

1 System.Text.Encoding Class

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

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 Represents a character encoding.

14 Inherits From: System.Object

15

16 **Library:** BCL

17

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

19

20 Description

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

25

26 The BCL includes the following types derived from `System.Text.Encoding`:

- 27 • `System.Text.ASCIIEncoding` - encodes Unicode characters as 7-bit ASCII
28 characters. This encoding only supports code points between U+0000 and U+007F
29 inclusive.
- 30 • `System.Text.UnicodeEncoding` - encodes each Unicode character as two
31 consecutive bytes. Both little-endian and big-endian byte orders are supported.
- 32 • `System.Text.UTF8Encoding` - encodes Unicode characters using the UTF-8 (UCS
33 Transformation Format, 8-bit form) encoding. This encoding supports all Unicode
34 character values.

35 An application can use the properties of this class such as `System.Text.Encoding.ASCII`,
36 `System.Text.Encoding.Default`, `System.Text.Encoding.Unicode`, and
37 `System.Text.Encoding.UTF8` to obtain encodings. Applications can initialize new instances

1 of `System.Text.Encoding` objects through the `System.Text.ASCIIEncoding`,
2 `System.Text.UnicodeEncoding`, and `System.Text.UTF8Encoding` classes.

3
4 Through an encoding, the `System.Text.Encoding.GetBytes` method is used to convert
5 arrays of Unicode characters to arrays of bytes, and the `System.Text.Encoding.GetChars`
6 method is used to convert arrays of bytes to arrays of Unicode characters. The
7 `System.Text.Encoding.GetBytes` and `System.Text.Encoding.GetChars` methods maintain
8 no state between conversions. When the data to be converted is only available in sequential
9 blocks (such as data read from a stream) or when the amount of data is so large that it
10 needs to be divided into smaller blocks, an application can choose to use a
11 `System.Text.Decoder` or a `System.Text.Encoder` to perform the conversion. Decoders and
12 encoders allow sequential blocks of data to be converted and they maintain the state
13 required to support conversions of data that spans adjacent blocks. Decoders and encoders
14 are obtained using the `System.Text.Encoding.GetDecoder` and
15 `System.Text.Encoding.GetEncoder` methods.

16
17 The core `System.Text.Encoding.GetBytes` and `System.Text.Encoding.GetChars` methods
18 require the caller to provide the destination buffer and ensure that the buffer is large
19 enough to hold the entire result of the conversion. When using these methods, either
20 directly on a `System.Text.Encoding` object or on an associated `System.Text.Decoder` or
21 `System.Text.Encoder`, an application can use one of two methods to allocate destination
22 buffers.

23 1. The `System.Text.Encoding.GetByteCount` and
24 `System.Text.Encoding.GetCharCount` methods can be used to compute the exact
25 size of the result of a particular conversion, and an appropriately sized buffer for that
26 conversion can then be allocated.

27 2. The `System.Text.Encoding.GetMaxByteCount` and
28 `System.Text.Encoding.GetMaxCharCount` methods can be used to compute the
29 maximum possible size of a conversion of a given number of characters or bytes,
30 regardless of the actual character or byte values, and a buffer of that size can then
31 be reused for multiple conversions.

32 The first method generally uses less memory, whereas the second method generally
33 executes faster.

34

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

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

Return Value

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

Exceptions

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

1

2

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

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

11 Summary

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

14 Parameters

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

15 16 Return Value

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

18 Exceptions

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

19
20

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 `System.Object` represent the
8 same type and value.

9 Parameters

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

10

11 Return Value

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

14 Description

15 [*Note:* This method overrides `System.Object.Equals`.]
16
17

18

1 Encoding.GetByteCount(System.Char[])

2 Method

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

8 Summary

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

10 Parameters

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

11 Return Value

13 A `System.Int32` containing the number of bytes needed to encode *chars*.

14 Behaviors

15 As described above.

17 How and When to Override

18 This method is overridden by types derived from `System.Text.Encoding` to return the
19 appropriate number of bytes for the particular encoding.

21 Usage

22 `System.Text.Encoding.GetByteCount` can be used to determine the exact number of
23 bytes that will be produced from encoding the given array of characters. An
24 appropriately sized buffer for that conversion can then be allocated.

25
26 Alternatively, `System.Text.Encoding.GetMaxByteCount` can be used to determine the
27 maximum number of bytes that will be produced from converting a given number of
28 characters, regardless of the actual character values. A buffer of that size can then be
29 reused for multiple conversions.

1
2 `System.Text.Encoding.GetByteCount` generally uses less memory and
3 `System.Text.Encoding.GetMaxByteCount` generally executes faster.

4 **Exceptions**

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

5

6

1 Encoding.GetByteCount(System.String)

2 Method

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

7 Summary

8 Returns the number of bytes required to encode the specified `System.String`.

9 Parameters

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

10

11 Return Value

12 A `System.Int32` containing the number of bytes needed to encode s.

13 Behaviors

14 As described above.

15

16 How and When to Override

17 This method is overridden by types derived from `System.Text.Encoding` to return the
18 appropriate number of bytes for the particular encoding.

19

20 Usage

21 `System.Text.Encoding.GetByteCount` can be used to determine the exact number of
22 bytes that will be produced from encoding the given `System.String`. An appropriately
23 sized buffer for that conversion can then be allocated.

24

25 Alternatively, `System.Text.Encoding.GetMaxByteCount` can be used to determine the
26 maximum number of bytes that will be produced from converting a given number of
27 characters, regardless of the actual character values. A buffer of that size can then be
28 reused for multiple conversions.

29

- 1 `System.Text.Encoding.GetByteCount` generally uses less memory and
- 2 `System.Text.Encoding.GetMaxByteCount` generally executes faster.

3 **Exceptions**

Exception	Condition
System.ArgumentNullException	s is null.

4

5

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

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

8 Summary

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

11 Parameters

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

12

13 Return Value

14 A System.Int32 containing the number of bytes required to encode the range in *chars*
15 from *index* to *index* + *count* - 1.

16 Behaviors

17 As described above.

18

19 How and When to Override

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

22

23 Usage

1 `System.Text.Encoding.GetByteCount` can be used to determine the exact the number
2 of bytes that will be produced from encoding a given range of characters. An
3 appropriately sized buffer for that conversion can then be allocated.

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

9
10 `System.Text.Encoding.GetByteCount` generally uses less memory and
11 `System.Text.Encoding.GetMaxByteCount` generally executes faster.

12 Exceptions

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

13

14

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

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

7 Summary

8 Encodes the specified System.Char array.

9 Parameters

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

10

11 Return Value

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

13 Behaviors

14 As described above.

15

16 How and When to Override

17 This method is overridden by types derived from System.Text.Encoding to perform the
18 encoding.

19

20 Exceptions

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

21

22

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

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

8 Summary

9 Encodes the specified range of the specified `System.Char` array.

10 Parameters

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

11

12 Return Value

13 A `System.Byte` array containing the encoded representation of the range in *chars* from
14 *index* to *index* + *count* - 1.

15 Behaviors

16 As described above.

17

18 How and When to Override

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

21

22 Exceptions

Exception	Condition
-----------	-----------

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

1

2

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

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

11 Summary

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

14 Parameters

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

15 16 Return Value

17 The number of bytes encoded into *bytes*.

18 Behaviors

19 As described above.

20

21 How and When to Override

1 This method is overridden by types derived from `System.Text.Encoding` to perform the
2 encoding.

3

4 Usage

5 `System.Text.Encoding.GetByteCount` can be used to determine the exact number of
6 bytes that will be produced for a given range of characters. Alternatively,
7 `System.Text.Encoding.GetMaxByteCount` can be used to determine the maximum
8 number of bytes that will be produced for a given number of characters, regardless of
9 the actual character values.

10

11 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

1 Encoding.GetBytes(System.String) Method

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

6 Summary

7 Encodes the specified System.String.

8 Parameters

Parameter	Description
s	The System.String to encode.

9 Return Value

11 A System.Byte array containing the encoded representation of s.

12 Behaviors

13 As described above.

15 How and When to Override

16 This method is overridden by types derived from System.Text.Encoding to perform the
17 encoding.

19 Exceptions

Exception	Condition
System.ArgumentNullException	s is null.

20
21

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

```
4 [ILAsm]  
5 .method public hidebysig virtual int32 GetBytes(string s, int32 charIndex,  
6 int32 charCount, class System.Byte[] bytes, int32 byteIndex)  
  
7 [C#]  
8 public virtual int GetBytes(string s, int charIndex, int charCount, byte[]  
9 bytes, int byteIndex)
```

10 Summary

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

13 Parameters

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

14 15 Return Value

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

17 Behaviors

18 As described above.

19

20 How and When to Override

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

1

2 Exceptions

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

3

4

1 Encoding.GetCharCount(System.Byte[])

2 Method

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

8 Summary

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

11 Parameters

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

12 Return Value

14 A `System.Int32` containing the number of characters produced by decoding *bytes*.

15 Behaviors

16 As described above.

18 How and When to Override

19 This method is overridden by types derived from `System.Text.Encoding` to return the
20 appropriate number of bytes for the particular encoding.

22 Usage

23 Use `System.Text.Encoding.GetCharCount` to determine the exact number of
24 characters that will be produced from converting a given byte array. An appropriately
25 sized buffer for that conversion can then be allocated.

26
27 Alternatively, use `System.Text.Encoding.GetMaxCharCount` to determine the maximum
28 number of characters that will be produced for a given number of bytes, regardless of
29 the actual byte values. A buffer of that size can then be reused for multiple conversions.

1
2 `System.Text.Encoding.GetCharCount` generally uses less memory and
3 `System.Text.Encoding.GetMaxCharCount` generally executes faster.

4 **Exceptions**

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

5

6

Encoding.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 `System.Byte` array.

Parameters

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

Behaviors

As described above.

How and When to Override

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

Usage

1 Use `System.Text.Encoding.GetCharCount` to determine the exact number of
2 characters that will be produced from converting a given range of bytes. An
3 appropriately sized buffer for that conversion can then be allocated.
4
5 Alternatively, use `System.Text.Encoding.GetMaxCharCount` to determine the maximum
6 number of characters that will be produced for a given number of bytes, regardless of
7 the actual byte values. A buffer of that size can then be reused for multiple conversions.
8
9 `System.Text.Encoding.GetCharCount` generally uses less memory and
10 `System.Text.Encoding.GetMaxCharCount` generally executes faster.

11 Exceptions

Exception	Condition
System.ArgumentNullException	<i>bytes</i> is null.
System.ArgumentOutOfRangeException	<i>index</i> and <i>count</i> do not specify a valid range in <i>bytes</i> (i.e. $(index + count) > bytes.Length$).

12

13

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

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

7 Summary

8 Decodes a System.Byte array.

9 Parameters

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

10

11 Return Value

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

13 Exceptions

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

14

15

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

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

8 Summary

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

10 Parameters

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

11

12 Return Value

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

15 Exceptions

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

16

17

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

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

11 Summary

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

14 Parameters

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

15 16 Return Value

17 The number of characters stored in *chars*.

18 Behaviors

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

21 How and When to Override

22 This method is overridden by types derived from System.Text.Encoding to perform the
23 particular decoding.

1

2 Usage

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

7

8 Exceptions

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

1 Encoding.GetDecoder() Method

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

6 Summary

7 Returns a `System.Text.Decoder` for the current instance.

8 Return Value

9 A `System.Text.Decoder` for the current instance.

10 Behaviors

11 As described above.

12

13 Default

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

17

18 How and When to Override

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

22

23 Usage

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

27

28

1 Encoding.GetEncoder() Method

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

6 Summary

7 Returns a `System.Text.Encoder` for the current instance.

8 Return Value

9 A `System.Text.Encoder` for the current encoding.

10 Behaviors

11 As described above.

12

13 Default

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

17

18 How and When to Override

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

21

22 Usage

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

26

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 A `System.Int32` containing the hash code for the current instance.

10 Description

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

12

13 [*Note:* This method overrides `System.Object.GetHashCode()`.]

14

15

16

1 Encoding.GetMaxByteCount(System.Int32)

2 Method

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

8 Summary

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

11 Parameters

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

12 Return Value

14 A `System.Int32` containing the maximum number of bytes required to encode
15 *charCount* characters.

16 Behaviors

17 As described above.

19 How and When to Override

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

23 Usage

24 `System.Text.Encoding.GetMaxByteCount` can be used to determine the minimum
25 buffer size for byte arrays passed to the `System.Text.Encoding.GetBytes` of the
26 current encoding. Using this minimum buffer size ensures that no buffer overflow
27 exceptions occur.

28

1 Encoding.GetMaxCharCount(System.Int32)

2 Method

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

8 Summary

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

11 Parameters

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

12 Return Value

14 A `System.Int32` containing the maximum number of characters that would be produced
15 by decoding *byteCount* bytes.

16 Behaviors

17 As described above.

19 How and When to Override

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

23 Usage

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

28

1 Encoding.GetPreamble() Method

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

6 Summary

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

9 Return Value

10 A `System.Byte` array that identifies the encoding used on a stream.

11 Description

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

17 Behaviors

18 As described above.
19

20 Default

21 The default implementation returns an empty `System.Byte` array.
22

23 How and When to Override

24 Override this method to return a `System.Byte` array containing the preamble
25 appropriate for the type derived from `System.Text.Encoding`.
26
27

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

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

7 Summary

8 Decodes the specified System.Byte array.

9 Parameters

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

10

11 Return Value

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

13 Behaviors

14 As described above.

15

16 How and When to Override

17 This method is overridden by particular character encodings.

18

19 Exceptions

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

20

21

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 <code>System.Byte</code> array to decode.
<i>index</i>	A <code>System.Int32</code> containing the starting index of <i>bytes</i> to decode.
<i>count</i>	A <code>System.Int32</code> 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
-----------	-----------

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

1

2

1 Encoding.ASCII Property

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

7 Summary

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

9 Description

10 This property is read-only.

11
12 [*Note:* ASCII characters can represent Unicode characters from U+0000 to U+007f,
13 inclusive.

14
15]

16

1 Encoding.BigEndianUnicode Property

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

7 Summary

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

9 Property Value

10 A System.Text.Encoding for the Unicode format in big-endian byte order.

11 Description

12 This property is read-only.

13
14 [Note: Unicode characters can be stored in two different byte orders, big-endian and
15 little-endian. On little-endian platforms such as those implemented on Intel processors,
16 it is generally more efficient to store Unicode characters in little-endian byte order.
17 However, many other platforms can store Unicode characters in big-endian byte order.
18 Unicode files can be distinguished by the presence of the byte order mark (U+FEFF),
19 which will be written as either 0xfe 0xff or 0xff 0xfe.

20
21 This encoding automatically detects a byte order mark and, if necessary, switches byte
22 orders.

23
24]

25

1 Encoding.Default Property

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

7 Summary

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

9 Property Value

10 A System.Text.Encoding for the ANSI code page of the current system.

11 Description

12 This property is read-only.

13

1 Encoding.Unicode Property

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

7 Summary

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

9 Property Value

10 A `System.Text.Encoding` for the Unicode format in little-endian byte order.

11 Description

12 This property is read-only.

13
14 [*Note:* Unicode characters can be stored in two different byte orders, big-endian and
15 little-endian. On little-endian platforms such as those implemented on Intel processors,
16 it is generally more efficient to store Unicode characters in little-endian byte order.
17 However, many other platforms can store Unicode characters in big-endian byte order.
18 Unicode files can be distinguished by the presence of the byte order mark (U+FEFF),
19 which will be written as either 0xfe 0xff or 0xff 0xfe.

20
21 This encoding automatically detects a byte order mark and, if necessary, switches byte
22 orders.

23
24]

25

1 Encoding.UTF8 Property

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

7 Summary

8 Gets an encoding for the UTF-8 format.

9 Property Value

10 A System.Text.Encoding for the UTF-8 format.

11 Description

12 This property is read-only.

13
14 [*Note:* For detailed information regarding UTF-8 encoding, see
15 System.Text.UTF8Encoding.

16
17]

18