**File and Stream I/O**

* 10minutes to read

File and stream I/O (input/output) refers to the transfer of data either to or from a storage medium. In .NET, the System.IO namespaces contain types that enable reading and writing, both synchronously and asynchronously, on data streams and files. These namespaces also contain types that perform compression and decompression on files, and types that enable communication through pipes and serial ports.

A file is an ordered and named collection of bytes that has persistent storage. When you work with files, you work with directory paths, disk storage, and file and directory names. In contrast, a stream is a sequence of bytes that you can use to read from and write to a backing store, which can be one of several storage mediums (for example, disks or memory). Just as there are several backing stores other than disks, there are several kinds of streams other than file streams, such as network, memory, and pipe streams.

**Files and directories**

You can use the types in the [System.IO](https://docs.microsoft.com/en-us/dotnet/api/system.io) namespace to interact with files and directories. For example, you can get and set properties for files and directories, and retrieve collections of files and directories based on search criteria.

For path naming conventions and the ways to express a file path for Windows systems, including with the DOS device syntax supported in .NET Core 1.1 and later and .NET Framework 4.6.2 and later, see [File path formats on Windows systems](https://docs.microsoft.com/en-us/dotnet/standard/io/file-path-formats).

Here are some commonly used file and directory classes:

* [File](https://docs.microsoft.com/en-us/dotnet/api/system.io.file) - provides static methods for creating, copying, deleting, moving, and opening files, and helps create a [FileStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream) object.
* [FileInfo](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo) - provides instance methods for creating, copying, deleting, moving, and opening files, and helps create a [FileStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream) object.
* [Directory](https://docs.microsoft.com/en-us/dotnet/api/system.io.directory) - provides static methods for creating, moving, and enumerating through directories and subdirectories.
* [DirectoryInfo](https://docs.microsoft.com/en-us/dotnet/api/system.io.directoryinfo) - provides instance methods for creating, moving, and enumerating through directories and subdirectories.
* [Path](https://docs.microsoft.com/en-us/dotnet/api/system.io.path) - provides methods and properties for processing directory strings in a cross-platform manner.

You should always provide robust exception handling when calling filesystem methods. For more information, see [Handling I/O errors](https://docs.microsoft.com/en-us/dotnet/standard/io/handling-io-errors).

In addition to using these classes, Visual Basic users can use the methods and properties provided by the [Microsoft.VisualBasic.FileIO.FileSystem](https://docs.microsoft.com/en-us/dotnet/api/microsoft.visualbasic.fileio.filesystem) class for file I/O.

See [How to: Copy Directories](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-copy-directories), [How to: Create a Directory Listing](https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/5cf8zcfh(v=vs.100)), and [How to: Enumerate Directories and Files](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-enumerate-directories-and-files).

**Streams**

The abstract base class [Stream](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream) supports reading and writing bytes. All classes that represent streams inherit from the [Stream](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream) class. The [Stream](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream) class and its derived classes provide a common view of data sources and repositories, and isolate the programmer from the specific details of the operating system and underlying devices.

Streams involve three fundamental operations:

* Reading - transferring data from a stream into a data structure, such as an array of bytes.
* Writing - transferring data to a stream from a data source.
* Seeking - querying and modifying the current position within a stream.

Depending on the underlying data source or repository, a stream might support only some of these capabilities. For example, the [PipeStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.pipes.pipestream) class does not support seeking. The [CanRead](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.canread), [CanWrite](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.canwrite), and [CanSeek](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.canseek) properties of a stream specify the operations that the stream supports.

Here are some commonly used stream classes:

* [FileStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream) – for reading and writing to a file.
* [IsolatedStorageFileStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.isolatedstorage.isolatedstoragefilestream) – for reading and writing to a file in isolated storage.
* [MemoryStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.memorystream) – for reading and writing to memory as the backing store.
* [BufferedStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.bufferedstream) – for improving performance of read and write operations.
* [NetworkStream](https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.networkstream) – for reading and writing over network sockets.
* [PipeStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.pipes.pipestream) – for reading and writing over anonymous and named pipes.
* [CryptoStream](https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.cryptostream) – for linking data streams to cryptographic transformations.

For an example of working with streams asynchronously, see [Asynchronous File I/O](https://docs.microsoft.com/en-us/dotnet/standard/io/asynchronous-file-i-o).

**Readers and writers**

The [System.IO](https://docs.microsoft.com/en-us/dotnet/api/system.io) namespace also provides types for reading encoded characters from streams and writing them to streams. Typically, streams are designed for byte input and output. The reader and writer types handle the conversion of the encoded characters to and from bytes so the stream can complete the operation. Each reader and writer class is associated with a stream, which can be retrieved through the class's BaseStream property.

Here are some commonly used reader and writer classes:

* [BinaryReader](https://docs.microsoft.com/en-us/dotnet/api/system.io.binaryreader) and [BinaryWriter](https://docs.microsoft.com/en-us/dotnet/api/system.io.binarywriter) – for reading and writing primitive data types as binary values.
* [StreamReader](https://docs.microsoft.com/en-us/dotnet/api/system.io.streamreader) and [StreamWriter](https://docs.microsoft.com/en-us/dotnet/api/system.io.streamwriter) – for reading and writing characters by using an encoding value to convert the characters to and from bytes.
* [StringReader](https://docs.microsoft.com/en-us/dotnet/api/system.io.stringreader) and [StringWriter](https://docs.microsoft.com/en-us/dotnet/api/system.io.stringwriter) – for reading and writing characters to and from strings.
* [TextReader](https://docs.microsoft.com/en-us/dotnet/api/system.io.textreader) and [TextWriter](https://docs.microsoft.com/en-us/dotnet/api/system.io.textwriter) – serve as the abstract base classes for other readers and writers that read and write characters and strings, but not binary data.

See [How to: Read Text from a File](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-read-text-from-a-file), [How to: Write Text to a File](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-write-text-to-a-file), [How to: Read Characters from a String](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-read-characters-from-a-string), and [How to: Write Characters to a String](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-write-characters-to-a-string).

**Asynchronous I/O operations**

Reading or writing a large amount of data can be resource-intensive. You should perform these tasks asynchronously if your application needs to remain responsive to the user. With synchronous I/O operations, the UI thread is blocked until the resource-intensive operation has completed. Use asynchronous I/O operations when developing Windows 8.x Store apps to prevent creating the impression that your app has stopped working.

The asynchronous members contain Async in their names, such as the [CopyToAsync](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.copytoasync), [FlushAsync](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.flushasync), [ReadAsync](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.readasync), and [WriteAsync](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.writeasync) methods. You use these methods with the async and await keywords.

For more information, see [Asynchronous File I/O](https://docs.microsoft.com/en-us/dotnet/standard/io/asynchronous-file-i-o).

**Compression**

Compression refers to the process of reducing the size of a file for storage. Decompression is the process of extracting the contents of a compressed file so they are in a usable format. The [System.IO.Compression](https://docs.microsoft.com/en-us/dotnet/api/system.io.compression) namespace contains types for compressing and decompressing files and streams.

The following classes are frequently used when compressing and decompressing files and streams:

* [ZipArchive](https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.ziparchive) – for creating and retrieving entries in the zip archive.
* [ZipArchiveEntry](https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.ziparchiveentry) – for representing a compressed file.
* [ZipFile](https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.zipfile) – for creating, extracting, and opening a compressed package.
* [ZipFileExtensions](https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.zipfileextensions) – for creating and extracting entries in a compressed package.
* [DeflateStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.deflatestream) – for compressing and decompressing streams using the Deflate algorithm.
* [GZipStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.gzipstream) – for compressing and decompressing streams in gzip data format.

See [How to: Compress and Extract Files](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-compress-and-extract-files).

**Isolated storage**

Isolated storage is a data storage mechanism that provides isolation and safety by defining standardized ways of associating code with saved data. The storage provides a virtual file system that is isolated by user, assembly, and (optionally) domain. Isolated storage is particularly useful when your application does not have permission to access user files. You can save settings or files for your application in a manner that is controlled by the computer's security policy.

Isolated storage is not available for Windows 8.x Store apps; instead, use application data classes in the [Windows.Storage](https://docs.microsoft.com/en-us/uwp/api/windows.storage) namespace. For more information, see [Application data](https://docs.microsoft.com/en-us/previous-versions/windows/apps/hh464917(v=win.10)).

The following classes are frequently used when implementing isolated storage:

* [IsolatedStorage](https://docs.microsoft.com/en-us/dotnet/api/system.io.isolatedstorage.isolatedstorage) – provides the base class for isolated storage implementations.
* [IsolatedStorageFile](https://docs.microsoft.com/en-us/dotnet/api/system.io.isolatedstorage.isolatedstoragefile) – provides an isolated storage area that contains files and directories.
* [IsolatedStorageFileStream](https://docs.microsoft.com/en-us/dotnet/api/system.io.isolatedstorage.isolatedstoragefilestream) - exposes a file within isolated storage.

See [Isolated Storage](https://docs.microsoft.com/en-us/dotnet/standard/io/isolated-storage).

**I/O operations in Windows Store apps**

.NET for Windows 8.x Store apps contains many of the types for reading from and writing to streams; however, this set does not include all the .NET I/O types.

Some important differences to note when using I/O operations in Windows 8.x Store apps:

* Types specifically related to file operations, such as [File](https://docs.microsoft.com/en-us/dotnet/api/system.io.file), [FileInfo](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo), [Directory](https://docs.microsoft.com/en-us/dotnet/api/system.io.directory) and [DirectoryInfo](https://docs.microsoft.com/en-us/dotnet/api/system.io.directoryinfo), are not included in the .NET for Windows 8.x Store apps. Instead, use the types in the [Windows.Storage](https://docs.microsoft.com/en-us/uwp/api/windows.storage) namespace of the Windows Runtime, such as [StorageFile](https://docs.microsoft.com/en-us/uwp/api/windows.storage.storagefile) and [StorageFolder](https://docs.microsoft.com/en-us/uwp/api/windows.storage.storagefolder).
* Isolated storage is not available; instead, use [application data](https://docs.microsoft.com/en-us/previous-versions/windows/apps/hh464917(v=win.10)).
* Use asynchronous methods, such as [ReadAsync](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.readasync) and [WriteAsync](https://docs.microsoft.com/en-us/dotnet/api/system.io.stream.writeasync), to prevent blocking the UI thread.
* The path-based compression types [ZipFile](https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.zipfile) and [ZipFileExtensions](https://docs.microsoft.com/en-us/dotnet/api/system.io.compression.zipfileextensions) are not available. Instead, use the types in the [Windows.Storage.Compression](https://docs.microsoft.com/en-us/uwp/api/windows.storage.compression) namespace.

You can convert between .NET Framework streams and Windows Runtime streams, if necessary. For more information, see [How to: Convert Between .NET Framework Streams and Windows Runtime Streams](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-convert-between-dotnet-streams-and-winrt-streams) or [WindowsRuntimeStreamExtensions](https://docs.microsoft.com/en-us/dotnet/api/system.io.windowsruntimestreamextensions).

For more information about I/O operations in a Windows 8.x Store app, see [Quickstart: Reading and writing files](https://docs.microsoft.com/en-us/previous-versions/windows/apps/hh758325(v=win.10)).

**I/O and security**

When you use the classes in the [System.IO](https://docs.microsoft.com/en-us/dotnet/api/system.io) namespace, you must follow operating system security requirements such as access control lists (ACLs) to control access to files and directories. This requirement is in addition to any [FileIOPermission](https://docs.microsoft.com/en-us/dotnet/api/system.security.permissions.fileiopermission) requirements. You can manage ACLs programmatically. For more information, see [How to: Add or Remove Access Control List Entries](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-add-or-remove-access-control-list-entries).

Default security policies prevent Internet or intranet applications from accessing files on the user's computer. Therefore, do not use the I/O classes that require a path to a physical file when writing code that will be downloaded over the internet or intranet. Instead, use [isolated storage](https://docs.microsoft.com/en-us/dotnet/standard/io/isolated-storage) for .NET applications.

A security check is performed only when the stream is constructed. Therefore, do not open a stream and then pass it to less-trusted code or application domains.

**Related topics**

* [Common I/O Tasks](https://docs.microsoft.com/en-us/dotnet/standard/io/common-i-o-tasks)  
  Provides a list of I/O tasks associated with files, directories, and streams, and links to relevant content and examples for each task.
* [Asynchronous File I/O](https://docs.microsoft.com/en-us/dotnet/standard/io/asynchronous-file-i-o)  
  Describes the performance advantages and basic operation of asynchronous I/O.
* [Isolated Storage](https://docs.microsoft.com/en-us/dotnet/standard/io/isolated-storage)  
  Describes a data storage mechanism that provides isolation and safety by defining standardized ways of associating code with saved data.
* [Pipes](https://docs.microsoft.com/en-us/dotnet/standard/io/pipe-operations)  
  Describes anonymous and named pipe operations in .NET.
* [Memory-Mapped Files](https://docs.microsoft.com/en-us/dotnet/standard/io/memory-mapped-files)  
  Describes memory-mapped files, which contain the contents of files on disk in virtual memory. You can use memory-mapped files to edit very large files and to create shared memory for interprocess communication.

**Common I/O Tasks**

The [System.IO](https://docs.microsoft.com/en-us/dotnet/api/system.io) namespace provides several classes that allow for various actions, such as reading and writing, to be performed on files, directories, and streams. For more information, see [File and Stream I/O](https://docs.microsoft.com/en-us/dotnet/standard/io/).

**Common File Tasks**

| **COMMON FILE TASKS** | |
| --- | --- |
| **To do this...** | **See the example in this topic...** |
| Create a text file | [File.CreateText](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.createtext) method  [FileInfo.CreateText](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo.createtext) method  [File.Create](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.create) method  [FileInfo.Create](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo.create) method |
| Write to a text file | [How to: Write Text to a File](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-write-text-to-a-file)  [How to: Write a Text File (C++/CLI)](https://docs.microsoft.com/en-us/cpp/dotnet/how-to-write-a-text-file-cpp-cli) |
| Read from a text file | [How to: Read Text from a File](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-read-text-from-a-file) |
| Append text to a file | [How to: Open and Append to a Log File](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-open-and-append-to-a-log-file)  [File.AppendText](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.appendtext) method  [FileInfo.AppendText](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo.appendtext) method |
| Rename or move a file | [File.Move](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.move) method  [FileInfo.MoveTo](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo.moveto) method |
| Delete a file | [File.Delete](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.delete) method  [FileInfo.Delete](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo.delete) method |
| Copy a file | [File.Copy](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.copy) method  [FileInfo.CopyTo](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo.copyto) method |
| Get the size of a file | [FileInfo.Length](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo.length) property |
| Get the attributes of a file | [File.GetAttributes](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.getattributes) method |
| Set the attributes of a file | [File.SetAttributes](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.setattributes) method |
| Determine whether a file exists | [File.Exists](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.exists) method |
| Read from a binary file | [How to: Read and Write to a Newly Created Data File](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-read-and-write-to-a-newly-created-data-file) |
| Write to a binary file | [How to: Read and Write to a Newly Created Data File](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-read-and-write-to-a-newly-created-data-file) |
| Retrieve a file name extension | [Path.GetExtension](https://docs.microsoft.com/en-us/dotnet/api/system.io.path.getextension) method |
| Retrieve the fully qualified path of a file | [Path.GetFullPath](https://docs.microsoft.com/en-us/dotnet/api/system.io.path.getfullpath) method |
| Retrieve the file name and extension from a path | [Path.GetFileName](https://docs.microsoft.com/en-us/dotnet/api/system.io.path.getfilename) method |
| Change the extension of a file | [Path.ChangeExtension](https://docs.microsoft.com/en-us/dotnet/api/system.io.path.changeextension) method |

**Common Directory Tasks**

| **COMMON DIRECTORY TASKS** | |
| --- | --- |
| **To do this...** | **See the example in this topic...** |
| Access a file in a special folder such as My Documents | [How to: Write Text to a File](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-write-text-to-a-file) |
| Create a directory | [Directory.CreateDirectory](https://docs.microsoft.com/en-us/dotnet/api/system.io.directory.createdirectory) method  [FileInfo.Directory](https://docs.microsoft.com/en-us/dotnet/api/system.io.fileinfo.directory) property |
| Create a subdirectory | [DirectoryInfo.CreateSubdirectory](https://docs.microsoft.com/en-us/dotnet/api/system.io.directoryinfo.createsubdirectory) method |
| Rename or move a directory | [Directory.Move](https://docs.microsoft.com/en-us/dotnet/api/system.io.directory.move) method  [DirectoryInfo.MoveTo](https://docs.microsoft.com/en-us/dotnet/api/system.io.directoryinfo.moveto) method |
| Copy a directory | [How to: Copy Directories](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-copy-directories) |
| Delete a directory | [Directory.Delete](https://docs.microsoft.com/en-us/dotnet/api/system.io.directory.delete) method  [DirectoryInfo.Delete](https://docs.microsoft.com/en-us/dotnet/api/system.io.directoryinfo.delete) method |
| See the files and subdirectories in a directory | [How to: Enumerate Directories and Files](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-enumerate-directories-and-files) |
| Find the size of a directory | [System.IO.Directory](https://docs.microsoft.com/en-us/dotnet/api/system.io.directory) class |
| Determine whether a directory exists | [Directory.Exists](https://docs.microsoft.com/en-us/dotnet/api/system.io.directory.exists) method |

**Recommended content**

**[FileStream Class (System.IO)](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream)**

Provides a Stream for a file, supporting both synchronous and asynchronous read and write operations.

**[FileStream.Write Method (System.IO)](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream.write)**

Writes a sequence of bytes from a read-only span to the current file stream and advances the current position within this file stream by the number of bytes written.

**[How to: Read and write to a newly created data file](https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-read-and-write-to-a-newly-created-data-file)**

Learn how to read and write to a newly created data file in .NET using the System.IO.BinaryReader and System.IO.BinaryWriter classes.

**[FileStream.Read Method (System.IO)](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream.read)**

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

**[File.OpenWrite(String) Method (System.IO)](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.openwrite)**

Opens an existing file or creates a new file for writing.

**[File.Create Method (System.IO)](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.create)**

Creates or overwrites a file in the specified path.

**[File.OpenRead(String) Method (System.IO)](https://docs.microsoft.com/en-us/dotnet/api/system.io.file.openread)**

Opens an existing file for reading.

**[FileStream.ReadByte Method (System.IO)](https://docs.microsoft.com/en-us/dotnet/api/system.io.filestream.readbyte)**

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