

# System.IO.TextReader Class

```
[ILAsm]
.class public abstract serializable TextReader extends
System.MarshalByRefObject implements System.IDisposable

[C#]
public abstract class TextReader: MarshalByRefObject, IDisposable
```

## 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

Represents an object that can read a sequential series of characters.

## Inherits From: System.MarshalByRefObject

**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

`System.IO.TextReader` is designed for character input, whereas the `System.IO.StreamReader` is designed for byte input and the `System.IO.StringReader` class is designed for reading from a string.

By default, a `System.IO.TextReader` is not thread safe. For information on creating a thread-safe `System.IO.TextReader`, see `System.IO.TextReader.Synchronized`.

# 1 TextReader() Constructor

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

## 6 Summary

7 Constructs a new instance of the `System.IO.TextReader` class.

8

# 1 TextReader.Null Field

```
2 [ILAsm]  
3 .field public static initOnly class System.IO.TextReader Null  
  
4 [C#]  
5 public static readonly TextReader Null
```

## 6 Summary

7 Provides a `System.IO.TextReader` with no data to read from.

## 8 Description

9 Reading from the `System.IO.TextReader.Null` text reader is similar to reading from  
10 the end of a stream:

- 11 • `System.IO.TextReader.Read()` and `System.IO.TextReader.Peek` methods return -1
- 12 • `System.IO.TextReader.Read(System.Char[], System.Int32, System.Int32)` and  
13 `System.IO.TextReader.ReadBlock` methods return zero
- 14 • `System.IO.TextReader.ReadLine` and `System.IO.TextReader.ReadToEnd` methods  
15 return null.

# 1 TextReader.Close() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual void Close()  
  
4 [C#]  
5 public virtual void Close()
```

## 6 Summary

7 Closes the current `System.IO.TextReader` instance and releases any system resources  
8 associated with it.

## 9 Description

10 *[Note:* After a call to `System.IO.TextReader.Close`, any IO operation on the current  
11 instance might throw an exception.  
12  
13 ]

## 14 Behaviors

15 This method is equivalent to `System.IO.TextReader.Dispose( true )`.

## 17 Usage

18 Use this method to close the current instance and free any resources associated with it.  
19  
20

# 1 TextReader.Dispose(System.Boolean) Method

```
2 [ILAsm]  
3 .method family hidebysig virtual void Dispose(bool disposing)  
4 [C#]  
5 protected virtual void Dispose(bool disposing)
```

## 6 Summary

7 Releases the unmanaged resources used by the `System.IO.TextReader` and optionally  
8 releases the managed resources.

## 9 Parameters

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

## 11 Description

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

15  
16 [Note: `System.IO.TextReader.Dispose` can be called multiple times by other objects.  
17 When overriding `System.IO.TextReader.Dispose(System.Boolean)`, be careful not to  
18 reference objects that have been previously disposed in an earlier call to  
19 `System.IO.TextReader.Dispose`.]  
20  
21  
22

# 1 TextReader.Peek() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual int32 Peek()  
  
4 [C#]  
5 public virtual int Peek()
```

## 6 Summary

7 Reads the next character without changing the state of the reader or the character  
8 source.

## 9 Return Value

10 The next character to be read, or -1 if no more characters are available.

## 11 Description

12 The position of the `System.IO.TextReader` in the source is not changed by this  
13 operation.

## 14 Behaviors

15 As described above.

## 17 Default

18 The default implementation returns -1.

## 20 Exceptions

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

# 1    `TextReader.Read(System.Char[],` 2    `System.Int32, System.Int32) Method`

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

## 8    **Summary**

9       Reads at most the specified number of characters from the current character source,  
10      and writes them to the provided character array.

## 11   **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 source.
<i>index</i>	A <code>System.Int32</code> that specifies the place in <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. If the end of the stream is reached before <i>count</i> of characters is read into <i>buffer</i> , this method returns.

## 13   **Return Value**

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

## 17   **Description**

18       `System.IO.TextReader.ReadBlock` is a blocking version of this method.

## 19   **Behaviors**

20       The provided character array can be changed only in the specified range.

## 22   **Exceptions**

Exception	Condition
<b>System.ArgumentNullException</b>	<i>buffer</i> is null.
<b>System.ArgumentException</b>	$(index + count) > buffer.Length$ .
<b>System.ArgumentOutOfRangeException</b>	$index < 0$ - or - $count < 0$ .
<b>System.IO.IOException</b>	An I/O error occurred.

1

2



# 1 TextReader.Read() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual int32 Read()  
  
4 [C#]  
5 public virtual int Read()
```

## 6 Summary

7 Reads the next character from the character source and advances the character position  
8 by one character.

## 9 Return Value

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

## 12 Behaviors

13 As described above.

## 15 Default

16 The default implementation returns -1.

## 18 Exceptions

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

# 1    **TextReader.ReadBlock(System.Char[],** 2    **System.Int32, System.Int32) Method**

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

## 8    **Summary**

9       Reads a specified number of characters from the current stream into a provided  
10      character array.

## 11   **Parameters**

Parameter	Description
<i>buffer</i>	A System.Char 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 source.
<i>index</i>	A System.Int32 that specifies the index in <i>buffer</i> at which to begin writing.
<i>count</i>	A System.Int32 that specifies the maximum number of characters to read.

## 13   **Return Value**

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

## 17   **Description**

18      The method blocks until either the specified number of characters are read, or no more  
19      characters are available in the source.

## 20   **Behaviors**

21      As described above.

## 23   **Exceptions**

Exception	Condition
<b>System.ArgumentNullException</b>	<i>buffer</i> is null.
<b>System.ArgumentException</b>	$(index + count - 1) > buffer.Length$ .
<b>System.ArgumentOutOfRangeException</b>	$index < 0$ - or - $count < 0$ .
<b>System.IO.IOException</b>	An I/O error occurred.

1

2

# 1 TextReader.ReadLine() Method

```
2 [ILAsm]
3 .method public hidebysig virtual string ReadLine()
4
5 [C#]
6 public virtual string ReadLine()
```

## 6 Summary

7 Reads a line of characters from the current character source.

## 8 Return Value

9 A `System.String` containing the next line from the input stream, or `null` if all lines have  
10 been read. The returned string does not contain the line terminating character.

## 11 Description

12 A line is defined as a sequence of characters followed by a carriage return (0x000d), a  
13 line feed (0x000a), `System.Environment.NewLine`, or the end of stream marker.

## 14 Behaviors

15 As described above.

## 17 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.
<b>System.ArgumentOutOfRangeException</b>	The number of characters in the next line is larger than <code>System.Int32.MaxValue</code> .

# 1 TextReader.ReadToEnd() Method

```
2 [ILAsm]  
3 .method public hidebysig virtual string ReadToEnd()  
  
4 [C#]  
5 public virtual string ReadToEnd()
```

## 6 Summary

7 Reads all characters from the current position in the character source to the end of the  
8 source.

## 9 Return Value

10 A string containing all characters from the current position to the end of the character  
11 source.

## 12 Behaviors

13 As described above.

## 15 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.
<b>System.ArgumentOutOfRangeException</b>	The number of characters from the current position to the end of the underlying stream is larger than <code>System.Int32.MaxValue</code> .

# 1

## 2 TextReader.Synchronized(System.IO.TextRea

### 3 der) Method

```
4 [ILAsm]  
5 .method public hidebysig static class System.IO.TextReader  
6 Synchronized(class System.IO.TextReader reader)  
  
7 [C#]  
8 public static TextReader Synchronized(TextReader reader)
```

#### 9 Summary

10 Creates a thread-safe wrapper around the specified `System.IO.TextReader` instance.

#### 11 Parameters

Parameter	Description
<i>reader</i>	The <code>System.IO.TextReader</code> to synchronize.

#### 12 Return Value

14 A thread-safe `System.IO.TextReader`.

#### 15 Description

16 This method returns a `System.IO.TextReader` instance that wraps around the specified  
17 `System.IO.TextReader` instance and restricts concurrent access to it by multiple  
18 threads.

#### 19 Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	The <i>reader</i> parameter is null.

# 1 TextReader.System.IDisposable.Dispose() 2 Method

```
3 [ILAsm]  
4 .method private final hidebysig virtual void System.IDisposable.Dispose()  
  
5 [C#]  
6 void IDisposable.Dispose()
```

## 7 Summary

8 Implemented to support the System.IDisposable interface. [Note: For more  
9 information, see System.IDisposable.Dispose.]

10