

# System.IO.StreamReader Class

```
[ILAsm]
.class public serializable StreamReader extends System.IO.TextReader

[C#]
public class StreamReader: TextReader
```

## 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)

## Implements:

- **System.IDisposable**

## Summary

Implements a `System.IO.Stream` that reads characters from a byte stream in a particular encoding.

## Inherits From: System.IO.TextReader

## Library: BCL

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

## Description

The `System.IO.StreamReader` class is designed for character input in a particular `System.Text.Encoding`, whereas subclasses of `System.IO.Stream` are designed for byte input and output.

[*Note:* `System.IO.StreamReader` defaults to UTF-8 encoding unless specified otherwise, instead of defaulting to the ANSI code page for the current system. UTF-8 handles Unicode characters correctly and provides consistent results on localized versions of the operating system.

When reading from a `System.IO.Stream`, it is more efficient to use a buffer that is the same size as the internal buffer of the stream.

By default, a `System.IO.StreamReader` is not thread safe. For a thread-safe wrapper, see `System.IO.TextReader.Synchronized`.

1  
2 ]  
3

# StreamReader(System.String, System.Text.Encoding, System.Boolean, System.Int32) Constructor

```
[ILAsm]  
public rtspecialname specialname instance void .ctor(string path, class  
System.Text.Encoding encoding, bool detectEncodingFromByteOrderMarks,  
int32 bufferSize)
```

```
[C#]  
public StreamReader(string path, Encoding encoding, bool  
detectEncodingFromByteOrderMarks, int bufferSize)
```

## Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified file name, with the specified character encoding, byte order mark detection option, and buffer size.

## Parameters

Parameter	Description
<i>path</i>	A <code>System.String</code> that specifies the complete file path to read.
<i>encoding</i>	A <code>System.Text.Encoding</code> that specifies the character encoding to use.
<i>detectEncodingFromByteOrderMarks</i>	A <code>System.Boolean</code> value that indicates whether the new <code>System.IO.StreamReader</code> is required to look for byte order marks at the beginning of the stream. Specify <code>true</code> to enable detection of byte order marks; otherwise, specify <code>false</code> .
<i>bufferSize</i>	A <code>System.Int32</code> that specifies the minimum buffer size, in number of 16-bit characters. If less than the minimum allowable size (128 characters), the minimum allowable size is used.

## Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property using *encoding*.

If requested, the current constructor detects the encoding by examining the first three bytes of the stream. The constructor automatically recognizes UTF-8, little-endian Unicode, and big-endian Unicode text if the file starts with the appropriate byte order marks. Otherwise, the user-provided encoding is used. See the `System.Text.Encoding.GetPreamble` method for more information.

[*Note:* *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system, this class might be able to access a physical device.

When reading from a `System.IO.Stream`, it is more efficient to use a buffer that is the same size as the internal buffer of the stream.

For information on the valid format and characters for path strings, see `System.IO.Path`.

]

## Exceptions

Exception	Condition
<b>System.IO.IOException</b>	<i>path</i> is in an invalid format or contains invalid characters.
<b>System.IO.DirectoryNotFoundException</b>	The directory information specified in <i>path</i> was not found.
<b>System.IO.FileNotFoundException</b>	The file specified in <i>path</i> was not found.
<b>System.ArgumentException</b>	<i>path</i> is an empty string ("").
<b>System.ArgumentNullException</b>	<i>path</i> or <i>encoding</i> is null.
<b>System.ArgumentOutOfRangeException</b>	<i>bufferSize</i> is less than or equal to zero.

# StreamReader(System.String, System.Text.Encoding, System.Boolean) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(string path, class
System.Text.Encoding encoding, bool detectEncodingFromByteOrderMarks)

[C#]
public StreamReader(string path, Encoding encoding, bool
detectEncodingFromByteOrderMarks)
```

## Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified file name, with the specified character encoding and byte order mark detection option.

## Parameters

Parameter	Description
<i>path</i>	A <code>System.String</code> that specifies the complete file path to read.
<i>encoding</i>	A <code>System.Text.Encoding</code> that specifies the character encoding to use.
<i>detectEncodingFromByteOrderMarks</i>	A <code>System.Boolean</code> value that indicates whether the new <code>System.IO.StreamReader</code> is required to look for byte order marks at the beginning of the stream. Specify <code>true</code> to enable detection of byte order marks; otherwise, specify <code>false</code> .

## Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property using *encoding*, and the internal buffer to the default size. [Note: The default buffer size is implementation defined.]

If requested, the current constructor detects the encoding by examining the first three bytes of the stream. The constructor automatically recognizes UTF-8, little-endian Unicode, and big-endian Unicode text if the file starts with the appropriate byte order marks. Otherwise, the user-provided encoding is used. See the

System.Text.Encoding.GetPreamble method for more information.

[*Note:* *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system, this class might be able to access a physical device.

For information on the valid format and characters for path strings, see System.IO.Path.

]

## Exceptions

Exception	Condition
System.IO.IOException	<i>path</i> is in an invalid format or contains invalid characters.
System.IO.DirectoryNotFoundException	The directory information specified in <i>path</i> was not found.
System.IO.FileNotFoundException	The file specified in <i>path</i> was not found.
System.ArgumentException	<i>path</i> is an empty string ("").
System.ArgumentNullException	<i>path</i> or <i>encoding</i> is null.

# StreamReader(System.String, System.Text.Encoding) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(string path, class
System.Text.Encoding encoding)

[C#]
public StreamReader(string path, Encoding encoding)
```

## Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified file name and with the specified character encoding.

## Parameters

Parameter	Description
<i>path</i>	A <code>System.String</code> that specifies the complete file path to read.
<i>encoding</i>	A <code>System.Text.Encoding</code> that specifies the character encoding to use.

## Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property using *encoding*, and the internal buffer to the default size. [Note: The default buffer size is implementation defined.]

[Note: *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system, this class might be able to access a physical device.

For information on the valid format and characters for path strings, see `System.IO.Path`.

]

## Exceptions

Exception	Condition
<b>System.IO.IOException</b>	<i>path</i> is in an invalid format or contains invalid characters.

<b>System.IO.DirectoryNotFoundException</b>	The directory information specified in <i>path</i> was not found.
<b>System.IO.FileNotFoundException</b>	The file specified in <i>path</i> was not found.
<b>System.ArgumentException</b>	<i>path</i> is an empty string ("").
<b>System.ArgumentNullException</b>	<i>path</i> or <i>encoding</i> is null.

1

2



# StreamReader(System.String, System.Boolean) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(string path, bool
detectEncodingFromByteOrderMarks)

[C#]
public StreamReader(string path, bool detectEncodingFromByteOrderMarks)
```

## Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified file name, with the specified byte order mark detection option.

## Parameters

Parameter	Description
<i>path</i>	A <code>System.String</code> that specifies the complete file path to read.
<i>detectEncodingFromByteOrderMarks</i>	A <code>System.Boolean</code> value that indicates whether the new <code>System.IO.StreamReader</code> is required to look for byte order marks at the beginning of the stream. Specify <code>true</code> to enable detection of byte order marks; otherwise, specify <code>false</code> .

## Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property to `System.Text.UTF8Encoding`, and the internal buffer to the default size. [Note: The default buffer size is implementation defined.]

If requested, the current constructor detects the encoding by examining the first three bytes of the stream. The constructor automatically recognizes UTF-8, little-endian Unicode, and big-endian Unicode text if the file starts with the appropriate byte order marks. Otherwise, UTF-8 encoding is used. See the `System.Text.Encoding.GetPreamble` method for more information.

[Note: *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system, this class might be able to access a physical device.

For information on the valid format and characters for path strings, see

```
1      System.IO.Path.  
2  
3      ]
```

#### 4 Exceptions

Exception	Condition
<b>System.IO.IOException</b>	<i>path</i> is in an invalid format or contains invalid characters.
<b>System.IO.DirectoryNotFoundException</b>	The directory information specified in <i>path</i> was not found.
<b>System.IO.FileNotFoundException</b>	The file specified in <i>path</i> was not found.
<b>System.ArgumentException</b>	<i>path</i> is an empty string ("").
<b>System.ArgumentNullException</b>	<i>path</i> is null.

5

6

# StreamReader(System.String) Constructor

```
[ILAsm]  
public rtspecialname specialname instance void .ctor(string path)  
  
[C#]  
public StreamReader(string path)
```

## Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified file name.

## Parameters

Parameter	Description
<i>path</i>	A <code>System.String</code> that specifies the complete file path to read.

## Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property to `System.Text.UTF8Encoding`, and the internal buffer to the default size. [Note: The default buffer size is implementation defined.]

[Note: *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system, this class might be able to access a physical device.

For information on the valid format and characters for path strings, see `System.IO.Path`.

]

## Exceptions

Exception	Condition
<b>System.IO.IOException</b>	<i>path</i> is in an invalid format or contains invalid characters.
<b>System.IO.DirectoryNotFoundException</b>	The directory information specified in <i>path</i> was not found.

<b>System.IO.FileNotFoundException</b>	The file specified in <i>path</i> was not found.
<b>System.ArgumentException</b>	<i>path</i> is an empty string ("").
<b>System.ArgumentNullException</b>	<i>path</i> is null.

1

2

# StreamReader(System.IO.Stream, System.Text.Encoding, System.Boolean, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.IO.Stream stream, class System.Text.Encoding encoding, bool
detectEncodingFromByteOrderMarks, int32 bufferSize)

[C#]
public StreamReader(Stream stream, Encoding encoding, bool
detectEncodingFromByteOrderMarks, int bufferSize)
```

## Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified stream, with the specified character encoding, byte order mark detection option, and buffer size.

## Parameters

Parameter	Description
<i>stream</i>	The <code>System.IO.Stream</code> to read.
<i>encoding</i>	A <code>System.Text.Encoding</code> that specifies the character encoding to use.
<i>detectEncodingFromByteOrderMarks</i>	A <code>System.Boolean</code> value that indicates whether the new <code>System.IO.StreamReader</code> is required to look for byte order marks at the beginning of the stream. Specify <code>true</code> to enable detection of byte order marks; otherwise, specify <code>false</code> .
<i>bufferSize</i>	A <code>System.Int32</code> that specifies the minimum buffer size, in number of 16-bit characters. If <i>bufferSize</i> is less than the minimum allowable size (128 characters), the minimum allowable size is used.

## Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property using *encoding* parameter the `System.IO.StreamReader.BaseStream` property using *stream*.

If requested, this constructor detects the encoding by examining the first three bytes of the stream. The constructor automatically recognizes UTF-8, little-endian Unicode, and big-endian Unicode text if the file starts with the appropriate byte order marks. Otherwise, the user-provided encoding is used. For more information, see the `System.Text.Encoding.GetPreamble` method.

[*Note:* When reading from a `System.IO.Stream`, it is more efficient to use a buffer that is the same size as the internal buffer of the stream.]

## Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>stream</i> does not support reading.
<b>System.ArgumentNullException</b>	<i>stream</i> or <i>encoding</i> is null.
<b>System.ArgumentOutOfRangeException</b>	<i>bufferSize</i> is less than or equal to zero.

# StreamReader(System.IO.Stream, System.Text.Encoding, System.Boolean) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.IO.Stream stream, class System.Text.Encoding encoding, bool
detectEncodingFromByteOrderMarks)

[C#]
public StreamReader(Stream stream, Encoding encoding, bool
detectEncodingFromByteOrderMarks)
```

## Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified stream, with the specified character encoding and byte order mark detection option.

## Parameters

Parameter	Description
<i>stream</i>	The <code>System.IO.Stream</code> to read.
<i>encoding</i>	A <code>System.Text.Encoding</code> that specifies the character encoding to use.
<i>detectEncodingFromByteOrderMarks</i>	A <code>System.Boolean</code> value that indicates whether the new <code>System.IO.StreamReader</code> is required to look for byte order marks at the beginning of the stream. Specify <code>true</code> to enable detection of byte order marks; otherwise, specify <code>false</code> .

## Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property using *encoding*, the `System.IO.StreamReader.BaseStream` property using *stream*, and the internal buffer to the default size. [Note: The default buffer size is implementation defined.]

If requested, this constructor detects the encoding by examining the first three bytes of *stream*. This constructor automatically recognizes UTF-8, little-endian Unicode, and big-endian Unicode text if the stream starts with the appropriate byte order marks.

1 Otherwise, the user-provided encoding is used. See the  
2 `System.Text.Encoding.GetPreamble` method for more information.

### 3 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>stream</i> does not support reading.
<b>System.ArgumentNullException</b>	<i>stream</i> or <i>encoding</i> is null.

4

5



# StreamReader(System.IO.Stream)

## Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.IO.Stream stream)

[C#]
public StreamReader(Stream stream)
```

### Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified stream.

### Parameters

Parameter	Description
<i>stream</i>	The <code>System.IO.Stream</code> to read.

### Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property to `System.Text.UTF8Encoding`, the `System.IO.StreamReader.BaseStream` property using *stream*, and the internal buffer to the default size. [*Note:* The default buffer size is implementation dependent.]

### Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>stream</i> does not support reading.
<b>System.ArgumentNullException</b>	<i>stream</i> is null.

# StreamReader(System.IO.Stream, System.Boolean) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.IO.Stream stream, bool detectEncodingFromByteOrderMarks)

[C#]
public StreamReader(Stream stream, bool detectEncodingFromByteOrderMarks)
```

## Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified stream, with the specified byte order mark detection option.

## Parameters

Parameter	Description
<i>stream</i>	The <i>stream</i> to read.
<i>detectEncodingFromByteOrderMarks</i>	A <code>System.Boolean</code> value that indicates whether the new <code>System.IO.StreamReader</code> is required to look for byte order marks at the beginning of the stream. Specify <code>true</code> to enable detection of byte order marks; otherwise, specify <code>false</code> .

## Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property to `System.Text.UTF8Encoding`, the `System.IO.StreamReader.BaseStream` property using *stream*, and the internal buffer to the default size. [Note: The default buffer size is implementation defined.]

If requested, the current constructor detects the encoding by examining the first three bytes of the stream. The constructor automatically recognizes UTF-8, little-endian Unicode, and big-endian Unicode text if the file starts with the appropriate byte order marks. Otherwise, UTF-8 encoding is used. For more information, see the `System.Text.Encoding.GetPreamble` method.

## Exceptions

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

<b>System.ArgumentException</b>	<i>stream</i> does not support reading.
<b>System.ArgumentNullException</b>	<i>stream</i> is null.

1

2

# StreamReader(System.IO.Stream, System.Text.Encoding) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.IO.Stream stream, class System.Text.Encoding encoding)

[C#]
public StreamReader(Stream stream, Encoding encoding)
```

## Summary

Constructs and initializes a new instance of the `System.IO.StreamReader` class for the specified stream with the specified character encoding.

## Parameters

Parameter	Description
<i>stream</i>	The <code>System.IO.Stream</code> to read.
<i>encoding</i>	A <code>System.Text.Encoding</code> that specifies the character encoding to use.

## Description

This constructor initializes the `System.IO.StreamReader.CurrentEncoding` property using *encoding*, the `System.IO.StreamReader.BaseStream` property using *stream*, and the internal buffer to the default size. [*Note:* The default buffer size is implementation defined.]

## Exceptions

Exception	Condition
<code>System.ArgumentException</code>	<i>stream</i> does not support reading.
<code>System.ArgumentNullException</code>	<i>stream</i> or <i>encoding</i> is null.

# StreamReader.Close() Method

```
[ILAsm]  
.method public hidebysig virtual void Close()  
  
[C#]  
public override void Close()
```

## Summary

Closes the current instance of `System.IO.StreamReader`, releasing any system resources associated with it.

## Description

Following a call to this method, operations on the current instance might raise exceptions.

*[Note: This version of `System.IO.StreamReader.Close` is equivalent to `System.IO.StreamReader.Dispose(true)`.*

This method overrides `System.IO.TextReader.Close`.

]

# StreamReader.DiscardBufferedData() Method

```
[ILAsm]  
.method public hidebysig instance void DiscardBufferedData()  
  
[C#]  
public void DiscardBufferedData()
```

## Summary

Allows a `System.IO.StreamReader` to discard its buffered data.

## Description

[*Note:* This method is useful when reading from a stream after seeking to a new position. If this method is not called and the internal buffer is not empty, a read attempt at the new location will first return data that is in the buffer before returning the text at the current position in the stream.]

# StreamReader.Dispose(System.Boolean) Method

```
[ILAsm]  
.method family hidebysig virtual void Dispose(bool disposing)  
  
[C#]  
protected override void Dispose(bool disposing)
```

## Summary

Releases the unmanaged resources used by the `System.IO.StreamReader` and optionally releases the managed resources.

## Parameters

Parameter	Description
<i>disposing</i>	true to release both managed and unmanaged resources; false to release only unmanaged resources.

## Description

When the *disposing* parameter is true, this method releases all resources held by any managed objects that this `System.IO.StreamReader` references. This method invokes the `Dispose()` method of each referenced object.

[Note: `System.IO.StreamReader.Dispose` can be called multiple times by other objects. When overriding `System.IO.StreamReader.Dispose(System.Boolean)`, be careful not to reference objects that have been previously disposed in an earlier call to `System.IO.StreamReader.Dispose`.

This method calls the dispose method of the base class, `System.IO.TextReader.Dispose(disposing)`.

]

# StreamReader.Peek() Method

```
[ILAsm]  
.method public hidebysig virtual int32 Peek()  
  
[C#]  
public override int Peek()
```

## Summary

Returns the next character in the underlying stream without advancing the position of the `System.IO.StreamReader` in the stream.

## Return Value

The next character from the character source as a `System.Int32`, or -1 if at the end of the stream.

## Description

[*Note:* This method returns -1 is when the end of the underlying stream is reached because a Unicode character can contain only values between hexadecimal 0x0000 to 0xFFFF (0 to 65535).

This method overrides `System.IO.TextReader.Peek`.

]

## Exceptions

Exception	Condition
<code>System.IO.IOException</code>	An I/O error occurred.



# StreamReader.Read() Method

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

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

## Summary

Reads the next character from the input stream and advances the character position by one character.

## Return Value

The next character from the character source represented as a `System.Int32`, or -1 if at the end of the stream.

## Description

[*Note:* This method returns -1 when the end of the underlying stream is reached because a Unicode character can contain only values between hexadecimal 0x0000 to 0xFFFF (0 to 65535).

This method overrides `System.IO.TextReader.Read`.

]

## Exceptions

Exception	Condition
<code>System.IO.IOException</code>	An I/O error occurred.

# StreamReader.Read(System.Char[], System.Int32, System.Int32) Method

```
[ILAsm]  
.method public hidebysig virtual int32 Read(char[] buffer, int32 index,  
int32 count)  
  
[C#]  
public override int Read(char[] buffer, int index, int count)
```

## Summary

Reads a maximum of *count* characters from the current stream into *buffer*, beginning at *index*.

## Parameters

Parameter	Description
<i>buffer</i>	A <code>System.Char</code> array. When this method returns, contains the specified character array with the values between <i>index</i> and ( <i>index</i> + <i>count</i> - 1) replaced by the characters read from the current instance.
<i>index</i>	A <code>System.Int32</code> that specifies the index of <i>buffer</i> at which to begin writing.
<i>count</i>	A <code>System.Int32</code> that specifies the maximum number of characters to read.

## Return Value

A `System.Int32` containing the number of characters that have been read, or zero if there are no more characters left to read. Can be less than *count* if the end of the stream has been reached.

## Description

[*Note:* This method returns after either *count* characters are read, or the end of the file is reached. `System.IO.TextReader.ReadBlock` is a blocking version of `System.IO.StreamReader.Read`.

This method overrides `System.IO.TextReader.Read`.

]

## Exceptions

Exception	Condition
<b>System.ArgumentException</b>	<i>buffer.Length - index &lt; count.</i>
<b>System.ArgumentNullException</b>	<i>buffer</i> is <code>null</code> .
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> or <i>count</i> is negative.
<b>System.IO.IOException</b>	An I/O error occurred.  -or-  The current stream is closed.

1

2

# StreamReader.ReadLine() Method

```
[ILAsm]  
.method public hidebysig virtual string ReadLine()  
  
[C#]  
public override string ReadLine()
```

## Summary

Reads a line of characters from the current stream and returns the data as a string.

## Return Value

A `System.String` containing the next line from the input stream, or `null` if the end of the input stream is reached.

## Description

[*Note:* This method defines a line as a sequence of characters followed by a carriage return (hexadecimal 0x000d), a line feed (hexadecimal 0x000a), or `System.Environment.NewLine`. The returned string does not contain the terminating character(s).

This method overrides `System.IO.TextReader.ReadLine`.

]

## Exceptions

Exception	Condition
<b>System.IO.IOException</b>	An I/O error occurred.
<b>System.OutOfMemoryException</b>	There is insufficient memory to allocate a buffer for the returned string.

# StreamReader.ReadToEnd() Method

```
[ILAsm]  
.method public hidebysig virtual string ReadToEnd()  
  
[C#]  
public override string ReadToEnd()
```

## Summary

Reads the stream from the current position to the end of the stream.

## Return Value

A `System.Int32` containing the rest of the stream as a string, from the current position to the end. If the current position is at the end of the stream, returns the empty string `""`.

## Description

[*Note:* This method overrides `System.IO.TextReader.ReadToEnd`.]  
]

## Exceptions

Exception	Condition
<b>System.IO.IOException</b>	An I/O error occurred.
<b>System.OutOfMemoryException</b>	There is insufficient memory to allocate a buffer for the returned string.

# StreamReader.BaseStream Property

```
[ILAsm]  
.property class System.IO.Stream BaseStream { public hidebysig virtual  
specialname class System.IO.Stream get_BaseStream() }  
  
[C#]  
public virtual Stream BaseStream { get; }
```

## Summary

Gets the underlying stream.

## Property Value

The underlying `System.IO.Stream` which the current `System.IO.StreamReader` instance is reading.

## Description

This property is read-only.

# StreamReader.CurrentEncoding Property

```
[ILAsm]  
.property class System.Text.Encoding CurrentEncoding { public hidebysig  
virtual specialname class System.Text.Encoding get_CurrentEncoding() }  
  
[C#]  
public virtual Encoding CurrentEncoding { get; }
```

## Summary

Gets the current character encoding that the current `System.IO.StreamReader` is using.

## Property Value

The current `System.Text.Encoding` used by the current reader.

## Description

This property is read-only.

The value returned by this property might change after the first call to a `System.IO.StreamReader.Read` method if encoding auto detection was specified to the constructor for the current instance.