System.IO.Stream Class

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

[C#]
public abstract class Stream: 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

Abstract base class for all stream implementations.

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

Streams involve three fundamental operations:

- You can read from streams. Reading is the transfer of data from a stream into a data structure, such as an array of bytes.
- You can write to streams. Writing is the transfer of data from a data structure into a stream.
- Streams can support seeking. Seeking is the querying and modifying of the current position within a stream. Seek capability depends on the kind of backing store a stream has. For example, network streams have no unified concept of a current position, and therefore typically do not support seeking.

All classes that represent streams inherit from the `System.IO.Stream` class. The `System.IO.Stream` class and its subclasses provide a generic view of data sources and repositories, isolating the programmer from the specific details of the operating system and underlying devices.

Subclasses are required to provide implementations only for the synchronous read and write methods. The asynchronous read and write methods are implemented via the synchronous ones. [*Note:* The `System.IO.Stream` synchronous read and write methods are `System.IO.Stream.Read` and `System.IO.Stream.Write`. The asynchronous read and write methods are `System.IO.Stream.BeginRead`, `System.IO.Stream.EndRead`, `System.IO.Stream.BeginWrite`, and `System.IO.Stream.EndWrite`.]

Depending on the underlying data source or repository, streams might support only some of these capabilities. An application can query a stream for its capabilities by using the `System.IO.Stream.CanRead`, `System.IO.Stream.CanWrite`, and `System.IO.Stream.CanSeek` properties.

The `System.IO.Stream.Read` and `System.IO.Stream.Write` methods read and write data in a variety of formats. For streams that support seeking, the `System.IO.Stream.Seek` and `System.IO.Stream.SetLength` methods, and the `System.IO.Stream.Position` and `System.IO.Stream.Length` properties can be used to query and modify the current position and length of a stream.

Some stream implementations perform local buffering of the underlying data to improve performance. For such streams, the `System.IO.Stream.Flush` method can be used to clear any internal buffers and ensure that all data has been written to the underlying data source or repository.

Calling `System.IO.Stream.Close` on a `System.IO.Stream` flushes any buffered data, essentially calling `System.IO.Stream.Flush` for you. `System.IO.Stream.Close` also releases operating system resources such as file handles, network connections, or memory used for any internal buffering.

If you need a `System.IO.Stream` with no backing store (i.e., a bit bucket), use `System.IO.Stream.Null`.

Stream() Constructor

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

[C#]
protected Stream()
```

Summary

Constructs a new instance of the `System.IO.Stream` class.

Stream.Null Field

```
[ILAsm]
.field public static initOnly class System.IO.Stream Null

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

Summary

Returns a `System.IO.Stream` with no backing store.

Description

[*Note:* `System.IO.Stream.Null` is used to redirect output to a stream that does not consume any operating system resources. When the methods of `System.IO.Stream` that provide writing are invoked on `System.IO.Stream.Null`, they simply return, and no data is written. `System.IO.Stream.Null` also implements a `System.IO.Stream.Read` method that returns zero without reading data.]

Stream.BeginRead(System.Byte[], System.Int32, System.Int32, System.AsyncCallback, System.Object) Method

```
[ILAsm]
.method public hidebysig virtual class System.IAsyncResult
BeginRead(class System.Byte[] buffer, int32 offset, int32 count,
class System.AsyncCallback callback, object state)

[C#]
public virtual IAsyncResult BeginRead(byte[] buffer, int offset, int
count, AsyncCallback callback, object state)
```

Summary

Begins an asynchronous read operation.

Parameters

Parameter	Description
buffer	The `System.Byte` array to read the data into.
offset	A `System.Int32` that specifies the byte offset in *buffer* at which to begin writing data read from the stream.
count	A `System.Int32` that specifies the maximum number of bytes to read from the stream.
callback	A `System.AsyncCallback` delegate to be called when the read is complete, or `null`.
state	An application-defined object, or `null`.

Return Value

A `System.IAsyncResult` that contains information about the asynchronous read operation, which could still be pending.

Description

This method starts an asynchronous read operation. To determine how many bytes were read and release resources allocated by this method, call the `System.IO.Stream.EndRead` method and specify the `System.IAsyncResult` object returned by this method. [*Note:* The `System.IO.Stream.EndRead` method should be called exactly once for each call to `System.IO.Stream.BeginRead`.]

If the *callback* parameter is not `null`, the method referenced by *callback* is invoked when the asynchronous operation completes. The `System.IAsyncResult` object returned by this method is passed as the argument to the method referenced by *callback*.

The current position in the stream is updated when the asynchronous read or write is issued, not when the I/O operation completes.

Multiple simultaneous asynchronous requests render the request completion order unspecified.

The *state* parameter can be any object that the caller wishes to have available for the duration of the asynchronous operation. This object is available via the `System.IAsyncResult.AsyncState` property of the object returned by this method.

[*Note:* Use the `System.IO.Stream.CanRead` property to determine whether the current instance supports reading.]

Behaviors

As described above.

Exceptions

Exception	Condition
System.NotSupportedException	The current `System.IO.Stream` does not support reading.
System.ObjectDisposedException	The stream is closed.
System.IO.IOException	An I/O error occurred.

Stream.BeginWrite(System.Byte[], System.Int32, System.Int32, System.AsyncCallback, System.Object) Method

```
[ILAsm]
.method public hidebysig virtual class System.IAsyncResult
BeginWrite(class System.Byte[] buffer, int32 offset, int32 count,
class System.AsyncCallback callback, object state)

[C#]
public virtual IAsyncResult BeginWrite(byte[] buffer, int offset,
int count, AsyncCallback callback, object state)
```

Summary

Begins an asynchronous write operation.

Parameters

Parameter	Description
buffer	The `System.Byte` array to be written to the current stream.
offset	A `System.Int32` that specifies the byte offset in *buffer* at which to begin copying bytes to the current stream.
count	A `System.Int32` that specifies the maximum number of bytes to be written to the current stream.
callback	A `System.AsyncCallback` delegate to be called when the write is complete, or `null`.
state	An application-defined object, or `null`.

Return Value

A `System.IAsyncResult` that represents the asynchronous write, which could still be pending.

Description

Pass the `System.IAsyncResult` returned by this method to `System.IO.Stream.EndWrite` to ensure that the write completes and frees resources appropriately. If an error occurs during an asynchronous write, an exception will not be thrown until `System.IO.Stream.EndWrite` is called with the `System.IAsyncResult` returned by this method. [*Note:* If a failure is detected from the underlying OS (such as if a floppy is ejected in the middle of the

operation), the results of the write operation are undefined.]

If the *callback* parameter is not `null`, the method referenced by *callback* is invoked when the asynchronous operation completes. The `System.IAsyncResult` object returned by this method is passed as the argument to the method referenced by *callback*.

The *state* parameter can be any object that the caller wishes to have available for the duration of the asynchronous operation. This object is available via the `System.IAsyncResult.AsyncState` property of the object returned by this method.

If a stream is writable, writing at the end of it expands the stream.

The current position in the stream is updated when you issue the asynchronous read or write, not when the I/O operation completes. Multiple simultaneous asynchronous requests render the request completion order uncertain.

[*Note: buffer* should generally be greater than 64 KB.

Use the `System.IO.Stream.CanWrite` property to determine whether the current instance supports writing.

]

Behaviors

As described above.

Exceptions

Exception	Condition
System.NotSupportedException	The current `System.IO.Stream` does not support writing.
System.ObjectDisposedException	The stream is closed.
System.IO.IOException	An I/O error occurred.

Stream.Close() Method

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

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

Summary

Closes the current stream and releases any resources associated with the current stream.

Description

Following a call to this method, a call to another operation on the same stream might result in an exception (such as `System.ObjectDisposedException`, for example). However, if the stream is already closed, a call to `System.IO.Stream.Close` throws no exceptions.

[*Note:* If this method is called while an asynchronous read or write is pending for a stream, the behavior of the stream is undefined.]

Behaviors

As described above.

Stream.CreateWaitHandle() Method

```
[ILAsm]
.method family hidebysig virtual class System.Threading.WaitHandle
CreateWaitHandle()

[C#]
protected virtual WaitHandle CreateWaitHandle()
```

Summary

Allocates a `System.Threading.WaitHandle` object.

Return Value

A reference to the allocated `System.Threading.WaitHandle`.

Description

When called for the first time this method creates a `System.Threading.WaitHandle` object and returns it. On subsequent calls, the `System.IO.Stream.CreateWaitHandle` method returns a reference to the same wait handle.

[*Note:* `System.IO.Stream.CreateWaitHandle` is useful if you implement the asynchronous methods and require a way of blocking in `System.IO.Stream.EndRead` or `System.IO.Stream.EndWrite` until the asynchronous operation is complete.]

Stream.EndRead(System.IAsyncResult) Method

```
[ILAsm]
.method public hidebysig virtual int32 EndRead(class
System.IAsyncResult asyncResult)

[C#]
public virtual int EndRead(IAsyncResult asyncResult)
```

Summary

Ends a pending asynchronous read request.

Parameters

Parameter	Description
asyncResult	The `System.IAsyncResult` object that references the pending asynchronous read request.

Return Value

A `System.Int32` that indicates the number of bytes read from the stream, between 0 and the number of bytes specified via the `System.IO.Stream.BeginRead` parameter *count*. Streams only return 0 at the end of the stream, otherwise, they block until at least 1 byte is available.

Description

`System.IO.Stream.EndRead` blocks until the I/O operation has completed.

Behaviors

As described above.

Exceptions

Exception	Condition
System.ArgumentNullException	*asyncResult* is `null`.
System.ArgumentException	*asyncResult* did not originate from a `System.IO.Stream.BeginRead` method on the current stream.

Stream.EndWrite(System.IAsyncResult) Method

```
[ILAsm]
.method public hidebysig virtual void EndWrite(class
System.IAsyncResult asyncResult)

[C#]
public virtual void EndWrite(IAsyncResult asyncResult)
```

Summary

Ends an asynchronous write operation.

Parameters

Parameter	Description
asyncResult	A `System.IAsyncResult` that references the outstanding asynchronous I/O request.

Description

`System.IO.Stream.EndWrite` is required to be called exactly once for every `System.IO.Stream.BeginWrite`. `System.IO.Stream.EndWrite` blocks until the write I/O operation has completed.

Behaviors

As described above.

Exceptions

Exception	Condition
System.ArgumentNullException	The *asyncResult* parameter is `null`.
System.ArgumentException	*asyncResult* did not originate from a `System.IO.Stream.BeginWrite` method on the current stream.

Stream.Flush() Method

```
[ILAsm]
.method public hidebysig virtual abstract void Flush()

[C#]
public abstract void Flush()
```

Summary

Flushes the internal buffer.

Description

[*Note:* Implementers should use this method to move any information from an underlying buffer to its destination. The `System.IO.Stream.Flush` method should clear the buffer, but the stream should not be closed. Depending upon the state of the object, the current position within the stream might need to be modified (for example, if the underlying stream supports seeking). For additional information see `System.IO.Stream.CanSeek`.]

Behaviors

As described above.

How and When to Override

Override `System.IO.Stream.Flush` on streams that implement a buffer.

Exceptions

Exception	Condition
System.IO.IOException	An I/O error occurs.
System.ObjectDisposedException	The stream is closed.

Stream.Read(System.Byte[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig virtual abstract int32 Read(class
System.Byte[] buffer, int32 offset, int32 count)

[C#]
public abstract int Read(byte[] buffer, int offset, int count)
```

Summary

Reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read.

Parameters

Parameter	Description
buffer	A `System.Byte` array. When this method returns, the elements between *offset* and *(offset + count - 1)* are replaced by the bytes read from the current source.
offset	A `System.Int32` that specifies the zero based byte offset in *buffer* at which to begin storing the data read from the current stream.
count	A `System.Int32` that specifies the maximum number of bytes to be read from the current stream.

Return Value

A `System.Int32` that specifies the total number of bytes read into the buffer, or zero if the end of the stream has been reached before any data can be read.

Description

[*Note:* Use the `System.IO.Stream.CanRead` property to determine whether the current instance supports reading.]

Behaviors

As described above.

Exceptions

Exception	Condition
System.ArgumentException	(*offset* + *count* - 1) is greater than the length of *buffer*.
System.ArgumentNullException	*buffer* is `null`.
System.ArgumentOutOfRangeException	*offset* or *count* is less than zero.
System.NotSupportedException	The current stream does not support reading.
System.ObjectDisposedException	The stream is closed.
System.IO.IOException	An I/O error occurred.

Stream.ReadByte() Method

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

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

Summary

Reads a byte from the stream and advances the position within the stream by one byte.

Return Value

The unsigned byte cast to a `System.Int32`, or -1 if at the end of the stream.

Description

Behaviors

As described above.

[*Note:* Use the `System.IO.Stream.CanRead` property to determine whether the current instance supports reading.]

Exceptions

Exception	Condition
System.NotSupportedException	The stream does not support reading.
System.ObjectDisposedException	The stream is closed.
System.IO.IOException	An I/O error has occurred.

Stream.Seek(System.Int64, System.IO.SeekOrigin) Method

```
[ILAsm]
.method public hidebysig virtual abstract int64 Seek(int64 offset,
valuetype System.IO.SeekOrigin origin)

[C#]
public abstract long Seek(long offset, SeekOrigin origin)
```

Summary

Changes the position within the current stream by the given offset, which is relative to the stated origin.

Parameters

Parameter	Description
offset	A `System.Int64` that specifies the byte offset relative to origin.
origin	A `System.IO.SeekOrigin` value indicating the reference point used to obtain the new position.

Return Value

A `System.Int64` that specifies the new position within the current stream.

Description

[*Note:* Use the `System.IO.Stream.CanSeek` property to determine whether the current instance supports seeking.]

Behaviors

If *offset* is negative, the new position is required to precede the position specified by *origin* by the number of bytes specified by *offset*. If *offset* is zero, the new position is required to be the position specified by *origin*. If *offset* is positive, the new position is required to follow the position specified by *origin* by the number of bytes specified by *offset*.

How and When to Override

Classes derived from `System.IO.Stream` that support seeking are required to override this method.

Exceptions

Exception	Condition
System.NotSupportedException	The stream does not support seeking, such as if the stream is constructed from a pipe or console output.
System.ObjectDisposedException	The stream is closed.
System.IO.IOException	An I/O error has occurred.

Stream.SetLength(System.Int64) Method

```
[ILAsm]
.method public hidebysig virtual abstract void SetLength(int64
value)

[C#]
public abstract void SetLength(long value)
```

Summary

Sets the length of the current stream.

Parameters

Parameter	Description
value	A `System.Int64` that specifies the desired length of the current stream in bytes.

Description

[*Note:* Use the `System.IO.Stream.CanWrite` property to determine whether the current instance supports writing, and the `System.IO.Stream.CanSeek` property to determine whether seeking is supported.]

Behaviors

If the specified value is less than the current length of the stream, the stream is truncated. If the specified value is larger than the current length of the stream, the stream is expanded. If the stream is expanded, the contents of the stream between the old and the new length are initialized to zeros.

Default

There is no default implementation.

How and When to Override

Classes derived from `System.IO.Stream` are required to support both writing and seeking for `System.IO.Stream.SetLength` to work.

Exceptions

Exception	Condition
System.NotSupportedException	The stream does not support both writing and seeking, such as if the stream is constructed from a pipe or console output.
System.ObjectDisposedException	The stream is closed.
System.IO.IOException	An I/O error occurred.

Stream.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`.]

Stream.Write(System.Byte[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig virtual abstract void Write(class
System.Byte[] buffer, int32 offset, int32 count)

[C#]
public abstract void Write(byte[] buffer, int offset, int count)
```

Summary

Writes a sequence of bytes to the current stream and advances the current position within the current stream by the number of bytes written.

Parameters

Parameter	Description
buffer	A `System.Byte` array containing the data to write.
offset	A `System.Int32` that specifies the zero based byte offset in *buffer* at which to begin copying bytes to the current stream.
count	A `System.Int32` that specifies the number of bytes to be written to the current stream.

Description

[*Note:* Use the `System.IO.Stream.CanWrite` property to determine whether the current instance supports writing.]

Behaviors

If the write operation is successful, the position within the stream advances by the number of bytes written. If an exception occurs, the position within the stream remains unchanged.

Exceptions

Exception	Condition
System.ArgumentException	(*offset* + *count*) is greater than the length of *buffer*.
System.ArgumentNullException	*buffer* is `null`.

System.ArgumentOutOfRangeException	*offset* or *count* is negative.
System.NotSupportedException	The stream does not support writing.
System.ObjectDisposedException	The stream is closed.
System.IO.IOException	An I/O error occurred.

Stream.WriteByte(System.Byte) Method

```
[ILAsm]
.method public hidebysig virtual void WriteByte(unsigned int8 value)

[C#]
public virtual void WriteByte(byte value)
```

Summary

Writes a `System.Byte` to the current position in the stream and advances the position within the stream by one byte.

Parameters

Parameter	Description
value	The `System.Byte` to write to the stream.

Description

[*Note:* Use the `System.IO.Stream.CanWrite` property to determine whether the current instance supports writing.]

Behaviors

As described above.

Exceptions

Exception	Condition
System.NotSupportedException	The stream does not support writing.
System.ObjectDisposedException	The stream is closed.
System.IO.IOException	An I/O error has occurred.

Stream.CanRead Property

```
[ILAsm]
.property bool CanRead { public hidebysig virtual abstract
specialname bool get_CanRead() }

[C#]
public abstract bool CanRead { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current stream supports reading.

Property Value

`true` if the stream supports reading; otherwise, `false`.

Description

If a class derived from `System.IO.Stream` does not support reading, the following methods throw a `System.NotSupportedException`: `System.IO.Stream.BeginRead`, `System.IO.Stream.Read` and `System.IO.Stream.ReadByte`.

Behaviors

As described above.

Stream.CanSeek Property

```
[ILAsm]
.property bool CanSeek { public hidebysig virtual abstract
specialname bool get_CanSeek() }

[C#]
public abstract bool CanSeek { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current stream supports seeking.

Property Value

`true` if the stream supports seeking; otherwise, `false`.

Description

If a class derived from `System.IO.Stream` does not support seeking, the following methods throw a `System.NotSupportedException`: `System.IO.Stream.Length`, `System.IO.Stream.SetLength`, `System.IO.Stream.Position`, or `System.IO.Stream.Seek`.

Behaviors

As described above.

Stream.CanWrite Property

```
[ILAsm]
.property bool CanWrite { public hidebysig virtual abstract
specialname bool get_CanWrite() }

[C#]
public abstract bool CanWrite { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current stream supports writing.

Property Value

`true` if the stream supports writing; otherwise, `false`.

Description

If a class derived from `System.IO.Stream` does not support writing, the following methods throw a `System.NotSupportedException`: `System.IO.Stream.Write`, `System.IO.Stream.WriteByte`, and `System.IO.Stream.BeginWrite`.

Behaviors

As described above.

Stream.Length Property

```
[ILAsm]
.property int64 Length { public hidebysig virtual abstract
specialname int64 get_Length() }

[C#]
public abstract long Length { get; }
```

Summary

Gets the length in bytes of the stream.

Property Value

A `System.Int64` value representing the length of the stream in bytes.

Description

[*Note:* Use the `System.IO.Stream.CanSeek` property to determine whether the current instance supports seeking.]

Behaviors

This property is read-only.

Exceptions

Exception	Condition
System.NotSupportedException	The stream does not support seeking.
System.ObjectDisposedException	The stream is closed.

Stream.Position Property

```
[ILAsm]
.property int64 Position { public hidebysig virtual abstract
specialname int64 get_Position() public hidebysig virtual abstract
specialname void set_Position(int64 value) }

[C#]
public abstract long Position { get; set; }
```

Summary

Gets or sets the position within the current stream.

Property Value

A `System.Int64` that specifies the current position within the stream.

Description

The stream is required to support seeking to get or set the position. [*Note:* Use the `System.IO.Stream.CanSeek` property to determine whether the current instance supports seeking.]

Classes that derive from `System.IO.Stream` are required to provide an implementation of this property.

Behaviors

As described above.

Exceptions

Exception	Condition
System.NotSupportedException	The stream does not support seeking.
System.ObjectDisposedException	The stream is closed.
System.IO.IOException	An I/O error has occurred.