# OpcFileStream Members

**Namespace:** Opc.UaFx.Client
**Assemblies:** Opc.UaFx.Advanced.dll, Opc.UaFx.Advanced.dll
The OpcFileStream type exposes the following members.

# Constructors

## OpcFileStream(OpcClient, OpcNodeId, OpcFileMode)

Initializes a new instance of the OpcFileStream class using the specified `client`, `fileNodeId` and `mode`.

**C#**

```
public OpcFileStream(OpcClient client, OpcNodeId fileNodeId, OpcFileMode mode)
```

**Parameters**

`client` OpcClient

The OpcClient to use to access the file node.

`fileNodeId` OpcNodeId

The OpcNodeId of the file node to access.

`mode` OpcFileMode

The OpcFileMode to use to access the file node.

**Exceptions**

ArgumentNullException

The `client` or `fileNodeId` is a null reference (Nothing in Visual Basic).

## OpcFileStream(OpcFileInfo, OpcFileMode)

Initializes a new instance of the OpcFileStream class using the specified `file` and `mode`.

**C#**

```
public OpcFileStream(OpcFileInfo file, OpcFileMode mode)
```

**Parameters**

`file` OpcFileInfo

The OpcFileInfo of the file node to access.

`mode` OpcFileMode

The OpcFileMode to use to access the file node.

#### Exceptions

ArgumentNullException

The `file` is a null reference (Nothing in Visual Basic).

## OpcFileStream(OpcFileNodeContext, OpcFileMode)

Initializes a new instance of the OpcFileStream class using the specified `context` and `mode`.

#### C#

```
public OpcFileStream(OpcFileNodeContext context, OpcFileMode mode)
```

#### Parameters

`context` OpcFileNodeContext

The OpcFileNodeContext of the file node to access.

`mode` OpcFileMode

The OpcFileMode to use to access the file node.

#### Exceptions

ArgumentNullException

The `context` is a null reference (Nothing in Visual Basic).

# Properties

## CanRead

Gets a value indicating whether the current stream supports reading.

#### C#

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

#### Property Value

Boolean

The value true if the stream supports reading; otherwise the value false.

#### Remarks

If the stream is closed, this property returns false.

# CanSeek

Gets a value indicating whether the current stream supports seeking.

**C#**

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

**Property Value**

Boolean

The value true if the stream supports seeking; otherwise the value false.

**Remarks**

If the stream is closed, this property returns false.

# CanWrite

Gets a value indicating whether the current stream supports writing.

**C#**

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

**Property Value**

Boolean

The value true if the stream supports writing; otherwise the value false.

**Remarks**

If the stream is closed, this property returns false.

# File

Gets the OpcFileInfo used to determine the file accessibility and its size.

**C#**

```
public OpcFileInfo File { get; }
```

**Property Value**

OpcFileInfo

The OpcFileInfo used by the OpcFileStream object.

# Handle

Gets a SafeOpcFileHandle object that represents the file handle for the file that the current OpcFileStream object encapsulates.

**C#**

```
public SafeOpcFileHandle Handle { get; }
```

**Property Value**

SafeOpcFileHandle

An object that represents the file handle for the file that the current OpcFileStream object encapsulates or a null reference (Nothing in Visual Basic) then the file stream has been disposed of or closed.

# Length

Gets the length in bytes of the stream.

**C#**

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

**Property Value**

Int64

A long value representing the length of the stream in bytes.

**Exceptions**

ObjectDisposedException

The stream has been disposed of.

# Position

Gets or sets the position within the current stream.

**C#**

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

**Property Value**

Int64

The current position within the stream.

#### Exceptions

ObjectDisposedException

The stream has been disposed of.

# Methods

## Close()

Closes the current stream and releases any resources associated with the current stream. Instead of calling this method, ensure that the stream is properly disposed.

#### C#

```
public override sealed void Close()
```

#### Remarks

Attempts to manipulate the stream after the stream has been closed might throw an ObjectDisposedException.

## Flush()

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

#### C#

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

## Read(Byte[], Int32, Int32)

Reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read.

#### C#

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

#### Parameters

`buffer` Byte[]

An array of bytes. When this method returns, the buffer contains the specified byte array with the values

between `offset` and (`offset` + `count` - 1) replaced by the bytes read from the current source.

`offset` Int32

The zero-based byte offset in `buffer` at which to begin storing the data read from the current stream.

`count` Int32

The maximum number of bytes to be read from the current stream.

#### Returns

Int32

The total number of bytes read into the `buffer`. This can be less than the number of bytes requested if that many bytes are not currently available, or zero (0) if the end of the stream has been reached.

#### Exceptions

ArgumentException

The sum of `offset` and `count` is larger than the `buffer` length.

ArgumentNullException

The `buffer` is a null reference (Nothing in Visual Basic).

ArgumentOutOfRangeException

The `offset` or `count` is negative.

NotSupportedException

The stream does not support reading.

ObjectDisposedException

The stream has been disposed of.

#### Remarks

Use the CanRead property to determine whether the current instance supports reading.

## Seek(Int64, SeekOrigin)

Sets the position within the current stream.

#### C#

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

#### Parameters

`offset` Int64

A byte offset relative to the `origin` parameter.

`origin` SeekOrigin

A value of type SeekOrigin indicating the reference point used to obtain the new position.

#### Returns

Int64

The new position within the current stream.

#### Exceptions

NotSupportedException

The stream does not support seeking.

ObjectDisposedException

The stream has been disposed of.

#### Remarks

Use the CanSeek property to determine whether the current instance supports seeking. Seeking to any location beyond the length of the stream is supported.

## SetLength(Int64)

Sets the length of the current stream.

#### C#

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

#### Parameters

`value` Int64

The desired length of the current stream in bytes.

#### Exceptions

ObjectDisposedException

The stream has been disposed of.

NotSupportedException

The stream does not support both writing and seeking.

### Remarks

Use the CanWrite property to determine whether the current instance supports writing, and the CanSeek property to determine whether seeking is supported. If the specified value is less than the current length of the stream, the stream is truncated. If the specified value is larger than the current length of the stream, the stream is expanded. If the stream is expanded, the contents of the stream between the old and the new length are not defined.

## 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.

### C#

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

### Parameters

`buffer` Byte[]

An array of bytes. This method copies count bytes from `buffer` to the current stream.

`offset` Int32

The zero-based byte offset in `buffer` at which to begin copying bytes to the current stream.

`count` Int32

The number of bytes to be written to the current stream.

### Exceptions

ArgumentException

The sum of `offset` and `count` is larger than the `buffer` length.

ArgumentNullException

The `buffer` is a null reference (Nothing in Visual Basic).

ArgumentOutOfRangeException

The `offset` or `count` is negative.

NotSupportedException

The stream does not support writing.

ObjectDisposedException

The stream has been disposed of.

### Remarks

Use the CanWrite property to determine whether the current instance supports writing.

# Table of Contents

***Constructors*** ..... 1
OpcFileStream(OpcClient, OpcNodeId, OpcFileMode) ..... 1
OpcFileStream(OpcFileInfo, OpcFileMode) ..... 1
OpcFileStream(OpcFileNodeContext, OpcFileMode) ..... 2
***Properties*** ..... 2
CanRead ..... 2
CanSeek ..... 3
CanWrite ..... 3
File ..... 3
Handle ..... 4
Length ..... 4
Position ..... 4
***Methods*** ..... 5
Close() ..... 5
Flush() ..... 5
Read(Byte[], Int32, Int32) ..... 5
Seek(Int64, SeekOrigin) ..... 6
SetLength(Int64) ..... 7
Write(Byte[], Int32, Int32) ..... 8