SDD

Wayne.Lib.IO

Table of Contents

1 Document information

File: SDD_Wayne.Lib.IO.doc

1.1 Revision history

Revision	Author	Date	Change description
1.0	Roger Månsson	2006-12-19	Created

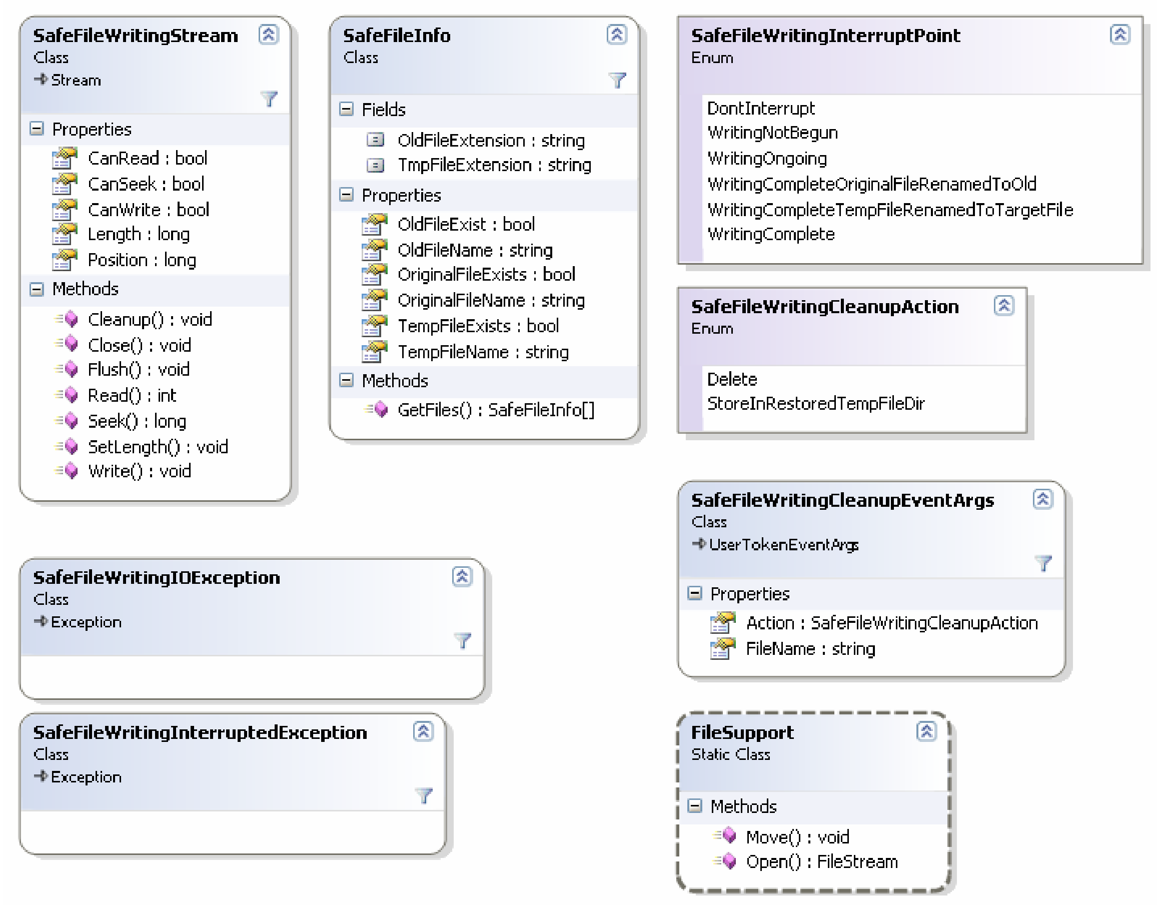
1.2 Purpose and scope

1.3 Abbreviations and acronyms

Abbreviation	Meaning

1.4 References

2 Overview


SafeFileWritingStream
Class
Stream
Properties
CanRead : bool
CanSeek : bool
CanWrite : bool
Length : long
Position : long
Methods
Cleanup() : void
Close() : void
Flush() : void
Read() : int
Seek() : long
SetLength() : void
Write() : void
SafeFileInfo
Class
Fields
OldFileExtension : string
TmpFileExtension : string
Properties
OldFileExist : bool
OldFileName : string
OriginalFileExists : bool
OriginalFileName : string
TempFileExists : bool
TempFileName : string
Methods
GetFiles() : SafeFileInfo[]
SafeFileWritingInterruptPoint
Enum
DontInterrupt
WritingNotBegun
WritingOngoing
WritingCompleteOriginalFileRenamedToOld
WritingCompleteTempFileRenamedToTargetFile
WritingComplete
SafeFileWritingCleanupAction
Enum
Delete
StoreInRestoredTempFileDir
SafeFileWritingCleanupEventArgs
Class
UserTokenEventArgs
Properties
Action : SafeFileWritingCleanupAction
FileName : string
SafeFileWritingIOException
Class
Exception
SafeFileWritingInterruptedException
Class
Exception
FileSupport
Static Class
Methods
Move() : void
Open() : FileStream

3 Namespace Wayne.Lib.IO

Classes

FileSupport	Support class for files.
SafeFileWritingCleanupEventArgs	Event argument that is used to notify and ask what action to take if a partially written file is found.
SafeFileWritingInterruptedException	Unit testing exception. Thrown when the processing was interrupted because of a deliberate interruption of the file writing for unit testing purposes.
SafeFileWritingIOException	Exception that is thrown when an unexpected file lock were found on any of the files that are involved in the safe file writing.
SafeFileWritingStream	A file writing stream that ensures that the files remain consistent even when a writing operation is interrupted. The writing will be performed to a temporary file that will be exchanged with the target file when the writing has completed. There is a static method, Cleanup that should be called when a program starts up that will clean up and restore the files in the best possible state.

Enumerations

SafeFileWritingCleanupAction	Possible actions to take when cleaning up files that has been written by SafeFileWritingStream, and interrupted.

3.1 Classes

3.1.1 Class FileSupport

```
abstract public class FileSupport : Object
```

Summary
Support class for files.
Methods

Move
```
public void Move(string sourceFileName, string destinationFileName, int
retries, int delayBetweenRetries);
```
Moves a file.

sourceFileName	
destinationFileName	
retries	
delayBetweenRetries	

Open
```
public IO.FileStream Open(string fileName, IO.FileMode fileMode,
IO.FileAccess fileAccess, IO.FileShare fileShare, int retries, int
delayBetweenRetries);
```
Opens a file.

fileName	

fileMode	
fileAccess	
fileShare	
retries	
delayBetweenRetries	

3.1.2 Class SafeFileWritingCleanupEventArgs

```
public class SafeFileWritingCleanupEventArgs : UserTokenEventArgs
```

Summary
Event argument that is used to notify and ask what action to take if a partially written file is found.
Properties

Action `Lib.IO.SafeFileWritingCleanupAction`	R/W	Sets or gets the action to take for the found temporary file.
FileName `string`	R	Name of the found temporary file.

3.1.3 Class SafeFileWritingInterruptedException

```
public class SafeFileWritingInterruptedException : Exception
```

Summary
Unit testing exception. Thrown when the processing was interrupted because of a deliberate interruption of the file writing for unit testing purposes.
Constructors

`public SafeFileWritingInterruptedException();` Constructor

`public SafeFileWritingInterruptedException(string message);` Constructor	
message	

`public SafeFileWritingInterruptedException(string message, Exception inner);` Constructor	
message	
inner	

3.1.4 Class SafeFileWritingIOException

```
public class SafeFileWritingIOException : Exception
```

Summary
Exception that is thrown when an unexpected file lock were found on any of the files that are involved in the safe file writing.
Constructors

`public SafeFileWritingIOException();` Constructor

`public SafeFileWritingIOException(string message);` Constructor	
message	

`public SafeFileWritingIOException(string message, Exception inner);` Constructor	
message	
inner	

3.1.5 Class SafeFileWritingStream

`public class SafeFileWritingStream : Stream`

Summary
A file writing stream that ensures that the files remain consistent even when a writing operation is interrupted. The writing will be performed to a temporary file that will be exchanged with the target file when the writing has completed. There is a static method, Cleanup that should be called when a program starts up that will clean up and restore the files in the best possible state.

Properties

CanRead `bool`	R	Always false, SafeFileStream is write-only
CanSeek `bool`	R	Always false, SafeFileStream is write-only
CanWrite `bool`	R	Always true, SafeFileStream is write-only
Length `long`	R	Gets the length in bytes of the stream.
Position `long`	R/W	Gets or sets the current position of this stream.

Constructors

`public SafeFileWritingStream(string fileName);` Creates a new instance of te SafeFileWriting Stream for the specified file.	
fileName	

Methods

Cleanup `public void Cleanup(string folderPath, string pattern, EventHandler{Wayne.Lib.IO.SafeFileWritingCleanupEventArgs} temporaryFileFoundCallback, object userToken);` All file types that is written with the SafeFileWritingStream should be cleaned at certain points to maintain the integrity. The typical place to place a call to this method is at the startup of a module. If the module wrote something and was interrupted by a program shutdown, it can rescue some data with this method.	
folderPath	
pattern	
temporaryFileFoundCallback	Delegate that is called when a temp file is found and asks for action to take. This delegate may be invoked several times.
userToken	Token that is returned in the invokation of the

	temporaryFileFoundCallback.

Close `public void Close();` Closes the stream and overwrites the target file.

Dispose `protected void Dispose(bool disposing);` Disposes the internal resources.	
disposing	

Flush `public void Flush();` Clears all buffers for this stream and causes any buffered data to be written to the underlying device.

Read `public int Read(Byte[] buffer, int offset, int count);` Not supported	
buffer	
offset	
count	

Seek `public long Seek(long offset, IO.SeekOrigin origin);` Not supported. Stream is Write-only.	
offset	
origin	

SetLength `public void SetLength(long value);` Not supported. Fast-forward writing only.	
value	

Write `public void Write(Byte[] buffer, int offset, int count);` Writes a block of bytes to this stream using data from a buffer.	
buffer	The buffer containing data to write to the stream.
offset	The zero-based byte offset in array at which to begin copying bytes to the current stream.
count	The maximum number of bytes to be written to the current stream.

3.2 Enumerations

3.2.1 Enumeration SafeFileWritingCleanupAction

Summary
Possible actions to take when cleaning up files that has been written by SafeFileWritingStream, and interrupted.
Fields

Delete	Just delete the file
StoreInRestoredTempFileDir	Delete the file, and store a copy in the Wayne restore temporary file folder.

4 Namespace Wayne.Lib.IO.UnitTest

Enumerations

SafeFileWritingInterruptPoint	Enumeration used for unit testing of the Safe File Writing stream

4.1 Enumerations

4.1.1 Enumeration SafeFileWritingInterruptPoint

Summary
Enumeration used for unit testing of the Safe File Writing stream
Fields

DontInterrupt	Dont interrupt the processing
WritingNotBegun	Interrupt before the writing has started
WritingOngoing	Interrupt when the writing has started
WritingCompleteOriginalFileRenamedToOld	Interrupt when the writing has finished, but the temp file has not yet been renamed to target file name.
WritingCompleteTempFileRenamedToTargetFile	Interrupt when the temp file has bee renamed to the target file, but the old file is not yet deleted.
WritingComplete	Interrupt after everything is done.