System.Text.Encoding Class

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

[C#]
public abstract class Encoding
```

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

Represents a character encoding.

Inherits From: System.Object

Library: BCL

Thread Safety: This type is safe for multithreaded operations.

Description

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

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

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

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

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

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

Encoding() Constructor

```
[ILAsm]
family rtspecialname specialname instance void .ctor()
```

```
[C#]
protected Encoding()
```

Summary

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

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
srcEncoding	The `System.Text.Encoding` that *bytes* is in.
dstEncoding	The `System.Text.Encoding` desired for the returned `System.Byte` array.
bytes	The `System.Byte` array containing the values to convert.
index	A `System.Int32` containing the first index of *bytes* from which to convert.
count	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	*srcEncoding*, *dstEncoding*, or *bytes* is `null`.
System.ArgumentOutOfRangeException	*index* and *count* do not denote a valid range in *bytes*.

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

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

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

Summary

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

Parameters

Parameter	Description
srcEncoding	The `System.Text.Encoding` that *bytes* is in.
dstEncoding	The `System.Text.Encoding` desired for the returned `System.Byte` array.
bytes	The `System.Byte` array containing the values to convert.

Return Value

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

Exceptions

Exception	Condition
System.ArgumentNullException	*srcEncoding*, *dstEncoding* or *bytes* is `null`.

Encoding.Equals(System.Object) Method

```
[ILAsm]
.method public hidebysig virtual bool Equals(object value)

[C#]
public override bool Equals(object value)
```

Summary

Determines whether the current instance and the specified `System.Object` represent the same type and value.

Parameters

Parameter	Description
value	The `System.Object` to compare to the current instance.

Return Value

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

Description

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

Encoding.GetByteCount(System.Char[]) Method

```
[ILAsm]
.method public hidebysig virtual int32 GetByteCount(class System.Char[]
chars)

[C#]
public virtual int GetByteCount(char[] chars)
```

Summary

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

Parameters

Parameter	Description
chars	The `System.Char` array to encode.

Return Value

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

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

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

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

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

Exceptions

Exception	Condition
System.ArgumentNullException	*chars* is `null`.

Encoding.GetByteCount(System.String) Method

```
[ILAsm]
.method public hidebysig virtual int32 GetByteCount(string s)

[C#]
public virtual int GetByteCount(string s)
```

Summary

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

Parameters

Parameter	Description
s	The `System.String` to decode.

Return Value

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

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

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

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

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

Exceptions

Exception	**Condition**
System.ArgumentNullException	*s* is `null`.

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

[ILAsm]
```
.method public hidebysig virtual abstract int32 GetByteCount(class
System.Char[] chars, int32 index, int32 count)
```

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

Summary

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

Parameters

Parameter	Description
chars	The `System.Char` array to encode.
index	A `System.Int32` containing the first index of *chars* to encode.
count	A `System.Int32` containing the number of characters to encode.

Return Value

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

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

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

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

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

Exceptions

Exception	**Condition**
System.ArgumentNullException	*chars* is `null`.
System.ArgumentOutOfRangeException	The number of bytes required to encode the specified elements in *chars* is greater than `System.Int32.MaxValue`. -or- *index* or *count* is less than zero. -or- *index* and *count* do not specify a valid range in *chars* (i.e. (*index* + *count*) > *chars*.Length).

Encoding.GetBytes(System.Char[]) Method

```
[ILAsm]
.method public hidebysig virtual class System.Byte[] GetBytes(class
System.Char[] chars)

[C#]
public virtual byte[] GetBytes(char[] chars)
```

Summary

Encodes the specified `System.Char` array.

Parameters

Parameter	Description
chars	The `System.Char` array to encode.

Return Value

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

Behaviors

As described above.

How and When to Override

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

Exceptions

Exception	Condition
System.ArgumentNullException	*chars* is `null`.

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

```
[ILAsm]
.method public hidebysig virtual class System.Byte[] GetBytes(class
System.Char[] chars, int32 index, int32 count)

[C#]
public virtual byte[] GetBytes(char[] chars, int index, int count)
```

Summary

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

Parameters

Parameter	Description
chars	The `System.Char` array to encode.
index	A `System.Int32` containing the first index of *chars* to encode.
count	A `System.Int32` containing the number of characters to encode.

Return Value

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

Behaviors

As described above.

How and When to Override

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

Exceptions

Exception	Condition

System.ArgumentNullException	*chars* is `null`.
System.ArgumentOutOfRangeException	*index* and *count* do not denote a valid range in *chars*.

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

```
[ILAsm]
.method public hidebysig virtual abstract int32 GetBytes(class
System.Char[] chars, int32 charIndex, int32 charCount, class System.Byte[]
bytes, int32 byteIndex)

[C#]
public abstract int GetBytes(char[] chars, int charIndex, int charCount,
byte[] bytes, int byteIndex)
```

Summary

Encodes the specified range of the specified `System.Char` array into the specified range of the specified `System.Byte` array.

Parameters

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

Return Value

The number of bytes encoded into *bytes*.

Behaviors

As described above.

How and When to Override

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

Usage

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

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.

Encoding.GetBytes(System.String) Method

```
[ILAsm]
.method public hidebysig virtual class System.Byte[] GetBytes(string s)

[C#]
public virtual byte[] GetBytes(string s)
```

Summary

Encodes the specified `System.String`.

Parameters

Parameter	Description
s	The `System.String` to encode.

Return Value

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

Behaviors

As described above.

How and When to Override

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

Exceptions

Exception	Condition
System.ArgumentNullException	*s* is `null`.

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

```
[ILAsm]
.method public hidebysig virtual int32 GetBytes(string s, int32 charIndex,
int32 charCount, class System.Byte[] bytes, int32 byteIndex)

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

Summary

Encodes the specified range of the specified `System.String` into the specified range of the specified `System.Byte` array.

Parameters

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

Return Value

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

Behaviors

As described above.

How and When to Override

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

Exceptions

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

Encoding.GetCharCount(System.Byte[]) Method

```
[ILAsm]
.method public hidebysig virtual int32 GetCharCount(class System.Byte[] bytes)

[C#]
public virtual int GetCharCount(byte[] bytes)
```

Summary

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

Parameters

Parameter	Description
bytes	The `System.Byte` array to decode.

Return Value

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

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

Exceptions

Exception	Condition
System.ArgumentNullException	*bytes* is `null`.

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
bytes	The `System.Byte` array to decode.
index	The first index in *bytes* to decode.
count	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

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

Exceptions

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

Encoding.GetChars(System.Byte[]) Method

[ILAsm]
```
.method public hidebysig virtual class System.Char[] GetChars(class
System.Byte[] bytes)
```

[C#]
```
public virtual char[] GetChars(byte[] bytes)
```

Summary

Decodes a `System.Byte` array.

Parameters

Parameter	Description
bytes	The `System.Byte` array to decode.

Return Value

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

Exceptions

Exception	Condition
System.ArgumentNullException	*bytes* is `null`.

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

```
[ILAsm]
.method public hidebysig virtual class System.Char[] GetChars(class
System.Byte[] bytes, int32 index, int32 count)

[C#]
public virtual char[] GetChars(byte[] bytes, int index, int count)
```

Summary

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

Parameters

Parameter	**Description**
bytes	The `System.Byte` array to decode.
index	A `System.Int32` containing the first index of *bytes* to decode.
count	A `System.Int32` containing the number of bytes to decode.

Return Value

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

Exceptions

Exception	**Condition**
System.ArgumentNullException	*bytes* is `null`.
System.ArgumentOutOfRangeException	*index* and *count* do not denote a valid range in the byte array.

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

```
[ILAsm]
.method public hidebysig virtual abstract int32 GetChars(class
System.Byte[] bytes, int32 byteIndex, int32 byteCount, class System.Char[]
chars, int32 charIndex)

[C#]
public abstract int GetChars(byte[] bytes, int byteIndex, int byteCount,
char[] chars, int charIndex)
```

Summary

Decodes the specified range of the specified `System.Byte` array into the specified range of the specified `System.Char` array.

Parameters

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

Return Value

The number of characters stored in *chars*.

Behaviors

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

How and When to Override

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

Usage

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

Exceptions

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

Encoding.GetDecoder() Method

```
[ILAsm]
.method public hidebysig virtual class System.Text.Decoder GetDecoder()

[C#]
public virtual Decoder GetDecoder()
```

Summary

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

Return Value

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

Behaviors

As described above.

Default

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

How and When to Override

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

Usage

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

Encoding.GetEncoder() Method

[ILAsm]
```
.method public hidebysig virtual class System.Text.Encoder GetEncoder()
```

[C#]
```
public virtual Encoder GetEncoder()
```

Summary

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

Return Value

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

Behaviors

As described above.

Default

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

How and When to Override

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

Usage

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

Encoding.GetHashCode() Method

[ILAsm]
```
.method public hidebysig virtual int32 GetHashCode()
```

[C#]
```
public override int GetHashCode()
```

Summary

Generates a hash code for the current instance.

Return Value

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

Description

The algorithm used to generate the hash code is unspecified.

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

Encoding.GetMaxByteCount(System.Int32) Method

```
[ILAsm]
.method public hidebysig virtual abstract int32 GetMaxByteCount(int32 charCount)

[C#]
public abstract int GetMaxByteCount(int charCount)
```

Summary

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

Parameters

Parameter	Description
charCount	A `System.Int32` containing the number of characters to encode.

Return Value

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

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

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

1

Encoding.GetMaxCharCount(System.Int32) Method

```
[ILAsm]
.method public hidebysig virtual abstract int32 GetMaxCharCount(int32
byteCount)

[C#]
public abstract int GetMaxCharCount(int byteCount)
```

Summary

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

Parameters

Parameter	Description
byteCount	A `System.Int32` containing the number of bytes to decode.

Return Value

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

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

1

Encoding.GetPreamble() Method

```
[ILAsm]
.method public hidebysig virtual class System.Byte[] GetPreamble()

[C#]
public virtual byte[] GetPreamble()
```

Summary

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

Return Value

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

Description

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

Behaviors

As described above.

Default

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

How and When to Override

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

Encoding.GetString(System.Byte[]) Method

```
[ILAsm]
.method public hidebysig virtual string GetString(class System.Byte[]
bytes)

[C#]
public virtual string GetString(byte[] bytes)
```

Summary

Decodes the specified `System.Byte` array.

Parameters

Parameter	Description
bytes	The `System.Byte` array to decode.

Return Value

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

Behaviors

As described above.

How and When to Override

This method is overridden by particular character encodings.

Exceptions

Exception	Condition
System.ArgumentNullException	*bytes* is `null`.

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**
bytes	The `System.Byte` array to decode.
index	A `System.Int32` containing the starting index of *bytes* to decode.
count	A `System.Int32` 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	*bytes* is `null`.
System.ArgumentOutOfRangeException	*index* and *count* do not denote a valid range in *bytes*.

Encoding.ASCII Property

[ILAsm]
```
.property class System.Text.Encoding ASCII { public hidebysig static
specialname class System.Text.Encoding get_ASCII() }
```

[C#]
```
public static Encoding ASCII { get; }
```

Summary

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

Description

This property is read-only.

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

]

Encoding.BigEndianUnicode Property

[ILAsm]
```
.property class System.Text.Encoding BigEndianUnicode { public hidebysig
static specialname class System.Text.Encoding get_BigEndianUnicode() }
```

[C#]
```
public static Encoding BigEndianUnicode { get; }
```

Summary

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

Property Value

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

Description

This property is read-only.

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

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

]

Encoding.Default Property

[ILAsm]
```
.property class System.Text.Encoding Default { public hidebysig static
specialname class System.Text.Encoding get_Default() }
```

[C#]
```
public static Encoding Default { get; }
```

Summary

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

Property Value

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

Description

This property is read-only.

Encoding.Unicode Property

[ILAsm]
```
.property class System.Text.Encoding Unicode { public hidebysig static
specialname class System.Text.Encoding get_Unicode() }
```

[C#]
```
public static Encoding Unicode { get; }
```

Summary

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

Property Value

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

Description

This property is read-only.

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

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

]

Encoding.UTF8 Property

[ILAsm]
```
.property class System.Text.Encoding UTF8 { public hidebysig static
specialname class System.Text.Encoding get_UTF8() }
```

[C#]
```
public static Encoding UTF8 { get; }
```

Summary

Gets an encoding for the UTF-8 format.

Property Value

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

Description

This property is read-only.

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

]