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:* 1.0.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**.

TextReader() Constructor

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

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

Summary

Constructs a new instance of the **System.IO.TextReader** class.

TextReader.Null Field

```
[ILASM]
.field public static initOnly class System.IO.TextReader
Null

[C#]
public static readonly TextReader Null
```

Summary

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

Description

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

- **System.IO.TextReader.Read**() and **System.IO.TextReader.Peek** methods return -1
- **System.IO.TextReader.Read (System.Char**[], **System.Int32**, **System.Int32**) and **System.IO.TextReader.ReadBlock** methods return zero
- **System.IO.TextReader.ReadLine** and **System.IO.TextReader.ReadToEnd** methods return **null**.

TextReader.Close() Method

```
[ILASM]
.method public hidebysig virtual void Close()

[C#]
public virtual void Close()
```

Summary

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

Description

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

Behaviors

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

Usage

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

TextReader.Dispose(System.Boolean) Method

```
[ILASM]
.method family hidebysig virtual void Dispose(bool
disposing)

[C#]
protected virtual void Dispose(bool disposing)
```

Summary

Releases the unmanaged resources used by the **System.IO.TextReader** and optionally releases the managed resources.

Parameters

Parameter	Description
disposing	**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.TextReader** references. This method invokes the **Dispose()** method of each referenced object.

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

TextReader.Peek() Method

```
[ILASM]
.method public hidebysig virtual int32 Peek()

[C#]
public virtual int Peek()
```

Summary

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

Return Value

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

Description

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

Behaviors

As described above.

Default

The default implementation returns -1.

Exceptions

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

TextReader.Read(System.Char[], System.Int32, System.Int32) Method

```
[ILASM]
.method public hidebysig virtual int32 Read(class
System.Char[] buffer, int32 index, int32 count)

[C#]
public virtual int Read(char[] buffer, int index, int
count)
```

Summary

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

Parameters

Parameter	Description
buffer	A **System.Char** array. When this method returns, contains the specified character array with the values between *index* and (*index* + *count* -1) replaced by the characters read from the current source.
index	A **System.Int32** that specifies the place in *buffer* at which to begin writing.
count	A **System.Int32** that specifies the maximum number of characters to read. If the end of the stream is reached before *count* of characters is read into *buffer*, this method returns.

Return Value

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

Description

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

Behaviors

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

Exceptions

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

TextReader.Read() Method

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

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

Summary

Reads the next character from the character source 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.

Behaviors

As described above.

Default

The default implementation returns -1.

Exceptions

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

TextReader.ReadBlock(System.Char[], System.Int32, System.Int32) Method

```
[ILASM]
.method public hidebysig virtual int32 ReadBlock(class
System.Char[] buffer, int32 index, int32 count)

[C#]
public virtual int ReadBlock(char[] buffer, int index, int
count)
```

Summary

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

Parameters

Parameter	Description
buffer	A **System.Char** array. When this method returns, contains the specified character array with the values between *index* and *(index + count)* replaced by the characters read from the current source.
index	A **System.Int32** that specifies the index in *buffer* at which to begin writing.
count	A **System.Int32** that specifies the maximum number of characters to read.

Return Value

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

Description

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

Behaviors

As described above.

Exceptions

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

1
2
3

TextReader.ReadLine() Method

```
[ILASM]
.method public hidebysig virtual string ReadLine()

[C#]
public virtual string ReadLine()
```

Summary

Reads a line of characters from the current character source.

Return Value

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

Description

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

Behaviors

As described above.

Exceptions

Exception	Condition
System.IO.IOException	An I/O error occurred.
System.OutOfMemoryException	There is insufficient memory to allocate a buffer for the returned string.
System.ArgumentOutOfRangeException	The number of characters in the next line is larger than **System.Int32.MaxValue**.

TextReader.ReadToEnd() Method

```
[ILASM]
.method public hidebysig virtual string ReadToEnd()

[C#]
public virtual string ReadToEnd()
```

Summary

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

Return Value

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

Behaviors

As described above.

Exceptions

Exception	Condition
System.IO.IOException	An I/O error occurred.
System.OutOfMemoryException	There is insufficient memory to allocate a buffer for the returned string.
System.ArgumentOutOfRangeException	The number of characters from the current position to the end of the underlying stream is larger than **System.Int32.MaxValue**.

TextReader.Synchronized(System.IO.TextReader) Method

```
[ILASM]
.method public hidebysig static class System.IO.TextReader
Synchronized(class System.IO.TextReader reader)

[C#]
public static TextReader Synchronized(TextReader reader)
```

Summary

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

Parameters

Parameter	Description
reader	The **System.IO.TextReader** to synchronize.

Return Value

A thread-safe **System.IO.TextReader**.

Description

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

Exceptions

Exception	Condition
System.ArgumentNullException	The *reader* parameter is **null**.

TextReader.System.IDisposable.Dispose() Method

```
[ILASM]
.method private final hidebysig virtual void
System.IDisposable.Dispose()

[C#]
void IDisposable.Dispose()
```

Summary

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