

System.Net.Sockets.NetworkStream Class

```
[ILAsm]
.class public NetworkStream extends System.IO.Stream

[C#]
public class NetworkStream: Stream
```

Assembly Info:

- *Name:* System
- *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 the standard stream mechanism to read and write network data through an instance of the `System.Net.Sockets.Socket` class.

Inherits From: System.IO.Stream

Library: Networking

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.Net.Sockets.NetworkStream` class allows network data to be read and written in the same manner as the `System.IO.Stream` class.

This class supports simultaneous synchronous and asynchronous access to the network data. Random access is not supported and thus the `System.Net.Sockets.NetworkStream.CanSeek` property always returns false.

The following properties and methods inherited from the `System.IO.Stream` class are not supported and throw a `System.NotSupportedException` exception when accessed:

- `System.Net.Sockets.NetworkStream.Length`
- `System.Net.Sockets.NetworkStream.Position`
- `System.Net.Sockets.NetworkStream.Seek`

- 1 • System.Net.Sockets.NetworkStream.SetLength
- 2 The System.Net.Sockets.NetworkStream.Flush method is reserved for future use but
- 3 does not throw an exception.
- 4

NetworkStream(System.Net.Sockets.Socket, System.IO.FileAccess, System.Boolean) Constructor

```
[ILAsm]  
public rtspecialname specialname instance void .ctor(class  
System.Net.Sockets.Socket socket, valuetype System.IO.FileAccess access,  
bool ownsSocket)
```

```
[C#]  
public NetworkStream(Socket socket, FileAccess access, bool ownsSocket)
```

Summary

Constructs and initializes a new instance of the `System.Net.Sockets.NetworkStream` class.

Parameters

Parameter	Description
<i>socket</i>	An instance of the <code>System.Net.Sockets.Socket</code> class.
<i>access</i>	One of the values of the <code>System.IO.FileAccess</code> enumeration.
<i>ownsSocket</i>	true if <i>socket</i> is owned by the current instance; otherwise, false.

Description

socket is required to be an instance of the `System.Net.Sockets.Socket` class with its `System.Net.Sockets.Socket.Connected` property equal to true, `System.Net.Sockets.Socket.Blocking` property equal to true, and `System.Net.Sockets.SocketType` equal to `System.Net.Sockets.SocketType.Stream`.

When *ownsSocket* is true, the current instance owns *socket*, meaning the `System.Net.Sockets.NetworkStream.Close` and `System.Net.Sockets.NetworkStream.Dispose` methods call the `System.Net.Sockets.Socket.Close` method of *socket*.

Exceptions

Exception	Condition
System.ArgumentNullException	<i>socket</i> is null.

System.IO.IOException	<p>The <code>System.Net.Sockets.Socket.Blocking</code> property of <i>socket</i> is false.</p> <p>-or-</p> <p>The <code>System.Net.Sockets.Socket.Connected</code> property of <i>socket</i> is false.</p> <p>-or-</p> <p>The <code>System.Net.Sockets.Socket.SocketType</code> property of <i>socket</i> is not <code>System.Net.Sockets.SocketType.Stream</code>.</p>
------------------------------	---

1

2

NetworkStream(System.Net.Sockets.Socket, System.IO.FileAccess) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.Net.Sockets.Socket socket, valuetype System.IO.FileAccess access)

[C#]
public NetworkStream(Socket socket, FileAccess access)
```

Summary

Constructs and initializes a new instance of the `System.Net.Sockets.NetworkStream` class.

Parameters

Parameter	Description
<i>socket</i>	An instance of the <code>System.Net.Sockets.Socket</code> class.
<i>access</i>	One of the values of the <code>System.IO.FileAccess</code> enumeration.

Description

This constructor is equivalent to `System.Net.Sockets.NetworkStream.NetworkStream(socket, access, false)`.

Exceptions

Exception	Condition
System.ArgumentNullException	<i>socket</i> is null.
System.IO.IOException	The <code>System.Net.Sockets.Socket.Blocking</code> property of <i>socket</i> is false.
	-or-
	The <code>System.Net.Sockets.Socket.Connected</code> property of <i>socket</i> is false.
	-or-
	The <code>System.Net.Sockets.Socket.SocketType</code> property

	of <i>socket</i> is not System.Net.Sockets.SocketType.Stream.
--	--

1

2

NetworkStream(System.Net.Sockets.Socket) Constructor

```
[ILAsm]  
public rtspecialname specialname instance void .ctor(class  
System.Net.Sockets.Socket socket)  
  
[C#]  
public NetworkStream(Socket socket)
```

Summary

Constructs and initializes a new instance of the `System.Net.Sockets.NetworkStream` class.

Parameters

Parameter	Description
<i>socket</i>	An instance of the <code>System.Net.Sockets.Socket</code> class.

Description

This constructor is equivalent to `System.Net.Sockets.NetworkStream.NetworkStream(socket, System.IO.FileAccess.ReadWrite, false)`.

Exceptions

Exception	Condition
System.ArgumentNullException	<i>socket</i> is null.
System.IO.IOException	The <code>System.Net.Sockets.Socket.Blocking</code> property of <i>socket</i> is false.
	-or-
	The <code>System.Net.Sockets.Socket.Connected</code> property of <i>socket</i> is false.
	-or-
	The <code>System.Net.Sockets.Socket.SocketType</code> property of <i>socket</i> is not

	<code>System.Net.Sockets.SocketType.Stream.</code>
--	--

1

2

NetworkStream(System.Net.Sockets.Socket, System.Boolean) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.Net.Sockets.Socket socket, bool ownsSocket)

[C#]
public NetworkStream(Socket socket, bool ownsSocket)
```

Summary

Constructs and initializes a new instance of the `System.Net.Sockets.NetworkStream` class.

Parameters

Parameter	Description
<i>socket</i>	An instance of the <code>System.Net.Sockets.Socket</code> class.
<i>ownsSocket</i>	true if <i>socket</i> is owned by the current instance; otherwise, false.

Description

This constructor is equivalent to `System.Net.Sockets.NetworkStream.NetworkStream(socket, System.IO.FileAccess.ReadWrite, ownsSocket)`.

Exceptions

Exception	Condition
System.ArgumentNullException	<i>socket</i> is null.
System.IO.IOException	The <code>System.Net.Sockets.Socket.Blocking</code> property of <i>socket</i> is false.
	-or-
	The <code>System.Net.Sockets.Socket.Connected</code> property of <i>socket</i> is false.
	-or-

	The <code>System.Net.Sockets.Socket.SocketType</code> property of <i>socket</i> is not <code>System.Net.Sockets.SocketType.Stream</code> .
--	--

1

2

1 **NetworkStream.BeginRead(System.Byte[],** 2 **System.Int32, System.Int32,** 3 **System.AsyncCallback, System.Object)** 4 **Method**

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

```
9    [C#]  
10   public override IAsyncResult BeginRead(byte[] buffer, int offset, int  
11   size, AsyncCallback callback, object state)
```

12 **Summary**

13 Begins an asynchronous operation to read data from the current instance.

14 **Parameters**

Parameter	Description
<i>buffer</i>	A System.Byte array to store data read from the stream.
<i>offset</i>	A System.Int32 containing the zero-based position in <i>buffer</i> at which to begin storing the data.
<i>size</i>	A System.Int32 containing the number of bytes to read.
<i>callback</i>	A System.AsyncCallback delegate, or null.
<i>state</i>	An application-defined object, or null.

15 **Return Value**

17 A System.IAsyncResult instance that contains information about the asynchronous
18 operation.

19 **Description**

20 To retrieve the results of the operation and release resources allocated by the
21 System.Net.Sockets.NetworkStream.BeginRead method, call the
22 System.Net.Sockets.NetworkStream.EndRead method, and specify the
23 System.IAsyncResult object returned by this method.
24

[*Note:* The `System.Net.Sockets.NetworkStream.EndRead` method should be called exactly once for each call to the `System.Net.Sockets.NetworkStream.BeginRead` method.]

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 method referenced by *callback* can retrieve the results of the operation by calling the `System.Net.Sockets.NetworkStream.EndRead` method.

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:* This method overrides `System.IO.Stream.BeginRead`.

]

Exceptions

Exception	Condition
System.ArgumentNullException	<i>buffer</i> is <code>null</code> .
System.ArgumentOutOfRangeException	<i>offset</i> < 0. -or- <i>offset</i> > <i>buffer.Length</i> . -or- <i>size</i> < 0. -or- <i>size</i> > <i>buffer.Length</i> - <i>offset</i> .
System.IO.IOException	An error occurred while accessing the underlying socket. [<i>Note:</i> Any exception thrown by the <code>System.Net.Sockets.Socket.BeginReceive</code> method is caught and rethrown as an <code>IOException</code> with the original exception stored in the <code>System.Exception.InnerException</code>

	property.]
System.ObjectDisposedException	The current instance has been disposed.

1

2 Example

3 For an outline of an asynchronous operation, see the
4 `System.Net.Sockets.Socket.BeginAccept` method. For the complete example, see the
5 `System.Net.Sockets.Socket` class overview.

6

1 **NetworkStream.BeginWrite(System.Byte[],** 2 **System.Int32, System.Int32,** 3 **System.AsyncCallback, System.Object)** 4 **Method**

```
5    [ILAsm]  
6    .method public hidebysig virtual class System.IAsyncResult  
7    BeginWrite(class System.Byte[] buffer, int32 offset, int32 size, class  
8    System.AsyncCallback callback, object state)  
  
9    [C#]  
10   public override IAsyncResult BeginWrite(byte[] buffer, int offset, int  
11   size, AsyncCallback callback, object state)
```

12 **Summary**

13 Begins an asynchronous operation to write data to the current instance.

14 **Parameters**

Parameter	Description
<i>buffer</i>	A System.Byte array containing data to write to the stream.
<i>offset</i>	A System.Int32 containing the zero-based position in <i>buffer</i> containing the starting location of the data to write.
<i>size</i>	A System.Int32 containing the number of bytes to write to the stream.
<i>callback</i>	A System.AsyncCallback delegate, or null.
<i>state</i>	An application-defined object, or null.

16 **Return Value**

17 A System.IAsyncResult instance that contains information about the asynchronous
18 operation.

19 **Description**

20 To release resources allocated by the System.Net.Sockets.NetworkStream.BeginWrite
21 method, call the System.Net.Sockets.NetworkStream.EndWrite method, and specify
22 the System.IAsyncResult object returned by this method.

23
24 [Note: The System.Net.Sockets.NetworkStream.EndWrite method should be called

exactly once for each call to the `System.Net.Sockets.NetworkStream.BeginWrite` method.]

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 method referenced by *callback* can retrieve the results of the operation by calling the `System.Net.Sockets.NetworkStream.EndWrite` method.

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:* This method overrides `System.IO.Stream.BeginWrite`.

]

Exceptions

Exception	Condition
System.ArgumentNullException	<i>buffer</i> is <code>null</code> .
System.ArgumentOutOfRangeException	<i>offset</i> < 0. -or- <i>offset</i> > <i>buffer.Length</i> . -or- <i>size</i> < 0. -or- <i>size</i> > <i>buffer.Length</i> - <i>offset</i> .
System.IO.IOException	An error occurred while accessing the underlying socket. [<i>Note:</i> Any exception thrown by the <code>System.Net.Sockets.Socket.BeginSend</code> method is caught and rethrown as an <code>IOException</code> with the original exception stored in the <code>System.Exception.InnerException</code> property.]

System.ObjectDisposedException	The current instance has been disposed.

1

2 Example

3 For an outline of an asynchronous operation, see the
 4 `System.Net.Sockets.Socket.BeginAccept` method. For the complete example, see the
 5 `System.Net.Sockets.Socket` class overview.

6

1 NetworkStream.Close() Method

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

6 Summary

7 Closes the stream and, if owned by the current instance, the underlying socket.

8 Description

9 This method calls `System.Net.Sockets.NetworkStream.Dispose(true)`, which frees
10 both managed and unmanaged resources used by the current instance. When the
11 underlying socket is owned by the current instance, the
12 `System.Net.Sockets.Socket.Close` method of the socket is called, which frees both
13 managed and unmanaged resources used by the socket.

14
15 [*Note:* Ownership of a socket is specified using the
16 `System.Net.Sockets.NetworkStream` constructor.

17
18 This method overrides `System.IO.Stream.Close`.

19
20]

NetworkStream.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 current instance and optionally releases the managed resources.

Parameters

Parameter	Description
<i>disposing</i>	A <code>System.Boolean</code> . Specify <code>true</code> to release both managed and unmanaged resources; specify <code>false</code> to release only unmanaged resources.

Description

[*Note:* Ownership of a socket is specified using the `System.Net.Sockets.NetworkStream` constructor.

The `System.Net.Sockets.NetworkStream.Close` method calls this method with the *disposing* parameter set to `true`. The finalizer calls this method with the *disposing* parameter set to `false`.

]

Behaviors

This method closes the current `System.Net.Sockets.NetworkStream` instance releasing all unmanaged resources allocated by the current instance. When the underlying socket is owned by the current instance, the `System.Net.Sockets.Socket.Close` method of the socket is called, which frees the managed and unmanaged resources used by the socket. When the *disposing* parameter is `true`, this method also releases all resources held by any other managed objects allocated by the current instance.

Default

This method closes the current `System.Net.Sockets.NetworkStream` instance releasing all unmanaged resources allocated by the current instance. When the underlying socket

1 is owned by the current instance, the `System.Net.Sockets.Socket.Close` method of
2 the socket is called, which frees the managed and unmanaged resources used by the
3 socket.

5 **How and When to Override**

6 The `System.Net.Sockets.Socket.Dispose` method can be called multiple times by
7 other objects. When overriding this method, do not reference objects that have been
8 previously disposed in an earlier call.

10 **Usage**

11 Use this method to release resources allocated by the current instance.

1

2 NetworkStream.EndRead(System.IAsyncResult)

3 It) Method

```
4 [ILAsm]  
5 .method public hidebysig virtual int32 EndRead(class System.IAsyncResult  
6 asyncResult)  
  
7 [C#]  
8 public override int EndRead(IAsyncResult asyncResult)
```

9 Summary

10 Ends an asynchronous call to read data from the current instance.

11 Parameters

Parameter	Description
<i>asyncResult</i>	A System.IAsyncResult object that holds the state information for the asynchronous operation.

13 Return Value

14 A System.Int32 containing the number of bytes read from the stream.

15 Description

16 This method blocks if the asynchronous operation has not completed.

17
18 The System.Net.Sockets.NetworkStream.EndRead method completes an asynchronous
19 request that was started with a call to the
20 System.Net.Sockets.NetworkStream.BeginRead method. The object specified for the
21 *asyncResult* parameter is required to be the same object as was returned by the
22 System.Net.Sockets.NetworkStream.BeginRead method call that began the request.

23
24 If the System.Net.Sockets.NetworkStream.EndRead method is invoked via the
25 System.AsyncCallback delegate specified to the
26 System.Net.Sockets.NetworkStream.BeginRead method, the *asyncResult* parameter is
27 the System.IAsyncResult argument passed to the delegate's method.

28
29 [Note: This method overrides System.IO.Stream.EndRead.

30
31]

32 Exceptions

Exception	Condition
System.ArgumentNullException	<i>asyncResult</i> is null.
System.IO.IOException	An error occurred while accessing the underlying socket. [<i>Note</i> : This method catches all exceptions thrown by the <code>System.Net.Sockets.Socket.EndReceive</code> method.]
System.ObjectDisposedException	The current instance has been disposed.

1

2 Example

3 For an outline of an asynchronous operation, see the
4 `System.Net.Sockets.Socket.BeginAccept` method. For the complete example, see the
5 `System.Net.Sockets.Socket` class overview.

6

NetworkStream.EndWrite(System.IAsyncResult) Method

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

Summary

Ends an asynchronous call to write data to the current instance.

Parameters

Parameter	Description
<i>asyncResult</i>	A <code>System.IAsyncResult</code> object that holds the state information for the asynchronous operation.

Description

This method blocks if the asynchronous operation has not completed.

The `System.Net.Sockets.NetworkStream.EndWrite` method completes an asynchronous request that was started with a call to the `System.Net.Sockets.NetworkStream.BeginWrite` method. The object specified for the *asyncResult* parameter is required to be the same object as was returned by the `System.Net.Sockets.NetworkStream.BeginWrite` method call that began the request.

If the `System.Net.Sockets.NetworkStream.EndWrite` method is invoked via the `System.AsyncCallback` delegate specified to the `System.Net.Sockets.NetworkStream.BeginWrite` method, the *asyncResult* parameter is the `System.IAsyncResult` argument passed to the delegate's method.

[Note: This method overrides `System.IO.Stream.EndWrite`.

]

Exceptions

Exception	Condition
<code>System.ArgumentNullException</code>	<i>asyncResult</i> is null.

System.IO.IOException	An error occurred while accessing the underlying socket. [<i>Note:</i> This method catches all exceptions thrown by the <code>System.Net.Sockets.Socket.EndSend</code> method.]
System.ObjectDisposedException	The current instance has been disposed.

1

2 Example

3 For an outline of an asynchronous operation, see the
4 `System.Net.Sockets.Socket.BeginAccept` method. For the complete example, see the
5 `System.Net.Sockets.Socket` class overview.

6

1 NetworkStream.Finalize() Method

```
2 [ILAsm]  
3 .method family hidebysig virtual void Finalize()  
  
4 [C#]  
5 ~NetworkStream()
```

6 Summary

7 Frees unmanaged resources used by the current instance.

8 Description

9 [Note: Application code does not call this method; it is automatically invoked during
10 garbage collection unless finalization by the garbage collector has been disabled. For
11 more information, see `System.GC.SuppressFinalize`, and `System.Object.Finalize`.
12

13 This method calls `System.Net.Sockets.NetworkStream.Dispose(false)`, which frees
14 unmanaged resources used by the current instance. When the underlying socket is
15 owned by the current instance, it is closed and the managed and unmanaged resources
16 used by the socket are freed.
17

18 Ownership of a socket is specified using the `System.Net.Sockets.NetworkStream`
19 constructor.
20

21 This method overrides `System.Object.Finalize`.
22

23]
24

1 **NetworkStream.Flush() Method**

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

6 **Summary**

7 This method is reserved for future use.

8 **Description**

9 Calling this method does not throw an exception.

10 [*Note:* This method overrides `System.IO.Stream.Flush`.

11]
12
13
14

1 **NetworkStream.Read(System.Byte[],**

2 **System.Int32, System.Int32) Method**

```
3    [ILAsm]  
4    .method public hidebysig virtual int32 Read(class System.Byte[] buffer,  
5    int32 offset, int32 size)
```

```
6    [C#]  
7    public override int Read(byte[] buffer, int offset, int size)
```

8 **Summary**

9 Reads data from the current instance and stores it in a data buffer.

10 **Parameters**

Parameter	Description
<i>buffer</i>	A System.Byte array to store data read from the stream.
<i>offset</i>	A System.Int32 containing the zero-based position in <i>buffer</i> at which to begin storing the data.
<i>size</i>	A System.Int32 containing the number of bytes to read.

12 **Return Value**

13 A System.Int32 containing the number of bytes read from the stream.

14 **Description**

15 When no incoming data is available, this method blocks and waits for data to arrive.

16
17 If the remote socket was shut down gracefully (System.Net.Sockets.Socket.Shutdown
18 was called on the socket or the System.Net.Sockets.SocketOptionName.Linger option
19 was enabled and System.Net.Sockets.Socket.Close was called on the socket) and all
20 data was received, this method immediately returns zero.

21
22 [Note: This method overrides System.IO.Stream.Read.

23
24]

25 **Exceptions**

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

System.ArgumentNullException	<i>buffer</i> is null.
System.ArgumentOutOfRangeException	<i>offset</i> < 0. -or- <i>offset</i> > <i>buffer</i> .Length. -or- <i>size</i> < 0. -or- <i>size</i> > <i>buffer</i> .Length - <i>offset</i> .
System.IO.IOException	An error occurred while accessing the underlying socket. [<i>Note:</i> This method catches all exceptions thrown by the <code>System.Net.Sockets.Socket.Receive</code> method.]
System.ObjectDisposedException	The current instance has been disposed.

1

2

1 **NetworkStream.Seek(System.Int64,**

2 **System.IO.SeekOrigin) Method**

```
3    [ILAsm]  
4    .method public hidebysig virtual int64 Seek(int64 offset, valuetype  
5    System.IO.SeekOrigin origin)  
  
6    [C#]  
7    public override long Seek(long offset, SeekOrigin origin)
```

8 **Summary**

9 Throws a `System.NotSupportedException`.

10 **Parameters**

Parameter	Description
<i>offset</i>	This parameter is not used.
<i>origin</i>	This parameter is not used.

12 **Description**

13 [Note: The `System.IO.Stream` base class uses this method to set the current position in
14 the stream. This functionality is not supported in the
15 `System.Net.Sockets.NetworkStream` class.

16 This method overrides `System.IO.Stream.Seek`.

17]
18]
19]

20 **Exceptions**

Exception	Condition
System.NotSupportedException	Any call to this method.

1 NetworkStream.SetLength(System.Int64)

2 Method

```
3 [ILAsm]  
4 .method public hidebysig virtual void SetLength(int64 value)  
  
5 [C#]  
6 public override void SetLength(long value)
```

7 Summary

8 Throws a `System.NotSupportedException`.

9 Parameters

Parameter	Description
<i>value</i>	This parameter is not used.

11 Description

12 [Note: The `System.IO.Stream` base class uses this method to set the length of the data
13 available on the stream. This functionality is not supported in the
14 `System.Net.Sockets.NetworkStream` class.

15 This method overrides `System.IO.Stream.SetLength`.

17]
18]

19 Exceptions

Exception	Condition
System.NotSupportedException	Any call to this method.

1 **NetworkStream.Write(System.Byte[],**

2 **System.Int32, System.Int32) Method**

```
3    [ILAsm]
4    .method public hidebysig virtual void Write(class System.Byte[] buffer,
5    int32 offset, int32 size)
6    [C#]
7    public override void Write(byte[] buffer, int offset, int size)
```

8 **Summary**

9 Writes data from a specific area of a data buffer to the current instance.

10 **Parameters**

Parameter	Description
<i>buffer</i>	A <code>System.Byte</code> array containing data to write to the stream.
<i>offset</i>	A <code>System.Int32</code> containing the zero-based position in <i>buffer</i> containing the starting location of the data to write.
<i>size</i>	A <code>System.Int32</code> containing the number of bytes to write to the stream.

12 **Description**

13 When no buffer space is available within the underlying protocol, this method blocks
14 unless the socket is in non-blocking mode.

15 [Note: This method overrides `System.IO.Stream.Write`.

16]

19 **Exceptions**

Exception	Condition
System.ArgumentNullException	<i>buffer</i> is null.
System.ArgumentOutOfRangeException	<i>offset</i> < 0. -or- <i>offset</i> > <i>buffer.Length</i> .

	<p>-or-</p> <p><i>size</i> < 0.</p> <p>-or-</p> <p><i>size</i> > <i>buffer.Length</i> - <i>offset</i>.</p>
System.IO.IOException	An error occurred while accessing the underlying socket. [<i>Note:</i> This method catches all exceptions thrown by the <code>System.Net.Sockets.Socket.Send</code> method.]
System.ObjectDisposedException	The current instance has been disposed.

1

2

1 NetworkStream.CanRead Property

```
2 [ILAsm]  
3 .property bool CanRead { public hidebysig virtual specialname bool  
4 get_CanRead() }  
  
5 [C#]  
6 public override bool CanRead { get; }
```

7 Summary

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

9 Property Value

10 `true` indicates that the current stream supports reading; `false`. indicates that the
11 current stream does not support reading.

12 Description

13 This property is read-only.

14
15 The value of this property is initially set by the `System.Net.Sockets.NetworkStream`
16 constructors.

17
18 [*Note:* This property overrides `System.IO.Stream.CanRead`.
19
20]

1 NetworkStream.CanSeek Property

```
2 [ILAsm]  
3 .property bool CanSeek { public hidebysig virtual specialname bool  
4 get_CanSeek() }  
  
5 [C#]  
6 public override bool CanSeek { get; }
```

7 Summary

8 Returns the `System.Boolean` value `false` to indicate that the
9 `System.Net.Sockets.NetworkStream` class cannot access a specific location in the data
10 stream.

11 Property Value

12 `false`.

13 Description

14 This property is read-only.

15
16 [*Note:* This property overrides `System.IO.Stream.CanSeek`.
17
18]

1 NetworkStream.CanWrite Property

```
2 [ILAsm]  
3 .property bool CanWrite { public hidebysig virtual specialname bool  
4 get_CanWrite() }  
  
5 [C#]  
6 public override bool CanWrite { get; }
```

7 Summary

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

9 Property Value

10 `true` indicates that the current stream supports writing; `false` indicates that the current
11 stream does not support writing.

12 Description

13 This property is read-only.

14
15 The value of this property is set by the `System.Net.Sockets.NetworkStream`
16 constructors.

17
18 [*Note:* This property overrides `System.IO.Stream.CanWrite`.
19
20]

1 NetworkStream.DataAvailable Property

```
2    [ILAsm]
3    .property bool DataAvailable { public hidebysig virtual specialname bool
4    get_DataAvailable() }

5    [C#]
6    public virtual bool DataAvailable { get; }
```

7 Summary

8 Gets a System.Boolean value indicating whether data is available to be read from the
9 underlying socket buffer.

10 Property Value

11 true indicates that data is available to be read; false indicates that there is no data
12 available to be read.

13 Description

14 This property is read-only.

15 Behaviors

16 As described above.

18 Default

19 Accessing this property causes a call to the System.Net.Sockets.Socket.Available
20 method of the underlying System.Net.Sockets.Socket instance. If the Available
21 method returns a non-zero value, indicating data is available to be read, this property
22 returns true; otherwise, this property returns false.

24 How and When to Override

25 Override this property to determine if data is available to be read in the underlying
26 socket buffer.

28 Exceptions

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

System.ObjectDisposedException

The current instance has been disposed.

1

2

1 NetworkStream.Length Property

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

5    [C#]
6    public override long Length { get; }
```

7 Summary

8 Throws a System.NotSupportedException.

9 Description

10 [Note: The System.IO.Stream base class implements this property to return the length
11 of the data available on the stream. This functionality is not supported in the
12 System.Net.Sockets.NetworkStream class.

13 This property overrides System.IO.Stream.Length.

14]

17 Exceptions

Exception	Condition
System.NotSupportedException	Any attempt to access this property.

1 NetworkStream.Position Property

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

6    [C#]
7    public override long Position { get; set; }
```

8 Summary

9 Throws a System.NotSupportedException.

10 Description

11 [*Note:* The System.IO.Stream base class implements this property to return or set the
12 current position in the stream. This functionality is not supported in the
13 System.Net.Sockets.NetworkStream class.

14 This property overrides System.IO.Stream.Position.

15]

18 Exceptions

Exception	Condition
System.NotSupportedException	Any attempt to access this property.