

System.IO.FileStream Class

```
[ILAsm]  
.class public FileStream extends System.IO.Stream  
  
[C#]  
public class FileStream: Stream
```

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

Exposes a `System.IO.Stream` around a file, supporting both synchronous and asynchronous read and write operations.

Inherits From: System.IO.Stream

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.FileStream` is used for reading and writing files on a file system, as well as other file-related operating system handles such as pipes, standard input, standard output. `System.IO.FileStream` buffers input and output for better performance.

The `System.IO.FileStream` class can open a file in one of two modes, either synchronously or asynchronously, with significant performance consequences for the synchronous methods (`System.IO.FileStream.Read` and `System.IO.FileStream.Write`) and the asynchronous methods (`System.IO.FileStream.BeginRead` and `System.IO.FileStream.BeginWrite`). Both sets of methods will work in either mode; however, the mode will affect the performance of these methods. `System.IO.FileStream` defaults to opening files synchronously, but provides a constructor to open files asynchronously.

When accessing files, a security check is performed when the file is created or

opened. The security check is typically not done again unless the file is closed and reopened. [Note: Checking permissions when the file is first accessed minimizes the impact of the security check on application performance (since opening a file happens once, while reading and writing can happen multiple times).]

Note that if an opened file is passed to an untrusted caller, the security system can, but is not required to prevent the caller from accessing the file.

`System.IO.FileStream` objects support random access to files using the `System.IO.FileStream.Seek` method, and the `System.IO.Stream.CanSeek` properties of `System.IO.FileStream` instances encapsulating files are set to `true`. The `System.IO.FileStream.Seek` method allows the read/write position to be moved to any position within the file. This is done with byte offset reference point parameters. The byte offset is relative to the seek reference point, which can be the beginning, the current position, or the end of the underlying file, as represented by the three values of the `System.IO.SeekOrigin` enumeration.

If a `System.IO.FileStream` encapsulates a device that does not support seeking, its `System.IO.FileStream.CanSeek` property is `false`. [Note: For additional information, see `System.IO.Stream.CanSeek`.]

[Note: The `System.IO.File` class provides methods for the creation of `System.IO.FileStream` objects based on file paths. The `System.IO.MemoryStream` class creates a stream from a byte array and functions similarly to a `System.IO.FileStream`.]

Example

The following example demonstrates the use of a `System.IO.FileStream` object.

[C#]

```
using System;
using System.IO;

class Directory {
    public static void Main(String[] args) {
        FileStream fs = new FileStream("log.txt", FileMode.OpenOrCreate,
FileAccess.Write);
        StreamWriter w = new StreamWriter(fs);
        w.BaseStream.Seek(0, SeekOrigin.End);    // Set the file pointer
to the end.

        Log ("Test1", w);
        Log ("Test2", w);

        w.Close(); // Close the writer and underlying file.
```

```

        fs = new FileStream("log.txt", FileMode.OpenOrCreate,
FileAccess.Read);

        StreamReader r = new StreamReader(fs);
        r.BaseStream.Seek(0, SeekOrigin.Begin);
        DumpLog (r);
    }

    public static void Log (String logMessage, StreamWriter w) {
        w.Write("Log Entry: ");
        w.WriteLine("{0} {1}", DateTime.Now.ToLongTimeString(),
DateTime.Now.ToLongDateString());
        w.WriteLine(":");
        w.WriteLine(":{0}", logMessage);
        w.WriteLine ( "-----");
        w.Flush();
    }

    public static void DumpLog (StreamReader r) {
        while (r.Peek() > -1) { // While not at the end of the file,
write to standard output.
            Console.WriteLine(r.ReadLine());
        }

        r.Close();
    }
}

```

Some example output is

```
Log Entry: 9:26:21 AM Friday, July 06, 2001
```

```
:
```

```
:Test1
```

```
-----
```

```
Log Entry: 9:26:21 AM Friday, July 06, 2001
```

```
:
```

```
:Test2
```

```
-----
```

FileStream(System.String, System.IO.FileMode, System.IO.FileAccess, System.IO.FileShare, System.Int32, System.Boolean) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(string path,
valuetype System.IO.FileMode mode, valuetype System.IO.FileAccess
access, valuetype System.IO.FileShare share, int32 bufferSize, bool
useAsync)

[C#]
public FileStream(string path, FileMode mode, FileAccess access,
FileShare share, int bufferSize, bool useAsync)
```

Summary

Constructs and initializes a new instance of the `System.IO.FileStream` class.

Parameters

| Parameter | Description |
|-------------------|--|
| <i>path</i> | A <code>System.String</code> containing the relative or absolute path for the file that the new <code>System.IO.FileStream</code> object will encapsulate. |
| <i>mode</i> | A <code>System.IO.FileMode</code> value that determines how to open or create the file. |
| <i>access</i> | A <code>System.IO.FileAccess</code> value that determines how the file can be accessed by the <code>System.IO.FileStream</code> object. This parameter is used to specify the initial values of the <code>System.IO.FileStream.CanRead</code> and <code>System.IO.FileStream.CanWrite</code> properties. |
| <i>share</i> | A <code>System.IO.FileShare</code> value that determines how the file will be shared by processes. |
| <i>bufferSize</i> | A <code>System.Int32</code> containing the desired buffer size in bytes. |
| <i>useAsync</i> | A <code>System.Boolean</code> value that specifies whether to use asynchronous I/O or synchronous I/O. If the underlying operating system does not support asynchronous I/O, the <code>System.IO.FileStream</code> ignores this parameter and uses synchronous I/O. |

Description

This constructor sets read/write access to the file.

[Note: *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system,

this class might be able to access a physical device.]

`System.IO.Stream.CanSeek` is true for all `System.IO.FileStream` objects that encapsulate files. If *path* indicates a device that does not support seeking, the `System.IO.FileStream.CanSeek` property on the resulting `System.IO.FileStream` is required to be false. For additional information, see `System.IO.Stream.CanSeek`.

Exceptions

| Exception | Condition |
|---|--|
| System.ArgumentNullException | <i>path</i> is null. |
| System.ArgumentException | <i>path</i> is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters. |
| System.ArgumentOutOfRangeException | <i>bufferSize</i> is less than or equal to zero. -or- <i>mode</i> , <i>access</i> , or <i>share</i> contain an invalid value. |
| System.IO.FileNotFoundException | <i>mode</i> is <code>System.IO.FileMode.Truncate</code> or <code>System.IO.FileMode.Open</code> , but the specified file cannot be found. If a different mode is specified and the file cannot be found, a new one is created. |
| System.IO.IOException | An I/O error occurred, such as specifying <code>System.IO.FileMode.CreateNew</code> and the file specified by <i>path</i> already exists. |
| System.Security.SecurityException | The caller does not have the required permission. |
| System.IO.DirectoryNotFoundException | The directory information specified by <i>path</i> does not exist. |
| System.UnauthorizedAccessException | The <i>access</i> requested is not permitted by the operating system for the specified <i>path</i> . |
| System.IO.PathTooLongException | The length of <i>path</i> or the absolute path information for <i>path</i> exceeds the system-defined maximum length. |

Permissions

| Permission | Description |
|---|--|
| System.Security.Permissions.FileIOPermission | Requires permission to read, write, and append to files. See <code>System.Security.Permissions.FileIOPermissionAccess.Read</code> , <code>System.Security.Permissions.FileIOPermissionAccess.Write</code> , and <code>System.Security.Permissions.FileIOPermissionAccess.Append</code> . |

FileStream(System.String, System.IO.FileMode, System.IO.FileAccess, System.IO.FileShare, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(string path,
valuetype System.IO.FileMode mode, valuetype System.IO.FileAccess
access, valuetype System.IO.FileShare share, int32 bufferSize)

[C#]
public FileStream(string path, FileMode mode, FileAccess access,
FileShare share, int bufferSize)
```

Summary

Constructs and initializes a new instance of the `System.IO.FileStream` class.

Parameters

| Parameter | Description |
|-------------------|---|
| <i>path</i> | A <code>System.String</code> containing the relative or absolute path for the file that the current <code>System.IO.FileStream</code> object will encapsulate. |
| <i>mode</i> | A <code>System.IO.FileMode</code> constant that determines how to open or create the file. |
| <i>access</i> | A <code>System.IO.FileAccess</code> value that determines how the file can be accessed by the <code>System.IO.FileStream</code> object. This parameter is used to specify the initial values of the <code>System.IO.FileStream.CanRead</code> and <code>System.IO.FileStream.CanWrite</code> properties. For additional information, see <code>System.IO.Stream.CanRead</code> and <code>System.IO.Stream.CanWrite</code> . |
| <i>share</i> | A <code>System.IO.FileShare</code> constant that determines how the file will be shared by processes. |
| <i>bufferSize</i> | A <code>System.Int32</code> containing the desired buffer size in bytes. |

Description

[*Note:* *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system, this class might be able to access a physical device.]

`System.IO.Stream.CanSeek` is true for all `System.IO.FileStream` objects that encapsulate files. If *path* indicates a device that does not support seeking, the `System.IO.FileStream.CanSeek` property on the resulting `System.IO.FileStream` is required to be false. For additional information, see `System.IO.Stream.CanSeek`.

Exceptions

| Exception | Condition |
|---|--|
| System.ArgumentNullException | The <i>path</i> parameter is null. |
| System.ArgumentException | <i>path</i> is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters. |
| System.ArgumentOutOfRangeException | <i>bufferSize</i> is less than or equal to zero. -or- <i>mode</i> , <i>access</i> , or <i>share</i> contain an invalid value. |
| System.IO.FileNotFoundException | <i>mode</i> is <code>System.IO.FileMode.Truncate</code> or <code>System.IO.FileMode.Open</code> , but the specified file cannot be found. If a different mode is specified and the file cannot be found, a new one is created. |
| System.IO.IOException | An I/O error occurred, such as specifying <code>System.IO.FileMode.CreateNew</code> and the file specified by <i>path</i> already exists. |
| System.Security.SecurityException | The caller does not have the required permission. |
| System.IO.DirectoryNotFoundException | The directory information specified in <i>path</i> does not exist. |
| System.UnauthorizedAccessException | The <i>access</i> requested is not permitted by the operating system for the specified <i>path</i> . |
| System.IO.PathTooLongException | The length of <i>path</i> or the absolute path information for <i>path</i> exceeds the system-defined maximum length. |

Permissions

| Permission | Description |
|------------|-------------|
|------------|-------------|

| | |
|--|---|
| System.Security.Permissions. FileIOPermission | Requires permission to read, write, and append to files. See System.Security.Permissions.FileIOPermissionAccess. Read, System.Security.Permissions.FileIOPermissionAccess. Write, and System.Security.Permissions.FileIOPermissionAccess. Append. |
|--|---|

FileStream(System.String, System.IO.FileMode, System.IO.FileAccess, System.IO.FileShare) Constructor

```
[ILAsm]  
public rtspecialname specialname instance void .ctor(string path,  
valuetype System.IO.FileMode mode, valuetype System.IO.FileAccess  
access, valuetype System.IO.FileShare share)
```

```
[C#]  
public FileStream(string path, FileMode mode, FileAccess access,  
FileShare share)
```

Summary

Constructs and initializes a new instance of the `System.IO.FileStream` class with the specified path, creation mode, access type, and sharing permission.

Parameters

| Parameter | Description |
|---------------|---|
| <i>path</i> | A <code>System.String</code> containing relative or absolute path for the file that the current <code>System.IO.FileStream</code> object will encapsulate. |
| <i>mode</i> | A <code>System.IO.FileMode</code> value that determines how to open or create the file. |
| <i>access</i> | A <code>System.IO.FileAccess</code> value that determines how the file can be accessed by the <code>System.IO.FileStream</code> object. This parameter is used to specify the initial values of the <code>System.IO.FileStream.CanRead</code> and <code>System.IO.FileStream.CanWrite</code> properties. For additional information, see <code>System.IO.Stream.CanRead</code> and <code>System.IO.Stream.CanWrite</code> . |
| <i>share</i> | A <code>System.IO.FileShare</code> value that determines how the file will be shared by processes. |

Description

This constructor sets read/write access to the file.

[*Note:* *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system, this class might be able to access a physical device.]

`System.IO.Stream.CanSeek` is true for all `System.IO.FileStream` objects that encapsulate files. If *path* indicates a device that does not support seeking, the `System.IO.FileStream.CanSeek` property on the resulting `System.IO.FileStream` is required to be false. For additional information, see `System.IO.Stream.CanSeek`.

Exceptions

| Exception | Condition |
|---|--|
| System.ArgumentNullException | <i>path</i> is null. |
| System.ArgumentException | <i>path</i> is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters. |
| System.IO.FileNotFoundException | <i>mode</i> is <code>System.IO.FileMode.Truncate</code> or <code>System.IO.FileMode.Open</code> , but the specified file cannot be found. If a different mode is specified and the file cannot be found, a new one is created. |
| System.IO.IOException | An I/O error occurred, such as specifying <code>System.IO.FileMode.CreateNew</code> and the file specified by <i>path</i> already exists. |
| System.Security.SecurityException | The caller does not have the required permission. |
| System.IO.DirectoryNotFoundException | The directory information specified by <i>path</i> does not exist. |
| System.UnauthorizedAccessException | The <i>access</i> requested is not permitted by the operating system for the specified <i>path</i> . |
| System.IO.PathTooLongException | The length of <i>path</i> or the absolute path information for <i>path</i> exceeds the system-defined maximum length. |
| System.ArgumentOutOfRangeException | <i>mode</i> , <i>access</i> , or <i>share</i> contains an invalid value. |

Permissions

| Permission | Description |
|---|--|
| System.Security.Permissions.FileIOPermission | Requires permission to read, write, and append to files. See <code>System.Security.Permissions.FileIOPermissionAccess.Read</code> , <code>System.Security.Permissions.FileIOPermissionAccess.</code> |

| | |
|--|--|
| | Write, and System.Security.Permissions.FileIOPermissionAccess. Append. |
|--|--|

FileStream(System.String, System.IO.FileMode, System.IO.FileAccess) Constructor

```
[ILAsm]  
public rtspecialname specialname instance void .ctor(string path,  
valuetype System.IO.FileMode mode, valuetype System.IO.FileAccess  
access)
```

```
[C#]  
public FileStream(string path, FileMode mode, FileAccess access)
```

Summary

Constructs and initializes a new instance of the `System.IO.FileStream` class with the specified path, creation mode, and access type.

Parameters

| Parameter | Description |
|---------------|--|
| <i>path</i> | A <code>System.String</code> containing the relative or absolute path for the file that the current <code>System.IO.FileStream</code> object will encapsulate. |
| <i>mode</i> | A <code>System.IO.FileMode</code> value that determines how to open or create the file. |
| <i>access</i> | A <code>System.IO.FileAccess</code> value that determines how the file can be accessed by the <code>System.IO.FileStream</code> object. This parameter is used to specify the initial values of the <code>System.IO.FileStream.CanRead</code> and <code>System.IO.FileStream.CanWrite</code> properties. |

Description

This constructor sets read/write access to the file. Requests to open the file for writing by the current or another thread will fail until the `System.IO.FileStream` object has been closed. Read attempts will succeed.

[*Note:* *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system, this class might be able to access a physical device.]

`System.IO.Stream.CanSeek` is true for all `System.IO.FileStream` objects that encapsulate files. If *path* indicates a device that does not support seeking, the `System.IO.FileStream.CanSeek` property on the resulting `System.IO.FileStream` is required to be false. For additional information, see `System.IO.Stream.CanSeek`.

Exceptions

| Exception | Condition |
|---|---|
| System.ArgumentNullException | <i>path</i> is null. |
| System.ArgumentException | <p><i>path</i> is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters.</p> <p>-or-</p> <p><i>access</i> specified Read and <i>mode</i> specified Create, CreateNew, Truncate or Append.</p> |
| System.IO.FileNotFoundException | <i>mode</i> is System.IO.FileMode.Truncate or System.IO.FileMode.Open, but the specified file was not found. If a different mode is specified and the file was not found, a new one is created. |
| System.IO.IOException | An I/O error occurred, such as specifying System.IO.FileMode.CreateNew when the file specified by <i>path</i> already exists. |
| System.Security.SecurityException | The caller does not have the required permission. |
| System.IO.DirectoryNotFoundException | The directory information specified by <i>path</i> does not exist. |
| System.UnauthorizedAccessException | <i>path</i> specified a read-only file and <i>access</i> is not Read, or <i>path</i> specified a directory. |
| System.IO.PathTooLongException | The length of <i>path</i> or the absolute path information for <i>path</i> exceeds the system-defined maximum length. |
| System.ArgumentOutOfRangeException | <i>mode</i> or <i>access</i> contain an invalid value. |

Permissions

| Permission | Description |
|---|--|
| System.Security.Permissions.FileIOPermission | Requires permission to read, write, and append to files. See System.Security.Permissions.FileIOPermissionAccess. |

| | |
|--|--|
| | Read, System.Security.Permissions.FileIOPermissionAccess. Write, and System.Security.Permissions.FileIOPermissionAccess. Append. |
|--|--|

FileStream(System.String, System.IO.FileMode) Constructor

```
[ILAsm]  
public rtspecialname specialname instance void .ctor(string path,  
valuetype System.IO.FileMode mode)  
  
[C#]  
public FileStream(string path, FileMode mode)
```

Summary

Constructs and initializes a new instance of the `System.IO.FileStream` class with the specified path and creation mode.

Parameters

| Parameter | Description |
|-------------|--|
| <i>path</i> | A <code>System.String</code> containing the relative or absolute path for the file that the current <code>System.IO.FileStream</code> object will encapsulate. |
| <i>mode</i> | A <code>System.IO.FileMode</code> value that determines how to open or create the file. |

Description

This constructor sets `System.IO.FileAccess.ReadWrite` access to the file, and the `System.IO.Stream.CanRead` and `System.IO.Stream.CanWrite` properties of the current instance are set to `true`.

[*Note:* *path* is not required to be a file stored on disk; it can be any part of a system that supports access via streams. For example, depending on the system, this class might be able to access a physical device.]

`System.IO.Stream.CanSeek` is `true` for all `System.IO.FileStream` objects that encapsulate files. If *path* specifies a device that does not support seeking, the `System.IO.FileStream.CanSeek` property of the resulting `System.IO.FileStream` is required to be `false`. [*Note:* For additional information, see `System.IO.Stream.CanSeek`.]

Requests to open the file for writing by the current or another thread will fail until the `System.IO.FileStream` object has been closed. Read attempts will succeed.

Exceptions

| Exception | Condition |
|---|--|
| System.ArgumentException | <i>path</i> is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters. |
| System.ArgumentNullException | <i>path</i> is null. |
| System.Security.SecurityException | The caller does not have the required permission. |
| System.IO.FileNotFoundException | <i>mode</i> is <code>System.IO.FileMode.Truncate</code> or <code>System.IO.FileMode.Open</code> , but the specified file cannot be found. If a different mode is specified and the file cannot be found, a new one is created. |
| System.IO.IOException | An I/O error occurred, such as specifying <code>System.IO.FileMode.CreateNew</code> when the file specified by <i>path</i> already exists. |
| System.IO.DirectoryNotFoundException | The directory information specified in <i>path</i> does not exist. |
| System.IO.PathTooLongException | The length of <i>path</i> or the absolute path information for <i>path</i> exceeds the system-defined maximum length. |
| System.ArgumentOutOfRangeException | <i>mode</i> contains an invalid value. |

Permissions

| Permission | Description |
|---|--|
| System.Security.Permissions.FileIOPermission | Requires permission to read, write, and append to files. See <code>System.Security.Permissions.FileIOPermissionAccess.Read</code> , <code>System.Security.Permissions.FileIOPermissionAccess.Write</code> , and <code>System.Security.Permissions.FileIOPermissionAccess.Append</code> . |

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

```
[ILAsm]  
.method public hidebysig virtual class System.IAsyncResult  
BeginRead(class System.Byte[] array, int32 offset, int32 numBytes,  
class System.AsyncCallback userCallback, object stateObject)  
  
[C#]  
public override IAsyncResult BeginRead(byte[] array, int offset, int  
numBytes, AsyncCallback userCallback, object stateObject)
```

Summary

Begins an asynchronous read.

Parameters

| Parameter | Description |
|---------------------|--|
| <i>array</i> | A System.Byte array that specifies the buffer to read data into. |
| <i>offset</i> | A System.Int32 containing the zero based byte offset in <i>array</i> at which to begin writing data read from the stream. |
| <i>numBytes</i> | A System.Int32 containing the maximum number of bytes to read. |
| <i>userCallback</i> | A System.AsyncCallback delegate that references the method to be called when the asynchronous read operation is completed. |
| <i>stateObject</i> | An application-defined object containing the status of the asynchronous read. |

Return Value

A System.IAsyncResult that references the asynchronous read.

Description

To determine the number of bytes read, call System.IO.Stream.EndRead with the returned System.IAsyncResult.

Multiple simultaneous asynchronous requests render the request completion order uncertain.

[Note: Use the System.IO.FileStream.CanRead property to determine whether

the current instance supports reading. For additional information, see `System.IO.Stream.CanRead`.

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

]

Exceptions

| Exception | Condition |
|---|---|
| System.ArgumentException | The sum of <i>offset</i> and <i>numBytes</i> is greater than the length of <i>array</i> . |
| System.ArgumentNullException | <i>array</i> is null. |
| System.ArgumentOutOfRangeException | <i>offset</i> or <i>numBytes</i> is negative. |
| System.IO.IOException | The asynchronous read operation attempted to read past the end of the file. |

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

```
[ILAsm]  
.method public hidebysig virtual class System.IAsyncResult  
BeginWrite(class System.Byte[] array, int32 offset, int32 numBytes,  
class System.AsyncCallback userCallback, object stateObject)
```

```
[C#]  
public override IAsyncResult BeginWrite(byte[] array, int offset,  
int numBytes, AsyncCallback userCallback, object stateObject)
```

Summary

Begins an asynchronous write operation.

Parameters

| Parameter | Description |
|---------------------|--|
| <i>array</i> | A System.Byte array buffer containing data to write to the current stream. |
| <i>offset</i> | A System.Int32 containing the zero-based byte offset in <i>array</i> , which marks the beginning of the data to written to the current stream. |
| <i>numBytes</i> | A System.Int32 containing the maximum number of bytes to write. |
| <i>userCallback</i> | A System.AsyncCallback delegate that references the method to be called when the asynchronous write operation is completed. |
| <i>stateObject</i> | An application-defined object containing the status of the asynchronous write. |

Return Value

A System.IAsyncResult that references the asynchronous write.

Description

Multiple simultaneous asynchronous requests render the request completion order uncertain.

[*Note:* Use the System.IO.FileStream.CanWrite property to determine whether the current instance supports writing. For additional information, see System.IO.Stream.CanWrite.

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

]

Exceptions

| Exception | Condition |
|---|---|
| System.ArgumentException | The sum of <i>offset</i> and <i>numBytes</i> is greater than the length of <i>array</i> . |
| System.ArgumentNullException | <i>array</i> is null. |
| System.ArgumentOutOfRangeException | <i>offset</i> or <i>numBytes</i> is negative. |
| System.SystemNotSupportedException | The stream does not support writing. |
| System.IO.IOException | An I/O error occurred. |

FileStream.Close() Method

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

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

Summary

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

Description

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

Any data previously written to the buffer is copied to the file before the file stream is closed, so it is not necessary to call `System.IO.FileStream.Flush` before invoking `Close`. Following a call to `Close`, any operations on the file stream might raise exceptions. Invoking this method on the same instance multiple times does not result in an exception.

Usage

The `System.IO.FileStream.Finalize` method invokes `Close` so that the file stream is closed before the garbage collector finalizes the object. However, objects writing to the `System.IO.FileStream`, such as a `System.IO.StreamWriter`, might not have flushed the data from their internal buffers to the `System.IO.FileStream` when the call to `Finalize` closes the stream. To prevent data loss, always call `Close` on the highest-level object.

[*Note:* This method overrides `System.IO.Stream.Close`.]

FileStream.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.FileStream` and optionally releases the managed resources.

Parameters

| Parameter | Description |
|------------------|---|
| <i>disposing</i> | Specify <code>true</code> to release both managed and unmanaged resources, or specify <code>false</code> 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.FileStream` references.

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

]

Exceptions

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

FileStream.EndRead(System.IAsyncResult) Method

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

Summary

Ends a pending asynchronous read request, and blocks until the read request has completed.

Parameters

| Parameter | Description |
|--------------------|---|
| <i>asyncResult</i> | The <code>System.IAsyncResult</code> object for the pending asynchronous request. |

Return Value

A `System.Int32` containing the number of bytes read from the stream. Returns 0 only if the end of the file has been reached, otherwise, this method blocks until at least one byte is available.

Description

`EndRead` will block until the I/O operation has completed.

[*Note:* This method overrides `System.IO.Stream.EndRead`.]

Exceptions

| Exception | Condition |
|---|--|
| <code>System.ArgumentNullException</code> | <i>asyncResult</i> is null. |
| <code>System.ArgumentException</code> | <i>asyncResult</i> was not returned by a call to <code>System.IO.FileStream.BeginRead</code> . |
| <code>System.InvalidOperationException</code> | <code>System.IO.FileStream.EndRead</code> was called |

| | |
|--|--|
| | multiple times with <i>asyncResult</i> . |
|--|--|

FileStream.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 write, blocking until the I/O operation has completed.

Parameters

| Parameter | Description |
|--------------------|--|
| <i>asyncResult</i> | The System.IAsyncResult object for the pending asynchronous request. |

Description

System.IO.FileStream.EndWrite will block until the I/O operation has completed.

[*Note:* This method overrides System.IO.Stream.EndWrite.]

Exceptions

| Exception | Condition |
|----------------------------------|---|
| System.ArgumentNullException | <i>asyncResult</i> is null. |
| System.ArgumentException | <i>asyncResult</i> was not returned by a call to System.IO.FileStream.BeginWrite. |
| System.InvalidOperationException | System.IO.FileStream.EndWrite was called multiple times with <i>asyncResult</i> . |

FileStream.Finalize() Method

```
[ILAsm]  
.method family hidebysig virtual void Finalize()  
  
[C#]  
~FileStream()
```

Summary

Releases the resources held by the current instance.

Description

`System.IO.FileStream.Finalize` closes the `System.IO.FileStream`.

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

This method overrides `System.Object.Finalize`.

]

FileStream.Flush() Method

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

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

Summary

Updates the underlying file with the current state of the buffer and subsequently clears the buffer.

Description

A `System.IO.FileStream` buffer can be used either for reading or writing. If data was copied to the buffer for writing, it is written to the file and the buffer is cleared.

If data was copied to the buffer for reading, and the `System.IO.Stream.CanSeek` property is `true`, the current position within the file is decremented by the number of unread bytes in the buffer. The buffer is then cleared.

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

Exceptions

| Exception | Condition |
|---------------------------------------|---|
| System.IO.IOException | An I/O error occurred. |
| System.ObjectDisposedException | The current instance has already been closed. |

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

```
[ILAsm]  
.method public hidebysig virtual int32 Read(class System.Byte[]  
array, int32 offset, int32 count)  
  
[C#]  
public override int Read(byte[] array, int offset, int count)
```

Summary

Reads a block of bytes from the stream and returns the data in the specified buffer.

Parameters

| Parameter | Description |
|---------------|---|
| <i>array</i> | A <code>System.Byte</code> array. When this method returns, the bytes between <i>offset</i> and (<i>offset</i> + <i>count</i> - 1) in <i>array</i> are replaced by the bytes read from the current stream. |
| <i>offset</i> | A <code>System.Int32</code> containing the byte offset in <i>array</i> at which to begin writing data read from the current stream. |
| <i>count</i> | A <code>System.Int32</code> containing maximum number of bytes to read. |

Return Value

A `System.Int32` containing the total number of bytes read into the buffer, or zero if the end of the stream is reached.

Description

The `System.IO.FileStream.Read` method returns zero only after reaching the end of the stream. Otherwise, `System.IO.FileStream.Read` always reads at least one byte from the stream before returning. If no data is available from the stream, this method blocks until at least one byte of data can be returned.

If the read operation is successful, the current position of the stream is advanced by the number of bytes read. If an exception occurs, the current position of the stream is unchanged.

[*Note:* Use the `System.IO.FileStream.CanRead` property to determine whether the current instance supports reading. For additional information, see `System.IO.Stream.CanRead`.]

[*Note:* This method overrides `System.IO.Stream.Read`.]

Exceptions

| Exception | Condition |
|---|---|
| System.ArgumentNullException | <i>array</i> is null. |
| System.ArgumentOutOfRangeException | <i>offset</i> or <i>count</i> is negative. |
| System.NotSupportedException | The current stream does not support reading. |
| System.IO.IOException | An I/O error occurred. |
| System.ArgumentException | <i>offset</i> + <i>count</i> is greater than the length of <i>array</i> . |
| System.ObjectDisposedException | The current stream is closed. |

FileStream.ReadByte() Method

```
[ILAsm]  
.method public hidebysig virtual int32 ReadByte()  
  
[C#]  
public override int ReadByte()
```

Summary

Reads a byte from the file and advances the read position one byte.

Return Value

The byte cast to a `System.Int32`, or -1 if the end of the stream has been reached.

Description

[*Note:* Use the `System.IO.FileStream.CanRead` property to determine whether the current instance supports reading. For additional information, see `System.IO.Stream.CanRead`.

This method overrides `System.IO.Stream.ReadByte`.

]

Exceptions

| Exception | Condition |
|---------------------------------------|--|
| System.ObjectDisposedException | The current stream is closed. |
| System.NotSupportedException | The current stream does not support reading. |

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

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

```
[C#]  
public override 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 |
|---------------|---|
| <i>offset</i> | A <code>System.Int64</code> containing the position relative to <i>origin</i> from which to begin seeking. |
| <i>origin</i> | A <code>System.IO.SeekOrigin</code> value specifying the beginning, the end, or the current position as a reference point for <i>offset</i> . |

Return Value

A `System.Int64` containing the new position in the stream.

Description

[*Note:* Use the `System.IO.FileStream.CanSeek` property to determine whether the current instance supports seeking. For additional information, see `System.IO.Stream.CanSeek`.]

Usage

The position can be set beyond the end of the stream.

[*Note:* This method overrides `System.IO.Stream.Seek`.]

Exceptions

| Exception | Condition |
|---------------------------------------|---|
| System.IO.IOException | An I/O error occurred. |
| System.NotSupportedException | The stream does not support seeking. |
| System.ArgumentException | Attempted seeking before the beginning of the stream or to more than one byte past the end of the stream. |
| System.ObjectDisposedException | The current stream is closed. |

FileStream.SetLength(System.Int64) Method

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

Summary

Sets the length of the current stream to the specified value.

Parameters

| Parameter | Description |
|--------------|--|
| <i>value</i> | A <code>System.Int64</code> that specifies the new length of the stream. |

Description

If *value* is less than the current length of the stream, the stream is truncated. If *value* is greater than the current length of the stream, the stream is expanded, and the contents of the stream between the old and the new length are undefined. A stream is required to support both writing and seeking to implement `System.IO.FileStream.SetLength`.

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

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

]

Exceptions

| Exception | Condition |
|---|--|
| <code>System.IO.IOException</code> | An I/O error occurred. |
| <code>System.NotSupportedException</code> | The current stream does not support writing and seeking. |
| <code>System.ArgumentOutOfRangeException</code> | <i>value</i> is less than zero. |

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

```
[ILAsm]  
.method public hidebysig virtual void Write(class System.Byte[]  
array, int32 offset, int32 count)  
  
[C#]  
public override void Write(byte[] array, int offset, int count)
```

Summary

Writes a block of bytes from a specified byte array to the current stream.

Parameters

| Parameter | Description |
|---------------|--|
| <i>array</i> | The <code>System.Byte</code> array to read. |
| <i>offset</i> | A <code>System.Int32</code> that specifies the byte offset in <i>array</i> at which to begin reading. |
| <i>count</i> | A <code>System.Int32</code> that specifies the maximum number of bytes to write to the current stream. |

Description

If the write operation is successful, the current position of the stream is advanced by the number of bytes written. If an exception occurs, the current position of the stream is unchanged.

[*Note:* Use the `System.IO.FileStream.CanWrite` property to determine whether the current instance supports writing. For additional information, see `System.IO.Stream.CanWrite`.

This method overrides `System.IO.Stream.Write`.

]

Exceptions

| Exception | Condition |
|---|---|
| System.ArgumentNullException | <i>array</i> is null. |
| System.ArgumentException | <i>offset</i> + <i>count</i> is greater than the length of <i>array</i> . |
| System.ArgumentOutOfRangeException | <i>offset</i> or <i>count</i> is negative. |

| | |
|---------------------------------------|--|
| System.ObjectDisposedException | An I/O error occurred. |
| System.NotSupportedException | The current stream does not support writing. |

FileStream.WriteByte(System.Byte) Method

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

Summary

Writes a byte to the current position in the file stream.

Parameters

| Parameter | Description |
|--------------|--|
| <i>value</i> | A <code>System.Byte</code> to write to the stream. |

Description

Usage

Use `System.IO.FileStream.WriteByte` method to write a byte to a `System.IO.FileStream` efficiently.

[*Note:* Use the `System.IO.FileStream.CanWrite` property to determine whether the current instance supports writing. For additional information, see `System.IO.Stream.CanWrite`.

This method overrides `System.IO.Stream.WriteByte`.

]

Exceptions

| Exception | Condition |
|---------------------------------------|--|
| System.ObjectDisposedException | The current stream is closed. |
| System.NotSupportedException | The current stream does not support writing. |

FileStream.CanRead Property

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

Summary

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

Property Value

`true` if the stream supports reading; `false` if the stream is closed or was opened with write-only access.

Description

This property is read-only.

[*Note:* This property overrides `System.IO.Stream.CanRead`.

If a class derived from `System.IO.Stream` does not support reading, the `Read` method throws a `System.NotSupportedException`.

]

FileStream.CanSeek Property

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

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

Summary

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

Property Value

true if the stream supports seeking; false if the stream is closed or if the `System.IO.FileStream` was constructed from an operating-system handle such as a pipe or output to the console.

Description

[*Note:* If a class derived from `System.IO.Stream` does not support seeking, a call to `System.IO.FileStream.Length` (both get and set), `System.IO.FileStream.Position`, or `System.IO.FileStream.Seek` throws a `System.NotSupportedException`.

This property overrides `System.IO.Stream.CanSeek`.

]

FileStream.CanWrite Property

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

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

Summary

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

Property Value

true if the stream supports writing; false if the stream is closed or was opened with read-only access.

Description

If a class derived from `System.IO.Stream` does not support writing, a call to `System.IO.FileStream.Write` or `System.IO.FileStream.BeginWrite` will throw a `System.NotSupportedException`.

[*Note:* This property overrides `System.IO.Stream.CanWrite`.]

FileStream.IsAsync Property

```
[ILAsm]  
.property bool IsAsync { public hidebysig virtual specialname bool  
get_IsAsync() }
```

```
[C#]  
public virtual bool IsAsync { get; }
```

Summary

Gets a `System.Boolean` value indicating whether the current instance was opened asynchronously or synchronously.

Property Value

true if the current `System.IO.FileStream` was opened asynchronously; otherwise, false.

Behaviors

This property is read-only.

FileStream.Length Property

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

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

Summary

Gets the length in bytes of the stream.

Property Value

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

Description

This property is read-only.

Exceptions

| Exception | Condition |
|-------------------------------------|---|
| System.NotSupportedException | <code>System.IO.FileStream.CanSeek</code> for this stream is false. |
| System.IO.IOException | An I/O error occurred, such as the file being closed. |

FileStream.Position Property

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

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

Summary

Gets or sets the current position of this stream.

Property Value

A `System.Int64` containing the current position of this stream.

Description

The position can be set beyond the end of the stream.

Exceptions

| Exception | Condition |
|---|--|
| System.NotSupportedException | The current stream does not support seeking. |
| System.IO.IOException | An I/O error occurred. |
| System.IO.EndOfStreamException | Attempted seeking past the end of a stream that does not support this. |
| System.ArgumentOutOfRangeException | The value specified for a set operation is negative. |