Table of Contents

Add your introductions here!

Namespace Manandre.IO

Classes

AsyncStream

AsyncStreamExtensions

AsyncStream class extensions

FollowingFileStream

Provides a System.IO.Stream for following a file being written, supporting both synchronous and asynchronous read operations.

Class AsyncStream

Inheritance

Object
MarshalByRefObject
Stream
AsyncStream
FollowingFileStream

Implements

IAsyncDisposable
IDisposable

Inherited Members

Stream.Null
Stream.Close()
Stream.CopyTo(Stream)
Stream.CopyTo(Stream, Int32)
Stream.CopyToAsync(Stream)
Stream.CopyToAsync(Stream, Int32)
Stream.CopyToAsync(Stream, Int32, CancellationToken)
Stream.CopyToAsync(Stream, CancellationToken)
Stream.CreateWaitHandle()
Stream.Dispose()
Stream.FlushAsync()
Stream.ObjectInvariant()
Stream.Read(Span<Byte>)
Stream.ReadAsync(Byte[], Int32, Int32)
Stream.ReadAsync(Memory<Byte>, CancellationToken)
Stream.ReadByte()
Stream.Seek(Int64, SeekOrigin)
Stream.SetLength(Int64)
Stream.Synchronized(Stream)
Stream.Write(ReadOnlySpan<Byte>)
Stream.WriteAsync(Byte[], Int32, Int32)
Stream.WriteAsync(ReadOnlyMemory<Byte>, CancellationToken)
Stream.WriteByte(Byte)
Stream.CanRead
Stream.CanSeek
Stream.CanTimeout
Stream.CanWrite
Stream.Length
Stream.Position
Stream.ReadTimeout
Stream.WriteTimeout
MarshalByRefObject.GetLifetimeService()
MarshalByRefObject.InitializeLifetimeService()
MarshalByRefObject.MemberwiseClone(Boolean)
Object.Equals(Object)
Object.Equals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()

Object.ReferenceEquals(Object, Object)
Object.ToString()

Namespace: Manandre.IO

Assembly: FollowingFileStream.dll

Syntax

```
public abstract class AsyncStream : Stream, IAsyncDisposable, IDisposable
```

Methods

BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)

Begins an asynchronous read operation. (Consider using AsyncStream.ReadAsync(System.Byte[],System.Int32,System.Int32,System.Threading.CancellationToken) instead.)

Declaration

```
public override sealed IAsyncResult BeginRead(byte[] buffer, int offset, int count, AsyncCallback callback,
object state)
```

Parameters

TYPE	NAME	DESCRIPTION
Byte[]	buffer	The buffer to read data into.
Int32	offset	The byte offset in array at which to begin reading.
Int32	count	The maximum number of bytes to read.
AsyncCallback	callback	The method to be called when the asynchronous read operation is completed.
Object	state	A user-provided object that distinguishes this particular asynchronous read request from other requests.

Returns

TYPE	DESCRIPTION
IAsyncResult	An object that references the asynchronous read.

Overrides

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

Exceptions

TYPE	CONDITION
ArgumentNullException	buffer is null.

TYPE	CONDITION
ArgumentException	offset and count describe an invalid range in array.
NotSupportedException	FollowingFileStream.CanRead for this stream is false.
InvalidOperationException	The stream is currently in use by a previous read operation.
ArgumentOutOfRangeException	offset or count is negative.
IOException	An asynchronous read was attempted past the end of the file.

BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)

Begins an asynchronous write operation. (Consider using AsyncStream.WriteAsync(System.Byte[],System.Int32,System.Int32,System.Threading.CancellationToken) instead.)

Declaration

```
public override sealed IAsyncResult BeginWrite(byte[] buffer, int offset, int count, AsyncCallback callback,
object state)
```

Parameters

TYPE	NAME	DESCRIPTION
Byte[]	buffer	The buffer to read data from.
Int32	offset	The byte offset in array at which to begin writing.
Int32	count	The maximum number of bytes to write.
AsyncCallback	callback	The method to be called when the asynchronous write operation is completed.
Object	state	A user-provided object that distinguishes this particular asynchronous write request from other requests.

Returns

TYPE	DESCRIPTION
IAsyncResult	An object that references the asynchronous write.

Overrides

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

Exceptions

TYPE	CONDITION
ArgumentNullException	buffer is null.
ArgumentException	offset and count describe an invalid range in array.
NotSupportedException	FollowingFileStream.CanWrite for this stream is false.
InvalidOperationException	The stream is currently in use by a previous write operation.
ArgumentOutOfRangeException	offset or count is negative.
IOException	An asynchronous write was attempted past the end of the file.

Dispose(Boolean)

Releases the unmanaged resources used by the FollowingFileStream and optionally releases the managed resources.

Declaration

```
protected override sealed void Dispose(bool disposing)
```

Parameters

TYPE	NAME	DESCRIPTION
Boolean	disposing	true to release both managed and unmanaged resources; false to release only unmanaged resources.

Overrides

Stream.Dispose(Boolean)

DisposeAsync()

Asynchronously releases all resources used by the AsyncStream.

Declaration

```
public override sealed ValueTask DisposeAsync()
```

Returns

TYPE	DESCRIPTION
ValueTask	

Overrides

Stream.DisposeAsync()

DisposeAsync(Boolean)

Asynchronously releases the unmanaged resources used by the FollowingFileStream and optionally releases the managed resources.

Declaration

```
protected virtual ValueTask DisposeAsync(bool disposing)
```

Parameters

TYPE	NAME	DESCRIPTION
Boolean	disposing	true to release both managed and unmanaged resources; false to release only unmanaged resources.

Returns

TYPE	DESCRIPTION
ValueTask	

EndRead(IAsyncResult)

Waits for the pending asynchronous read operation to complete. (Consider using AsyncStream.ReadAsync(System.Byte[],System.Int32,System.Int32,System.Threading.CancellationToken) instead.)

Declaration

```
public override sealed int EndRead(IAsyncResult asyncResult)
```

Parameters

TYPE	NAME	DESCRIPTION
IAsyncResult	asyncResult	The reference to the pending asynchronous request to wait for.

Returns

TYPE	DESCRIPTION
Int32	The number of bytes read from the stream, between 0 and the number of bytes you requested. Streams only return 0 at the end of the stream, otherwise, they should block until at least 1 byte is available.

Overrides

Stream.EndRead(IAsyncResult)

Exceptions

TYPE	CONDITION
ArgumentNullException	asyncResult is null.

TYPE	CONDITION
ArgumentException	This System.IAsyncResult object was not created by calling AsyncStream.BeginRead(System.Byte[],System.Int32,System.Int32,System.AsyncCallback,System.Object) on this class.
InvalidOperationException	AsyncStream.EndRead(System.IAsyncResult) is called multiple times.
IOException	The stream is closed or an internal error has occurred.

EndWrite(IAsyncResult)

Waits for the pending asynchronous write operation to complete. (Consider using AsyncStream.WriteAsync(System.Byte[],System.Int32,System.Int32,System.Threading.CancellationToken) instead.)

Declaration

```
public override sealed void EndWrite(IAsyncResult asyncResult)
```

Parameters

TYPE	NAME	DESCRIPTION
IAsyncResult	asyncResult	The reference to the pending asynchronous request to wait for.

Overrides

Stream.EndWrite(IAsyncResult)

Exceptions

TYPE	CONDITION
ArgumentNullException	asyncResult is null.
ArgumentException	This System.IAsyncResult object was not created by calling AsyncStream.BeginWrite(System.Byte[],System.Int32,System.Int32,System.AsyncCallback,System.Object) on this class.
InvalidOperationException	AsyncStream.EndWrite(System.IAsyncResult) is called multiple times.
IOException	The stream is closed or an internal error has occurred.

Flush()

Clears all buffers for this stream and causes any buffered data to be written to the underlying device.

Declaration

```
public override sealed void Flush()
```

Overrides

Stream.Flush()

Exceptions

TYPE	CONDITION
IOException	The stream is closed or an internal error has occurred.

FlushAsync(CancellationToken)

Asynchronously clears all buffers for this stream, causes any buffered data to be written to the underlying device, and monitors cancellation requests.

Declaration

```
public abstract override Task FlushAsync(CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	The token to monitor for cancellation requests. The default value is System.Threading.CancellationToken.None.

Returns

TYPE	DESCRIPTION
Task	A task that represents the asynchronous flush operation.

Overrides

Stream.FlushAsync(CancellationToken)

Read(Byte[], Int32, Int32)

Reads a block of bytes from the stream and writes the data in a given buffer.

Declaration

```
public override sealed int Read(byte[] buffer, int offset, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
Byte[]	buffer	When this method returns, contains the specified byte array with the values between offset and (offset + count - 1) replaced by the bytes read from the current source.
Int32	offset	The byte offset in array at which the read bytes will be placed.

TYPE	NAME	DESCRIPTION
Int32	count	The maximum number of bytes to read.

Returns

TYPE	DESCRIPTION
Int32	The total number of bytes read into the buffer. This might be less than the number of bytes requested if that number of bytes are not currently available, or zero if the end of the stream is reached.

Overrides

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

Exceptions

TYPE	CONDITION
ArgumentNullException	buffer is null.
ArgumentException	offset and count describe an invalid range in array.
NotSupportedException	AsyncStream.CanRead for this stream is false.
IOException	An I/O error occurred.
ArgumentOutOfRangeException	offset or count is negative.
ObjectDisposedException	Methods were called after the stream was closed.

ReadAsync(Byte[], Int32, Int32, CancellationToken)

Asynchronously reads a sequence of bytes from the current stream, advances the position within the stream by the number of bytes read, and monitors cancellation requests.

Declaration

```
public abstract override Task<int> ReadAsync(byte[] buffer, int offset, int count, CancellationToken
cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
Byte[]	buffer	The buffer to write the data into.

TYPE	NAME	DESCRIPTION
Int32	offset	The byte offset in buffer at which to begin writing data from the stream.
Int32	count	The maximum number of bytes to read.
CancellationToken	cancellationToken	The token to monitor for cancellation requests.

Returns

TYPE	DESCRIPTION
Task<Int32>	A task that represents the asynchronous read operation. The value of the TResult parameter contains the total number of bytes read into the buffer. The result value can be less than the number of bytes requested if the number of bytes currently available is less than the requested number, or it can be 0 (zero) if the end of the stream has been reached.

Overrides

Stream.ReadAsync(Byte[], Int32, Int32, CancellationToken)

Exceptions

TYPE	CONDITION
ArgumentNullException	buffer is null.
ArgumentException	offset and count describe an invalid range in array.
NotSupportedException	FollowingFileStream.CanRead for this stream is false.
InvalidOperationException	The stream is currently in use by a previous read operation.
ArgumentOutOfRangeException	offset or count is negative.
ObjectDisposedException	Methods were called after the stream was closed.

Synchronized(AsyncStream)

Synchronized version of an async stream

Declaration

```
public static AsyncStream Synchronized(AsyncStream stream)
```

Parameters

TYPE	NAME	DESCRIPTION
AsyncStream	stream	Stream to synchronize

Returns

TYPE	DESCRIPTION
AsyncStream	

Write(Byte[], Int32, Int32)

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

Declaration

```
public override sealed void Write(byte[] buffer, int offset, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
Byte[]	buffer	An array of bytes. This method copies count bytes from buffer to the current stream.
Int32	offset	The zero-based byte offset in buffer at which to begin copying bytes to the current stream.
Int32	count	The number of bytes to be written to the current stream.

Overrides

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

Exceptions

TYPE	CONDITION
ArgumentNullException	buffer is null.
ArgumentException	offset and count describe an invalid range in array.
NotSupportedException	AsyncStream.CanWrite for this stream is false.
IOException	An I/O error occurred.
ArgumentOutOfRangeException	offset or count is negative.

TYPE	CONDITION
ObjectDisposedException	Methods were called after the stream was closed.

WriteAsync(Byte[], Int32, Int32, CancellationToken)

Asynchronously writes a sequence of bytes to the current stream, advances the current position within this stream by the number of bytes written, and monitors cancellation requests.

Declaration

```
public abstract override Task WriteAsync(byte[] buffer, int offset, int count, CancellationToken
cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
Byte[]	buffer	An array of bytes. This method copies count bytes from buffer to the current stream.
Int32	offset	The zero-based byte offset in buffer at which to begin copying bytes to the current stream.
Int32	count	The number of bytes to be written to the current stream.
CancellationToken	cancellationToken	The token to monitor for cancellation requests. The default value is System.Threading.CancellationToken.None.

Returns

TYPE	DESCRIPTION
Task	

Overrides

Stream.WriteAsync(Byte[], Int32, Int32, CancellationToken)

Exceptions

TYPE	CONDITION
ArgumentNullException	buffer is null.
ArgumentException	offset and count describe an invalid range in array.
NotSupportedException	AsyncStream.CanWrite for this stream is false.

TYPE	CONDITION
ArgumentOutOfRangeException	offset or count is negative.
ObjectDisposedException	Methods were called after the stream was closed.
InvalidOperationException	The stream is currently in use by a previous write operation.

Implements

System.IAsyncDisposable
System.IDisposable

Extension Methods

AsyncStreamExtensions.Synchronized(AsyncStream)

Class AsyncStreamExtensions

AsyncStream class extensions

Inheritance

Object
AsyncStreamExtensions

Inherited Members

Object.Equals(Object)
Object.Equals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()
Object.ReferenceEquals(Object, Object)
Object.ToString()

Namespace: Manandre.IO

Assembly: FollowingFileStream.dll

Syntax

```
public static class AsyncStreamExtensions
```

Methods

Synchronized(AsyncStream)

Synchronized version of an async stream

Declaration

```
public static AsyncStream Synchronized(this AsyncStream stream)
```

Parameters

TYPE	NAME	DESCRIPTION
AsyncStream	stream	Stream to synchronize

Returns

TYPE	DESCRIPTION
AsyncStream	

Class FollowingFileStream

Provides a System.IO.Stream for following a file being written, supporting both synchronous and asynchronous read operations.

Inheritance

Object
MarshalByRefObject
Stream
AsyncStream
FollowingFileStream

Implements

IAsyncDisposable
IDisposable

Inherited Members

AsyncStream.BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)
AsyncStream.BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)
AsyncStream.EndRead(IAsyncResult)
AsyncStream.EndWrite(IAsyncResult)
AsyncStream.Flush()
AsyncStream.Read(Byte[], Int32, Int32)
AsyncStream.Write(Byte[], Int32, Int32)
AsyncStream.DisposeAsync()
AsyncStream.Dispose(Boolean)
AsyncStream.Synchronized(AsyncStream)
Stream.Null
Stream.Close()
Stream.CopyTo(Stream)
Stream.CopyTo(Stream, Int32)
Stream.CopyToAsync(Stream)
Stream.CopyToAsync(Stream, Int32)
Stream.CopyToAsync(Stream, Int32, CancellationToken)
Stream.CopyToAsync(Stream, CancellationToken)
Stream.CreateWaitHandle()
Stream.Dispose()
Stream.FlushAsync()
Stream.ObjectInvariant()
Stream.Read(Span<Byte>)
Stream.ReadAsync(Byte[], Int32, Int32)
Stream.ReadAsync(Memory<Byte>, CancellationToken)
Stream.ReadByte()
Stream.Synchronized(Stream)
Stream.Write(ReadOnlySpan<Byte>)
Stream.WriteAsync(Byte[], Int32, Int32)
Stream.WriteAsync(ReadOnlyMemory<Byte>, CancellationToken)
Stream.WriteByte(Byte)
Stream.WriteTimeout
MarshalByRefObject.GetLifetimeService()
MarshalByRefObject.InitializeLifetimeService()
MarshalByRefObject.MemberwiseClone(Boolean)
Object.Equals(Object)
Object.Equals(Object, Object)

Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()
Object.ReferenceEquals(Object, Object)
Object.ToString()

Namespace: Manandre.IO

Assembly: FollowingFileStream.dll

Syntax

```
public class FollowingFileStream : AsyncStream, IAsyncDisposable, IDisposable
```

Constructors

FollowingFileStream(String)

Initializes a new instance of the FollowingFileStream class with the specified path.

Declaration

```
public FollowingFileStream(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
String	path	A relative or absolute path for the file that the current FollowingFileStream object will encapsulate.

Exceptions

TYPE	CONDITION
ArgumentException	path is an empty string (""), contains only white space, or contains one or more invalid characters. -or- path refers to a non-file device, such as "con:", "com1:", "lpt1:", etc. in an NTFS environment.
NotSupportedException	path refers to a non-file device, such as "con:", "com1:", "lpt1:", etc. in a non-NTFS environment.
ArgumentNullException	path is null.
SecurityException	The caller does not have the required permission.
FileNotFoundException	The file cannot be found. The file must already exist.
IOException	The stream has been closed.
DirectoryNotFoundException	The specified path is invalid, such as being on an unmapped drive.

TYPE	CONDITION
PathTooLongException	The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters.

FollowingFileStream(String, Int32, Boolean)

Initializes a new instance of the FollowingFileStream class with the specified path, buffer size, and synchronous or asynchronous state.

Declaration

```
public FollowingFileStream(string path, int bufferSize, bool useAsync)
```

Parameters

TYPE	NAME	DESCRIPTION
String	path	A relative or absolute path for the file that the current FollowingFileStream object will encapsulate.
Int32	bufferSize	A positive System.Int32 value greater than 0 indicating the buffer size. The default buffer size is 4096.
Boolean	useAsync	Specifies whether to use asynchronous I/O or synchronous I/O. However, note that the underlying operating system might not support asynchronous I/O, so when specifying true, the handle might be opened synchronously depending on the platform. When opened asynchronously, the System.IO.FileStream.BeginRead(System.Byte[],System.Int32,System.Int32,System.AsyncCallback,System.Object) and System.IO.FileStream.BeginWrite(System.Byte[],System.Int32,System.Int32,System.AsyncCallback,System.Object) methods perform better on large reads or writes, but they might be much slower for small reads or writes. If the application is designed to take advantage of asynchronous I/O, set the useAsync parameter to true. Using asynchronous I/O correctly can speed up applications by as much as a factor of 10, but using it without redesigning the application for asynchronous I/O can decrease performance by as much as a factor of 10.

Exceptions

TYPE	CONDITION
ArgumentException	path is an empty string (""), contains only white space, or contains one or more invalid characters. -or- path refers to a non-file device, such as "con:", "com1:", "lpt1:", etc. in an NTFS environment.
NotSupportedException	path refers to a non-file device, such as "con:", "com1:", "lpt1:", etc. in a non-NTFS environment.
ArgumentNullException	path is null.
ArgumentOutOfRangeException	bufferSize is negative or zero.

TYPE	CONDITION
SecurityException	The caller does not have the required permission.
FileNotFoundException	The file cannot be found. The file must already exist.
IOException	The stream has been closed.
DirectoryNotFoundException	The specified path is invalid, such as being on an unmapped drive.
PathTooLongException	The specified path, file name, or both exceed the system-defined maximum length. For example, on Windows-based platforms, paths must be less than 248 characters, and file names must be less than 260 characters.

Properties

CanRead

Gets a value indicating whether the current stream supports reading.

Declaration

```
public override bool CanRead { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	true if the stream supports reading; false if the stream is closed.

Overrides

Stream.CanRead

CanSeek

Gets a value indicating whether the current stream supports seeking.

Declaration

```
public override bool CanSeek { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	true if the stream supports seeking; false if the stream is closed.

Overrides

Stream.CanSeek

CanTimeout

Declaration

```
public override bool CanTimeout { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	

Overrides

Stream.CanTimeout

CanWrite

Gets a value indicating whether the current stream supports writing.

Declaration

```
public override bool CanWrite { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	Always false.

Overrides

Stream.CanWrite

IsAsync

Gets a value indicating whether the FollowingFileStream was opened asynchronously or synchronously.

Declaration

```
public virtual bool IsAsync { get; }
```

Property Value

TYPE	DESCRIPTION
Boolean	true if the FollowongFileStream was opened asynchronously; otherwise, false.

Length

Gets the length in bytes of the stream.

Declaration

```
public override long Length { get; }
```

Property Value

TYPE	DESCRIPTION

TYPE	DESCRIPTION
Int64	A long value representing the length of the stream in bytes.

Overrides

Stream.Length

Exceptions

TYPE	CONDITION
NotSupportedException	FollowingFileStream.CanSeek for this stream is false.
IOException	An I/O error, such as the file being closed, occurred.

Name

Gets the name of the FollowingFileStream that was passed to the constructor.

Declaration

```
public virtual string Name { get; }
```

Property Value

TYPE	DESCRIPTION
String	A string that is the name of the FollowingFileStream.

Position

Gets or sets the current position of this stream.

Declaration

```
public override long Position { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int64	The current position of this stream.

Overrides

Stream.Position

Exceptions

TYPE	CONDITION
NotSupportedException	FollowingFileStream.CanSeek for this stream is false.

TYPE	CONDITION
IOException	An I/O error, such as the file being closed, occurred.
ArgumentOutOfRangeException	Attempted to set the position to a negative value.
EndOfStreamException	Attempted seeking past the end of a stream that does not support this.

ReadTimeout

Declaration

```
public override int ReadTimeout { get; set; }
```

Property Value

TYPE	DESCRIPTION
Int32	

Overrides

Stream.ReadTimeout

Methods

DisposeAsync(Boolean)

Releases the unmanaged resources used by the FollowingFileStream and optionally releases the managed resources.

Declaration

```
protected override ValueTask DisposeAsync(bool disposing)
```

Parameters

TYPE	NAME	DESCRIPTION
Boolean	disposing	true to release both managed and unmanaged resources; false to release only unmanaged resources.

Returns

TYPE	DESCRIPTION
ValueTask	

Overrides

AsyncStream.DisposeAsync(Boolean)

FlushAsync(CancellationToken)

Clears buffers for this stream and causes any buffered data to be written to the file.

Declaration

```
public override Task FlushAsync(CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task	

Overrides

AsyncStream.FlushAsync(CancellationToken)

Exceptions

TYPE	CONDITION
NotSupportedException	Not supported

ReadAsync(Byte[], Int32, Int32, CancellationToken)

Asynchronously reads a sequence of bytes from the current stream, advances the position within the stream by the number of bytes read, and monitors cancellation requests.

Declaration

```
public override Task<int> ReadAsync(byte[] buffer, int offset, int count, CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
Byte[]	buffer	The buffer to write the data into.
Int32	offset	The byte offset in buffer at which to begin writing data from the stream.
Int32	count	The maximum number of bytes to read.
CancellationToken	cancellationToken	The token to monitor for cancellation requests.

Returns

TYPE	DESCRIPTION
Task<Int32>	A task that represents the asynchronous read operation. The value of the TResult parameter contains the total number of bytes read into the buffer. The result value can be less than the number of bytes requested if the number of bytes currently available is less than the requested number, or it can be 0 (zero) if the end of the stream has been reached.

Overrides

AsyncStream.ReadAsync(Byte[], Int32, Int32, CancellationToken)

Exceptions

TYPE	CONDITION
ArgumentNullException	buffer is null.
ArgumentException	offset and count describe an invalid range in array.
NotSupportedException	FollowingFileStream.CanRead for this stream is false.
InvalidOperationException	The stream is currently in use by a previous read operation.
ArgumentOutOfRangeException	offset or count is negative.
ObjectDisposedException	Methods were called after the stream was closed.

Seek(Int64, SeekOrigin)

Sets the current position of this stream to the given value.

Declaration

```
public override long Seek(long offset, SeekOrigin origin)
```

Parameters

TYPE	NAME	DESCRIPTION
Int64	offset	The point relative to origin from which to begin seeking.
SeekOrigin	origin	Specifies the beginning, the end, or the current position as a reference point for offset, using a value of type System.IO.SeekOrigin.

Returns

TYPE	DESCRIPTION
Int64	The new position in the stream.

Overrides

Stream.Seek(Int64, SeekOrigin)

Exceptions

TYPE	CONDITION
NotSupportedException	FollowingFileStream.CanSeek for this stream is false.
IOException	An I/O error, such as the file being closed, occurred.
ArgumentException	Seeking is attempted before the beginning of the stream.
ObjectDisposedException	Methods were called after the stream was closed.

SetLength(Int64)

Sets the length of this stream to the given value.

Declaration

```
public override void SetLength(long value)
```

Parameters

TYPE	NAME	DESCRIPTION
Int64	value	

Overrides

Stream.SetLength(Int64)

Exceptions

TYPE	CONDITION
NotSupportedException	Not supported

WriteAsync(Byte[], Int32, Int32, CancellationToken)

Asynchronously writes a block of bytes to the file stream.

Declaration

```
public override Task WriteAsync(byte[] buffer, int offset, int count, CancellationToken cancellationToken)
```

Parameters

TYPE	NAME	DESCRIPTION
Byte[]	buffer	
Int32	offset	
Int32	count	

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	

Returns

TYPE	DESCRIPTION
Task	

Overrides

AsyncStream.WriteAsync(Byte[], Int32, Int32, CancellationToken)

Exceptions

TYPE	CONDITION
NotSupportedException	Not supported

Implements

System.IAsyncDisposable
System.IDisposable

Extension Methods

AsyncStreamExtensions.Synchronized(AsyncStream)

Namespace Manandre.Threading

Classes

AsyncLock

This is the async-ready almost-equivalent of the lock keyword or the Mutex type, similar to Stephen Toub's AsyncLock.

Class AsyncLock

This is the async-ready almost-equivalent of the lock keyword or the Mutex type, similar to Stephen Toub's AsyncLock.

Inheritance

Object
AsyncLock

Implements

IDisposable

Inherited Members

Object.Equals(Object)
Object.Equals(Object, Object)
Object.GetHashCode()
Object.GetType()
Object.MemberwiseClone()
Object.ReferenceEquals(Object, Object)
Object.ToString()

Namespace: Manandre.Threading

Assembly: FollowingFileStream.dll

Syntax

```
public sealed class AsyncLock : IDisposable
```

Remarks

It's only almost equivalent because the lock keyword permits reentrancy, which is not currently possible to do with an async-ready lock. An AsyncLock is either taken or not. The lock can be asynchronously acquired by calling LockAsync, and it is released by disposing the result of that task.

Methods

Dispose()

Dispose method to release the lock.

Declaration

```
public void Dispose()
```

Lock(CancellationToken)

Synchronous method to request lock.

Declaration

```
public AsyncLock Lock(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	AsyncLock takes an optional CancellationToken, which can be used to cancel the acquiring of the lock.

Returns

TYPE	DESCRIPTION
AsyncLock	An instance of AsyncLock

LockAsync(CancellationToken)

Asynchronous method to request lock.

Declaration

```
public Task<AsyncLock> LockAsync(CancellationToken cancellationToken = default(CancellationToken))
```

Parameters

TYPE	NAME	DESCRIPTION
CancellationToken	cancellationToken	AsyncLock takes an optional CancellationToken, which can be used to cancel the acquiring of the lock.

Returns

TYPE	DESCRIPTION
Task<AsyncLock>	The task returned from LockAsync will enter the Completed state when it has acquired the AsyncLock. That same task will enter the Canceled state if the CancellationToken is signaled before the wait is satisfied; in that case, the AsyncLock is not taken by that task.

Implements

System.IDisposable