s</i>	A <b>System.String</b> to encode.
<i>charIndex</i>	A <b>System.Int32</b> containing the first index of <i>s</i> from which to encode.
<i>charCount</i>	A <b>System.Int32</b> containing the number of characters of <i>s</i> to encode.
<i>bytes</i>	The <b>System.Byte</b> array to encode into.
<i>byteIndex</i>	A <b>System.Int32</b> containing the first index of <i>bytes</i> to encode into.

17  
18  
19

## Return Value

20 A **System.Int32** containing the number of bytes encoded into *bytes*.

## 21 Behaviors

22 As described above.

## 23 How and When to Override

24 This method is overridden by types derived from  
25 **System.Text.Encoding** to perform the encoding.

## 26 Exceptions

27  
28

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

1  
2  
3

# 1 Encoding.GetBytes(System.String)

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual class System.Byte[]  
5 GetBytes(string s)  
  
6 [C#]  
7 public virtual byte[] GetBytes(string s)
```

### 8 Summary

9 Encodes the specified **System.String**.

### 10 Parameters

11  
12

Parameter	Description
s	The <b>System.String</b> to encode.

13  
14  
15

### 14 Return Value

16 A **System.Byte** array containing the encoded representation of s.

### 17 Behaviors

18 As described above.

### 19 How and When to Override

20 This method is overridden by types derived from  
21 **System.Text.Encoding** to perform the encoding.

### 22 Exceptions

23  
24

Exception	Condition
<b>System.ArgumentNullException</b>	s is <b>null</b> .

25  
26  
27

# 1 Encoding.GetBytes(System.Char[], 2 System.Int32, System.Int32, 3 System.Byte[], System.Int32) Method

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

## 11 Summary

12 Encodes the specified range of the specified **System.Char** array into  
13 the specified range of the specified **System.Byte** array.

## 14 Parameters

15  
16

Parameter	Description
<i>chars</i>	A <b>System.Char</b> array to encode.
<i>charIndex</i>	A <b>System.Int32</b> containing the first index of <i>chars</i> to encode.
<i>charCount</i>	A <b>System.Int32</b> containing the number of characters to encode.
<i>bytes</i>	A <b>System.Byte</b> array to encode into.
<i>byteIndex</i>	A <b>System.Int32</b> containing the first index of <i>bytes</i> to encode into.

17  
18  
19

## Return Value

20 The number of bytes encoded into *bytes*.

## 21 Behaviors

22 As described above.

## 23 How and When to Override

24 This method is overridden by types derived from  
25 **System.Text.Encoding** to perform the encoding.

## 26 Usage

1 **System.Text.Encoding.GetByteCount** can be used to determine the  
2 exact number of bytes that will be produced for a given range of  
3 characters. Alternatively, **System.Text.Encoding.GetMaxByteCount**  
4 can be used to determine the maximum number of bytes that will be  
5 produced for a given number of characters, regardless of the actual  
6 character values.

7 **Exceptions**

8  
9

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

10  
11  
12

# 1 Encoding.GetBytes(System.Char[], 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual class System.Byte[]  
5 GetBytes(class System.Char[] chars, int32 index, int32  
6 count)  
  
7 [C#]  
8 public virtual byte[] GetBytes(char[] chars, int index, int  
9 count)
```

## 10 Summary

11 Encodes the specified range of the specified **System.Char** array.

## 12 Parameters

13  
14

Parameter	Description
<i>chars</i>	The <b>System.Char</b> array to encode.
<i>index</i>	A <b>System.Int32</b> containing the first index of <i>chars</i> to encode.
<i>count</i>	A <b>System.Int32</b> containing the number of characters to encode.

15  
16  
17

## 16 Return Value

18 A **System.Byte** array containing the encoded representation of the  
19 range in *chars* from *index* to *index + count*.

## 20 Behaviors

21 As described above.

## 22 How and When to Override

23 This method is overridden by types derived from  
24 **System.Text.Encoding** to perform the encoding.

## 25 Exceptions

26  
27

Exception	Condition
<b>System.ArgumentNullException</b>	<i>chars</i> is <b>null</b> .
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> and <i>count</i> do not denote a valid range in <i>chars</i> .

1  
2  
3

# 1 Encoding.GetBytes(System.Char[])

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual class System.Byte[]  
5 GetBytes(class System.Char[] chars)  
  
6 [C#]  
7 public virtual byte[] GetBytes(char[] chars)
```

### 8 Summary

9 Encodes the specified **System.Char** array.

### 10 Parameters

11  
12

Parameter	Description
<i>chars</i>	The <b>System.Char</b> array to encode.

13

### 14 Return Value

15

16 A **System.Byte** array containing the encoded representation of *chars*.

### 17 Behaviors

18 As described above.

### 19 How and When to Override

20 This method is overridden by types derived from  
21 **System.Text.Encoding** to perform the encoding.

### 22 Exceptions

23

24

Exception	Condition
<b>System.ArgumentNullException</b>	<i>chars</i> is <b>null</b> .

25

26

27

# 1 Encoding.GetCharCount(System.Byte[], 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract int32  
5 GetCharCount(class System.Byte[] bytes, int32 index, int32  
6 count)  
  
7 [C#]  
8 public abstract int GetCharCount(byte[] bytes, int index,  
9 int count)
```

## 10 Summary

11 Determines the exact number of characters that will be produced by  
12 decoding the specified range of the specified **System.Byte** array.

## 13 Parameters

Parameter	Description
<i>bytes</i>	The <b>System.Byte</b> array to decode.
<i>index</i>	The first index in <i>bytes</i> to decode.
<i>count</i>	The number of bytes to decode.

## 16 Return Value

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

## 22 Behaviors

23 As described above.

## 24 How and When to Override

25 This method is overridden by types derived from  
26 **System.Text.Encoding** to return the appropriate number of bytes for  
27 the particular encoding.

## 28 Usage

29 Use **System.Text.Encoding.GetCharCount** to determine the exact  
30 number of characters that will be produced from converting a given  
31 range of bytes. An appropriately sized buffer for that conversion can

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

then be allocated.

Alternatively, use **System.Text.Encoding.GetMaxCharCount** to determine the maximum number of characters that will be produced for a given number of bytes, regardless of the actual byte values. A buffer of that size can then be reused for multiple conversions.

**System.Text.Encoding.GetCharCount** generally uses less memory and **System.Text.Encoding.GetMaxCharCount** generally executes faster.

11 **Exceptions**  
12  
13

Exception	Condition
<b>System.ArgumentNullException</b>	<i>bytes</i> is <b>null</b> .
<b>System.ArgumentOutOfRangeException</b>	<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> ).

14  
15  
16

# 1 Encoding.GetCharCount(System.Byte[])

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 GetCharCount(class  
5 System.Byte[] bytes)  
  
6 [C#]  
7 public virtual int GetCharCount(byte[] bytes)
```

### 8 Summary

9 Determines the exact number of characters that will be produced by  
10 decoding the specified **System.Byte** array.

### 11 Parameters

12  
13

Parameter	Description
<i>bytes</i>	The <b>System.Byte</b> array to decode.

14  
15  
16

### Return Value

17 A **System.Int32** containing the number of characters produced by  
18 decoding *bytes*.

### 19 Behaviors

20 As described above.

### 21 How and When to Override

22 This method is overridden by types derived from  
23 **System.Text.Encoding** to return the appropriate number of bytes for  
24 the particular encoding.

### 25 Usage

26 Use **System.Text.Encoding.GetCharCount** to determine the exact  
27 number of characters that will be produced from converting a given  
28 byte array. An appropriately sized buffer for that conversion can then  
29 be allocated.

30  
31  
32  
33  
34

Alternatively, use **System.Text.Encoding.GetMaxCharCount** to  
determine the maximum number of characters that will be produced  
for a given number of bytes, regardless of the actual byte values. A  
buffer of that size can then be reused for multiple conversions.

1  
2  
3  
4

**System.Text.Encoding.GetCharCount** generally uses less memory and **System.Text.Encoding.GetMaxCharCount** generally executes faster.

5  
6  
7

#### Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>bytes</i> is <b>null</b> .

8  
9  
10

# 1 Encoding.GetChars(System.Byte[], 2 System.Int32, System.Int32, 3 System.Char[], System.Int32) Method

```
4 [ILASM]  
5 .method public hidebysig virtual abstract int32  
6 GetChars(class System.Byte[] bytes, int32 byteIndex, int32  
7 byteCount, class System.Char[] chars, int32 charIndex)  
  
8 [C#]  
9 public abstract int GetChars(byte[] bytes, int byteIndex,  
10 int byteCount, char[] chars, int charIndex)
```

## 11 Summary

12 Decodes the specified range of the specified **System.Byte** array into  
13 the specified range of the specified **System.Char** array.

## 14 Parameters

15  
16

Parameter	Description
<i>bytes</i>	The <b>System.Byte</b> array to decode.
<i>byteIndex</i>	A <b>System.Int32</b> containing the first index of <i>bytes</i> to decode.
<i>byteCount</i>	A <b>System.Int32</b> containing the number of bytes to decode.
<i>chars</i>	The <b>System.Char</b> array to decode into.
<i>charIndex</i>	A <b>System.Int32</b> containing the first index of <i>chars</i> to decode into.

17  
18  
19

## Return Value

20 The number of characters stored in *chars*.

## 21 Behaviors

22 This method requires the caller to provide the destination buffer and  
23 ensure that the buffer is large enough to hold the entire result of the  
24 conversion.

## 25 How and When to Override

26 This method is overridden by types derived from  
27 **System.Text.Encoding** to perform the particular decoding.

## 28 Usage

1 When using this method directly on a **System.Text.Encoding** object  
2 or on an associated **System.Text.Decoder** or  
3 **System.Text.Encoder**, use **System.Text.Encoding.GetCharCount**  
4 or **System.Text.Encoding.GetMaxCharCount** to allocate destination  
5 buffers.

6 **Exceptions**

7  
8

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

9  
10  
11

# 1 Encoding.GetChars(System.Byte[], 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual class System.Char[]  
5 GetChars(class System.Byte[] bytes, int32 index, int32  
6 count)  
  
7 [C#]  
8 public virtual char[] GetChars(byte[] bytes, int index, int  
9 count)
```

## 10 Summary

11 Decodes the specified range of the specified **System.Byte** array.

## 12 Parameters

13  
14

Parameter	Description
<i>bytes</i>	The <b>System.Byte</b> array to decode.
<i>index</i>	A <b>System.Int32</b> containing the first index of <i>bytes</i> to decode.
<i>count</i>	A <b>System.Int32</b> containing the number of bytes to decode.

15  
16  
17

## 16 Return Value

18 A **System.Char** array containing the decoded representation of the  
19 range in *bytes* between *index* to *index + count*.

## 20 Exceptions

21  
22

Exception	Condition
<b>System.ArgumentNullException</b>	<i>bytes</i> is <b>null</b> .
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> and <i>count</i> do not denote a valid range in the byte array.

23  
24  
25

# 1 Encoding.GetChars(System.Byte[])

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual class System.Char[]  
5 GetChars(class System.Byte[] bytes)  
6  
7 [C#]  
8 public virtual char[] GetChars(byte[] bytes)
```

### 8 Summary

9 Decodes a **System.Byte** array.

### 10 Parameters

11  
12

Parameter	Description
<i>bytes</i>	The <b>System.Byte</b> array to decode.

13  
14  
15

### 14 Return Value

16 A **System.Char** array produced by decoding *bytes*.

### 17 Exceptions

18  
19

Exception	Condition
<b>System.ArgumentNullException</b>	<i>bytes</i> is <b>null</b> .

20  
21  
22

# 1 Encoding.GetDecoder() Method

```
2 [ILASM]  
3 .method public hidebysig virtual class System.Text.Decoder  
4 GetDecoder()  
5 [C#]  
6 public virtual Decoder GetDecoder()
```

## 7 Summary

8 Returns a **System.Text.Decoder** for the current instance.

## 9 Return Value

10

11 A **System.Text.Decoder** for the current instance.

## 12 Behaviors

13 As described above.

## 14 Default

15 The default implementation returns a **System.Text.Decoder** that  
16 forwards calls made to the **System.Text.Encoding.GetCharCount**  
17 and **System.Text.Encoding.GetChars** methods to the corresponding  
18 methods of the current instance.

## 19 How and When to Override

20 Encoding that requires state to be maintained between successive  
21 conversions should override this method and return an instance of an  
22 appropriate **System.Text.Decoder** implementation.

## 23 Usage

24 Unlike the **System.Text.Encoding.GetChars** methods, a  
25 **System.Text.Decoder** can convert partial sequences of bytes into  
26 partial sequences of characters by maintaining the appropriate state  
27 between the conversions.

28

# 1 Encoding.GetEncoder() Method

```
2 [ILASM]  
3 .method public hidebysig virtual class System.Text.Encoder  
4 GetEncoder()  
  
5 [C#]  
6 public virtual Encoder GetEncoder()
```

## 7 Summary

8 Returns a **System.Text.Encoder** for the current instance.

## 9 Return Value

10

11 A **System.Text.Encoder** for the current encoding.

## 12 Behaviors

13 As described above.

## 14 Default

15 The default implementation returns a **System.Text.Encoder** that  
16 forwards calls made to the **System.Text.Encoding.GetByteCount**  
17 and **System.Text.Encoding.GetBytes** methods to the corresponding  
18 methods of the current instance.

## 19 How and When to Override

20 Types derived from **System.Text.Encoding** override this method to  
21 return an instance of an appropriate **System.Text.Encoder**.

## 22 Usage

23 Unlike the **System.Text.Encoding.GetBytes** method, a  
24 **System.Text.Encoder** can convert partial sequences of characters  
25 into partial sequences of bytes by maintaining the appropriate state  
26 between the conversions.

27

# 1 Encoding.GetHashCode() Method

```
2 [ILASM]  
3 .method public hidebysig virtual int32 GetHashCode()  
4 [C#]  
5 public override int GetHashCode()
```

## 6 Summary

7 Generates a hash code for the current instance.

## 8 Return Value

9

10 A **System.Int32** containing the hash code for the current instance.

## 11 Description

12 The algorithm used to generate the hash code is unspecified.

13

14 [*Note:* This method overrides **System.Object.GetHashCode.**]

15

# 1 Encoding.GetMaxByteCount(System.Int32 2 ) Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract int32  
5 GetMaxByteCount(int32 charCount)  
  
6 [C#]  
7 public abstract int GetMaxByteCount(int charCount)
```

## 8 Summary

9 Returns the maximum number of bytes required to encode the  
10 specified number of characters, regardless of the actual character  
11 values.

## 12 Parameters

13  
14

Parameter	Description
<i>charCount</i>	A <b>System.Int32</b> containing the number of characters to encode.

15  
16  
17

## 16 Return Value

18 A **System.Int32** containing the maximum number of bytes required  
19 to encode *charCount* characters.

## 20 Behaviors

21 As described above.

## 22 How and When to Override

23 This method is overridden by types derived from  
24 **System.Text.Encoding** to return the appropriate number of bytes for  
25 the particular encoding.

## 26 Usage

27 **System.Text.Encoding.GetMaxByteCount** can be used to  
28 determine the minimum buffer size for byte arrays passed to the  
29 **System.Text.Encoding.GetBytes** of the current encoding. Using this  
30 minimum buffer size ensures that no buffer overflow exceptions occur.

31

# 1 Encoding.GetMaxCharCount(System.Int32 2 ) Method

```
3 [ILASM]  
4 .method public hidebysig virtual abstract int32  
5 GetMaxCharCount(int32 byteCount)  
  
6 [C#]  
7 public abstract int GetMaxCharCount(int byteCount)
```

## 8 Summary

9 Returns the maximum number of characters produced by decoding the  
10 specified number of bytes, regardless of the actual byte values.

## 11 Parameters

12  
13

Parameter	Description
<i>byteCount</i>	A <b>System.Int32</b> containing the number of bytes to decode.

14  
15  
16

## 15 Return Value

17 A **System.Int32** containing the maximum number of characters that  
18 would be produced by decoding *byteCount* bytes.

## 19 Behaviors

20 As described above.

## 21 How and When to Override

22 This method is overridden by types derived from  
23 **System.Text.Encoding** to return the appropriate number of bytes for  
24 the particular encoding.

## 25 Usage

26 **System.Text.Encoding.GetMaxCharCount** can be used to  
27 determine the minimum buffer size for byte arrays passed to the  
28 **System.Text.Encoding.GetChars** of the current encoding. Using this  
29 minimum buffer size ensures that no buffer overflow exceptions will  
30 occur.

31

# 1 Encoding.GetPreamble() Method

```
2 [ILASM]  
3 .method public hidebysig virtual class System.Byte[]  
4 GetPreamble()  
  
5 [C#]  
6 public virtual byte[] GetPreamble()
```

## 7 Summary

8 Returns the bytes used at the beginning of a **System.IO.Stream** to  
9 determine which **System.Text.Encoding** the stream was created  
10 with.

## 11 Return Value

12

13 A **System.Byte** array that identifies the encoding used on a stream.

## 14 Description

15 [*Note:* The preamble can be the Unicode byte order mark (U+FEFF  
16 written in the appropriate encoding) or any other type of identifying  
17 marks. This method can return an empty array.]

## 18 Behaviors

19 As described above.

## 20 Default

21 The default implementation returns an empty **System.Byte** array.

## 22 How and When to Override

23 Override this method to return a **System.Byte** array containing the  
24 preamble appropriate for the type derived from  
25 **System.Text.Encoding**.

26

# Encoding.GetString(System.Byte[], System.Int32, System.Int32) Method

```
[ILASM]
.method public hidebysig virtual string GetString(class
System.Byte[] bytes, int32 index, int32 count)

[C#]
public virtual string GetString(byte[] bytes, int index,
int count)
```

## Summary

Decodes the specified range of the specified **System.Byte** array.

## Parameters

Parameter	Description
<i>bytes</i>	The <b>System.Byte</b> array to decode.
<i>index</i>	A <b>System.Int32</b> containing the starting index of <i>bytes</i> to decode.
<i>count</i>	A <b>System.Int32</b> containing the number of bytes to decode.

## Return Value

A **System.String** containing the decoded representation of the range of *bytes* from *index* to *index* + *count*.

## Behaviors

As described above.

## How and When to Override

This method is overridden by particular character encodings.

## Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>bytes</i> is <b>null</b> .
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> and <i>count</i> do not denote a valid range in <i>bytes</i> .

# 1 Encoding.GetString(System.Byte[])

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual string GetString(class  
5 System.Byte[] bytes)  
  
6 [C#]  
7 public virtual string GetString(byte[] bytes)
```

### 8 Summary

9 Decodes the specified **System.Byte** array.

### 10 Parameters

11  
12

Parameter	Description
<i>bytes</i>	The <b>System.Byte</b> array to decode.

13  
14  
15

### 14 Return Value

16 A **System.String** containing the decoded representation of *bytes*.

### 17 Behaviors

18 As described above.

### 19 How and When to Override

20 This method is overridden by particular character encodings.

### 21 Exceptions

22  
23

Exception	Condition
<b>System.ArgumentNullException</b>	<i>bytes</i> is <b>null</b> .

24  
25  
26

# 1 Encoding.ASCII Property

```
2 [ILASM]
3 .property class System.Text.Encoding ASCII { public
4 hidebysig static specialname class System.Text.Encoding
5 get_ASCII() }
6
7 [C#]
8 public static Encoding ASCII { get; }
```

## 8 Summary

9 Gets an encoding for the ASCII (7-bit) character set.

## 10 Description

11 This property is read-only.

12

13 [*Note:* ASCII characters can represent Unicode characters from  
14 U+0000 to U+007f, inclusive.]

15

# 1 Encoding.BigEndianUnicode Property

```
2 [ILASM]
3 .property class System.Text.Encoding BigEndianUnicode {
4 public hidebysig static specialname class
5 System.Text.Encoding get_BigEndianUnicode() }
6
7 [C#]
8 public static Encoding BigEndianUnicode { get; }
```

## 8 Summary

9 Gets an encoding for the Unicode format in big-endian byte order.

## 10 Property Value

11

12 A **System.Text.Encoding** for the Unicode format in big-endian byte  
13 order.

## 14 Description

15 This property is read-only.

16

17 [*Note:* Unicode characters can be stored in two different byte orders,  
18 big-endian and little-endian. On little-endian platforms such as those  
19 implemented on Intel processors, it is generally more efficient to store  
20 Unicode characters in little-endian byte order. However, many other  
21 platforms can store Unicode characters in big-endian byte order.  
22 Unicode files can be distinguished by the presence of the byte order  
23 mark (U+FEFF), which will be written as either 0xfe 0xff or 0xff 0xfe.

24

25 This encoding automatically detects a byte order mark and, if  
26 necessary, switches byte orders.]

27

# 1 Encoding.Default Property

```
2 [ILASM]
3 .property class System.Text.Encoding Default { public
4 hidebysig static specialname class System.Text.Encoding
5 get_Default() }
6
7 [C#]
8 public static Encoding Default { get; }
```

## 8 Summary

9 Gets an encoding for the ANSI code page of the current system.

## 10 Property Value

11

12 A **System.Text.Encoding** for the ANSI code page of the current  
13 system.

## 14 Description

15 This property is read-only.

16

# 1 Encoding.Unicode Property

```
2 [ILASM]
3 .property class System.Text.Encoding Unicode { public
4 hidebysig static specialname class System.Text.Encoding
5 get_Unicode() }
6
7 [C#]
8 public static Encoding Unicode { get; }
```

## 8 Summary

9 Gets an encoding for the Unicode format in little-endian byte order.

## 10 Property Value

11

12 A **System.Text.Encoding** for the Unicode format in little-endian byte  
13 order.

## 14 Description

15 This property is read-only.

16

17 [*Note:* Unicode characters can be stored in two different byte orders,  
18 big-endian and little-endian. On little-endian platforms such as those  
19 implemented on Intel processors, it is generally more efficient to store  
20 Unicode characters in little-endian byte order. However, many other  
21 platforms can store Unicode characters in big-endian byte order.  
22 Unicode files can be distinguished by the presence of the byte order  
23 mark (U+FEFF), which will be written as either 0xfe 0xff or 0xff 0xfe.

24

25 This encoding automatically detects a byte order mark and, if  
26 necessary, switches byte orders.]

27

# 1 Encoding.UTF8 Property

```
2 [ILASM]
3 .property class System.Text.Encoding UTF8 { public
4 hidebysig static specialname class System.Text.Encoding
5 get_UTF8() }
6
7 [C#]
8 public static Encoding UTF8 { get; }
```

## 8 Summary

9 Gets an encoding for the UTF-8 format.

## 10 Property Value

11

12 A **System.Text.Encoding** for the UTF-8 format.

## 13 Description

14 This property is read-only.

15

16 [*Note:* For detailed information regarding UTF-8 encoding, see  
17 **System.Text.UTF8Encoding.**]

18