ICSharpCode SharpZipLib Class Library
# ISharpCode.SharpZipLib Namespace

## Namespace hierarchy

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SharpZipBaseException</td>
<td>SharpZipBaseException is the base exception class for the SharpZipLibrary. All library exceptions are derived from this.</td>
</tr>
</tbody>
</table>
ICSharpCode SharpZipLib Class Library
SharpZipBaseException Class

SharpZipBaseException is the base exception class for the SharpZipLibrary. All library exceptions are derived from this.

For a list of all members of this type, see SharpZipBaseException Members.

System.Object System.Exception System.ApplicationException

ICSharpCode.SharpZipLib.SharpZipBaseException
   ISharpCode.SharpZipLib.BZip2.BZip2Exception
   ISharpCode.SharpZipLib.GZip.GZipException
   ISharpCode.SharpZipLib.Tar.TarException

public class SharpZipBaseException : ApplicationException

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Remarks

NOTE: Not all exceptions thrown will be derived from this class. A variety of other exceptions are possible for example ArgumentNullException

Requirements

Namespace: ISharpCode.SharpZipLib


See Also

SharpZipBaseException Members | ISharpCode.SharpZipLib Namespace
# SharpZipBaseException Members

## SharpZipBaseException overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="SharpZipBaseException" /></td>
<td>Overloaded. Initializes a new instance of the SharpZipBaseException class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="HelpLink" /> (inherited from Exception)</td>
<td>Gets or sets a link to the help file associated with this exception.</td>
</tr>
<tr>
<td><img src="image" alt="InnerException" /> (inherited from Exception)</td>
<td>Gets the Exception instance that caused the current exception.</td>
</tr>
<tr>
<td><img src="image" alt="Message" /> (inherited from Exception)</td>
<td>Gets a message that describes the current exception.</td>
</tr>
<tr>
<td><img src="image" alt="Source" /> (inherited from Exception)</td>
<td>Gets or sets the name of the application or the object that causes the error.</td>
</tr>
<tr>
<td><img src="image" alt="StackTrace" /> (inherited from Exception)</td>
<td>Gets a string representation of the frames on the call stack at the time the current exception was thrown.</td>
</tr>
<tr>
<td><img src="image" alt="TargetSite" /> (inherited from Exception)</td>
<td>Gets the method that throws the current exception.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Equals" /> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><img src="image" alt="GetBaseException" /> (inherited from Exception)</td>
<td>When overridden in a derived class, returns the Exception that</td>
</tr>
</tbody>
</table>
is the root cause of one or more subsequent exceptions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetObjectData</strong> (inherited from Exception)</td>
<td>When overridden in a derived class, sets the SerializationInfo with information about the exception.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Exception)</td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SharpZipBaseException</strong></td>
<td>Overloaded. Initializes a new instance of the SharpZipBaseException class.</td>
</tr>
</tbody>
</table>

### Protected Instance Properties

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HResult</strong> (inherited from Exception)</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>
See Also

SharpZipBaseException Class | ICSharpCode.SharpZipLib
Namespace
ICSharpCode SharpZipLib Class Library
**SharpZipBaseException Constructor**

Deserialization constructor

**Overload List**

Initializes a new instance of the SharpZipBaseException class.

```csharp
public SharpZipBaseException();
```

Deserialization constructor

```csharp
protected SharpZipBaseException(SerializationInfo, StreamingContext);
```

Initializes a new instance of the SharpZipBaseException class with a specified error message.

```csharp
public SharpZipBaseException(string);
```

Initializes a new instance of the SharpZipBaseException class with a specified error message and a reference to the inner exception that is the cause of this exception.

```csharp
public SharpZipBaseException(string, Exception);
```

**See Also**

[SharpZipBaseException Class](#) | [ICSharpCode.SharpZipLib Namespace](#)
SharpZipBaseException Constructor (SerializationInfo, StreamingContext)

Deserialization constructor

```csharp
protected SharpZipBaseException(
    SerializationInfo info,
    StreamingContext context
);
```

Parameters

- **info**
  - `SerializationInfo` for this constructor

- **context**
  - `StreamingContext` for this constructor

See Also

- [SharpZipBaseException Class](#)
- [ICSharpCode.SharpZipLib Namespace](#)
- [SharpZipBaseException Constructor Overload List](#)
SharpZipBaseException Constructor ()

Initializes a new instance of the SharpZipBaseException class.

```
public SharpZipBaseException();
```

See Also

SharpZipBaseException Class | ICSharpCode.SharpZipLib Namespace | SharpZipBaseException Constructor Overload List
ICSharpCode SharpZipLib Class Library
SharpZipBaseException Constructor (String)

Initializes a new instance of the SharpZipBaseException class with a specified error message.

```csharp
public SharpZipBaseException(
    string msg
);
```

See Also

[SharpZipBaseException Class](#) | [ICSharpCode.SharpZipLib Namespace](#) | [SharpZipBaseException Constructor Overload List](#)
SharpZipBaseException Constructor (String, Exception)

Initializes a new instance of the SharpZipBaseException class with a specified error message and a reference to the inner exception that is the cause of this exception.

```csharp
public SharpZipBaseException(
    string message,
    Exception innerException
);
```

Parameters

- `message`  
  Error message string

- `innerException`  
  The inner exception

See Also

- [SharpZipBaseException Class](#) | [ICSharpCode.SharpZipLib Namespace](#) | [SharpZipBaseException Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
## ISharpCode.SharpZipLib.BZip2 Namespace

### Namespace hierarchy

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BZip2</td>
<td>A helper class to simplify compressing and decompressing streams.</td>
</tr>
<tr>
<td>BZip2Constants</td>
<td>Defines internal values for both compression and decompression</td>
</tr>
<tr>
<td>BZip2Exception</td>
<td>BZip2Exception represents exceptions specific to Bzip2 algorithm</td>
</tr>
<tr>
<td>BZip2InputStream</td>
<td>An input stream that decompresses files in the BZip2 format</td>
</tr>
<tr>
<td>BZip2OutputStream</td>
<td>An output stream that compresses into the BZip2 format including file header chars into another stream.</td>
</tr>
</tbody>
</table>
BZip2 Class

A helper class to simplify compressing and decompressing streams. For a list of all members of this type, see BZip2 Members.


public sealed class BZip2

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

Namespace: ISharpCode.SharpZipLib.BZip2


See Also

BZip2 Members | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
## BZip2 Members

### BZip2 overview

<table>
<thead>
<tr>
<th>Public Static Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✷ Compress</td>
<td>Compress <code>inStream</code> sending result to <code>outStream</code></td>
</tr>
<tr>
<td>✷ Decompress</td>
<td>Decompress <code>inStream</code> writing decompressed data to the <code>outStream</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Public Instance Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✷ Equals (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td>✷ GetHashCode (inherited from <code>Object</code>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>✷ GetType (inherited from <code>Object</code>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td>✷ ToString (inherited from <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

### See Also

[BZip2 Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2 Methods

The methods of the BZip2 class are listed below. For a complete list of BZip2 class members, see the BZip2 Members topic.

Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compress</td>
<td>Compress inStream sending result to outStream</td>
</tr>
<tr>
<td>Decompress</td>
<td>Decompress inStream writing decompressed data to the outStream</td>
</tr>
</tbody>
</table>

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

See Also

BZip2 Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2.Compress Method

Compress inStream sending result to outStream

```csharp
public static void Compress(
    Stream inStream,
    Stream outStream,
    int blockSize
);
```

Parameters

inStream
- The stream to compress.

outStream
- The stream to write compressed data to.

blockSize
- The block size to use.

See Also

BZip2 Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2.Decompress Method

Decompress *inStream* writing decompressed data to the *outStream*

```java
public static void Decompress(
    Stream inStream,
    Stream outStream
);
```

**Parameters**

*inStream*  
The stream containing data to decompress.

*outStream*  
The stream to write decompressed data to.

**See Also**

[BZip2 Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2Constants Class

Defines internal values for both compression and decompression
For a list of all members of this type, see BZip2Constants Members.

System.Object
ICSharpCode.SharpZipLib.BZip2.BZip2Constants

public sealed class BZip2Constants

Thread Safety
Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

Namespace: IICSharpCode.SharpZipLib.BZip2

See Also

BZip2Constants Members | IICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
## BZip2Constants Members

### BZip2Constants overview

#### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseBlockSize</td>
<td>When multiplied by compression parameter (1-9) gives the block size for compression 9 gives the best compesssion but uses the most memory.</td>
</tr>
<tr>
<td>G_SIZE</td>
<td>Backend constant</td>
</tr>
<tr>
<td>MAX_ALPHA_SIZE</td>
<td>Backend constant</td>
</tr>
<tr>
<td>MAX_CODE_LEN</td>
<td>Backend constant</td>
</tr>
<tr>
<td>MAX_SELECTORS</td>
<td>Backend constant</td>
</tr>
<tr>
<td>N_GROUPS</td>
<td>Backend constant</td>
</tr>
<tr>
<td>N_ITERS</td>
<td>Backend constant</td>
</tr>
<tr>
<td>NUM_OVERSHOOT_BYTES</td>
<td>Backend constant</td>
</tr>
<tr>
<td>rNNums</td>
<td>Random numbers used to randomise repetitive blocks</td>
</tr>
<tr>
<td>RUNA</td>
<td>Backend constant</td>
</tr>
<tr>
<td>RUNB</td>
<td>Backend constant</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

[BZip2Constants Class] | [ICSharpCode.SharpZipLib.BZip2 Namespace]
The fields of the **BZip2Constants** class are listed below. For a complete list of **BZip2Constants** class members, see the **BZip2Constants Members** topic.

### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseBlockSize</td>
<td>When multiplied by compression parameter (1-9) gives the block size for compression 9 gives the best compression but uses the most memory.</td>
</tr>
<tr>
<td>G_SIZE</td>
<td>Backend constant</td>
</tr>
<tr>
<td>MAX_ALPHA_SIZE</td>
<td>Backend constant</td>
</tr>
<tr>
<td>MAX_CODE_LEN</td>
<td>Backend constant</td>
</tr>
<tr>
<td>MAX_SELECTORS</td>
<td>Backend constant</td>
</tr>
<tr>
<td>N_GROUPS</td>
<td>Backend constant</td>
</tr>
<tr>
<td>N_ITERS</td>
<td>Backend constant</td>
</tr>
<tr>
<td>NUM_OVERSHOOT_BYTES</td>
<td>Backend constant</td>
</tr>
<tr>
<td>rNums</td>
<td>Random numbers used to randomise repetitive blocks</td>
</tr>
<tr>
<td>RUNA</td>
<td>Backend constant</td>
</tr>
<tr>
<td>RUNB</td>
<td>Backend constant</td>
</tr>
</tbody>
</table>

**See Also**

[BZip2Constants Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2Constants.baseBlockSize Field

When multiplied by compression parameter (1-9) gives the block size for compression 9 gives the best compression but uses the most memory.

```csharp
public const int baseBlockSize = 100000;
```

See Also

[BZip2Constants Class] | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2Constants.G_SIZE Field

Backend constant

```csharp
public const int G_SIZE = 50;
```

See Also

BZip2Constants Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2Constants.MAX_ALPHA_SIZE Field

Backend constant

```csharp
public const int MAX_ALPHA_SIZE = 258;
```

See Also

ICSharpCode SharpZipLib Class Library
BZip2Constants.MAX_CODE_LEN Field

Backend constant

```csharp
public const int MAX_CODE_LEN = 23;
```

See Also

[BZip2Constants Class] | [ICSharpCode.SharpZipLib.BZip2 Namespace]
ICSharpCode SharpZipLib Class Library
Backend constant

```csharp
public const int MAX_SELECTORS = 18002;
```

See Also

[BZip2Constants Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2Constants.N_GROUPS Field

Backend constant

```csharp
public const int N_GROUPS = 6;
```

See Also

[BZip2Constants Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
BZip2Constants.N_ITERS Field

Backend constant

```csharp
public const int N_ITERS = 4;
```

See Also

BZip2Constants Class | ISharpCode.SharpZipLib.BZip2 Namespace
BZip2Constants.NUM_OVERSHOOT_BYTES Field

Backend constant

```csharp
public const int NUM_OVERSHOOT_BYTES = 20;
```

See Also

BZip2Constants Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
**BZip2Constants.rNums Field**

Random numbers used to randomise repetitive blocks

```csharp
public static readonly int[] rNums;
```

See Also

- [BZip2Constants Class](https://icsharpcode.net/SharpZipLib.BZip2)
- [ICSharpCode.SharpZipLib.BZip2 Namespace](https://icsharpcode.net/SharpZipLib.BZip2_Namespace)
ICSharpCode SharpZipLib Class Library
BZip2Constants.RUNA Field

Backend constant

```csharp
public const int RUNA = 0;
```

See Also

[BZip2Constants Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2Constants.RUNB Field

Backend constant

```
public const int RUNB = 1;
```

See Also

[BZip2Constants Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2Exception Class

BZip2Exception represents exceptions specific to Bzip2 algorithm. For a list of all members of this type, see BZip2Exception Members.

System.Object System.Exception System.ApplicationException

ICSharpCode.SharpZipLib.SharpZipBaseException
ICSharpCode.SharpZipLib.BZip2.BZip2Exception

public class BZip2Exception : SharpZipBaseException

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

Namespace: ISharpCode.SharpZipLib.BZip2

See Also

BZip2Exception Members | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
# BZip2Exception Members

## BZip2Exception overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BZip2Exception</strong></td>
<td>Overloaded. Initializes a new instance of the BZip2Exception class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HelpLink</strong> (inherited from Exception)</td>
<td>Gets or sets a link to the help file associated with this exception.</td>
</tr>
<tr>
<td><strong>InnerException</strong> (inherited from Exception)</td>
<td>Gets the <code>Exception</code> instance that caused the current exception.</td>
</tr>
<tr>
<td><strong>Message</strong> (inherited from Exception)</td>
<td>Gets a message that describes the current exception.</td>
</tr>
<tr>
<td><strong>Source</strong> (inherited from Exception)</td>
<td>Gets or sets the name of the application or the object that causes the error.</td>
</tr>
<tr>
<td><strong>StackTrace</strong> (inherited from Exception)</td>
<td>Gets a string representation of the frames on the call stack at the time the current exception was thrown.</td>
</tr>
<tr>
<td><strong>TargetSite</strong> (inherited from Exception)</td>
<td>Gets the method that throws the current exception.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>GetBaseException</strong> (inherited from <code>Exception</code>)</td>
<td>When overridden in a derived class, returns the <code>Exception</code> that</td>
</tr>
</tbody>
</table>
is the root cause of one or more subsequent exceptions.

<table>
<thead>
<tr>
<th><strong>GetHashCode</strong> (inherited from Object)</th>
<th>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetObjectData</strong> (inherited from Exception)</td>
<td>When overridden in a derived class, sets the SerializationInfo with information about the exception.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Exception)</td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

**Protected Instance Constructors**

| **BZip2Exception** | Overloaded. Initializes a new instance of the BZip2Exception class. |

**Protected Instance Properties**

| **HResult** (inherited from Exception) | Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. |

**Protected Instance Methods**

| **Finalize** (inherited from Object) | Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. |
| **MemberwiseClone** (inherited from Object) | Creates a shallow copy of the current Object. |
See Also

BZip2Exception Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
**BZip2Exception Constructor**

Deserialization constructor

**Overload List**

Initialise a new instance of BZip2Exception.

```
public BZip2Exception();
```

Deserialization constructor

```
protected BZip2Exception(SerializationInfo, StreamingContext);
```

Initialise a new instance of BZip2Exception with its message set to message.

```
public BZip2Exception(string);
```

Initialise an instance of BZip2Exception

```
public BZip2Exception(string, Exception);
```

See Also

[BZip2Exception Class] | [ICSharpCode.SharpZipLib.BZip2 Namespace]
ICSharpCode SharpZipLib Class Library
**BZip2Exception Constructor (SerializationInfo, StreamingContext)**

Deserialization constructor

```csharp
protected BZip2Exception(
    SerializationInfo info,
    StreamingContext context
);
```

**Parameters**

- `info`
  - `SerializationInfo` for this constructor

- `context`
  - `StreamingContext` for this constructor

**See Also**

- BZip2Exception Class | ISharpCode.SharpZipLib.BZip2 Namespace | BZip2Exception Constructor Overload List
BZip2Exception Constructor ()

Initialise a new instance of BZip2Exception.

```java
public BZip2Exception();
```

See Also

[BZip2Exception Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#) | [BZip2Exception Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
BZip2Exception Constructor (String)

Initialise a new instance of BZip2Exception with its message set to message.

```java
public BZip2Exception(
    string message
);
```

Parameters

- `message`
  The message describing the error.

See Also

BZip2Exception Class | ISharpCode.SharpZipLib.BZip2 Namespace | BZip2Exception Constructor Overload List
BZip2Exception Constructor (String, Exception)

Initialise an instance of BZip2Exception

```java
public BZip2Exception(
    string message,
    Exception exception
);
```

**Parameters**

- `message`
  - A message describing the error.

- `exception`
  - The exception that is the cause of the current exception.

**See Also**

- [BZip2Exception Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#) | [BZip2Exception Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
**BZip2InputStream Class**

An input stream that decompresses files in the BZip2 format

For a list of all members of this type, see [BZip2InputStream Members](#).

```csharp
```

```csharp
public class BZip2InputStream : Stream
```

**Thread Safety**

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** [ICSharpCode.SharpZipLib.BZip2](#)

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

**See Also**

[BZip2InputStream Members](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
## BZip2InputStream Members

### BZip2InputStream overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BZip2InputStream</code> Constructor</td>
<td>Construct instance for reading from stream</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CanRead</code></td>
<td>Gets a value indicating if the stream supports reading</td>
</tr>
<tr>
<td><code>CanSeek</code></td>
<td>Gets a value indicating whether the current stream supports seeking.</td>
</tr>
<tr>
<td><code>CanWrite</code></td>
<td>Gets a value indicating whether the current stream supports writing. This property always returns false</td>
</tr>
<tr>
<td><code>IsStreamOwner</code></td>
<td>Get/set flag indicating ownership of underlying stream. When the flag is true <code>Close</code> will close the underlying stream also.</td>
</tr>
<tr>
<td><code>Length</code></td>
<td>Gets the length in bytes of the stream.</td>
</tr>
<tr>
<td><code>Position</code></td>
<td>Gets or sets the streams position. Setting the position is not supported and will throw a NotSupportException</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BeginRead</code></td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><code>BeginWrite</code></td>
<td>Begins an asynchronous write</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the stream, releasing any associated resources.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the stream.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Read a sequence of bytes and advances the read position by one byte.</td>
</tr>
<tr>
<td><strong>ReadByte</strong></td>
<td>Read a byte from stream advancing position</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>Set the streams position. This</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Sets the length of this stream to the given value. This operation is not supported and will throw a NotSupportedException.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes a block of bytes to this stream using data from a buffer. This operation is not supported and will throw a NotSupportedException.</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Writes a byte to the current position in the file stream. This operation is not supported and will throw a NotSupportedException.</td>
</tr>
</tbody>
</table>

Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateWaitHandle</strong> (inherited from Stream)</td>
<td>Allocates a WaitHandle object.</td>
</tr>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

See Also

BZip2InputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream Constructor

Construct instance for reading from stream

```java
public BZip2InputStream(
    Stream stream
);
```

Parameters

*stream*

Data source

See Also

BZip2InputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputSteam Properties

The properties of the **BZip2InputSteam** class are listed below. For a complete list of **BZip2InputSteam** class members, see the [BZip2InputSteam Members](#) topic.

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CanRead</strong></td>
<td>Gets a value indicating if the stream supports reading</td>
</tr>
<tr>
<td><strong>CanSeek</strong></td>
<td>Gets a value indicating whether the current stream supports seeking.</td>
</tr>
<tr>
<td><strong>CanWrite</strong></td>
<td>Gets a value indicating whether the current stream supports writing.</td>
</tr>
<tr>
<td><strong>CanWrite</strong></td>
<td>This property always returns false</td>
</tr>
<tr>
<td><strong>IsStreamOwner</strong></td>
<td>Get/set flag indicating ownership of underlying stream. When the flag is true</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>Gets the length in bytes of the stream.</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>Gets or sets the streams position. Setting the position is not supported and</td>
</tr>
</tbody>
</table>

See Also

[**BZip2InputSteam Class**](#) | [**ICSharpCode.SharpZipLib.BZip2 Namespace**](#)
ICSharpCode SharpZipLib Class Library
BZip2InputStream.CanRead Property

Gets a value indicating if the stream supports reading

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

See Also

- BZip2InputStream Class
- ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream.CanSeek Property

Gets a value indicating whether the current stream supports seeking.

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

See Also

BZip2InputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream.CanWrite Property

Gets a value indicating whether the current stream supports writing. This property always returns false.

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

See Also

BZip2InputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
Get/set flag indicating ownership of underlying stream. When the flag is true, **Close** will close the underlying stream also.

```csharp
public bool IsStreamOwner {get; set;}
```

See Also

[BZip2InputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2InputStream.Length Property

Gets the length in bytes of the stream.

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

See Also

BZip2InputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream.Position Property

Gets or sets the streams position. Setting the position is not supported and will throw a NotSupportedException

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any attempt to set the position</td>
</tr>
</tbody>
</table>

See Also

BZip2InputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
## BZip2InputSteam Methods

The methods of the **BZip2InputSteam** class are listed below. For a complete list of **BZip2InputSteam** class members, see the [BZip2InputSteam Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong></td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong></td>
<td>Begins an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the stream, releasing any associated resources.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong></td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong></td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong></td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the stream.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong></td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong></td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Read a sequence of bytes and advances the read position by one byte.</td>
</tr>
<tr>
<td><strong>ReadByte</strong></td>
<td>Read a byte from stream advancing position</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>Set the streams position. This operation is not supported and will throw a NotSupportedException</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Sets the length of this stream to the given value. This operation is not supported and will throw a NotSupportedException</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes a block of bytes to this stream using data from a buffer. This operation is not supported and will throw a NotSupportedException</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Writes a byte to the current position in the file stream. This operation is not supported and will throw a NotSupportedException</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

| **CreateWaitHandle** (inherited from Stream) | Allocates a WaitHandle object. |
| **Finalize** (inherited from Object)        | Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. |
| **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

[BZip2InputStream Class] | [ICSharpCode.SharpZipLib.BZip2 Namespace]
ICSharpCode SharpZipLib Class Library
BZip2InputStream.Close Method

Closes the stream, releasing any associated resources.

```csharp
public override void Close();
```

See Also

BZip2InputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream.Flush Method

Flushes the stream.

```csharp
public override void Flush();
```

See Also

BZip2InputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream.Read Method

Read a sequence of bytes and advances the read position by one byte.

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

### Parameters

- `buffer`  
  Array of bytes to store values in

- `offset`  
  Offset in array to begin storing data

- `count`  
  The maximum number of bytes to read

### Return Value

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.

### See Also

BZip2InputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream.ReadByte Method

Read a byte from stream advancing position

```
public override int ReadByte();
```

Return Value

byte read or -1 on end of stream

See Also

BZip2InputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream.Seek Method

Set the streams position. This operation is not supported and will throw a NotSupportedException

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

[BZip2InputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
Sets the length of this stream to the given value. This operation is not supported and will throw a NotSupportedException or IOException.

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

- BZip2InputStream Class
- ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream.Write Method

Writes a block of bytes to this stream using data from a buffer. This operation is not supported and will throw a NotSupportedException.

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

BZip2InputStream Class  |  ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2InputStream.WriteByte Method

Writes a byte to the current position in the file stream. This operation is not supported and will throw a NotSupportedException

```csharp
public override void WriteByte(
  byte value
);
```

Parameters

**value**

The value to write.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

BZip2InputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
**BZip2OutputStream Class**

An output stream that compresses into the BZip2 format including file header chars into another stream.

For a list of all members of this type, see [BZip2OutputStream Members](#).


```plaintext
public class BZip2OutputStream : Stream
```

**Thread Safety**

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are *not* guaranteed to be thread-safe.

**Requirements**

- **Namespace**: [ICSharpCode.SharpZipLib.BZip2](#)

**See Also**

[BZip2OutputStream Members](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2OutputStream Members

**BZip2OutputStream overview**

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>BZip2OutputStream</th>
<th>Overloaded. Initializes a new instance of the BZip2OutputStream class.</th>
</tr>
</thead>
</table>

**Public Instance Properties**

<table>
<thead>
<tr>
<th>BytesWritten</th>
<th>Get the number of bytes written to the output.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Gets a value indicating whether the current stream supports reading</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Gets a value indicating whether the current stream supports seeking</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Gets a value indicating whether the current stream supports writing</td>
</tr>
<tr>
<td>IsStreamOwner</td>
<td>Get/set flag indicating ownership of underlying stream. When the flag is true <strong>Close</strong> will close the underlying stream also.</td>
</tr>
<tr>
<td>Length</td>
<td>Gets the length in bytes of the stream</td>
</tr>
<tr>
<td>Position</td>
<td>Gets or sets the current position of this stream.</td>
</tr>
</tbody>
</table>

**Public Instance Methods**

<table>
<thead>
<tr>
<th>BeginRead (inherited from Stream)</th>
<th>Begins an asynchronous read operation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from Stream)</td>
<td>Begins an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>End the current block and end compression. Close the stream and free any resources.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flush output buffers</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Read a block of bytes</td>
</tr>
<tr>
<td><strong>ReadByte</strong></td>
<td>Read a byte from the stream advancing the position.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>Sets the current position of this stream to the given value.</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Sets the length of this stream to the given value.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Write a block of bytes to the stream</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Write a byte to the stream</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateWaitHandle</strong> (inherited from <strong>Stream</strong>)</td>
<td>Allocates a <strong>WaitHandle</strong> object.</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Releases the unmanaged resources used by the <strong>BZip2OutputStream</strong> and optionally releases the managed resources.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Ensures that resources are freed and other cleanup operations are performed when the garbage collector reclains the <strong>BZip2OutputStream</strong>.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

[BZip2OutputStream Class] | [ICSharpCode.SharpZipLib.BZip2 Namespace]
**BZip2OutputStream Constructor**

Construct a default output stream with maximum block size

**Overload List**

Construct a default output stream with maximum block size

```csharp
public BZip2OutputStream(Stream);
```

Initialise a new instance of the `BZip2OutputStream` for the specified stream, using the given blocksize.

```csharp
public BZip2OutputStream(Stream,int);
```

**See Also**

[BZip2OutputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
Construct a default output stream with maximum block size

```java
public BZip2OutputStream(
    Stream stream
);
```

Parameters

`stream`

The stream to write BZip data onto.

See Also

BZip2OutputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace | BZip2OutputStream Constructor Overload List
ICSharpCode SharpZipLib Class Library
BZip2OutputStream Constructor (Stream, Int32)

Initialise a new instance of the BZip2OutputStream for the specified stream, using the given blocksize.

```csharp
public BZip2OutputStream(
    Stream stream,
    int blockSize
);
```

Parameters

- `stream`
  The stream to write compressed data to.

- `blockSize`
  The block size to use.

Remarks

Valid block sizes are in the range 1..9, with 1 giving the lowest compression and 9 the highest.

See Also

BZip2OutputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace | BZip2OutputStream Constructor Overload List
ICSharpCode SharpZipLib Class Library
# BZip2OutputStream Properties

The properties of the **BZip2OutputStream** class are listed below. For a complete list of **BZip2OutputStream** class members, see the [BZip2OutputStream Members](#) topic.

## Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>❌ BytesWritten</td>
<td>Get the number of bytes written to the output.</td>
</tr>
<tr>
<td>❌ CanRead</td>
<td>Gets a value indicating whether the current stream supports reading.</td>
</tr>
<tr>
<td>❌ CanSeek</td>
<td>Gets a value indicating whether the current stream supports seeking.</td>
</tr>
<tr>
<td>❌ CanWrite</td>
<td>Gets a value indicating whether the current stream supports writing.</td>
</tr>
<tr>
<td>❌ IsStreamOwner</td>
<td>Get/set flag indicating ownership of underlying stream. When the flag is true Close will close the underlying stream also.</td>
</tr>
<tr>
<td>❌ Length</td>
<td>Gets the length in bytes of the stream.</td>
</tr>
<tr>
<td>❌ Position</td>
<td>Gets or sets the current position of this stream.</td>
</tr>
</tbody>
</table>

## See Also

- [BZip2OutputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
Get the number of bytes written to the output.

```csharp
public int BytesWritten {get;}
```

See Also

[BZip2OutputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.CanRead Property

Gets a value indicating whether the current stream supports reading

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

See Also

- BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
BZip2OutputStream.CanSeek Property

Gets a value indicating whether the current stream supports seeking

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

See Also

BZip2OutputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
BZip2OutputStream.CanWrite Property

Gets a value indicating whether the current stream supports writing

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

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.IsStreamOwner Property

Get/set flag indicating ownership of underlying stream. When the flag is true Close will close the underlying stream also.

```csharp
public bool IsStreamOwner {get; set;}
```

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
BZip2OutputStream.Length Property

Gets the length in bytes of the stream

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

See Also

- BZip2OutputStream Class
- ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.Position Property

Gets or sets the current position of this stream.

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

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
## BZip2OutputStream Methods

The methods of the BZip2OutputStream class are listed below. For a complete list of BZip2OutputStream class members, see the BZip2OutputStream Members topic.

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead (inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td>BeginWrite (inherited from Stream)</td>
<td>Begins an asynchronous write operation.</td>
</tr>
<tr>
<td>Close</td>
<td>End the current block and end compression. Close the stream and free any resources</td>
</tr>
<tr>
<td>CreateObjRef (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td>EndRead (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td>EndWrite (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>Flush</td>
<td>Flush output buffers</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetLifetimeService (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from <strong>MarshalByRefObject</strong>)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Read a block of bytes</td>
</tr>
<tr>
<td><strong>ReadByte</strong></td>
<td>Read a byte from the stream advancing the position.</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>Sets the current position of this stream to the given value.</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Sets the length of this stream to the given value.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Write a block of bytes to the stream</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Write a byte to the stream</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateWaitHandle</strong> (inherited from <strong>Stream</strong>)</td>
<td>Allocates a <strong>WaitHandle</strong> object.</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Releases the unmanaged resources used by the <strong>BZip2OutputStream</strong> and optionally releases the managed resources.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Ensures that resources are freed and other cleanup operations are performed when the garbage collector reclaims the <strong>BZip2OutputStream</strong>.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>
See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.Close Method

End the current block and end compression. Close the stream and free any resources

```csharp
public override void Close();
```

See Also

BZip2OutputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.Dispose Method

Releases the unmanaged resources used by the BZip2OutputStream and optionally releases the managed resources.

```csharp
protected virtual void Dispose(
    bool disposing
);
```

Parameters

_disposing_
true to release both managed and unmanaged resources; false to release only unmanaged resources.

See Also

BZip2OutputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.Finalize Method

Ensures that resources are freed and other cleanup operations are performed when the garbage collector reclaims the BZip2OutputStream.

protected override void Finalize();

See Also

BZip2OutputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.Flush Method

Flush output buffers

```java
public override void Flush();
```

See Also

[BZip2OutputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
BZip2OutputStream.Read Method

Read a block of bytes

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

See Also

[See Also](#) [BZip2OutputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.ReadByte Method

Read a byte from the stream advancing the position.

```csharp
public override int ReadByte();
```

See Also

- [BZip2OutputStream Class](#)
- [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
BZip2OutputStream.Seek Method

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

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

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
Sets the length of this stream to the given value.

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

See Also

[BZip2OutputStream Class] | [ICSharpCode.SharpZipLib.BZip2 Namespace]
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.Write Method

Write a block of bytes to the stream

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

See Also

BZip2OutputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode SharpZipLib Class Library
BZip2OutputStream.WriteByte Method

Write a byte to the stream.

```csharp
public override void WriteByte(byte value);
```

See Also

BZip2OutputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
Adler32

Computes Adler32 checksum for a stream of data. An Adler32 checksum is not as reliable as a CRC32 checksum, but a lot faster to compute. The specification for Adler32 may be found in RFC 1950. ZLIB (Data Format Specification version 3.3) From that document: (Adler-32 checksum) This contains a checksum value of the data (excluding any dictionary data) computed according to the algorithm. This algorithm is a 32-bit extension and improvement to the Fletcher algorithm, used in the ITU-T X.224 / ISO 8073 standard. The algorithm is composed of two sums accumulated per byte: \( s_1 \) is the sum of all \( s_1 \) values. Both sums are done modulo 65521, \( s_1 \) initialized to 1, \( s_2 \) to zero. The Adler-32 checksum is stored as \( s_2 \times 65536 + s_1 \) in most-significant-byte first (network) order.

8.2. The Adler-32 algorithm

The Adler-32 algorithm is much faster than the CRC32 algorithm yet still provides an extremely low probability of undetected errors. The modulo operation time is negligible. If the bytes are \( a, b, c \), the second sum is \( 3a + 2b + c + 3 \), and so is position and order sensitive, unlike the first sum, which is just a checksum. That 65521 is prime is important to avoid a possible large class of two-byte errors that leave the checksum unchanged. (The Fletcher checksum uses 255, which is not prime and makes the Fletcher check insensitive to single byte changes.) The sum \( s_1 \) is initialized to 1 instead of zero to make the length part of \( s_2 \), so that the length does not have to be checked separately. (Any sequence of zeroes has a Fletcher checksum of zero.)

Crc32

Generate a table for a byte-wise 32-bit CRC calculation of the polynomial:
\[ x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + \]

Polynomials over GF(2) are represented in binary, one bit per coefficient, with the lowest powers in the most significant bit. Then exclusive-or is just exclusive-or, and multiplying a polynomial by \( x \) is a right shift by one. If we call the above polynomial \( p \), and represent a byte as a polynomial with the lowest power in the most significant bit (so...
polynomial \( x^7 + x^3 + x + 1 \), then the CRC is \( (q \cdot x^{32}) \mod p \), where \( \mod \) means the remainder after dividing a by b. This calculation is done using a shift-register method of multiplying and taking the remainder. The register is initialized to zero, and for each incoming bit, \( x^{32} \) is added to the register if the bit is a one (where \( x^{32} \mod p = p + x^{32} \)). The register is multiplied \( \mod p \) by \( x \) (which is shifting right by one) if the bit shifted out is a one. We start with the highest power (least significant bit) of \( q \) and repeat for all eight bits of \( q \). The table is simply the CRC of all possible eight bit values. This is all the information needed to generate CRC's on data a byte at a time for all combinations of CRC register values and incoming bytes.

| StrangeCRC | Bzip2 checksum algorithm |

### Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IChecksum</td>
<td>Interface to compute a data checksum used by checked input/output streams. A data checksum can be updated by one byte or with a byte array. After each update the value of the current checksum can be returned by calling <code>getValue</code>. The complete checksum object can also be reset so it can be used again with new data.</td>
</tr>
</tbody>
</table>
ICSharpCode SharpZipLib Class Library
Adler32 Class

Computes Adler32 checksum for a stream of data. An Adler32 checksum is not as reliable as a CRC32 checksum, but a lot faster to compute. The specification for Adler32 may be found in RFC 1950. ZLIB Compressed Data Format Specification version 3.3).

From that document: "ADLER32 (Adler-32 checksum) This contains a checksum value of the uncompressed data (excluding any dictionary data) computed according to Adler-32 algorithm. This algorithm is a 32-bit extension and improvement of the Fletcher algorithm, used in the ITU-T X.224 / ISO 8073 standard. Adler-32 is composed of two sums accumulated per byte: s1 is the sum of all bytes, s2 is the sum of all s1 values. Both sums are done modulo 65521. s1 is initialized to 1, s2 to zero. The Adler-32 checksum is stored as s2*65536 + s1 in most-significant-byte first (network) order." "8.2. The Adler-32 algorithm The Adler-32 algorithm is much faster than the CRC32 algorithm yet still provides an extremely low probability of undetected errors. The modulo on unsigned long accumulators can be delayed for 5552 bytes, so the modulo operation time is negligible. If the bytes are a, b, c, the second sum is 3a + 2b + c + 3, and so is position and order sensitive, unlike the first sum, which is just a checksum. That 65521 is prime is important to avoid a possible large class of two-byte errors that leave the check unchanged. (The Fletcher checksum uses 255, which is not prime and which also makes the Fletcher check insensitive to single byte changes 0 - 255.) The sum s1 is initialized to 1 instead of zero to make the length of the sequence part of s2, so that the length does not have to be checked separately. (Any sequence of zeroes has a Fletcher checksum of zero.)"

For a list of all members of this type, see Adler32 Members.


public sealed class Adler32 : IChecksum

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed
to be thread-safe.

Requirements


See Also

Adler32 Members | ICSHarpCode.SharpZipLib.Checksums Namespace
# Adler32 Members

## Adler32 overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adler32 Constructor</strong></td>
<td>Creates a new instance of the Adler32 class. The checksum starts off with a value of 1.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value</strong></td>
<td>Returns the Adler32 data checksum computed so far.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <code>Object</code>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <code>Object</code>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets the Adler32 checksum to the initial value.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Overloaded. Updates the checksum with the bytes taken from the array.</td>
</tr>
</tbody>
</table>

### See Also

- Adler32 Class
- ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
Adler32 Constructor

Creates a new instance of the Adler32 class. The checksum starts off with a value of 1.

```csharp
public Adler32();
```

See Also

Adler32 Properties

The properties of the Adler32 class are listed below. For a complete list of Adler32 class members, see the Adler32 Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Returns the Adler32 data checksum computed so far.</td>
</tr>
</tbody>
</table>

See Also

Adler32 Class | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
**Adler32.Value Property**

Returns the Adler32 data checksum computed so far.

```csharp
public long Value {get;}
```

**Implements**

IChecksum.Value

**See Also**

Adler32 Class | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
The methods of the Adler32 class are listed below. For a complete list of Adler32 class members, see the Adler32 Members topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets the Adler32 checksum to the initial value.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Overloaded. Updates the checksum with the bytes taken from the array.</td>
</tr>
</tbody>
</table>

See Also

Adler32 Class | ISharpCode.SharpZipLib.Checksums Namespace
Adler32.Reset Method

Resets the Adler32 checksum to the initial value.

```csharp
public void Reset();
```

Implements

IChecksum.Reset

See Also

Adler32 Class | ICSharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
Adler32.Update Method

Updates the checksum with an array of bytes.

Overload List

Updates the checksum with an array of bytes.

public void Update(byte[]);

Updates the checksum with the bytes taken from the array.

public void Update(byte[],int,int);

Updates the checksum with a byte value.

public void Update(int);

See Also

Adler32 Class | ISharpCode.SharpZipLib.Checksums Namespace
Adler32.Update Method (Byte[])

Updates the checksum with an array of bytes.

```csharp
public void Update(byte[] buffer);
```

Parameters

buffer

- The source of the data to update with.

Implements

IChecksum.Update

See Also

Adler32.Update Method (Byte[], Int32, Int32)

Updates the checksum with the bytes taken from the array.

```csharp
public void Update(
    byte[] buffer,
    int offset,
    int count
);
```

Parameters

- **buffer**
  - an array of bytes

- **offset**
  - the start of the data used for this update

- **count**
  - the number of bytes to use for this update

Implements

- IChecksum.Update

See Also

ICSharpCode SharpZipLib Class Library
Adler32.Update Method (Int32)

Updates the checksum with a byte value.

```csharp
public void Update(
    int value
);
```

Parameters

value

The data value to add. The high byte of the int is ignored.

Implements

IChecksum.Update

See Also

ICSharpCode SharpZipLib Class Library
Crc32 Class

Generate a table for a byte-wise 32-bit CRC calculation on the polynomial:
\[ x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1. \]
Polynomials over GF(2) are represented in binary, one bit per coefficient, with the lowest powers in the most significant bit. Then adding polynomials is just exclusive-or, and multiplying a polynomial by \( x \) is a right shift by one. If we call the above polynomial \( p \), and represent a byte as the polynomial \( q \), also with the lowest power in the most significant bit (so the byte 0xb1 is the polynomial \( x^7 + x^3 + x + 1 \)), then the CRC is \( (q \times x^{32}) \mod p \), where \( a \mod b \) means the remainder after dividing \( a \) by \( b \). This calculation is done using the shift-register method of multiplying and taking the remainder. The register is initialized to zero, and for each incoming bit, \( x^{32} \) is added mod \( p \) to the register if the bit is a one (where \( x^{32} \mod p = p + x^{32} = x^{26} + \ldots + 1 \)), and the register is multiplied mod \( p \) by \( x \) (which is shifting right by one and adding \( x^{32} \mod p \) if the bit shifted out is a one). We start with the highest power (least significant bit) of \( q \) and repeat for all eight bits of \( q \). The table is simply the CRC of all possible eight bit values. This is all the information needed to generate CRC's on data a byte at a time for all combinations of CRC register values and incoming bytes.

For a list of all members of this type, see Crc32 Members.


public sealed class Crc32 : IChecksum

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

Assembly: ICSharpCode.SharpZipLib (in
ICSharpCode.SharpZipLib.dll

See Also

Crc32 Members | ICSharpCode.SharpZipLib.Checksums
Namespace
ICSharpCode SharpZipLib Class Library
# Crc32 Members

## Crc32 overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crc32 Constructor</td>
<td>Initializes a new instance of the Crc32 class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Returns the CRC32 data checksum computed so far.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>Reset</td>
<td>Resets the CRC32 data checksum as if no update was ever called.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td>Update</td>
<td>Overloaded. Adds the byte array to the data checksum.</td>
</tr>
</tbody>
</table>

### See Also

- Crc32 Class
- ISharpCode.SharpZipLib.Checksums Namespace
Crc32 Constructor

Initializes a new instance of the Crc32 class.

```csharp
public Crc32();
```

See Also

Crc32 Class | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
Crc32 Properties

The properties of the Crc32 class are listed below. For a complete list of Crc32 class members, see the Crc32 Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Returns the CRC32 data checksum computed so far.</td>
</tr>
</tbody>
</table>

See Also

Crc32 Class | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
Crc32.Value Property

Returns the CRC32 data checksum computed so far.

```csharp
public long Value {get; set;}
```

Implements

IChecksum.Value

See Also

Crc32 Class | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
Crc32 Methods

The methods of the Crc32 class are listed below. For a complete list of Crc32 class members, see the Crc32 Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets the CRC32 data checksum as if no update was ever called.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Overloaded. Adds the byte array to the data checksum.</td>
</tr>
</tbody>
</table>

See Also

Crc32 Class | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
Crc32.Reset Method

Resets the CRC32 data checksum as if no update was ever called.

```
public void Reset();
```

Implements

IChecksum.Reset

See Also

Crc32 Class | ISharpCode.SharpZipLib.Checksums Namespace
Crc32.Update Method

Updates the checksum with the bytes taken from the array.

Overload List

Updates the checksum with the bytes taken from the array.

  public void Update(byte[]):

Adds the byte array to the data checksum.

  public void Update(byte[], int, int):

Updates the checksum with the int bval.

  public void Update(int):

See Also

Crc32 Class | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
Crc32.Update Method (Byte[])  

Updates the checksum with the bytes taken from the array.

```csharp
public void Update(byte[] buffer);
```

Parameters

- **buffer**
  - buffer an array of bytes

Implements

- IChecksum.Update

See Also

ICSharpCode SharpZipLib Class Library
Crc32.Update Method (Byte[], Int32, Int32)

Adds the byte array to the data checksum.

```csharp
public void Update(
    byte[] buffer,
    int offset,
    int count
);
```

Parameters

- **buffer**
  - The buffer which contains the data

- **offset**
  - The offset in the buffer where the data starts

- **count**
  - The number of data bytes to update the CRC with.

Implements

- IChecksum.Update

See Also

ICSharpCode SharpZipLib Class Library
Crc32.Update Method (Int32)

Updates the checksum with the int bval.

```csharp
public void Update(
    int value
);
```

Parameters

- **value**
  the byte is taken as the lower 8 bits of value

Implements

- `IChecksum.Update`

See Also

ICSharpCode SharpZipLib Class Library
**IChecksum Interface**

Interface to compute a data checksum used by checked input/output streams. A data checksum can be updated by one byte or with a byte array. After each update the value of the current checksum can be returned by calling

`getValue`

The complete checksum object can also be reset so it can be used again with new data.

For a list of all members of this type, see **IChecksum Members**.

```plaintext
public interface IChecksum
```

**Types that implement IChecksum**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Adler32 | Computes Adler32 checksum for a stream of data. An Adler32 is not as reliable as a CRC32 checksum, but a lot faster to compute. The specification for Adler32 may be found in RFC 1950. ZLIB Compressed Data Format Specification version 3.3) From that document: (Adler-32 checksum) This contains a checksum value of the uncompressed data (excluding any dictionary data) computed according to the Adler-32 algorithm. This algorithm is a 32-bit extension and improvement of the Fletcher algorithm, used in the ITU-T X.224 / ISO 8073 standard. Adler-32 is composed of two sums accumulated per byte: s1 is the sum of all bytes, s2 is the sum of all s1 values. Both sums are done modulo 65521. s1 is initialized to 1, s2 to zero. The Adler-32 checksum is stored as s2*65536+s1 in most-significant-byte first (network) order. "8.2. The Adler-32 algorithm The Adler-32 algorithm is much faster than the CRC32, yet still provides an extremely low probability of undetected errors. If the modulo on unsigned long accumulators can be delayed for 5552 bytes, the modulo operation time is negligible. If the bytes are a, b, c, the new checksum is 3a + 2b + c + 3, and so is position and order sensitive. This is the sum, which is just a checksum. That 65521 is prime is important to avoid a possible large class of two-byte errors that leave the checksum unchanged. (The Fletcher checksum uses 255, which is not prime at all).
makes the Fletcher check insensitive to single byte changes. The sum \( s_1 \) is initialized to 1 instead of zero to make the length part of \( s_2 \), so that the length does not have to be checked separately. (Any sequence of zeroes has a Fletcher checksum of zero.)

**Crc32**

Generate a table for a byte-wise 32-bit CRC calculation on the polynomial:

\[ x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1. \]

Polynomials over \( \text{GF}(2) \) are represented in binary, one bit per coefficient, with the lowest powers in the most significant bit. Then addition is just exclusive-or, and multiplying a polynomial by \( x \) is a right shift by one. If we call the above polynomial \( p \), and represent a byte with the lowest power in the most significant bit (so polynomial \( x^7+x^3+x+1 \)), then the CRC is \((q \times x^{32}) \mod p\), meaning the remainder after dividing \( a \) by \( b \). This calculation is done using the shift-register method of multiplying and taking the remainder, initialized to zero, and for each incoming bit, \( x^{32} \) is added to the register if the bit is a one (where \( x^{32} \mod p = p+x^{32} \)), and the register is multiplied \( \mod p \) by \( x \) (which is shifting right one). We start with the highest power (least significant bit) of \( q \) and repeat for all eight bits, simply the CRC of all possible eight bit values. This is all the information needed to generate CRC's on data a byte at a time for all CRC register values and incoming bytes.

**StrangeCRC**

Bzip2 checksum algorithm

### Requirements

**Namespace:** [ICSharpCode.SharpZipLib.Checksums](#)

**Assembly:** ICSHarpCode.SharpZipLib (in ICSHarpCode.SharpZipLib.dll)

### See Also

[IChecksum Members](#) | [ICSharpCode.SharpZipLib.Checksums Namespace](#)
### IChecksum Members

**IChecksum overview**

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value</strong></td>
<td>Returns the data checksum computed so far.</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reset</strong></td>
<td>Resets the data checksum as if no update was ever called.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Overloaded. Adds the byte array to the data checksum.</td>
</tr>
</tbody>
</table>

### See Also

- [IChecksum Interface](#)
- [ICSharpCode.SharpZipLib.Checksums Namespace](#)
ICSharpCode SharpZipLib Class Library
IChecksum Properties

The properties of the IChecksum interface are listed below. For a complete list of IChecksum interface members, see the IChecksum Members topic.

Public Instance Properties

| Value | Returns the data checksum computed so far. |

See Also

IChecksum Interface | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
IChecksum.Value Property

Returns the data checksum computed so far.

```csharp
long Value {get;}
```

See Also

IChecksum Interface | ISharpCode.SharpZipLib.Checksums Namespace
IChecksum Methods

The methods of the IChecksum interface are listed below. For a complete list of IChecksum interface members, see the IChecksum Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>Resets the data checksum as if no update was ever called.</td>
</tr>
<tr>
<td>Update</td>
<td>Overloaded. Adds the byte array to the data checksum.</td>
</tr>
</tbody>
</table>

See Also

IChecksum Interface | ISharpCode.SharpZipLib.Checksums Namespace
**ICSharpCode SharpZipLib Class Library**
IChecksum.Reset Method

Resets the data checksum as if no update was ever called.

```csharp
void Reset();
```

See Also

[IChecksum Interface](#) | [ICSharpCode.SharpZipLib.Checksums Namespace](#)
ICSharpCode SharpZipLib Class Library
IChecksum.Update Method

Updates the data checksum with the bytes taken from the array.

Overload List

Updates the data checksum with the bytes taken from the array.

void Update(byte[]);

Adds the byte array to the data checksum.

void Update(byte[],int,int);

Adds one byte to the data checksum.

void Update(int);

See Also

IChecksum Interface | ISharpCode.SharpZipLib.Checksums
Namespace
ICSharpCode SharpZipLib Class Library
IChecksum.Update Method (Byte[])  

Updates the data checksum with the bytes taken from the array.

```csharp
void Update(
    byte[] buffer
);
```

**Parameters**

*buffer*

buffer an array of bytes

**See Also**

ICSharpCode SharpZipLib Class Library
IChecksum.Update Method (Byte[], Int32, Int32)

Adds the byte array to the data checksum.

```csharp
void Update(
    byte[] buffer,
    int offset,
    int count
);
```

Parameters

- `buffer`  
  The buffer which contains the data

- `offset`  
  The offset in the buffer where the data starts

- `count`  
  The number of data bytes to add.

See Also

- IChecksum Interface  
- IChecksum.Update Overload List
ICSharpCode SharpZipLib Class Library
IChecksum.Update Method (Int32)

Adds one byte to the data checksum.

```csharp
void Update(int value);
```

Parameters

- **value**
  the data value to add. The high byte of the int is ignored.

See Also

ICSharpCode SharpZipLib Class Library
StrangeCRC Class

Bzip2 checksum algorithm
For a list of all members of this type, see StrangeCRC Members.

System.Object

```
public class StrangeCRC : IChecksum
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

StrangeCRC Members | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
## StrangeCRC Members

### StrangeCRC overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>StrangeCRC Constructor</th>
<th>Initialise a default instance of StrangeCRC</th>
</tr>
</thead>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Value</th>
<th>Get the current Crc value.</th>
</tr>
</thead>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Equals (inherited from Object)</th>
<th>Determines whether the specified Object is equal to the current Object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>Reset</td>
<td>Reset the state of Crc.</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td>Update</td>
<td>Overloaded. Update Crc based on a portion of a block of data</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Finalize (inherited from Object)</th>
<th>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="MemberwiseClone" /> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

[StrangeCRC Class](#) | [ICSharpCode.SharpZipLib.Checksums Namespace](#)
ICSharpCode SharpZipLib Class Library
StrangeCRC Constructor

Initialise a default instance of StrangeCRC

public StrangeCRC();

See Also

StrangeCRC Class | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
StrangeCRC Properties

The properties of the **StrangeCRC** class are listed below. For a complete list of **StrangeCRC** class members, see the **StrangeCRC Members** topic.

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Get the current Crc value.</td>
</tr>
</tbody>
</table>

See Also

**StrangeCRC Class** | **ICSharpCode.SharpZipLib.Checksums Namespace**
ICSharpCode SharpZipLib Class Library
StrangeCRC.Value Property

Get the current Crc value.

```csharp
public long Value {get;}
```

Implements

IChecksum.Value

See Also

StrangeCRC Class | ICSharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
StrangeCRC Methods

The methods of the StrangeCRC class are listed below. For a complete list of StrangeCRC class members, see the StrangeCRC Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>(inherited from Object) Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>(inherited from Object) Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>(inherited from Object) Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Reset the state of Crc.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>(inherited from Object) Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Overloaded. Update Crc based on a portion of a block of data</td>
</tr>
</tbody>
</table>

Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>(inherited from Object) Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>(inherited from Object) Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

See Also

StrangeCRC Class | ISharpCode.SharpZipLib.Checksums Namespace
| ISharpCode SharpZipLib Class Library |
StrangeCRC.Reset Method

Reset the state of Crc.

```csharp
public void Reset();
```

Implements

[IChecksum.Reset](#)

See Also

StrangeCRC Class  |  ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
StrangeCRC.Update Method

Update Crc based on a block of data

Overload List

Update Crc based on a block of data

public void Update(byte[]);

Update Crc based on a portion of a block of data

public void Update(byte[],int,int);

Update the Crc value.

public void Update(int);

See Also

StrangeCRC Class | ISharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
StrangeCRC.Update Method (Byte[])  

Update Crc based on a block of data

```csharp
public void Update(byte[] buffer);
```

Implements

IChecksum.Update

See Also

ICSharpCode SharpZipLib Class Library
StrangeCRC.Update Method (Byte[], Int32, Int32)

Update Crc based on a portion of a block of data

```csharp
public void Update(
    byte[] buffer,
    int offset,
    int count
);
```

Parameters

- **buffer**
  - block of data

- **offset**
  - index of first byte to use

- **count**
  - number of bytes to use

Implements

- IChecksum.Update

See Also

ICSharpCode SharpZipLib Class Library
StrangeCRC.Update Method (Int32)

Update the Crc value.

```java
public void Update(int value);
```

Parameters

- `value` data update is based on

Implements

IChecksum.Update

See Also

StrangeCRC Class | ISharpCode.SharpZipLib.Checksums Namespace | StrangeCRC.Update Overload List
### ICSharpCode.SharpZipLib.Core Namespace

#### Namespace hierarchy

#### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DirectoryEventArgs</strong></td>
<td>Event arguments for directories.</td>
</tr>
<tr>
<td><strong>ExtendedPathFilter</strong></td>
<td>ExtendedPathFilter filters based on name, file size, and the last write time of the file.</td>
</tr>
<tr>
<td><strong>FileSystemScanner</strong></td>
<td>FileSystemScanner provides facilities scanning of files and directories.</td>
</tr>
<tr>
<td><strong>NameAndSizeFilter</strong></td>
<td><strong>Obsolete.</strong> NameAndSizeFilter filters based on name and file size.</td>
</tr>
<tr>
<td><strong>NameFilter</strong></td>
<td>NameFilter is a string matching class which allows for both positive and negative matching. A filter is a sequence of independant regular expressions separated by semicolons ';' Each expression can be prefixed by a plus '+' sign or a minus '-' sign to denote the expression is intended to include or exclude names. If neither a plus or minus sign is found include is the default A given name is tested for inclusion before checking exclusions. Only names matching an include spec and not matching an exclude spec are deemed to match the filter. An empty filter matches any</td>
</tr>
</tbody>
</table>
PathFilter filters directories and files using a form of regular expressions by full path name. See NameFilter for more detail on filtering.

**ScanEventArgs**
Event arguments for scanning.

**ScanFailureEventArgs**
Arguments passed when scan failures are detected.

**StreamUtils**
Provides simple Stream" utilities.

## Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INameTransform</strong></td>
<td>INameTransform defines how file system names are transformed for use with archives.</td>
</tr>
<tr>
<td><strong>IScanFilter</strong></td>
<td>Scanning filters support filtering of names.</td>
</tr>
</tbody>
</table>

## Delegates

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DirectoryFailureDelegate</strong></td>
<td>Delegate invoked when a directory failure is detected.</td>
</tr>
<tr>
<td><strong>FileFailureDelegate</strong></td>
<td>Delegate invoked when a file failure is detected.</td>
</tr>
<tr>
<td><strong>ProcessDirectoryDelegate</strong></td>
<td>Delegate invoked when a directory is processed.</td>
</tr>
<tr>
<td><strong>ProcessFileDelegate</strong></td>
<td>Delegate invoked when a file is processed.</td>
</tr>
</tbody>
</table>
DirectoryEventArgs Class

Event arguments for directories.

For a list of all members of this type, see DirectoryEventArgs Members.

System.Object    System.EventArgs
    ISharpCode.SharpZipLib.Core.ScanEventArgs

public class DirectoryEventArgs : ScanEventArgs

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

Namespace: ISharpCode.SharpZipLib.Core


See Also

ICSharpCode SharpZipLib Class Library
## DirectoryEventArgs Members

### DirectoryEventArgs overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContinueRunning</td>
<td>Get set a value indicating if scanning should continue or not.</td>
</tr>
<tr>
<td>HasMatchingFiles</td>
<td>Get a value indicating if the directory contains any matching files or not.</td>
</tr>
<tr>
<td>Name</td>
<td>The name for this event.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform</td>
</tr>
</tbody>
</table>
other cleanup operations before the **Object** is reclaimed by garbage collection.

| 📌 **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

[DirectoryEventArgs Class] | [ICSharpCode.SharpZipLib.Core Namespace]
ICSharpCode SharpZipLib Class Library
DirectoryEventArgs Constructor

Initialize an instance of DirectoryEventArgs.

```csharp
public DirectoryEventArgs(
    string name,
    bool hasMatchingFiles
);
```

**Parameters**

- **name**
  - The name for this directory.

- **hasMatchingFiles**
  - Flag value indicating if any matching files are contained in this directory.

**See Also**

- DirectoryEventArgs Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
The properties of the **DirectoryEventArgs** class are listed below. For a complete list of **DirectoryEventArgs** class members, see the [DirectoryEventArgs Members](#) topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ContinueRunning</strong></td>
<td>Get set a value indicating if scanning should continue or not.</td>
</tr>
<tr>
<td><strong>HasMatchingFiles</strong></td>
<td>Get a value indicating if the directory contains any matching files or not.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>The name for this event.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
**DirectoryEventArgs.HasMatchingFiles Property**

Get a value indicating if the directory contains any matching files or not.

```csharp
public bool HasMatchingFiles {get;}
```

See Also

**DirectoryFailureDelegate Delegate**

Delegate invoked when a directory failure is detected.

```csharp
public delegate void DirectoryFailureDelegate(
    object sender,
    ScanFailureEventArgs e
);
```

**Requirements**

**Namespace:** ICSharpCode.SharpZipLib.Core

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

**See Also**

ICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
ExtendedPathFilter Class

ExtendedPathFilter filters based on name, file size, and the last write time of the file.

For a list of all members of this type, see ExtendedPathFilter Members.

    ISharpCode.SharpZipLib.Core.ExtendedPathFilter

```csharp
public class ExtendedPathFilter : PathFilter
```

### Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

### Remarks

Provides an example of how to customise filtering.

### Requirements

**Namespace:** ISharpCode.SharpZipLib.Core

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

### See Also

ExtendedPathFilter Members  |  ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
### ExtendedPathFilter Members

#### ExtendedPathFilter overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ExtendedPathFilter</strong></td>
<td>Overloaded. Initializes a new instance of the ExtendedPathFilter class.</td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MaxDate</strong></td>
<td>Get/set the maximum DateTime value that will match for this filter.</td>
</tr>
<tr>
<td><strong>MaxSize</strong></td>
<td>Get/set the maximum size for a file that will match this filter.</td>
</tr>
<tr>
<td><strong>MinDate</strong></td>
<td>Get/set the minimum DateTime value that will match for this filter.</td>
</tr>
<tr>
<td><strong>MinSize</strong></td>
<td>Get/set the minimum size for a file that will match this filter.</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>IsMatch</strong></td>
<td>Test a filename to see if it matches the filter.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents</td>
</tr>
<tr>
<td>Protected Instance Methods</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

[ExtendedPathFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
**ExtendedPathFilter Constructor**

Initialise a new instance of ExtendedPathFilter.

**Overload List**

Initialise a new instance of ExtendedPathFilter.

    public ExtendedPathFilter(string, DateTime, DateTime);

Initialise a new instance of ExtendedPathFilter.

    public ExtendedPathFilter(string, long, long);

Initialise a new instance of ExtendedPathFilter.

    public ExtendedPathFilter(string, long, long, DateTime, DateTime);

**See Also**

[ExtendedPathFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ExtendedPathFilter Constructor (String, Int64, Int64)

Initialise a new instance of ExtendedPathFilter.

```csharp
public ExtendedPathFilter(
    string filter,
    long minSize,
    long maxSize
);
```

Parameters

- `filter`
  The filter to apply.

- `minSize`
  The minimum file size to include.

- `maxSize`
  The maximum file size to include.

See Also

- [ExtendedPathFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#) | [ExtendedPathFilter Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
ExtendedPathFilter Constructor (String, DateTime, DateTime)

Initialise a new instance of ExtendedPathFilter.

```csharp
public ExtendedPathFilter(
    string filter,
    DateTime minDate,
    DateTime maxDate
);
```

Parameters

* filter  
  The filter to apply.

* minDate  
  The minimum `DateTime` to include.

* maxDate  
  The maximum `DateTime` to include.

See Also

[ExtendedPathFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#) | [ExtendedPathFilter Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
ExtendedPathFilter Constructor (String, Int64, Int64, DateTime, DateTime)

Initialise a new instance of ExtendedPathFilter.

```java
public ExtendedPathFilter(
    string filter,
    long minSize,
    long maxSize,
    DateTime minDate,
    DateTime maxDate
);
```

Parameters

- **filter**
  The filter to apply.

- **minSize**
  The minimum file size to include.

- **maxSize**
  The maximum file size to include.

- **minDate**
  The minimum DateTime to include.

- **maxDate**
  The maximum DateTime to include.

See Also

- [ExtendedPathFilter Class](#)
- [ICSharpCode.SharpZipLib.Core Namespace](#)
- [ExtendedPathFilter Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
ExtendedPathFilter Properties

The properties of the ExtendedPathFilter class are listed below. For a complete list of ExtendedPathFilter class members, see the ExtendedPathFilter Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxDate</td>
<td>Get/set the maximum DateTime value that will match for this filter.</td>
</tr>
<tr>
<td>MaxSize</td>
<td>Get/set the maximum size for a file that will match this filter.</td>
</tr>
<tr>
<td>MinDate</td>
<td>Get/set the minimum DateTime value that will match for this filter.</td>
</tr>
<tr>
<td>MinSize</td>
<td>Get/set the minimum size for a file that will match this filter.</td>
</tr>
</tbody>
</table>

See Also

ExtendedPathFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
**ExtendedPathFilter.MaxDate Property**

Get/set the maximum `DateTime` value that will match for this filter.

```csharp
public System.DateTime MaxDate {get; set;}
```

**Remarks**

Files with a LastWrite time greater than this value are excluded by the filter.

**See Also**

[ExtendedPathFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
ExtendedPathFilter.MaxSize Property

Get/set the maximum size for a file that will match this filter.

```csharp
public long MaxSize {get; set;}
```

See Also

[ExtendedPathFilter Class](/ICSharpCode.SharpZipLib.Core/Namespace)
ICSharpCode SharpZipLib Class Library
ExtendedPathFilter.MinDate Property

Get/set the minimum DateTime value that will match for this filter.

```csharp
public System.DateTime MinDate {get; set;}
```

Remarks

Files with a LastWrite time less than this value are excluded by the filter.

See Also

[ExtendedPathFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
ExtendedPathFilter.MinSize Property

Get/set the minimum size for a file that will match this filter.

```csharp
public long MinSize {get; set;}
```

See Also

[ExtendedPathFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
The methods of the `ExtendedPathFilter` class are listed below. For a complete list of `ExtendedPathFilter` class members, see the `ExtendedPathFilter Members` topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>GetHashCode</code> (inherited from <code>Object</code>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><code>GetType</code> (inherited from <code>Object</code>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>IsMatch</code></td>
<td>Test a filename to see if it matches the filter.</td>
</tr>
<tr>
<td><code>ToString</code> (inherited from <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

### See Also

- `ExtendedPathFilter Class`
- `ICSharpCode.SharpZipLib.Core Namespace`
ICSharpCode SharpZipLib Class Library
ExtendedPathFilter.IsMatch Method

Test a filename to see if it matches the filter.

```csharp
public override bool IsMatch(string name);
```

Parameters

`name`  
The filename to test.

Return Value

True if the filter matches, false otherwise.

Implements

`IScanFilter.IsMatch`

See Also

`ExtendedPathFilter Class` | `ICSharpCode.SharpZipLib.Core Namespace`
ICSharpCode SharpZipLib Class Library
Delegate invoked when a file failure is detected.

```csharp
public delegate void FileFailureDelegate(
    object sender,
    ScanFailureEventArgs e
);
```

Requirements

- **Namespace**: ISharpCode.SharpZipLib.Core

See Also

- ISharpCode.SharpZipLib.Core Namespace
**FileSystemScanner Class**

FileSystemScanner provides facilities scanning of files and directories.

For a list of all members of this type, see FileSystemScanner Members.

**System.Object**


```csharp
public class FileSystemScanner
```

**Thread Safety**

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

**Requirements**

**Namespace:** ICSharpCode.SharpZipLib.Core

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

**See Also**

FileSystemScanner Members | ICSharpCode.SharpZipLib.Core Namespace
FileSystemScanner Members

FileSystemScanner overview

Public Instance Constructors

| FileSystemScanner | Overloaded. Initializes a new instance of the FileSystemScanner class. |

Public Instance Fields

| DirectoryFailure | Delegate to invoke when a directory failure is detected. |
| FileFailure      | Delegate to invoke when a file failure is detected. |
| ProcessDirectory | Delegate to invoke when a directory is processed. |
| ProcessFile      | Delegate to invoke when a file is processed. |

Public Instance Methods

<p>| Equals (inherited from Object) | Determines whether the specified Object is equal to the current Object. |
| GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| GetType (inherited from Object) | Gets the Type of the current instance. |
| OnDirectoryFailure | Raise the DirectoryFailure event. |
| OnFileFailure | Raise the FileFailure event. |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OnProcessFile</code></td>
<td>Raise the ProcessFile event.</td>
</tr>
<tr>
<td><code>Scan</code></td>
<td>Scan a directory.</td>
</tr>
<tr>
<td><code>ToString</code> (inherited from <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

## Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

### See Also

- [FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
**FileSystemScanner Constructor**

Initialise a new instance of **FileSystemScanner**

**Overload List**

Initialise a new instance of **FileSystemScanner**

```java
public FileSystemScanner(IScanFilter);
```

Initialise a new instance of **FileSystemScanner**

```java
public FileSystemScanner(IScanFilter, IScanFilter);
```

Initialise a new instance of **FileSystemScanner**

```java
public FileSystemScanner(string);
```

Initialise a new instance of **FileSystemScanner**

```java
public FileSystemScanner(string, string);
```

**See Also**

[FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
Initialise a new instance of `FileSystemScanner`

```
public FileSystemScanner(
    string filter
);
```

**Parameters**

`filter`

The `file filter` to apply when scanning.

**See Also**

- [FileSystemScanner Class](#)
- [ICSharpCode.SharpZipLib.Core Namespace](#)
- [FileSystemScanner Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
FileSystemScanner Constructor (String, String)

Initialise a new instance of FileSystemScanner

```csharp
public FileSystemScanner(
    string fileFilter,
    string directoryFilter
);
```

Parameters

*fileFilter*
The file filter to apply.

*directoryFilter*
The directory filter to apply.

See Also

ICSharpCode SharpZipLib Class Library
FileSystemScanner Constructor (IScanFilter)

Initialise a new instance of FileSystemScanner

```csharp
public FileSystemScanner(
    IScanFilter fileFilter
);
```

Parameters

**fileFilter**

The file IScanFilter filter to apply.

See Also

ICSharpCode SharpZipLib Class Library
Initialise a new instance of FileSystemScanner

```java
public FileSystemScanner(
    IScanFilter fileFilter,
    IScanFilter directoryFilter
);
```

Parameters

- **fileFilter**
  - The file IScanFilter filter to apply.

- **directoryFilter**
  - The directory IScanFilter filter to apply.

See Also

ICSharpCode SharpZipLib Class Library
**FileSystemScanner Fields**

The fields of the `FileSystemScanner` class are listed below. For a complete list of `FileSystemScanner` class members, see the `FileSystemScanner Members` topic.

### Public Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DirectoryFailure</code></td>
<td>Delegate to invoke when a directory failure is detected.</td>
</tr>
<tr>
<td><code>FileFailure</code></td>
<td>Delegate to invoke when a file failure is detected.</td>
</tr>
<tr>
<td><code>ProcessDirectory</code></td>
<td>Delegate to invoke when a directory is processed.</td>
</tr>
<tr>
<td><code>ProcessFile</code></td>
<td>Delegate to invoke when a file is processed.</td>
</tr>
</tbody>
</table>

**See Also**

- [FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
FileSystemScanner.DirectoryFailure Field

Delegate to invoke when a directory failure is detected.

```csharp
public DirectoryFailureDelegate DirectoryFailure;
```

See Also

[FileSystemScanner Class | ISharpCode.SharpZipLib.Core Namespace]
ICSharpCode SharpZipLib Class Library
Delegate to invoke when a file failure is detected.

```csharp
public FileFailureDelegate FileFailure;
```

See Also

[FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
Delegate to invoke when a directory is processed.

```csharp
```

See Also

- FileSystemScanner Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
**FileSystemScanner.ProcessFile Field**

Delegate to invoke when a file is processed.

```csharp
public ProcessFileDelegate ProcessFile;
```

See Also

[FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
**FileSystemScanner Methods**

The methods of the `FileSystemScanner` class are listed below. For a complete list of `FileSystemScanner` class members, see the `FileSystemScanner Members` topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified <a href="#">Object</a> is equal to the current <a href="#">Object</a>.</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <a href="#">Type</a> of the current instance.</td>
</tr>
<tr>
<td><code>OnDirectoryFailure</code></td>
<td>Raise the DirectoryFailure event.</td>
</tr>
<tr>
<td><code>OnFileFailure</code></td>
<td>Raise the FileFailure event.</td>
</tr>
<tr>
<td><code>OnProcessFile</code></td>
<td>Raise the ProcessFile event.</td>
</tr>
<tr>
<td><code>Scan</code></td>
<td>Scan a directory.</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Returns a <a href="#">String</a> that represents the current <a href="#">Object</a>.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the <a href="#">Object</a> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the</td>
</tr>
</tbody>
</table>
from Object) | current Object.

See Also

FileSystemScanner Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
Raise the DirectoryFailure event.

```csharp
public void OnDirectoryFailure(string directory, Exception e);
```

**Parameters**

- **directory**
  - The directory name.

- **e**
  - The exception detected.

**See Also**

- [FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
Raise the FileFailure event.

```csharp
public void OnFileFailure(
    string file,
    Exception e
);
```

**Parameters**

- **file**
  The file name.

- **e**
  The exception detected.

**See Also**

- [FileSystemScanner Class](#)
- [ICSharpCode.SharpZipLib.Core Namespace](#)
Raise the ProcessDirectory event.

```csharp
public void OnProcessDirectory(
    string directory,
    bool hasMatchingFiles
);
```

**Parameters**

- `directory`  
  The directory name.

- `hasMatchingFiles`  
  Flag indicating if the directory has matching files.

**See Also**

- [FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
Raise the ProcessFile event.

```csharp
public void OnProcessFile(string file);
```

Parameters

- **file**
  - The file name.

See Also

- [FileSystemScanner Class](#)
- [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
FileSystemScanner.Scan Method

Scan a directory.

```csharp
public void Scan(
    string directory,
    bool recurse
);
```

Parameters

- **directory**
  The base directory to scan.

- **recurse**
  True to recurse subdirectories, false to scan a single directory.

See Also

- FileSystemScanner Class | ISharpCode.SharpZipLib.Core
- Namespace
ICSharpCode SharpZipLib Class Library
**INameTransform Interface**

INameTransform defines how file system names are transformed for use with archives.

For a list of all members of this type, see [INameTransform Members](#).

```csharp
public interface INameTransform
```

**Types that implement INameTransform**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipNameTransform</td>
<td>ZipNameTransform transforms names as per the Zip file naming convention.</td>
</tr>
</tbody>
</table>

**Requirements**

**Namespace:** [ICSharpCode.SharpZipLib.Core](#)

**Assembly:** ICLSharpCode.SharpZipLib (in ICLSharpCode.SharpZipLib.dll)

**See Also**

[INameTransform Members](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
INameTransform Members

INameTransform overview

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransformDirectory</td>
<td>Given a directory name determine the transformed value.</td>
</tr>
<tr>
<td>TransformFile</td>
<td>Given a file name determine the transformed value.</td>
</tr>
</tbody>
</table>

See Also

INameTransform Interface | ICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
The methods of the `INameTransform` interface are listed below. For a complete list of `INameTransform` interface members, see the `INameTransform Members` topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TransformDirectory</code></td>
<td>Given a directory name determine the transformed value.</td>
</tr>
<tr>
<td><code>TransformFile</code></td>
<td>Given a file name determine the transformed value.</td>
</tr>
</tbody>
</table>

See Also

- `INameTransform Interface` | `ICSharpCode.SharpZipLib.Core Namespace`
ICSharpCode SharpZipLib Class Library
Given a directory name determine the transformed value.

```csharp
string TransformDirectory(string name);
```

**Parameters**

`name`  
The name to transform.

**Return Value**

The transformed directory name

**See Also**

INameTransform Interface | ISharpCode.SharpZipLib.Core Namespace
**INameTransform.TransformFile Method**

Given a file name determine the transformed value.

```csharp
string TransformFile(
    string name
);
```

**Parameters**

*name*

The name to transform.

**Return Value**

The transformed file name.

**See Also**

[INameTransform Interface](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
### IScanFilter Interface

Scanning filters support filtering of names.

For a list of all members of this type, see [IScanFilter Members](#).

---

**public interface IScanFilter**

### Types that implement IScanFilter

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ExtendedPathFilter</strong></td>
<td>ExtendedPathFilter filters based on name, file size, and the last write time of the file.</td>
</tr>
<tr>
<td><strong>NameAndSizeFilter</strong></td>
<td><strong>Obsolete.</strong> NameAndSizeFilter filters based on name and file size.</td>
</tr>
<tr>
<td><strong>NameFilter</strong></td>
<td>NameFilter is a string matching class which allows for both positive and negative matching. A filter is a sequence of independent regular expressions separated by semicolons ';'. Each expression can be prefixed by a plus '+' sign or a minus '-' sign to denote the expression is intended to include or exclude names. If neither a plus or minus sign is found include is the default. A given name is tested for inclusion before checking exclusions. Only names matching an include spec and not matching an exclude spec are deemed to match the filter. An empty filter matches any name.</td>
</tr>
</tbody>
</table>
PathFilter

PathFilter filters directories and files using a form of regular expressions by full path name. See NameFilter for more detail on filtering.

Requirements

Namespace: ISharpCode.SharpZipLib.Core


See Also

IScanFilter Members | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
IScanFilter Members

IScanFilter overview

Public Instance Methods

| `IsMatch` | Test a name to see if it 'matches' the filter. |

See Also

IScanFilter Interface | ICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
The methods of the IScanFilter interface are listed below. For a complete list of IScanFilter interface members, see the IScanFilter Members topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsMatch</td>
<td>Test a name to see if it 'matches' the filter.</td>
</tr>
</tbody>
</table>

See Also

ICanFilter Interface | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
IScanFilter.IsMatch Method

Test a name to see if it 'matches' the filter.

```csharp
bool IsMatch(
    string name
);
```

Parameters

- `name`:
  The name to test.

Return Value

Returns true if the name matches the filter, false if it does not match.

See Also

IScanFilter Interface | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
NameAndSizeFilter Class

NOTE: This class is now obsolete.
Use ExtendedPathFilter instead

NameAndSizeFilter filters based on name and file size.
For a list of all members of this type, see NameAndSizeFilter Members.

ISharpCode.SharpZipLib.Core.NameAndSizeFilter

```csharp
public class NameAndSizeFilter : PathFilter
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Remarks

A sample showing how filters might be extended.

Requirements

Namespace: ISharpCode.SharpZipLib.Core

See Also

NameAndSizeFilter Members  |  ISharpCode.SharpZipLib.Core Namespace
# NameAndSizeFilter Members

## NameAndSizeFilter overview

## Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameAndSizeFilter Constructor</td>
<td>Initialise a new instance of NameAndSizeFilter.</td>
</tr>
</tbody>
</table>

## Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSize</td>
<td>Get/set the maximum size for a file that will match this filter.</td>
</tr>
<tr>
<td>MinSize</td>
<td>Get/set the minimum size for a file that will match this filter.</td>
</tr>
</tbody>
</table>

## Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>IsMatch</td>
<td>Test a filename to see if it matches the filter.</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

## Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by</td>
</tr>
</tbody>
</table>
garbage collection.

| MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

See Also

NameAndSizeFilter Class | ICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
Initialise a new instance of NameAndSizeFilter.

```csharp
public NameAndSizeFilter(
    string filter,
    long minSize,
    long maxSize
);
```

**Parameters**

- `filter`
  - The filter to apply.

- `minSize`
  - The minimum file size to include.

- `maxSize`
  - The maximum file size to include.

**See Also**

- NameAndSizeFilter Class | ISharpCode.ZipLib.Core Namespace
NameAndSizeFilter Properties

The properties of the NameAndSizeFilter class are listed below. For a complete list of NameAndSizeFilter class members, see the NameAndSizeFilter Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MaxSize</strong></td>
<td>Get/set the maximum size for a file that will match this filter.</td>
</tr>
<tr>
<td><strong>MinSize</strong></td>
<td>Get/set the minimum size for a file that will match this filter.</td>
</tr>
</tbody>
</table>

See Also

NameAndSizeFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
NameAndSizeFilter.MaxSize Property

Get/set the maximum size for a file that will match this filter.

```csharp
public long MaxSize {get; set;}
```

See Also

NameAndSizeFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
NameAndSizeFilter.MinSize Property

Get/set the minimum size for a file that will match this filter.

```csharp
public long MinSize {get; set;}
```

See Also

[NameAndSizeFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
NameAndSizeFilter Methods

The methods of the NameAndSizeFilter class are listed below. For a complete list of NameAndSizeFilter class members, see the NameAndSizeFilter Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>IsMatch</strong></td>
<td>Test a filename to see if it matches the filter.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

NameAndSizeFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
NameAndSizeFilter.IsMatch Method

Test a filename to see if it matches the filter.

```csharp
public override bool IsMatch(string name);
```

Parameters

- `name`  
  The filename to test.

Return Value

True if the filter matches, false otherwise.

Implements

- `IScanFilter.IsMatch`

See Also

- `NameAndSizeFilter Class`  
  `ICSharpCode.SharpZipLib.Core Namespace`
ICSharpCode SharpZipLib Class Library
NameFilter Class

NameFilter is a string matching class which allows for both positive and negative matching. A filter is a sequence of independant regular expressions separated by semi-colons ';'. Each expression can be prefixed by a plus '+' sign or a minus '-' sign to denote the expression is intended to include or exclude names. If neither a plus or minus sign is found include is the default. A given name is tested for inclusion before checking exclusions. Only names matching an include spec and not matching an exclude spec are deemed to match the filter. An empty filter matches any name.

For a list of all members of this type, see NameFilter Members.


public class NameFilter : IScanFilter

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Example

The following expression includes all name ending in '.dat' with the exception of 'dummy.dat' "+\.dat$;-\^dummy\.dat$"

Requirements


See Also

NameFilter Members | ICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
## NameFilter Members

### NameFilter overview

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsValidExpression</td>
<td>Test a string to see if it is a valid regular expression.</td>
</tr>
<tr>
<td>IsValidFilterExpression</td>
<td>Test an expression to see if it is valid as a filter.</td>
</tr>
</tbody>
</table>

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameFilter Constructor</td>
<td>Construct an instance based on the filter expression passed</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>IsExcluded</td>
<td>Test a value to see if it is excluded by the filter.</td>
</tr>
<tr>
<td>IsIncluded</td>
<td>Test a value to see if it is included by the filter.</td>
</tr>
<tr>
<td>IsMatch</td>
<td>Test a value to see if it matches the filter.</td>
</tr>
<tr>
<td>ToString</td>
<td>Convert this filter to its string equivalent.</td>
</tr>
</tbody>
</table>
Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

[NameFilter Class] | [ICSharpCode.SharpZipLib.Core Namespace]
ICSharpCode SharpZipLib Class Library
NameFilter Constructor

Construct an instance based on the filter expression passed

```csharp
public NameFilter(
    string filter
);
```

Parameters

- `filter`  
  The filter expression.

See Also

- [NameFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
**NameFilter Methods**

The methods of the `NameFilter` class are listed below. For a complete list of `NameFilter` class members, see the [NameFilter Members](#) topic.

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsValidExpression</code></td>
<td>Test a string to see if it is a valid regular expression.</td>
</tr>
<tr>
<td><code>IsValidFilterExpression</code></td>
<td>Test an expression to see if it is valid as a filter.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>GetHashCode</code> (inherited from <code>Object</code>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><code>GetType</code> (inherited from <code>Object</code>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>IsExcluded</code></td>
<td>Test a value to see if it is excluded by the filter.</td>
</tr>
<tr>
<td><code>IsIncluded</code></td>
<td>Test a value to see if it is included by the filter.</td>
</tr>
<tr>
<td><code>IsMatch</code></td>
<td>Test a value to see if it matches the filter.</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Convert this filter to its string equivalent.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform</td>
</tr>
</tbody>
</table>
other cleanup operations before the **Object** is reclaimed by garbage collection.

| ⚡ **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

[NameFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
NameFilter.IsExcluded Method

Test a value to see if it is excluded by the filter.

```csharp
public bool IsExcluded(
    string name
);
```

Parameters

- `name` The value to test.

Return Value

True if the value is excluded, false otherwise.

See Also

- [NameFilter Class](#)
- [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
NameFilter.IsIncluded Method

Test a value to see if it is included by the filter.

```csharp
public bool IsIncluded(
    string name
);
```

Parameters

*name*

The value to test.

Return Value

True if the value is included, false otherwise.

See Also

NameFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
NameFilter.IsMatch Method

Test a value to see if it matches the filter.

```csharp
public bool IsMatch(
    string name
);
```

Parameters

- `name`:
  The value to test.

Return Value

True if the value matches, false otherwise.

Implements

- IScanFilter.IsMatch

See Also

- NameFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
NameFilter.IsValidExpression Method

Test a string to see if it is a valid regular expression.

```csharp
public static bool IsValidExpression(
        string expression
    );
```

Parameters

`expression`

The expression to test.

Return Value

True if expression is a valid Regex false otherwise.

See Also

[NameFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
NameFilter.IsValidFilterExpression Method

Test an expression to see if it is valid as a filter.

```csharp
public static bool IsValidFilterExpression(string toTest);
```

Parameters

toTest
The filter expression to test.

Return Value
True if the expression is valid, false otherwise.

See Also

NameFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
NameFilter.ToString Method

Convert this filter to its string equivalent.

```csharp
public override string ToString();
```

Return Value

The string equivalent for this filter.

See Also

[NameFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
PathFilter Class

PathFilter filters directories and files using a form of regular expressions by full path name. See NameFilter for more detail on filtering.

For a list of all members of this type, see PathFilter Members.

  ISharpCode.SharpZipLib.Core.ExtendedPathFilter
  ISharpCode.SharpZipLib.Core.NameAndSizeFilter

public class PathFilter : IScanFilter

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

Namespace: ISharpCode.SharpZipLib.Core

See Also

PathFilter Members  |  ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
# PathFilter Members

## PathFilter overview

## Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathFilter Constructor</td>
<td>Initialise a new instance of PathFilter.</td>
</tr>
</tbody>
</table>

## Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>IsMatch</td>
<td>Test a name to see if it matches the filter.</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

## Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

## See Also

PathFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
PathFilter Constructor

Initialise a new instance of PathFilter.

```csharp
public PathFilter(string filter);
```

Parameters

*filter*

The NameFilter filter expression to apply.

See Also

PathFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
PathFilter Methods

The methods of the PathFilter class are listed below. For a complete list of PathFilter class members, see the PathFilter Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>IsMatch</strong></td>
<td>Test a name to see if it matches the filter.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

See Also

PathFilter Class | ICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
PathFilter.IsMatch Method

Test a name to see if it matches the filter.

```csharp
public virtual bool IsMatch(
    string name
);
```

Parameters

- **name**
  The name to test.

Return Value

- True if the name matches, false otherwise.

Implements

- IScanFilter.IsMatch

See Also

- PathFilter Class | ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
Delegate invoked when a directory is processed.

```csharp
public delegate void ProcessDirectoryDelegate(
    object sender,
    DirectoryEventArgs e
);
```

Requirements

**Namespace:** [ICSharpCode.SharpZipLib.Core](#)

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

See Also

[ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
Delegate invoked when a file is processed.

```csharp
public delegate void ProcessFileDelegate(
    object sender,
    ScanEventArgs e
);
```

Requirements

**Namespace:** IICSharpCode.SharpZipLib.Core

**Assembly:** IICSharpCode.SharpZipLib (in IICSharpCode.SharpZipLib.dll)

See Also

IICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
**ScanEventArgs Class**

Event arguments for scanning.

For a list of all members of this type, see [ScanEventArgs Members](#).

System.Object  System.EventArgs  
ICSharpCode.SharpZipLib.Core.ScanEventArgs  

```csharp
public class ScanEventArgs : EventArgs
```

**Thread Safety**

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** ISharpCode.SharpZipLib.Core

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

**See Also**

[ScanEventArgs Members](#)  |  ISharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
### ScanEventArgs Members

#### ScanEventArgs overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ScanEventArgs Constructor</strong></td>
<td>Initialise a new instance of <strong>ScanEventArgs</strong></td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ContinueRunning</strong></td>
<td>Get set a value indicating if scanning should continue or not.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>The name for this event.</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

#### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited)</td>
<td>Creates a shallow copy of the <strong>Object</strong>.</td>
</tr>
</tbody>
</table>
from Object)               current Object.

See Also

ScanEventArgs Class | ICSharpCode.SharpZipLib.Core Namespace
ScanEventArgs Constructor

Initialise a new instance of `ScanEventArgs`

```csharp
public ScanEventArgs(
    string name
);
```

Parameters

* name

See Also

`ScanEventArgs Class` | `ICSharpCode.SharpZipLib.Core Namespace`
ICSharpCode SharpZipLib Class Library
ScanEventArgs Properties

The properties of the `ScanEventArgs` class are listed below. For a complete list of `ScanEventArgs` class members, see the `ScanEventArgs Members` topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ContinueRunning</code></td>
<td>Get set a value indicating if scanning should continue or not.</td>
</tr>
<tr>
<td><code>Name</code></td>
<td>The name for this event.</td>
</tr>
</tbody>
</table>

See Also

[ScanEventArgs Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
ScanEventArgs.ContinueRunning Property

Get set a value indicating if scanning should continue or not.

```csharp
public bool ContinueRunning {get; set;}
```

See Also

ScanEventArgs Class | ISharpCode.SharpZipLib.Core Namespace
ScanEventArgs.Name Property

The name for this event.

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

See Also

ScanEventArgs Class | ICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
ScanFailureEventArgs Class

Arguments passed when scan failures are detected.
For a list of all members of this type, see ScanFailureEventArgs Members.

System.Object
ICSharpCode.SharpZipLib.Core.ScanFailureEventArgs

public class ScanFailureEventArgs

Thread Safety
Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements
Namespace: ICSHarpCode.SharpZipLib.Core

See Also
ScanFailureEventArgs Members | ICSHarpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
## ScanFailureEventArgs Members

### ScanFailureEventArgs overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScanFailureEventArgs Constructor</td>
<td>Initialise a new instance of ScanFailureEventArgs</td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContinueRunning</td>
<td>Get / set a value indicating whether scanning should continue.</td>
</tr>
<tr>
<td>Exception</td>
<td>The applicable exception.</td>
</tr>
<tr>
<td>Name</td>
<td>The applicable name.</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

#### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
</tbody>
</table>
**MemberwiseClone** (inherited from **Object**)  |  Creates a shallow copy of the current **Object**.

**See Also**

[ScanFailureEventArgs Class]  |  [ICSharpCode.SharpZipLib.Core Namespace]
ScanFailureEventArgs Constructor

Initialise a new instance of `ScanFailureEventArgs`

```csharp
public ScanFailureEventArgs(
    string name,
    Exception e
);
```

Parameters

- `name`
  - The name to apply.
- `e`
  - The exception to use.

See Also

- [ScanFailureEventArgs Class](#)
- [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
ScanFailureEventArgs Properties

The properties of the `ScanFailureEventArgs` class are listed below. For a complete list of `ScanFailureEventArgs` class members, see the `ScanFailureEventArgs Members` topic.

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ContinueRunning</code></td>
<td>Get / set a value indicating whether scanning should continue.</td>
</tr>
<tr>
<td><code>Exception</code></td>
<td>The applicable exception.</td>
</tr>
<tr>
<td><code>Name</code></td>
<td>The applicable name.</td>
</tr>
</tbody>
</table>

See Also

[ScanFailureEventArgs Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ScanFailureEventArgs.ContinueRunning Property

Get / set a value indicating whether scanning should continue.

```csharp
public bool ContinueRunning {get; set;}
```

See Also

- [ScanFailureEventArgs Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
ScanFailureEventArgs.Exception Property

The applicable exception.

```csharp
public System.Exception Exception {get;}
```

See Also

ScanFailureEventArgs Class | ICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
ScanFailureEventArgs.Name Property

The applicable name.

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

See Also

[ScanFailureEventArgs Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
StreamUtils Class

Provides simple Stream utilities.
For a list of all members of this type, see StreamUtils Members.


public sealed class StreamUtils

Thread Safety
Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements
Namespace: ISharpCode.SharpZipLib.Core

See Also
StreamUtils Members | ISharpCode.SharpZipLib.Core Namespace
### StreamUtils Members

#### StreamUtils overview

**Public Static Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Copy</strong></td>
<td>Copy the contents of one Stream to another.</td>
</tr>
<tr>
<td><strong>ReadFully</strong></td>
<td>Overloaded. Read from a Stream ensuring all the required data is read.</td>
</tr>
</tbody>
</table>

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

**See Also**

[StreamUtils Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
StreamUtils Methods

The methods of the **StreamUtils** class are listed below. For a complete list of **StreamUtils** class members, see the **StreamUtils Members** topic.

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Copy</strong></td>
<td>Copy the contents of one <strong>Stream</strong> to another.</td>
</tr>
<tr>
<td><strong>ReadFully</strong></td>
<td>Overloaded. Read from a <strong>Stream</strong> ensuring all the required data is read.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

- [StreamUtils Class](#) | [ICSSharpCode.SharpZipLib.Core Namespace](#)
ICSharpCode SharpZipLib Class Library
StreamUtils.Copy Method

Copy the contents of one Stream to another.

```csharp
public static void Copy(
    Stream source,
    Stream destination,
    byte[] buffer
);
```

Parameters

- `source`
  The stream to source data from.

- `destination`
  The stream to write data to.

- `buffer`
  The buffer to use during copying.

See Also

- StreamUtils Class
- ISharpCode.SharpZipLib.Core Namespace
StreamUtils.ReadFully Method

Read from a Stream ensuring all the required data is read.

Overload List

Read from a Stream ensuring all the required data is read.

    public static void ReadFully(Stream,byte[]);

Read from a Stream ensuring all the required data is read.

    public static void ReadFully(Stream,byte[],int,int);

See Also

StreamUtils Class | ICSharpCode.SharpZipLib.Core Namespace
ICSharpCode SharpZipLib Class Library
StreamUtils.ReadFully Method (Stream, Byte[])  
Read from a Stream ensuring all the required data is read.

```csharp
public static void ReadFully(
    Stream stream,
    byte[] buffer
);
```

Parameters

- **stream**
  - The stream to read.

- **buffer**
  - The buffer to fill.

See Also

ICSharpCode SharpZipLib Class Library
StreamUtils.ReadFully Method (Stream, Byte[], Int32, Int32)

Read from a Stream ensuring all the required data is read.

```csharp
public static void ReadFully(
    Stream stream,
    byte[] buffer,
    int offset,
    int count
);
```

Parameters

- **stream**
  The stream to read data from.

- **buffer**
  The buffer to store data in.

- **offset**
  The offset at which to begin storing data.

- **count**
  The number of bytes of data to store.

See Also

- [StreamUtils Class](#)
- [ICSharpCode.SharpZipLib.Core Namespace](#)
- [StreamUtils.ReadFully Overload List](#)
ICSharpCode SharpZipLib Class Library
### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PkzipClassic</strong></td>
<td>PkzipClassic embodies the classic or original encryption facilities used in Pkzip archives. While it has been superceded by more recent and more powerful algorithms, its still in use and is viable for preventing casual snooping.</td>
</tr>
<tr>
<td><strong>PkzipClassicManaged</strong></td>
<td>Defines a wrapper object to access the Pkzip algorithm. This class cannot be inherited.</td>
</tr>
</tbody>
</table>
ICSharpCode SharpZipLib Class Library
**PkzipClassic Class**

PkzipClassic embodies the classic or original encryption facilities used in Pkzip archives. While it has been superceded by more recent and more powerful algorithms, it's still in use and is viable for preventing casual snooping.

For a list of all members of this type, see [PkzipClassic Members](#).

```csharp
public abstract class PkzipClassic : SymmetricAlgorithm
```

**Thread Safety**

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** [ICSharpCode.SharpZipLib.Encryption](#)

**Assembly:** IICSharpCode.SharpZipLib (in IICSharpCode.SharpZipLib.dll)

**See Also**

ICSharpCode SharpZipLib Class Library
# PkzipClassic Members

## PkzipClassic overview

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateKeys</td>
<td>Generates new encryption keys based on given seed</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BlockSize</strong></td>
<td>(inherited from SymmetricAlgorithm) Gets or sets the block size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td><strong>FeedbackSize</strong></td>
<td>(inherited from SymmetricAlgorithm) Gets or sets the feedback size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td><strong>IV</strong></td>
<td>(inherited from SymmetricAlgorithm) Gets or sets the initialization vector (IV) for the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>(inherited from SymmetricAlgorithm) Gets or sets the secret key for the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>KeySize</strong></td>
<td>(inherited from SymmetricAlgorithm) Gets or sets the size of the secret key used by the symmetric algorithm in bits.</td>
</tr>
<tr>
<td><strong>LegalBlockSizes</strong></td>
<td>(inherited from SymmetricAlgorithm) Gets the block sizes that are supported by the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>LegalKeySizes</strong></td>
<td>(inherited from SymmetricAlgorithm) Gets the key sizes that are supported by the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>Mode</strong></td>
<td>(inherited from SymmetricAlgorithm) Gets or sets the mode for operation of the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>Padding</strong></td>
<td>(inherited from SymmetricAlgorithm) Gets or sets the padding mode used in the symmetric algorithm.</td>
</tr>
</tbody>
</table>
Public Instance Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✷ <strong>Clear</strong></td>
<td>Releases all resources used by the <strong>SymmetricAlgorithm</strong>.</td>
</tr>
<tr>
<td>✷ <strong>CreateDecryptor</strong></td>
<td>Overloaded. When overridden in a derived class, creates a symmetric decryptor object with the specified <strong>Key</strong> and initialization vector (<strong>IV</strong>).</td>
</tr>
<tr>
<td>✷ <strong>CreateEncryptor</strong></td>
<td>Overloaded. When overridden in a derived class, creates a symmetric encryptor object with the specified <strong>Key</strong> and initialization vector (<strong>IV</strong>).</td>
</tr>
<tr>
<td>✷ <strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>✷ <strong>GenerateIV</strong></td>
<td>When overridden in a derived class, generates a random initialization vector (<strong>IV</strong>) to be used for the algorithm.</td>
</tr>
<tr>
<td>✷ <strong>GenerateKey</strong></td>
<td>When overridden in a derived class, generates a random <strong>Key</strong> to be used for the algorithm.</td>
</tr>
<tr>
<td>✷ <strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>✷ <strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>✷ <strong>ToString</strong></td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>✷ <strong>ValidKeySize</strong></td>
<td>Determines whether the</td>
</tr>
<tr>
<td>SymmetricAlgorithm</td>
<td>specified key size is valid for the current algorithm.</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------</td>
</tr>
</tbody>
</table>

**Protected Instance Constructors**

<table>
<thead>
<tr>
<th><img src="image" alt="Constructor" /></th>
<th>PkzipClassic Constructor</th>
<th>Initializes a new instance of the PkzipClassic class.</th>
</tr>
</thead>
</table>

**Protected Instance Fields**

<table>
<thead>
<tr>
<th><img src="image" alt="Field" /></th>
<th>BlockSizeValue (inherited from SymmetricAlgorithm)</th>
<th>Represents the block size of the cryptographic operation in bits.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Field" /></td>
<td>FeedbackSizeValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the feedback size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td><img src="image" alt="Field" /></td>
<td>IVValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the initialization vector (IV) for the symmetric algorithm.</td>
</tr>
<tr>
<td><img src="image" alt="Field" /></td>
<td>KeySizeValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the size of the secret key used by the symmetric algorithm in bits.</td>
</tr>
<tr>
<td><img src="image" alt="Field" /></td>
<td>KeyValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the secret key for the symmetric algorithm.</td>
</tr>
<tr>
<td><img src="image" alt="Field" /></td>
<td>LegalBlockSizesValue (inherited from SymmetricAlgorithm)</td>
<td>Specifies the block sizes that are supported by the symmetric algorithm.</td>
</tr>
<tr>
<td><img src="image" alt="Field" /></td>
<td>LegalKeySizesValue (inherited from SymmetricAlgorithm)</td>
<td>Specifies the key sizes that are supported by the symmetric algorithm.</td>
</tr>
<tr>
<td><img src="image" alt="Field" /></td>
<td>ModeValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the cipher mode used in the symmetric algorithm.</td>
</tr>
<tr>
<td><img src="image" alt="Field" /></td>
<td>PaddingValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the padding mode used in the symmetric algorithm.</td>
</tr>
</tbody>
</table>
Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dispose</strong></td>
<td>Releases the unmanaged resources used by the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.symmetricsymmetricalgorithm">SymmetricAlgorithm</a> and optionally releases the managed resources.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Releases the unmanaged resources used by the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.symmetricsymmetricalgorithm">SymmetricAlgorithm</a>.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.</td>
</tr>
</tbody>
</table>

See Also


---

For more information, refer to the [PKZipClassic Class](https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.pkzipclassic) and the [ICSharpCode.SharpZipLib.Encryption Namespace](https://icsharpcode.net/compactzip/framework/).
PkzipClassic Constructor

Initializes a new instance of the PkzipClassic class.

```java
protected PkzipClassic();
```

See Also

- PkzipClassic Class
- ISharpCode.SharpZipLib.Encryption Namespace
ICSharpCode SharpZipLib Class Library
The methods of the **PkzipClassic** class are listed below. For a complete list of **PkzipClassic** class members, see the **PkzipClassic Members** topic.

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GenerateKeys</strong></td>
<td>Generates new encryption keys based on given seed</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clear</strong></td>
<td>Releases all resources used by the <strong>SymmetricAlgorithm</strong>.</td>
</tr>
<tr>
<td><strong>CreateDecryptor</strong></td>
<td>Overloaded. When overridden in a derived class, creates a symmetric decryptor object with the specified <strong>Key</strong> and initialization vector (<strong>IV</strong>).</td>
</tr>
<tr>
<td><strong>CreateEncryptor</strong></td>
<td>Overloaded. When overridden in a derived class, creates a symmetric encryptor object with the specified <strong>Key</strong> and initialization vector (<strong>IV</strong>).</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GenerateIV</strong></td>
<td>When overridden in a derived class, generates a random initialization vector (<strong>IV</strong>) to be used for the algorithm.</td>
</tr>
<tr>
<td><strong>GenerateKey</strong></td>
<td>When overridden in a derived class, generates a random <strong>Key</strong> to be used for the algorithm.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use.</td>
</tr>
</tbody>
</table>
in hashing algorithms and data structures like a hash table.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Get Type</strong> (inherited from Object)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>To S t r i n g</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Valid Key Size</strong> (inherited from SymmetricAlgorithm)</td>
<td>Determines whether the specified key size is valid for the current algorithm.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dispose</strong> (inherited from SymmetricAlgorithm)</td>
<td>Releases the unmanaged resources used by the <strong>SymmetricAlgorithm</strong> and optionally releases the managed resources.</td>
</tr>
<tr>
<td><strong>Finalize</strong> (inherited from SymmetricAlgorithm)</td>
<td>Releases the unmanaged resources used by the <strong>SymmetricAlgorithm</strong>.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
Generates new encryption keys based on given seed

```csharp
public static byte[] GenerateKeys(byte[] seed);
```

See Also

ICSharpCode SharpZipLib Class Library
**PkzipClassicManaged Class**

Defines a wrapper object to access the Pkzip algorithm. This class cannot be inherited.

For a list of all members of this type, see [PkzipClassicManaged Members](#).

```csharp
```

```csharp
public sealed class PkzipClassicManaged :
    PkzipClassic
```

**Thread Safety**

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** ISharpCode.SharpZipLib.Encryption

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

**See Also**

- [PkzipClassicManaged Members](#)
- [ICSharpCode.SharpZipLib.Encryption Namespace](#)
ICSharpCode SharpZipLib Class Library
**PkzipClassicManaged Members**

**PkzipClassicManaged overview**

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PkzipClassicManaged Constructor</strong></td>
<td>Initializes a new instance of the PkzipClassicManaged class.</td>
</tr>
</tbody>
</table>

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BlockSize</strong></td>
<td>Get / set the applicable block size in bits.</td>
</tr>
<tr>
<td><strong>FeedbackSize</strong> (inherited from SymmetricAlgorithm)</td>
<td>Gets or sets the feedback size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td><strong>IV</strong> (inherited from SymmetricAlgorithm)</td>
<td>Gets or sets the initialization vector (IV) for the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>Get / set the key value applicable.</td>
</tr>
<tr>
<td><strong>KeySize</strong> (inherited from SymmetricAlgorithm)</td>
<td>Gets or sets the size of the secret key used by the symmetric algorithm in bits.</td>
</tr>
<tr>
<td><strong>LegalBlockSizes</strong></td>
<td>Get an array of legal block sizes.</td>
</tr>
<tr>
<td><strong>LegalKeySizes</strong></td>
<td>Get an array of legal key sizes.</td>
</tr>
<tr>
<td><strong>Mode</strong> (inherited from SymmetricAlgorithm)</td>
<td>Gets or sets the mode for operation of the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>Padding</strong> (inherited from SymmetricAlgorithm)</td>
<td>Gets or sets the padding mode used in the symmetric algorithm.</td>
</tr>
</tbody>
</table>

**Public Instance Methods**
<table>
<thead>
<tr>
<th>Method (inherited from SymmetricAlgorithm)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clear</strong></td>
<td>Releases all resources used by the <a href="#">SymmetricAlgorithm</a>.</td>
</tr>
<tr>
<td><strong>CreateDecryptor</strong></td>
<td>Overloaded. Create a decryptor.</td>
</tr>
<tr>
<td><strong>CreateDecryptor</strong> (inherited from SymmetricAlgorithm)</td>
<td>Overloaded. Creates a symmetric decryptor object with the current <a href="#">Key</a> and initialization vector (<a href="#">IV</a>).</td>
</tr>
<tr>
<td><strong>CreateEncryptor</strong></td>
<td>Overloaded. Create an encryptor.</td>
</tr>
<tr>
<td><strong>CreateEncryptor</strong> (inherited from SymmetricAlgorithm)</td>
<td>Overloaded. Creates a symmetric encryptor object with the current <a href="#">Key</a> and initialization vector (<a href="#">IV</a>).</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified <a href="#">Object</a> is equal to the current <a href="#">Object</a>.</td>
</tr>
<tr>
<td><strong>GenerateIV</strong></td>
<td>Generate an initial vector.</td>
</tr>
<tr>
<td><strong>GenerateKey</strong></td>
<td>Generate a new random key.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the <a href="#">Type</a> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <a href="#">String</a> that represents the current <a href="#">Object</a>.</td>
</tr>
<tr>
<td><strong>ValidKeySize</strong> (inherited from SymmetricAlgorithm)</td>
<td>Determines whether the specified key size is valid for the current algorithm.</td>
</tr>
</tbody>
</table>

See Also

[![PkzipClassicManaged Class](https://i.imgur.com/3z5y.png)](https://i.imgur.com/3z5y.png) | ![ICSharpCode.SharpZipLib.Encryption Namespace](https://i.imgur.com/3z5y.png) |
ICSharpCode SharpZipLib Class Library
**PkzipClassicManaged Constructor**

Initializes a new instance of the `PkzipClassicManaged` class.

```csharp
public PkzipClassicManaged();
```

See Also

[PkzipClassicManaged Class](#) | [ICSharpCode.SharpZipLib.Encryption Namespace](#)
ICSharpCode SharpZipLib Class Library
The properties of the PkzipClassicManaged class are listed below. For a complete list of PkzipClassicManaged class members, see the PkzipClassicManaged Members topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockSize</td>
<td>Get / set the applicable block size in bits.</td>
</tr>
<tr>
<td>FeedbackSize</td>
<td>Gets or sets the feedback size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td>IV</td>
<td>Gets or sets the initialization vector (IV) for the symmetric algorithm.</td>
</tr>
<tr>
<td>Key</td>
<td>Get / set the key value applicable.</td>
</tr>
<tr>
<td>KeySize</td>
<td>Gets or sets the size of the secret key used by the symmetric algorithm in bits.</td>
</tr>
<tr>
<td>LegalBlockSizes</td>
<td>Get an array of legal block sizes.</td>
</tr>
<tr>
<td>LegalKeySizes</td>
<td>Get an array of legal key sizes.</td>
</tr>
<tr>
<td>Mode</td>
<td>Gets or sets the mode for operation of the symmetric algorithm.</td>
</tr>
<tr>
<td>Padding</td>
<td>Gets or sets the padding mode used in the symmetric algorithm.</td>
</tr>
</tbody>
</table>

See Also

- PkzipClassicManaged Class
ICSharpCode SharpZipLib Class Library
**PkgzipClassicManaged.BlockSize Property**

Get / set the applicable block size in bits.

```csharp
public override int BlockSize {get; set;}
```

**Remarks**

The only valid block size is 8.

**See Also**

Get / set the key value applicable.

```csharp
public override byte[] Key {get; set;}
```

See Also

- PkzipClassicManaged Class
ICSharpCode SharpZipLib Class Library
Get an array of legal block sizes.

```csharp
```

See Also

- **PkzipClassicManaged Class**  |  **ICSharpCode.SharpZipLib.Encryption Namespace**
Get an array of legal key sizes.


See Also

PkzipClassicManaged Class | ICSharpCode.SharpZipLib.Encryption Namespace
ICSharpCode SharpZipLib Class Library
The methods of the **PkzipClassicManaged** class are listed below. For a complete list of **PkzipClassicManaged** class members, see the **PkzipClassicManaged Members** topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clear</strong> (inherited from <strong>SymmetricAlgorithm</strong>)</td>
<td>Releases all resources used by the <strong>SymmetricAlgorithm</strong>.</td>
</tr>
<tr>
<td><strong>CreateDecryptor</strong></td>
<td>Overloaded. Create a decryptor.</td>
</tr>
<tr>
<td><strong>CreateDecryptor</strong> (inherited from <strong>SymmetricAlgorithm</strong>)</td>
<td>Overloaded. Creates a symmetric decryptor object with the current <strong>Key</strong> and initialization vector (<strong>IV</strong>).</td>
</tr>
<tr>
<td><strong>CreateEncryptor</strong></td>
<td>Overloaded. Create an encryptor.</td>
</tr>
<tr>
<td><strong>CreateEncryptor</strong> (inherited from <strong>SymmetricAlgorithm</strong>)</td>
<td>Overloaded. Creates a symmetric encryptor object with the current <strong>Key</strong> and initialization vector (<strong>IV</strong>).</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GenerateIV</strong></td>
<td>Generate an initial vector.</td>
</tr>
<tr>
<td><strong>GenerateKey</strong></td>
<td>Generate a new random key.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>
ValidKeySize (inherited from SymmetricAlgorithm)

Determines whether the specified key size is valid for the current algorithm.

See Also

PkzipClassicManaged Class | ISharpCode.SharpZipLib.Encryption Namespace
Create a decryptor.

**Overload List**

Inherited from [SymmetricAlgorithm](#).

```csharp
public virtual ICryptoTransform CreateDecryptor();
```

Create a decryptor.

```csharp
public override ICryptoTransform CreateDecryptor(byte[], byte[]);
```

**See Also**

[PkzipClassicManaged Class](#) | [ICSharpCode.SharpZipLib.Encryption Namespace](#)
ICSharpCode SharpZipLib Class Library
Create a decryptor.

```csharp
public override ICryptoTransform CreateDecryptor(byte[] rgbKey, byte[] rgbIV);
```

Parameters

- **rgbKey**
  Keys to use for this new decryptor.

- **rgbIV**
  Initialisation vector for the new decryptor.

Return Value

Returns a new decryptor.

See Also

- PkzipClassicManaged Class
- PkzipClassicManaged.CreateDecryptor Overload List
ICSharpCode SharpZipLib Class Library
Create an encryptor.

**Overload List**

Inherited from [SymmetricAlgorithm](#).

- **public virtual ICryptoTransform CreateEncryptor();**

Create an encryptor.

- **public override ICryptoTransform CreateEncryptor(byte[], byte[]);**

**See Also**

- [PkzipClassicManaged Class](#) | [ICSharpCode.SharpZipLib.Encryption Namespace](#)
ICSharpCode SharpZipLib Class Library
Create an encryptor.

```csharp
public override ICryptoTransform CreateEncryptor(byte[] rgbKey, byte[] rgbIV);
```

**Parameters**

- `rgbKey`:
  - The key to use for this encryptor.

- `rgbIV`:
  - Initialisation vector for the new encryptor.

**Return Value**

Returns a new PkzipClassic encryptor

**See Also**

Generate an initial vector.

```csharp
public override void GenerateIV();
```

See Also

[PkzipClassicManaged Class](#) | [ICSharpCode.SharpZipLib.Encryption Namespace](#)
ICSharpCode SharpZipLib Class Library
Generate a new random key.

```csharp
public override void GenerateKey();
```

See Also

[PkzipClassicManaged Class](#) | [ICSharpCode.SharpZipLib.Encryption Namespace](#)
ICSharpCode SharpZipLib Class Library
### Namespace hierarchy

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GZipConstants</strong></td>
<td>This class contains constants used for gzip.</td>
</tr>
<tr>
<td><strong>GZipException</strong></td>
<td>GZipException represents a Gzip specific exception</td>
</tr>
<tr>
<td><strong>GZipInputStream</strong></td>
<td>This filter stream is used to decompress a &quot;GZIP&quot; format stream. The &quot;GZIP&quot; format is described in RFC 1952. author of the original java version: John Leuner</td>
</tr>
<tr>
<td><strong>GZipOutputStream</strong></td>
<td>This filter stream is used to compress a stream into a &quot;GZIP&quot; stream. The &quot;GZIP&quot; format is described in RFC 1952. author of the original java version: John Leuner</td>
</tr>
</tbody>
</table>
GZipConstants Class

This class contains constants used for gzip.

For a list of all members of this type, see GZipConstants Members.


|public sealed class GZipConstants|

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

Namespace: ISharpCode.SharpZipLib.GZip


See Also

GZipConstants Members | ISharpCode.SharpZipLib.GZip Namespace
GZipConstants Members

GZipConstants overview

Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCOMMENT</td>
<td>Flag bit mask indicating comment is present</td>
</tr>
<tr>
<td>FEXTRA</td>
<td>Flag bit mask for extra</td>
</tr>
<tr>
<td>FHCRC</td>
<td>Flag bitmask for Crc</td>
</tr>
<tr>
<td>FNAME</td>
<td>Flag bitmask for name</td>
</tr>
<tr>
<td>FTEXT</td>
<td>Flag bit mask for text</td>
</tr>
<tr>
<td>GZIP_MAGIC</td>
<td>Magic number found at start of GZIP header</td>
</tr>
</tbody>
</table>

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

See Also

GZipConstants Class | ISharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
The fields of the **GZipConstants** class are listed below. For a complete list of **GZipConstants** class members, see the **GZipConstants Members** topic.

### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ FCOMMENT</td>
<td>Flag bit mask indicating comment is present</td>
</tr>
<tr>
<td>$ FEXTRA</td>
<td>Flag bit mask for extra</td>
</tr>
<tr>
<td>$ FHCRC</td>
<td>Flag bitmask for Crc</td>
</tr>
<tr>
<td>$ FNAME</td>
<td>Flag bitmask for name</td>
</tr>
<tr>
<td>$ FTEXT</td>
<td>Flag bit mask for text</td>
</tr>
<tr>
<td>$ GZIP_MAGIC</td>
<td>Magic number found at start of GZIP header</td>
</tr>
</tbody>
</table>

See Also

[**GZipConstants Class**](#) | [**ICSharpCode.SharpZipLib.GZip Namespace**](#)
ICSharpCode SharpZipLib Class Library
GZipConstants.FCOMMENT Field

flag bit mask indicating comment is present

public const int FCOMMENT = 16;

See Also

GZipConstants Class | ISharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
**GZipConstants.FEXTRA Field**

Flag bit mask for extra

```csharp
public const int FEXTRA = 4;
```

See Also

[GZipConstants Class] | [ICSharpCode.SharpZipLib.GZip Namespace]
ICSharpCode SharpZipLib Class Library
GZipConstants.FHCRC Field

Flag bitmask for Crc

```csharp
public const int FHCRC = 2;
```

See Also

GZipConstants Class | ISharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
GZipConstants.FNAME Field

flag bitmask for name

```csharp
public const int FNAME = 8;
```

See Also

GZipConstants Class | ISharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
**GZipConstants.FTEXT Field**

Flag bit mask for text

```csharp
public const int FTEXT = 1;
```

See Also

[GZipConstants Class] | [ICSharpCode.SharpZipLib.GZip Namespace]
ICSharpCode SharpZipLib Class Library
GZipConstants.GZIP_MAGIC Field

Magic number found at start of GZIP header

```csharp
public const int GZIP_MAGIC = 8075;
```

See Also

GZipConstants Class | ISharpCode.SharpZipLib.GZip Namespace
GZipException Class

GZipException represents a Gzip specific exception

For a list of all members of this type, see GZipException Members.

System.Object    System.Exception
    System.ApplicationException
        ISharpCode.SharpZipLib.SharpZipBaseException
        ISharpCode.SharpZipLib.GZip.GZipException

public class GZipException : SharpZipBaseException

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

    Namespace: ISharpCode.SharpZipLib.GZip

See Also

GZipException Members | ISharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
## GZipException Members

### GZipException overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GZipException</strong></td>
<td>Overloaded. Initializes a new instance of the GZipException class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HelpLink</strong> (inherited from Exception)</td>
<td>Gets or sets a link to the help file associated with this exception.</td>
</tr>
<tr>
<td><strong>InnerException</strong> (inherited from Exception)</td>
<td>Gets the Exception instance that caused the current exception.</td>
</tr>
<tr>
<td><strong>Message</strong> (inherited from Exception)</td>
<td>Gets a message that describes the current exception.</td>
</tr>
<tr>
<td><strong>Source</strong> (inherited from Exception)</td>
<td>Gets or sets the name of the application or the object that causes the error.</td>
</tr>
<tr>
<td><strong>StackTrace</strong> (inherited from Exception)</td>
<td>Gets a string representation of the frames on the call stack at the time the current exception was thrown.</td>
</tr>
<tr>
<td><strong>TargetSite</strong> (inherited from Exception)</td>
<td>Gets the method that throws the current exception.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetBaseException</strong> (inherited from Exception)</td>
<td>When overridden in a derived class, returns the Exception that</td>
</tr>
</tbody>
</table>
is the root cause of one or more subsequent exceptions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetObjectData (inherited from Exception)</td>
<td>When overridden in a derived class, sets the SerializationInfo with information about the exception.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString (inherited from Exception)</td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GZipException</td>
<td>Overloaded. Initializes a new instance of the GZipException class.</td>
</tr>
</tbody>
</table>

### Protected Instance Properties

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HResult (inherited from Exception)</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>
See Also

GZipException Class | ISharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
GZipException Constructor

Deserialization constructor

**Overload List**

- Initialise a new instance of GZipException
  
  ```csharp
  public GZipException();
  ```

  Deserialization constructor
  
  ```csharp
  protected GZipException(SerializationInfo, StreamingContext);
  ```

  Initialise a new instance of GZipException with its message string.
  
  ```csharp
  public GZipException(string);
  ```

  Initialise a new instance of `GZipException`.
  
  ```csharp
  public GZipException(string, Exception);
  ```

**See Also**

- [GZipException Class](#)
- [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
GZipException Constructor (SerializationInfo, StreamingContext)

Deserialization constructor

protected GZipException(
    SerializationInfo info,
    StreamingContext context
);

Parameters

info
    SerializationInfo for this constructor

context
    StreamingContext for this constructor

See Also

GZipException Class | ISharpCode.SharpZipLib.GZip Namespace
| GZipException Constructor Overload List
ICSharpCode SharpZipLib Class Library
GZipException Constructor ()

Initialise a new instance of GZipException

```csharp
public GZipException();
```

See Also

- GZipException Class
- ICSharpCode.SharpZipLib.GZip Namespace
- GZipException Constructor Overload List
ICSharpCode SharpZipLib Class Library
**GZipException Constructor (String)**

Initialise a new instance of GZipException with its message string.

```csharp
public GZipException(
    string message
);
```

**Parameters**

*message*

A `String` that describes the error.

**See Also**

- GZipException Class
- ISharpCode.SharpZipLib.GZip Namespace
- GZipException Constructor Overload List
Initialise a new instance of `GZipException`.

```csharp
public GZipException(
    string message,
    Exception innerException
);
```

**Parameters**

- **message**
  A `String` that describes the error.

- **innerException**
  The `Exception` that caused this exception.

**See Also**

- `GZipException Class`
- `ICSharpCode.SharpZipLib.GZip Namespace`
- `GZipException Constructor Overload List`
ICSharpCode SharpZipLib Class Library
GZipInputStream Class

This filter stream is used to decompress a "GZIP" format stream. The "GZIP" format is described baseInputStream RFC 1952. author of the original java version : John Leuner

For a list of all members of this type, see GZipInputStream Members.

System.Object  System.MarshalByRefObject
System.IO.Stream
ICSharpCode.SharpZipLib.GZip.GZip.GZipInputStream

public class GZipInputStream : InflaterInputStream

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Example

This sample shows how to unzip a gzipped file

```csharp
using System;
using System.IO;

using ISharpCode.SharpZipLib.Core;
using ISharpCode.SharpZipLib.GZip;

class MainClass
{
    public static void Main(string[] args)
    {
        using (Stream inStream = new GZipInputStream(File.OpenRead(args[0])))
            using (FileStream outStream = File.Create(Path.GetFileNameWithoutExtension(args[0])))
            {
                byte[] buffer = new byte[4096];
StreamUtils.Copy(inStream, outStream, buffer)
}
}

Requirements


**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

See Also

GZipInputStream Members

**GZipInputStream overview**

Public Instance Constructors

<table>
<thead>
<tr>
<th><strong>GZipInputStream</strong></th>
<th>Overloaded. Initializes a new instance of the GZipInputStream class.</th>
</tr>
</thead>
</table>

Public Instance Properties

| **Available** (inherited from InflaterInputStream) | Returns 0 once the end of the stream (EOF) has been reached. Otherwise returns 1. |
| **CanRead** (inherited from InflaterInputStream) | Gets a value indicating whether the current stream supports reading |
| **CanSeek** (inherited from InflaterInputStream) | Gets a value of false indicating seeking is not supported for this stream. |
| **CanWrite** (inherited from InflaterInputStream) | Gets a value of false indicating that this stream is not writeable. |
| **IsStreamOwner** (inherited from InflaterInputStream) | Get/set flag indicating ownership of underlying stream. When the flag is true, `Close` will close the underlying stream also. |
| **Length** (inherited from InflaterInputStream) | A value representing the length of the stream in bytes. |
| **Position** (inherited from InflaterInputStream) | The current position within the stream. Throws a `NotSupportedException` when attempting to set the position |

Public Instance Methods
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from InflaterInputStream)</td>
<td>Entry point to begin an asynchronous write. Always throws a NotSupportedException.</td>
</tr>
<tr>
<td><strong>Close</strong> (inherited from InflaterInputStream)</td>
<td>Closes the input stream. When IsStreamOwner is true the underlying stream is also closed.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Flush</strong> (inherited from InflaterInputStream)</td>
<td>Flushes the baseInputStream</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InitializeLifetimeService</code></td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><code>Read</code></td>
<td>Reads uncompressed data into an array of bytes</td>
</tr>
<tr>
<td><code>.ReadByte</code></td>
<td>Reads a byte from the stream and advances the position within the stream by one byte, or returns -1 if at the end of the stream.</td>
</tr>
<tr>
<td><code>Seek</code></td>
<td>Sets the position within the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td><code>SetLength</code></td>
<td>Set the length of the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td><code>Skip</code></td>
<td>Skip specified number of bytes of uncompressed data</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>Write</code></td>
<td>Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException</td>
</tr>
<tr>
<td><code>WriteByte</code></td>
<td>Writes one byte to the current stream and advances the current position Always throws a NotSupportedException</td>
</tr>
</tbody>
</table>

**Protected Instance Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>baseInputStream</code></td>
<td>Base stream the inflater reads from.</td>
</tr>
<tr>
<td><code>crc</code></td>
<td>CRC-32 value for</td>
</tr>
</tbody>
</table>
uncompressed data

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>csize</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>The compressed size</td>
</tr>
<tr>
<td><code>eos</code></td>
<td>Indicates end of stream</td>
</tr>
<tr>
<td><code>inf</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Decompressor for this stream</td>
</tr>
<tr>
<td><code>inputBuffer</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Input buffer for this stream.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateWaitHandle</code> (inherited from <code>Stream</code>)</td>
<td>Allocates a <code>WaitHandle</code> object.</td>
</tr>
<tr>
<td><code>Fill</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Fills the buffer with more data to decompress.</td>
</tr>
<tr>
<td><code>Finalize</code> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>StopDecrypting</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Clear any cryptographic state.</td>
</tr>
</tbody>
</table>

**See Also**

- [GZipInputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
## GZipInputStream Constructor

Creates a GzipInputStream with the default buffer size

## Overload List

 Creates a GzipInputStream with the default buffer size

```java
public GZipInputStream(Stream);
```

Creates a GZIPInputStream with the specified buffer size

```java
public GZipInputStream(Stream,int);
```

## See Also

- [GZipInputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
GZipInputStream Constructor (Stream)

Creates a GzipInputStream with the default buffer size

```java
public GZipInputStream(
    Stream baseInputStream
);
```

Parameters

**baseInputStream**
The stream to read compressed data from (baseInputStream GZIP format)

See Also

GZipInputStream Class | ICSharpCode.SharpZipLib.GZip Namespace | GZipInputStream Constructor Overload List
ICSharpCode SharpZipLib Class Library
GZipInputStream Constructor (Stream, Int32)

Creates a GZipInputStream with the specified buffer size

```csharp
public GZipInputStream(
    Stream baseInputStream,
    int size
);
```

Parameters

- `baseInputStream`  
  The stream to read compressed data from (baseInputStream GZIP format)

- `size`  
  Size of the buffer to use

See Also

ICSharpCode SharpZipLib Class Library
GZipInputStream Fields

The fields of the GZipInputStream class are listed below. For a complete list of GZipInputStream class members, see the GZipInputStream Members topic.

Protected Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>baseInputStream</code></td>
<td>Base stream the inflater reads from.</td>
</tr>
<tr>
<td><code>crc</code></td>
<td>CRC-32 value for uncompressed data</td>
</tr>
<tr>
<td><code>csize</code></td>
<td>The compressed size</td>
</tr>
<tr>
<td><code>eos</code></td>
<td>Indicates end of stream</td>
</tr>
<tr>
<td><code>inf</code></td>
<td>Decompressor for this stream</td>
</tr>
<tr>
<td><code>inputBuffer</code></td>
<td>Input buffer for this stream.</td>
</tr>
</tbody>
</table>

See Also

GZipInputStream Class | ICSharpCode.SharpZipLib.GZip Namespace
**GZipInputStream.crc Field**

CRC-32 value for uncompressed data

```java
protected Crc32 crc;
```

See Also

- [GZipInputStream Class](#)
- [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
GZipInputStream.eos Field

Indicates end of stream

```
protected bool eos;
```

See Also

GZipInputStream Class | ICSharpCode.SharpZipLib.GZip Namespace
The methods of the **GZipInputStream** class are listed below. For a complete list of **GZipInputStream** class members, see the **GZipInputStream Members** topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from InflaterInputStream)</td>
<td>Entry point to begin an asynchronous write. Always throws a NotSupportedException.</td>
</tr>
<tr>
<td><strong>Close</strong> (inherited from InflaterInputStream)</td>
<td>Closes the input stream. When IsStreamOwner is true the underlying stream is also closed.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Flush</strong> (inherited from InflaterInputStream)</td>
<td>Flushes the baseInputStream</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data</td>
</tr>
</tbody>
</table>
structures like a hash table.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetLifetimeService</strong></td>
<td>(inherited from MarshalByRefObject) Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>(inherited from Object) Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong></td>
<td>(inherited from MarshalByRefObject) Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Reads uncompressed data into an array of bytes</td>
</tr>
<tr>
<td><strong>ReadByte</strong></td>
<td>(inherited from Stream) Reads a byte from the stream and advances the position within the stream by one byte, or returns -1 if at the end of the stream.</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>(inherited from InflaterInputStream) Sets the position within the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>(inherited from InflaterInputStream) Set the length of the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td><strong>Skip</strong></td>
<td>(inherited from InflaterInputStream) Skip specified number of bytes of uncompressed data</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>(inherited from Object) Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>(inherited from InflaterInputStream) Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>(inherited from InflaterInputStream) Writes one byte to the current stream and advances the</td>
</tr>
<tr>
<td>current position Always throws a NotSupportedException</td>
<td></td>
</tr>
</tbody>
</table>

### Protected Instance Methods

- **CreateWaitHandle** (inherited from Stream)
  - Allocates a **WaitHandle** object.

- **Fill** (inherited from InflaterInputStream)
  - Fills the buffer with more data to decompress.

- **Finalize** (inherited from Object)
  - Allows an **Object** to attempt to free resources and perform other cleanup operations before the **Object** is reclaimed by garbage collection.

- **MemberwiseClone** (inherited from Object)
  - Creates a shallow copy of the current **Object**.

- **StopDecrypting** (inherited from InflaterInputStream)
  - Clear any cryptographic state.

### See Also

- [GZipInputStream Class](ICSharpCode.SharpZipLib.GZip)
- [Namespace](ICSharpCode.SharpZipLib.GZip)
ICSharpCode SharpZipLib Class Library
GZipInputStream.Read Method

Reads uncompressed data into an array of bytes

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

Parameters

- **buffer**
  The buffer to read uncompressed data into

- **offset**
  The offset indicating where the data should be placed

- **count**
  The number of uncompressed bytes to be read

See Also

- GZipInputStream Class
- ICSharpCode.SharpZipLib.GZip Namespace
GZipOutputStream Class

This filter stream is used to compress a stream into a "GZIP" stream. The "GZIP" format is described in RFC 1952. author of the original java version : John Leuner

For a list of all members of this type, see GZipOutputStream Members.

System.Object System.MarshalByRefObject System.IO.Stream
ICSharpCode.SharpZipLib.GZip.GZip.GZipOutputStream

public class GZipOutputStream : DeflaterOutputStream

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Example

This sample shows how to gzip a file

using System;
using System.IO;

using ISharpCode.SharpZipLib.GZip;
using ISharpCode.SharpZipLib.Core;

class MainClass
{
    public static void Main(string[] args)
    {
        using (Stream s = new GZipOutputStream(File.Create(args[0] + ".gz")))
        using (FileStream fs = File.OpenRead(args[0]))
        { byte[] writeData = new byte[4096];
            
        
    
}
Streamutils.Copy(s, fs, writeData);

Requirements

**Namespace:** ISharpCode.SharpZipLib.GZip

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

See Also

[GZipOutputStream Members](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
# GZipOutputStream Members

## GZipOutputStream overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GZipOutputStream</td>
<td>Overloaded. Initializes a new instance of the GZipOutputStream class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanPatchEntries</td>
<td>Allows client to determine if an entry can be patched after its added.</td>
</tr>
<tr>
<td>CanRead</td>
<td>Gets value indicating stream can be read from.</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Gets a value indicating if seeking is supported for this stream. This property always returns false.</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Get value indicating if this stream supports writing.</td>
</tr>
<tr>
<td>IsStreamOwner</td>
<td>Get/set flag indicating ownership of the underlying stream. When the flag is true Close will close the underlying stream also.</td>
</tr>
<tr>
<td>Length</td>
<td>Get current length of stream.</td>
</tr>
<tr>
<td>Password</td>
<td>Get/set the password used for encryption.</td>
</tr>
<tr>
<td>Position</td>
<td>Gets the current position within the stream.</td>
</tr>
</tbody>
</table>

### Public Instance Methods
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous reads are not supported, a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous writes are not supported, a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Writes remaining compressed output data to the output stream and closes it.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Finish compression and write any footer information required to stream</td>
</tr>
<tr>
<td><strong>Flush</strong> (inherited from DeflaterOutputStream)</td>
<td>Flushes the stream by calling flush() on the deflater and then on the underlying stream. This ensures that all bytes are flushed.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetLevel</td>
<td>Get the current compression level.</td>
</tr>
<tr>
<td>GetLifetimeService (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td>InitializeLifetimeService (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td>Read (inherited from DeflaterOutputStream)</td>
<td>Read a block of bytes from stream</td>
</tr>
<tr>
<td>ReadByte (inherited from DeflaterOutputStream)</td>
<td>Read a byte from stream advancing position by one</td>
</tr>
<tr>
<td>Seek (inherited from DeflaterOutputStream)</td>
<td>Sets the current position of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td>SetLength (inherited from DeflaterOutputStream)</td>
<td>Sets the length of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td>SetLevel</td>
<td>Sets the active compression level (1-9). The new level will be activated immediately.</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
<tr>
<td>Write</td>
<td>Write given buffer to output updating crc</td>
</tr>
<tr>
<td>WriteByte (inherited from DeflaterOutputStream)</td>
<td>Writes a single byte to the compressed output stream.</td>
</tr>
</tbody>
</table>

**Protected Instance Fields**
| **baseOutputStream** (inherited from DeflaterOutputStream) | Base stream the deflater depends on. |
| **crc** | CRC-32 value for uncompressed data |
| **def** (inherited from DeflaterOutputStream) | The deflater which is used to deflate the stream. |

## Protected Instance Methods

| **CreateWaitHandle** (inherited from Stream) | Allocates a **WaitHandle** object. |
| **Deflate** (inherited from DeflaterOutputStream) | Deflates everything in the input buffers. This will call `def.deflate()` until all bytes from the input buffers are processed. |
| **EncryptBlock** (inherited from DeflaterOutputStream) | Encrypt a block of data |
| **EncryptByte** (inherited from DeflaterOutputStream) | Encrypt a single byte |
| **Finalize** (inherited from Object) | Allows an **Object** to attempt to free resources and perform other cleanup operations before the **Object** is reclaimed by garbage collection. |
| **InitializePassword** (inherited from DeflaterOutputStream) | Initializes encryption keys based on given password |
| **MemberwiseClone** (inherited from Object) | Creates a shallow copy of the current **Object**. |
| **UpdateKeys** (inherited from DeflaterOutputStream) | Update encryption keys |

See Also
ICSharpCode SharpZipLib Class Library
GZipOutputStream Constructor

Creates a GzipOutputStream with the default buffer size

Overload List

Creates a GzipOutputStream with the default buffer size

```csharp
public GZipOutputStream(Stream);
```

Creates a GZipOutputStream with the specified buffer size

```csharp
public GZipOutputStream(Stream, int);
```

See Also

[GZipOutputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
GZipOutputStream Constructor (Stream)

Creates a GzipOutputStream with the default buffer size

```java
public GZipOutputStream(
    Stream baseOutputStream
);
```

Parameters

`baseOutputStream`
The stream to read data (to be compressed) from

See Also

GZipOutputStream Class | ISharpCode.SharpZipLib.GZip Namespace | GZipOutputStream Constructor Overload List
ICSharpCode SharpZipLib Class Library
GZipOutputStream Constructor (Stream, Int32)

Creates a GZipOutputStream with the specified buffer size

```csharp
public GZipOutputStream(
    Stream baseOutputStream,
    int size
);
```

Parameters

*baseOutputStream*
  The stream to read data (to be compressed) from

*size*
  Size of the buffer to use

See Also

GZipOutputStream Class | ICSharpCode.SharpZipLib.GZip Namespace | GZipOutputStream Constructor Overload List
ICSharpCode SharpZipLib Class Library
GZipOutputStream Fields

The fields of the **GZipOutputStream** class are listed below. For a complete list of **GZipOutputStream** class members, see the [GZipOutputStream Members](#) topic.

**Protected Instance Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>baseOutputStream</strong> (inherited from DeflaterOutputStream)</td>
<td>Base stream the deflater depends on.</td>
</tr>
<tr>
<td><strong>crc</strong></td>
<td>CRC-32 value for uncompressed data</td>
</tr>
<tr>
<td><strong>def</strong> (inherited from DeflaterOutputStream)</td>
<td>The deflater which is used to deflate the stream.</td>
</tr>
</tbody>
</table>

**See Also**

[GZipOutputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
**GZipOutputStream.crc Field**

CRC-32 value for uncompressed data

```java
protected Crc32 crc;
```

**See Also**

[NGZipOutputStream Class](https://icsharpcode.net/SharpZipLib.GZipNamespace) | [ICSharpCode.SharpZipLib.GZipNamespace](https://icsharpcode.net/SharpZipLib.GZipNamespace)
ICSharpCode SharpZipLib Class Library
The methods of the **GZipOutputStream** class are listed below. For a complete list of **GZipOutputStream** class members, see the [GZipOutputStream Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous reads are not supported a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous writes are not supported, a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Writes remaining compressed output data to the output stream and closes it.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Finish compression and write any footer information required to stream</td>
</tr>
<tr>
<td><strong>Flush</strong> (inherited from)</td>
<td>Flushes the stream by calling</td>
</tr>
<tr>
<td>Method/Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>DeflaterOutputStream</strong></td>
<td>flush() on the deflater and then on the underlying stream. This ensures that all bytes are flushed.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLevel</strong></td>
<td>Get the current compression level.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong> (inherited from DeflaterOutputStream)</td>
<td>Read a block of bytes from stream</td>
</tr>
<tr>
<td><strong>ReadByte</strong> (inherited from DeflaterOutputStream)</td>
<td>Read a byte from stream advancing position by one</td>
</tr>
<tr>
<td><strong>Seek</strong> (inherited from DeflaterOutputStream)</td>
<td>Sets the current position of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><strong>SetLength</strong> (inherited from DeflaterOutputStream)</td>
<td>Sets the length of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><strong>SetLevel</strong></td>
<td>Sets the active compression level (1-9). The new level will be activated immediately.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Write given buffer to output updating crc</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Writes a single byte to the compressed output stream.</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>(inherited from DeflaterOutputStream)</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateWaitHandle</strong></td>
<td>Allocates a <strong>WaitHandle</strong> object.</td>
</tr>
<tr>
<td><strong>CreateWaitHandle</strong></td>
<td>(inherited from Stream)</td>
</tr>
<tr>
<td><strong>Deflate</strong></td>
<td>Deflates everything in the input buffers. This will call</td>
</tr>
<tr>
<td><strong>Deflate</strong></td>
<td><strong>def.deflate()</strong></td>
</tr>
<tr>
<td><strong>Deflate</strong></td>
<td>until all bytes from the input buffers are processed.</td>
</tr>
<tr>
<td><strong>Deflate</strong></td>
<td>(inherited from DeflaterOutputStream)</td>
</tr>
<tr>
<td><strong>EncryptBlock</strong></td>
<td>Encrypt a block of data</td>
</tr>
<tr>
<td><strong>EncryptBlock</strong></td>
<td>(inherited from DeflaterOutputStream)</td>
</tr>
<tr>
<td><strong>EncryptByte</strong></td>
<td>Encrypt a single byte</td>
</tr>
<tr>
<td><strong>EncryptByte</strong></td>
<td>(inherited from DeflaterOutputStream)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>(inherited from Object)</td>
</tr>
<tr>
<td><strong>InitializePassword</strong></td>
<td>Initializes encryption keys based on given password</td>
</tr>
<tr>
<td><strong>InitializePassword</strong></td>
<td>(inherited from DeflaterOutputStream)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>(inherited from Object)</td>
</tr>
<tr>
<td><strong>UpdateKeys</strong></td>
<td>Update encryption keys</td>
</tr>
<tr>
<td><strong>UpdateKeys</strong></td>
<td>(inherited from DeflaterOutputStream)</td>
</tr>
</tbody>
</table>

**See Also**

[GZipOutputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
GZipOutputStream.Close Method

Writes remaining compressed output data to the output stream and closes it.

```java
public override void Close();
```

See Also

GZipOutputStream Class | ISharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
GZipOutputStream.Finish Method

Finish compression and write any footer information required to stream

```
public override void Finish();
```

See Also

GZipOutputStream Class | ISharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
GZipOutputStream.GetLevel Method

Get the current compression level.

```csharp
public int GetLevel();
```

Return Value
The current compression level.

See Also
GZipOutputStream Class | ISharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
Sets the active compression level (1-9). The new level will be activated immediately.

```csharp
public void SetLevel(
    int level
);
```

### Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentOutOfRangeException</code></td>
<td>Level specified is not supported.</td>
</tr>
</tbody>
</table>

### See Also

- [GZipOutputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
GZipOutputStream.Write Method

Write given buffer to output updating crc

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

Parameters

- **buffer**
  - Buffer to write

- **offset**
  - Offset of first byte in buf to write

- **count**
  - Number of bytes to write

See Also

- [GZipOutputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
ICSharpCode SharpZipLib Class Library
# ICSHarpCode.SharpZipLib.Tar Namespace

## Namespace hierarchy

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidHeaderException</code></td>
<td>This exception is used to indicate that there is a problem with a TAR archive header.</td>
</tr>
<tr>
<td><code>TarArchive</code></td>
<td>The TarArchive class implements the concept of a 'Tape Archive'. A tar archive is a series of entries, each of which represents a file system object. Each entry in the archive consists of a header block followed by 0 or more data blocks. Directory entries consist only of the header block, and are followed by entries for the directory's contents. File entries consist of a header followed by the number of blocks needed to contain the file's contents. All entries are written on block boundaries. Blocks are 512 bytes long. TarArchives are instantiated in either read or write mode, based upon whether they are instantiated with an <code>InputStream</code> or an <code>OutputStream</code>. Once</td>
</tr>
<tr>
<td>instantiated TarArchives read/write mode can not be changed. There is currently no support for random access to tar archives. However, it seems that subclassing TarArchive, and using the TarBuffer.CurrentRecord and TarBuffer.CurrentBlock properties, this would be rather trivial.</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td><strong>TarBuffer</strong></td>
<td></td>
</tr>
<tr>
<td>The TarBuffer class implements the tar archive concept of a buffered input stream. This concept goes back to the days of blocked tape drives and special io devices. In the C# universe, the only real function that this class performs is to ensure that files have the correct &quot;record&quot; size, or other tars will complain. You should never have a need to access this class directly. TarBuffers are created by Tar IO Streams.</td>
<td></td>
</tr>
<tr>
<td><strong>TarEntry</strong></td>
<td></td>
</tr>
<tr>
<td>This class represents an entry in a Tar archive. It consists of the entry's header, as well as the entry's File. Entries can be instantiated in one of three ways, depending on how they are to be used.</td>
<td></td>
</tr>
</tbody>
</table>
TarEntries that are created from the header bytes read from an archive are instantiated with the TarEntry(byte[]) constructor. These entries will be used when extracting from or listing the contents of an archive. These entries have their header filled in using the header bytes. They also set the File to null, since they reference an archive entry not a file.

TarEntries that are created from files that are to be written into an archive are instantiated with the CreateEntryFromFile(string) pseudo constructor. These entries have their header filled in using the File's information. They also keep a reference to the File for convenience when writing entries.

Finally, TarEntries can be constructed from nothing but a name. This allows the programmer to construct the entry by hand, for instance when only an InputStream is available for writing to the archive, and the header information is constructed from other information. In this case the
header fields are set to defaults and the File is set to null.

**TarHeader**

**TarException**

TarExceptions are used for exceptions specific to tar classes and code.

**TarHeader**

This class encapsulates the Tar Entry Header used in Tar Archives. The class also holds a number of tar constants, used mostly in headers.

**TarInputStream**

The TarInputStream reads a UNIX tar archive as an InputStream. methods are provided to position at each successive entry in the archive, and the read each entry as a normal input stream using read().

**TarInputStream.EntryFactoryAdapter**

Standard entry factory class creating instances of the class TarEntry

**TarOutputStream**

The TarOutputStream writes a UNIX tar archive as an OutputStream. Methods are provided to put entries, and then write their contents by writing to this stream using write().

## Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Delegate</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>TarInputstream.IEntryFactory</td>
<td>This interface is provided, along with the method SetEntryFactory, to allow the programmer to have their own TarEntry subclass instantiated for the entries return from GetNextEntry.</td>
</tr>
</tbody>
</table>

## Delegates

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProgressMessageHandler</td>
<td>Used to advise clients of 'events' while processing archives</td>
</tr>
</tbody>
</table>
ICSharpCode SharpZipLib Class Library
InvalidHeaderException Class

This exception is used to indicate that there is a problem with a TAR archive header.

For a list of all members of this type, see `InvalidHeaderException Members`.


```csharp
public class InvalidHeaderException : TarException
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

**Namespace:** ICSharpCode.SharpZipLib.Tar

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

See Also

InvalidHeaderException Members  |  ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
# InvalidHeaderException Members

## InvalidHeaderException overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidHeaderException</code></td>
<td>Overloaded. Initializes a new instance of the InvalidHeaderException class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>HelpLink</code> (inherited from Exception)</td>
<td>Gets or sets a link to the help file associated with this exception.</td>
</tr>
<tr>
<td><code>InnerException</code> (inherited from Exception)</td>
<td>Gets the Exception instance that caused the current exception.</td>
</tr>
<tr>
<td><code>Message</code> (inherited from Exception)</td>
<td>Gets a message that describes the current exception.</td>
</tr>
<tr>
<td><code>Source</code> (inherited from Exception)</td>
<td>Gets or sets the name of the application or the object that causes the error.</td>
</tr>
<tr>
<td><code>StackTrace</code> (inherited from Exception)</td>
<td>Gets a string representation of the frames on the call stack at the time the current exception was thrown.</td>
</tr>
<tr>
<td><code>TargetSite</code> (inherited from Exception)</td>
<td>Gets the method that throws the current exception.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><code>GetBaseException</code> (inherited from Exception)</td>
<td>When overridden in a derived class, returns the Exception that</td>
</tr>
</tbody>
</table>
is the root cause of one or more subsequent exceptions.

<table>
<thead>
<tr>
<th>Method (inherited from)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHashCode (Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetObjectData (Exception)</td>
<td>When overridden in a derived class, sets the <code>SerializationInfo</code> with information about the exception.</td>
</tr>
<tr>
<td>GetType (Object)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td>ToString (Exception)</td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidHeaderException</td>
<td>Overloaded. Initializes a new instance of the <code>InvalidHeaderException</code> class.</td>
</tr>
</tbody>
</table>

### Protected Instance Properties

<table>
<thead>
<tr>
<th>Method (inherited from)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HResult (Exception)</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method (inherited from)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize (Object)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>MemberwiseClone (Object)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>
See Also

InvalidHeaderException Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
InvalidHeaderException Constructor

Deserialization constructor

**Overload List**

Initialise a new instance of the InvalidHeaderException class.

```csharp
public InvalidHeaderException();
```

Deserialization constructor

```csharp
protected InvalidHeaderException(SerializationInfo, StreamingContext);
```

Initialises a new instance of the InvalidHeaderException class with a specified message.

```csharp
public InvalidHeaderException(string);
```

Initialise a new instance of InvalidHeaderException

```csharp
public InvalidHeaderException(string, Exception);
```

**See Also**

[InvalidHeaderException Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
InvalidHeaderException Constructor (SerializationInfo, StreamingContext)

Deserialization constructor

```csharp
protected InvalidHeaderException(
    SerializationInfo information,
    StreamingContext context
);
```

Parameters

- **information**
  - `SerializationInfo` for this constructor

- **context**
  - `StreamingContext` for this constructor

See Also

- `InvalidHeaderException Class`
- `ICSharpCode.SharpZipLib.Tar Namespace`
- `InvalidHeaderException Constructor Overload List`
ICSharpCode SharpZipLib Class Library
InvalidHeaderException Constructor ()

Initialise a new instance of the InvalidHeaderException class.

public InvalidHeaderException();

See Also

InvalidHeaderException Class  |  ICSharpCode.SharpZipLib.Tar Namespace  |  InvalidHeaderException Constructor Overload List
ICSharpCode SharpZipLib Class Library
**InvalidHeaderException Constructor (String)**

Initialises a new instance of the InvalidHeaderException class with a specified message.

```csharp
public InvalidHeaderException(
    string message
);
```

**Parameters**

- `message`
  Message describing the exception cause.

**See Also**

- [InvalidHeaderException Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#) | [InvalidHeaderException Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
InvalidHeaderException Constructor (String, Exception)

 Initialise a new instance of InvalidHeaderException

```csharp
public InvalidHeaderException(
    string message,
    Exception exception
);
```

Parameters

- **message**
  Message describing the problem.

- **exception**
  The exception that is the cause of the current exception.

See Also

- InvalidHeaderException Class | ICSharpCode.SharpZipLib.Tar Namespace | InvalidHeaderException Constructor Overload List
### ProgressMessageHandler Delegate

Used to advise clients of 'events' while processing archives

```csharp
public delegate void ProgressMessageHandler(
    TarArchive archive,
    TarEntry entry,
    string message
);
```

### Requirements

**Namespace:** ICSharpCode.SharpZipLib.Tar

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

### See Also

ICSharpCode.SharpZipLib.Tar Namespace
The TarArchive class implements the concept of a 'Tape Archive'. A tar archive is a series of entries, each of which represents a file system object. Each entry in the archive consists of a header block followed by 0 or more data blocks. Directory entries consist only of the header block, and are followed by entries for the directory's contents. File entries consist of a header followed by the number of blocks needed to contain the file's contents. All entries are written on block boundaries. Blocks are 512 bytes long. TarArchives are instantiated in either read or write mode, based upon whether they are instantiated with an InputStream or an OutputStream. Once instantiated TarArchives read/write mode can not be changed. There is currently no support for random access to tar archives. However, it seems that subclassing TarArchive, and using the TarBuffer.CurrentRecord and TarBuffer.CurrentBlock properties, this would be rather trivial.

For a list of all members of this type, see TarArchive Members.


public class TarArchive : IDisposable

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

TarArchive Members | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
# TarArchive Members

## TarArchive overview

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="CreateInputTarArchive" /></td>
<td>Overloaded. The InputStream based constructors create a TarArchive for the purposes of extracting or listing a tar archive. Thus, use these constructors when you wish to extract files from or list the contents of an existing tar archive.</td>
</tr>
<tr>
<td><img src="image" alt="CreateOutputTarArchive" /></td>
<td>Overloaded. Create a TarArchive for writing to, using the default blocking factor</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="ApplyUserInfoOverrides" /></td>
<td>Get or set a value indicating if overrides defined by SetUserInfo should be applied.</td>
</tr>
<tr>
<td><img src="image" alt="AsciiTranslate" /></td>
<td>Get/set the ascii file translation flag. If ascii file translation is true, then the file is checked to see if it a binary file or not. If the flag is true and the test indicates it is ascii text file, it will be translated. The translation converts the local operating system's concept of line ends into the UNIX line end, 'n', which is the defacto standard for a TAR archive. This makes text files compatible with UNIX.</td>
</tr>
<tr>
<td><img src="image" alt="Groupld" /></td>
<td>Get the archive group id. See</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th><strong>GroupName</strong></th>
<th>Get the archive group name. See <a href="#">ApplyUserInfoOverrides</a> for detail on how to allow setting values on a per entry basis.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PathPrefix</strong></td>
<td>PathPrefix is added to entry names as they are written if the value is not null. A slash character is appended after PathPrefix</td>
</tr>
<tr>
<td><strong>RecordSize</strong></td>
<td>Get the archive's record size. Tar archives are composed of a series of RECORDS each containing a number of BLOCKS. This allowed tar archives to match the IO characteristics of the physical device being used. Archives are expected to be properly &quot;blocked&quot;.</td>
</tr>
<tr>
<td><strong>RootPath</strong></td>
<td>RootPath is removed from entry names if it is found at the beginning of the name.</td>
</tr>
<tr>
<td><strong>UserId</strong></td>
<td>Get the archive user id. See <a href="#">ApplyUserInfoOverrides</a> for detail on how to allow setting values on a per entry basis.</td>
</tr>
<tr>
<td><strong>UserName</strong></td>
<td>Get the archive user name. See <a href="#">ApplyUserInfoOverrides</a> for detail on how to allow setting values on a per entry basis.</td>
</tr>
</tbody>
</table>

**Public Instance Methods**
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Close</strong></td>
<td>Closes the archive and releases any associated resources.</td>
</tr>
<tr>
<td><strong>CloseArchive</strong></td>
<td>Obsolete. Close the archive.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>ExtractContents</strong></td>
<td>Perform the &quot;extract&quot; command and extract the contents of the archive.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ListContents</strong></td>
<td>Perform the &quot;list&quot; command for the archive contents. NOTE That this method uses the progress event to actually list the contents. If the progress display event is not set, nothing will be listed!</td>
</tr>
<tr>
<td><strong>SetAsciiTranslation</strong></td>
<td>Obsolete. Set the ascii file translation flag.</td>
</tr>
<tr>
<td><strong>SetKeepOldFiles</strong></td>
<td>Set the flag that determines whether existing files are kept, or overwritten during extraction.</td>
</tr>
<tr>
<td><strong>SetUserInfo</strong></td>
<td>Set user and group information that will be used to fill in the tar archive's entry headers. This information based on that available for the linux operating system, which is not always available on other operating systems.</td>
</tr>
</tbody>
</table>
systems. TarArchive allows the programmer to specify values to be used in their place. `ApplyUserInfoOverrides` is set to true by this call.

<table>
<thead>
<tr>
<th><strong>ToString</strong> (inherited from <strong>Object</strong>)</th>
<th>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WriteEntry</strong></td>
<td>Write an entry to the archive. This method will call the <code>putNextEntry</code> and then write the contents of the entry, and finally call <code>closeEntry()</code> for entries that are files. For directories, it will call <code>putNextEntry()</code>, and then, if the recurse flag is true, process each entry that is a child of the directory.</td>
</tr>
</tbody>
</table>

### Public Instance Events

| **ProgressMessageEvent** | Client hook allowing detailed information to be reported during processing |

### Protected Instance Constructors

| **TarArchive** | Overloaded. Initializes a new instance of the TarArchive class. |

### Protected Instance Methods

| **Dispose** | Releases the unmanaged resources used by the FileStream and optionally releases the managed resources. |
| **Finalize** | Ensures that resources are freed and other cleanup |
Operations are performed when the garbage collector reclaims the **TarArchive**.

| **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |
| **OnProgressMessageEvent** | Raises the ProgressMessage event |

**See Also**

[TarArchive Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
## TarArchive Constructor

Constructor for a default [TarArchive](#).

### Overload List

Constructor for a default [TarArchive](#).

```java
protected TarArchive();
```

Initialise a TarArchive for input.

```java
protected TarArchive(TarInputStream);
```

Initialise a TarArchive for output.

```java
protected TarArchive(TarOutputStream);
```

### See Also

[TarArchive Class](#) | ISharpCode.SharpZipLib.Tar Namespace
TarArchive Constructor ()

Constructor for a default TarArchive.

protected TarArchive();

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace | TarArchive Constructor Overload List
ICSharpCode SharpZipLib Class Library
TarArchive Constructor (TarInputStream)

Initialise a TarArchive for input.

```csharp
protected TarArchive(
    TarInputStream stream
);
```

Parameters

- `stream` The `TarInputStream` to use for input.

See Also

- [TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#) | [TarArchive Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
TarArchive Constructor (TarOutputStream)

Initialise a TarArchive for output.

```java
protected TarArchive(
    TarOutputStream stream
);
```

Parameters

*stream*

The `TarOutputStream` to use for output.

See Also

[TarArchive Class] | [ICSharpCode.SharpZipLib.Tar Namespace] | TarArchive Constructor Overload List
ICSharpCode SharpZipLib Class Library
**TarArchive Properties**

The properties of the **TarArchive** class are listed below. For a complete list of **TarArchive** class members, see the **TarArchive Members** topic.

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyUserInfoOverrides</strong></td>
<td>Get or set a value indicating if overrides defined by <strong>SetUserInfo</strong> should be applied.</td>
</tr>
<tr>
<td><strong>AsciiTranslate</strong></td>
<td>Get/set the ascii file translation flag. If ascii file translation is true, then the file is checked to see if it a binary file or not. If the flag is true and the test indicates it is ascii text file, it will be translated. The translation converts the local operating system's concept of line ends into the UNIX line end, '\n', which is the defacto standard for a TAR archive. This makes text files compatible with UNIX.</td>
</tr>
<tr>
<td><strong>GroupId</strong></td>
<td>Get the archive group id. See <strong>ApplyUserInfoOverrides</strong> for detail on how to allow setting values on a per entry basis.</td>
</tr>
<tr>
<td><strong>GroupName</strong></td>
<td>Get the archive group name. See <strong>ApplyUserInfoOverrides</strong> for detail on how to allow setting values on a per entry basis.</td>
</tr>
<tr>
<td><strong>PathPrefix</strong></td>
<td>PathPrefix is added to entry names as they are written if the value is not null. A slash character is appended after PathPrefix</td>
</tr>
</tbody>
</table>
Get the archive's record size. Tar archives are composed of a series of RECORDS each containing a number of BLOCKS. This allowed tar archives to match the IO characteristics of the physical device being used. Archives are expected to be properly "blocked".

RootPath is removed from entry names if it is found at the beginning of the name.

Get the archive user id. See ApplyUserInfoOverrides for detail on how to allow setting values on a per entry basis.

Get the archive user name. See ApplyUserInfoOverrides for detail on how to allow setting values on a per entry basis.

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
TarArchive.ApplyUserInfoOverrides Property

Get or set a value indicating if overrides defined by SetUserInfo should be applied.

```csharp
public bool ApplyUserInfoOverrides {get; set;}
```

Remarks

If overrides are not applied then the values as set in each header will be used.

See Also

TarArchive Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarArchive.AsciiTranslate Property

Get/set the ascii file translation flag. If ascii file translation is true, then the file is checked to see if it a binary file or not. If the flag is true and the test indicates it is ascii text file, it will be translated. The translation converts the local operating system's concept of line ends into the UNIX line end, '\n', which is the defacto standard for a TAR archive. This makes text files compatible with UNIX.

```csharp
public bool AsciiTranslate {get; set;}
```

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
Get the archive group id. See [ApplyUserInfoOverrides](#) for detail on how to allow setting values on a per entry basis.

```csharp
public int GroupId {get;}
```

See Also

[TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarArchive.GroupName Property**

Get the archive group name. See [ApplyUserInfoOverrides](#) for detail on how to allow setting values on a per entry basis.

```csharp
public string GroupName {get;}
```

See Also

[TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarArchive.PathPrefix Property**

PathPrefix is added to entry names as they are written if the value is not null. A slash character is appended after PathPrefix.

```csharp
public string PathPrefix {get; set;}
```

See Also

[TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarArchive.RecordSize Property

Get the archive's record size. Tar archives are composed of a series of RECORDS each containing a number of BLOCKS. This allowed tar archives to match the IO characteristics of the physical device being used. Archives are expected to be properly "blocked".

```csharp
public int RecordSize {get;}
```

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
RootPath is removed from entry names if it is found at the beginning of the name.

```csharp
public string RootPath {get; set;}
```

See Also

[TarArchive Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarArchive.UserId Property

Get the archive user id. See [ApplyUserInfoOverrides](#) for detail on how to allow setting values on a per entry basis.

```csharp
public int UserId {get;}
```

See Also

[TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
TarArchive.getUserName Property

Get the archive user name. See [ApplyUserInfoOverrides](#) for detail on how to allow setting values on a per entry basis.

```csharp
public string UserName {get;}
```

See Also

[TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
TarArchive Methods

The methods of the TarArchive class are listed below. For a complete list of TarArchive class members, see the TarArchive Members topic.

Public Static Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateInputTarArchive</code></td>
<td>Overloaded. The InputStream based constructors create a TarArchive for the purposes of extracting or listing a tar archive. Thus, use these constructors when you wish to extract files from or list the contents of an existing tar archive.</td>
</tr>
<tr>
<td><code>CreateOutputTarArchive</code></td>
<td>Overloaded. Create a TarArchive for writing to, using the default blocking factor</td>
</tr>
</tbody>
</table>

Public Instance Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Close</code></td>
<td>Closes the archive and releases any associated resources.</td>
</tr>
<tr>
<td><code>CloseArchive</code></td>
<td>Obsolete. Close the archive.</td>
</tr>
<tr>
<td><code>Equals</code> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><code>ExtractContents</code></td>
<td>Perform the &quot;extract&quot; command and extract the contents of the archive.</td>
</tr>
<tr>
<td><code>GetHashCode</code> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><code>GetType</code> (inherited from)</td>
<td>Gets the Type of the current</td>
</tr>
</tbody>
</table>

<p>|</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListContents</td>
<td>Perform the &quot;list&quot; command for the archive contents. NOTE That this method uses the progress event to actually list the contents. If the progress display event is not set, nothing will be listed!</td>
</tr>
<tr>
<td>SetAsciiTranslation</td>
<td>Obsolete. Set the ascii file translation flag.</td>
</tr>
<tr>
<td>SetKeepOldFiles</td>
<td>Set the flag that determines whether existing files are kept, or overwritten during extraction.</td>
</tr>
<tr>
<td>SetUserInfo</td>
<td>Set user and group information that will be used to fill in the tar archive's entry headers. This information based on that available for the linux operating system, which is not always available on other operating systems. TarArchive allows the programmer to specify values to be used in their place. ApplyUserInfoOverrides is set to true by this call.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td>WriteEntry</td>
<td>Write an entry to the archive. This method will call the putNextEntry and then write the contents of the entry, and finally call closeEntry() for entries that are files. For directories, it will call putNextEntry(), and then, if the recurse flag is true, process each entry that is a child of the</td>
</tr>
</tbody>
</table>
Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✨ Dispose</td>
<td>Releases the unmanaged resources used by the FileStream and optionally releases the managed resources.</td>
</tr>
<tr>
<td>✨ Finalize</td>
<td>Ensures that resources are freed and other cleanup operations are performed when the garbage collector reclaimsthe TarArchive.</td>
</tr>
<tr>
<td>✨ MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>✨ OnProgressMessageEvent</td>
<td>Raises the ProgressMessage event</td>
</tr>
</tbody>
</table>

See Also

TarArchive Class | ISharpCode.SharpZipLib.Tar Namespace
TarArchive.Close Method

Closes the archive and releases any associated resources.

```csharp
public virtual void Close();
```

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
NOTE: This method is now obsolete.
Use Close instead

Close the archive.

```csharp
public void CloseArchive();
```

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
The InputStream based constructors create a TarArchive for the purposes of extracting or listing a tar archive. Thus, use these constructors when you wish to extract files from or list the contents of an existing tar archive.

**Overload List**

The InputStream based constructors create a TarArchive for the purposes of extracting or listing a tar archive. Thus, use these constructors when you wish to extract files from or list the contents of an existing tar archive.

```java
public static TarArchive CreateInputTarArchive(Stream);
```

Create TarArchive for reading setting block factor

```java
public static TarArchive CreateInputTarArchive(Stream,int);
```

**See Also**

[TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
The InputStream based constructors create a TarArchive for the purposes of extracting or listing a tar archive. Thus, use these constructors when you wish to extract files from or list the contents of an existing tar archive.

```csharp
public static TarArchive CreateInputTarArchive(Stream inputStream);
```

See Also

[TarArchive Class] | [ICSharpCode.SharpZipLib.Tar Namespace] | [TarArchive.CreateInputTarArchive Overload List]
ICSharpCode SharpZipLib Class Library
Create TarArchive for reading setting block factor

```csharp
public static TarArchive CreateInputTarArchive(
    Stream inputStream,
    int blockFactor
);
```

**Parameters**

- `inputStream`  
  Stream for tar archive contents

- `blockFactor`  
  The blocking factor to apply

**Return Value**

TarArchive

**See Also**

- [TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#) | [TarArchive.CreateInputTarArchive Overload List](#)
ICSharpCode SharpZipLib Class Library
TarArchive.CreateOutputTarArchive Method

Create a TarArchive for writing to, using the default blocking factor

Overload List

Create a TarArchive for writing to, using the default blocking factor

```csharp
public static TarArchive CreateOutputTarArchive(Stream);
```
Create a TarArchive for writing to

```csharp
public static TarArchive CreateOutputTarArchive(Stream,int);
```

See Also

- TarArchive Class
- ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarArchive.CreateOutputTarArchive Method (Stream)**

Create a TarArchive for writing to, using the default blocking factor

```csharp
public static TarArchive CreateOutputTarArchive(Stream outputStream);
```

**Parameters**

`outputStream`  
Stream to write to

**See Also**

ICSharpCode SharpZipLib Class Library
Create a TarArchive for writing to

```csharp
public static TarArchive CreateOutputTarArchive(
    Stream outputStream,
    int blockFactor
);
```

**Parameters**

- `outputStream`
  The stream to write to

- `blockFactor`
  The blocking factor to use for buffering.

**See Also**

- [TarArchive Class](#)
- [ICSharpCode.SharpZipLib.Tar Namespace](#)
- [TarArchive.CreateOutputTarArchive Overload List](#)
ICSharpCode SharpZipLib Class Library
TarArchive.Dispose Method

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

```csharp
protected virtual void Dispose(bool disposing);
```

Parameters

- `disposing`  
  true to release both managed and unmanaged resources; false to release only unmanaged resources.

See Also

- TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarArchive.ExtractContents Method

Perform the "extract" command and extract the contents of the archive.

```csharp
public void ExtractContents(
    string destinationDirectory
);
```

Parameters

destinationDirectory
The destination directory into which to extract.

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
Ensures that resources are freed and other cleanup operations are performed when the garbage collector reclaims the TarArchive.

```cpp
protected override void Finalize();
```

See Also

TarArchive Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarArchive.ListContents Method**

Perform the "list" command for the archive contents. NOTE That this method uses the **progress event** to actually list the contents. If the progress display event is not set, nothing will be listed!

```
public void ListContents();
```

**See Also**

[TarArchive Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarArchive.OnProgressMessageEvent Method

Raises the ProgressMessage event

```csharp
protected virtual void OnProgressMessageEvent(TarEntry entry, string message);
```

**Parameters**

- `entry`
  TarEntry for this event

- `message`
  message for this event. Null is no message

**See Also**

[TarArchive Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarArchive.SetAsciiTranslation Method

NOTE: This method is now obsolete.

Use the AsciiTranslate property

Set the ascii file translation flag.

```csharp
public void SetAsciiTranslation(
    bool asciiTranslate
);
```

Parameters

`asciiTranslate`
If true, translate ascii text files.

See Also

TarArchive Class  |  ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarArchive.SetKeepOldFiles Method

Set the flag that determines whether existing files are kept, or overwritten during extraction.

```csharp
public void SetKeepOldFiles(
    bool keepOldFiles
);
```

Parameters

*keepOldFiles*

If true, do not overwrite existing files.

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarArchive.SetUserInfo Method**

Set user and group information that will be used to fill in the tar archive's entry headers. This information based on that available for the linux operating system, which is not always available on other operating systems. TarArchive allows the programmer to specify values to be used in their place. `ApplyUserInfoOverrides` is set to true by this call.

```csharp
public void SetUserInfo(
    int userId,
    string userName,
    int groupId,
    string groupName
);
```

**Parameters**

- `userId`
  - The user id to use in the headers.

- `userName`
  - The user name to use in the headers.

- `groupId`
  - The group id to use in the headers.

- `groupName`
  - The group name to use in the headers.

**See Also**

- `TarArchive Class`
- `ICSharpCode.SharpZipLib.Tar Namespace`
ICSharpCode SharpZipLib Class Library
TarArchive.WriteEntry Method

Write an entry to the archive. This method will call the putNextEntry and then write the contents of the entry, and finally call closeEntry() for entries that are files. For directories, it will call putNextEntry(), and then, if the recurse flag is true, process each entry that is a child of the directory.

```csharp
public void WriteEntry(
    TarEntry sourceEntry,
    bool recurse
);
```

Parameters

sourceEntry
  The TarEntry representing the entry to write to the archive.

recurse
  If true, process the children of directory entries.

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarArchive Events

The events of the TarArchive class are listed below. For a complete list of TarArchive class members, see the TarArchive Members topic.

Public Instance Events

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProgressBarEvent</td>
<td>Client hook allowing detailed information to be reported during processing</td>
</tr>
</tbody>
</table>

See Also

TarArchive Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarArchive.ProgressMessageEventArgs Event**

Client hook allowing detailed information to be reported during processing.

```csharp
public event ProgressMessageEventHandler ProgressMessage;
```

See Also

[TarArchive Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
The TarBuffer class implements the tar archive concept of a buffered input stream. This concept goes back to the days of blocked tape drives and special io devices. In the C# universe, the only real function that this class performs is to ensure that files have the correct "record" size, or other tars will complain.

You should never have a need to access this class directly. TarBuffers are created by Tar IO Streams.

For a list of all members of this type, see TarBuffer Members.


public class TarBuffer

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

Namespace: ISharpCode.SharpZipLib.Tar


See Also

TarBuffer Members  |  ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
## TarBuffer Members

### TarBuffer overview

#### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S BlockSize</td>
<td>The size of a block in a tar archive in bytes.</td>
</tr>
<tr>
<td>S DefaultBlockFactor</td>
<td>The number of blocks in a default record.</td>
</tr>
<tr>
<td>S DefaultRecordSize</td>
<td>The size in bytes of a default record.</td>
</tr>
</tbody>
</table>

#### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S CreateInputTarBuffer</td>
<td>Overloaded. Create TarBuffer for reading with default BlockFactor.</td>
</tr>
<tr>
<td>S CreateOutputTarBuffer</td>
<td>Overloaded. Construct TarBuffer for writing with default BlockFactor.</td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S BlockFactor</td>
<td>Get the Blocking factor for the buffer</td>
</tr>
<tr>
<td>S CurrentBlock</td>
<td>Get the current block number, within the current record, zero based.</td>
</tr>
<tr>
<td>S CurrentRecord</td>
<td>Get the current record number.</td>
</tr>
<tr>
<td>S RecordSize</td>
<td>Get the record size for this buffer</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S Close</td>
<td>Close the TarBuffer. If this is an output buffer, also flush the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetBlockFactor</strong></td>
<td><strong>Obsolete.</strong> Get the TAR Buffer's block factor.</td>
</tr>
<tr>
<td><strong>GetCurrentBlockNum</strong></td>
<td><strong>Obsolete.</strong> Get the current block number, within the current record, zero based.</td>
</tr>
<tr>
<td><strong>GetCurrentRecordNum</strong></td>
<td><strong>Obsolete.</strong> Get the current record number.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetRecordSize</strong></td>
<td><strong>Obsolete.</strong> Get the TAR Buffer's record size.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>IsEOFBlock</strong></td>
<td>Determine if an archive block indicates End of Archive. End of archive is indicated by a block that consists entirely of null bytes. All remaining blocks for the record should also be null's. However some older tars only do a couple of null blocks (Old GNU tar for one) and also partial records</td>
</tr>
<tr>
<td><strong>ReadBlock</strong></td>
<td>Read a block from the input stream.</td>
</tr>
<tr>
<td><strong>SkipBlock</strong></td>
<td>Skip over a block on the input stream.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>WriteBlock</strong></td>
<td>Overloaded. Write a block of data to the archive.</td>
</tr>
</tbody>
</table>

**Protected Instance Constructors**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TarBuffer Constructor</strong></td>
<td>Construct a default TarBuffer</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarBuffer Constructor

Construct a default TarBuffer

```java
protected TarBuffer();
```

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarBuffer Fields

The fields of the `TarBuffer` class are listed below. For a complete list of `TarBuffer` class members, see the `TarBuffer Members` topic.

Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BlockSize</code></td>
<td>The size of a block in a tar archive in bytes.</td>
</tr>
<tr>
<td><code>DefaultBlockFactor</code></td>
<td>The number of blocks in a default record.</td>
</tr>
<tr>
<td><code>DefaultRecordSize</code></td>
<td>The size in bytes of a default record.</td>
</tr>
</tbody>
</table>

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
**TarBuffer.BlockSize Field**

The size of a block in a tar archive in bytes.

```csharp
public const int BlockSize = 512;
```

Remarks

This is 512 bytes.

See Also

[TarBuffer Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarBuffer.DefaultBlockFactor Field**

The number of blocks in a default record.

```csharp
public const int DefaultBlockFactor = 20;
```

**Remarks**

The default value is 20 blocks per record.

**See Also**

[ICSharpCode.SharpZipLib.Tar Namespace](#)
The size in bytes of a default record.

```csharp
public const int DefaultRecordSize = 10240;
```

**Remarks**

The default size is 10KB.

**See Also**

TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarBuffer Properties

The properties of the TarBuffer class are listed below. For a complete list of TarBuffer class members, see the TarBuffer Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockFactor</td>
<td>Get the Blocking factor for the buffer</td>
</tr>
<tr>
<td>CurrentBlock</td>
<td>Get the current block number, within the current record, zero based.</td>
</tr>
<tr>
<td>CurrentRecord</td>
<td>Get the current record number.</td>
</tr>
<tr>
<td>RecordSize</td>
<td>Get the record size for this buffer</td>
</tr>
</tbody>
</table>

See Also

TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarBuffer.BlockFactor Property

Get the Blocking factor for the buffer

```csharp
public int BlockFactor {get;}
```

See Also

TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
Get the current block number, within the current record, zero based.

```csharp
public int CurrentBlock {get;}
```

**See Also**

[TarBuffer Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarBuffer.CurrentRecord Property

Get the current record number.

```csharp
public int CurrentRecord {get;}
```

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
TarBuffer.RecordSize Property

Get the record size for this buffer

```csharp
public int RecordSize {get;}
```

See Also

TarBuffer Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
## TarBuffer Methods

The methods of the **TarBuffer** class are listed below. For a complete list of **TarBuffer** class members, see the [TarBuffer Members](#) topic.

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateInputTarBuffer</strong></td>
<td>Overloaded. Create TarBuffer for reading with default BlockFactor</td>
</tr>
<tr>
<td><strong>CreateOutputTarBuffer</strong></td>
<td>Overloaded. Construct TarBuffer for writing with default BlockFactor</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Close</strong></td>
<td>Close the TarBuffer. If this is an output buffer, also flush the current block before closing.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetBlockFactor</strong></td>
<td><strong>Obsolete.</strong> Get the TAR Buffer's block factor</td>
</tr>
<tr>
<td><strong>GetCurrentBlockNum</strong></td>
<td><strong>Obsolete.</strong> Get the current block number, within the current record, zero based.</td>
</tr>
<tr>
<td><strong>GetCurrentRecordNum</strong></td>
<td><strong>Obsolete.</strong> Get the current record number.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetRecordSize</strong></td>
<td><strong>Obsolete.</strong> Get the TAR Buffer's record size.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>IsEOFBlock</code></td>
<td>Determine if an archive block indicates End of Archive. End of archive is indicated by a block that consists entirely of null bytes. All remaining blocks for the record should also be null's. However some older tars only do a couple of null blocks (Old GNU tar for one) and also partial records.</td>
</tr>
<tr>
<td><code>ReadBlock</code></td>
<td>Read a block from the input stream.</td>
</tr>
<tr>
<td><code>SkipBlock</code></td>
<td>Skip over a block on the input stream.</td>
</tr>
<tr>
<td><code>ToString (inherited from Object)</code></td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>WriteBlock</code></td>
<td>Overloaded. Write a block of data to the archive.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize (inherited from Object)</code></td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone (inherited from Object)</code></td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

**See Also**

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
TarBuffer.Close Method

Close the TarBuffer. If this is an output buffer, also flush the current block before closing.

```java
public void Close();
```

See Also

TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarBuffer.CreateInputTarBuffer Method

Create TarBuffer for reading with default BlockFactor

**Overload List**

Create TarBuffer for reading with default BlockFactor

    public static TarBuffer CreateInputTarBuffer(Stream);

Construct TarBuffer for reading inputStream setting BlockFactor

    public static TarBuffer CreateInputTarBuffer(Stream,int);

See Also

[ TarBuffer Class ] | [ ISharpCode.SharpZipLib.Tar Namespace ]
TarBuffer.CreateInputTarBuffer Method (Stream)

Create TarBuffer for reading with default BlockFactor

```java
public static TarBuffer CreateInputTarBuffer(Stream inputStream);
```

Parameters

`inputStream`
Stream to buffer

Return Value

TarBuffer

See Also

ICSharpCode SharpZipLib Class Library
**TarBuffer.CreateInputTarBuffer Method (Stream, Int32)**

Construct TarBuffer for reading inputStream setting BlockFactor

```csharp
public static TarBuffer CreateInputTarBuffer(
    Stream inputStream,
    int blockFactor
);
```

**Parameters**

* inputStream
  Stream to buffer

* blockFactor
  Blocking factor to apply

**Return Value**

TarBuffer

**See Also**

TarBuffer.CreateOutputTarBuffer Method

Construct TarBuffer for writing with default BlockFactor

Overload List

Construct TarBuffer for writing with default BlockFactor

public static TarBuffer CreateOutputTarBuffer(Stream);

Construct TarBuffer for writing Tar output to streams.

public static TarBuffer CreateOutputTarBuffer(Stream,int);

See Also

TarBuffer Class  |  ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarBuffer.CreateOutputTarBuffer Method (Stream)

Construct TarBuffer for writing with default BlockFactor

```java
public static TarBuffer CreateOutputTarBuffer(
    Stream outputStream
);
```

Parameters

`outputStream`
output stream for buffer

Return Value

TarBuffer

See Also

ICSharpCode SharpZipLib Class Library
Construct TarBuffer for writing Tar output to streams.

```csharp
public static TarBuffer CreateOutputTarBuffer(
    Stream outputStream,
    int blockFactor
);
```

**Parameters**

- `outputStream`  
  Output stream to write to.

- `blockFactor`  
  Blocking factor to apply

**Return Value**

TarBuffer

**See Also**

**NOTE:** This method is now obsolete.

Use BlockFactor property instead

---

Get the TAR Buffer's block factor

```csharp
public int GetBlockFactor();
```

See Also

[TarBuffer Class]([link](#)) | [ICSharpCode.SharpZipLib.Tar Namespace]([link](#))
ICSharpCode SharpZipLib Class Library
NOTE: This method is now obsolete.
Use CurrentBlock property instead

Get the current block number, within the current record, zero based.

```csharp
public int GetCurrentBlockNum();
```

Return Value

The current zero based block number.

Remarks

The absolute block number = (record number * block factor) + block number.

See Also

TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
NOTE: This method is now obsolete.
Use CurrentRecord property instead

Get the current record number.

```java
public int GetCurrentRecordNum();
```

Return Value

The current zero based record number.

See Also

TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace
**NOTE:** This method is now obsolete.
Use `RecordSize` property instead

Get the TAR Buffer's record size.

```csharp
public int GetRecordSize();
```

See Also

[TarBuffer Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarBuffer.IsEOFBlock Method

Determine if an archive block indicates End of Archive. End of archive is indicated by a block that consists entirely of null bytes. All remaining blocks for the record should also be null's However some older tars only do a couple of null blocks (Old GNU tar for one) and also partial records

```csharp
public bool IsEOFBlock(byte[] block);
```

Parameters

*block*

The data block to check.

See Also

TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarBuffer.ReadBlock Method

Read a block from the input stream.

```csharp
public byte[] ReadBlock();
```

Return Value

The block of data read.

See Also

[ TarBuffer Class ](https://icsharpcode.net/SharpZipLib/) | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarBuffer.SkipBlock Method

Skip over a block on the input stream.

```csharp
public void SkipBlock();
```

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarBuffer.WriteBlock Method

Write a block of data to the archive.

**Overload List**

Write a block of data to the archive.

```java
public void WriteBlock(byte[]);
```

Write an archive record to the archive, where the record may be inside of a larger array buffer. The buffer must be "offset plus record size" long.

```java
public void WriteBlock(byte[], int);
```

**See Also**

TarBuffer Class | ISharpCode.SharpZipLib.Tar.Namespace
ICSharpCode SharpZipLib Class Library
TarBuffer.WriteBlock Method (Byte[])  

Write a block of data to the archive.

```csharp
public void WriteBlock(byte[] block);
```

Parameters

`block`

The data to write to the archive.

See Also

- TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace | TarBuffer.WriteBlock Overload List
TarBuffer.WriteBlock Method (Byte[], Int32)

Write an archive record to the archive, where the record may be inside of a larger array buffer. The buffer must be "offset plus record size" long.

```csharp
public void WriteBlock(
    byte[] buffer,
    int offset
);
```

Parameters

- **buffer**
  - The buffer containing the record data to write.

- **offset**
  - The offset of the record data within buffer.

See Also

- [TarBuffer Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#) | [TarBuffer.WriteBlock Overload List](#)
ICSharpCode SharpZipLib Class Library
TarEntry Class

This class represents an entry in a Tar archive. It consists of the entry's header, as well as the entry's File. Entries can be instantiated in one of three ways, depending on how they are to be used.

TarEntries that are created from the header bytes read from an archive are instantiated with the TarEntry(byte[]) constructor. These entries will be used when extracting from or listing the contents of an archive. These entries have their header filled in using the header bytes. They also set the File to null, since they reference an archive entry not a file.

TarEntries that are created from files that are to be written into an archive are instantiated with the CreateEntryFromFile(string) pseudo constructor. These entries have their header filled in using the File's information. They also keep a reference to the File for convenience when writing entries.

Finally, TarEntries can be constructed from nothing but a name. This allows the programmer to construct the entry by hand, for instance when only an InputStream is available for writing to the archive, and the header information is constructed from other information. In this case the header fields are set to defaults and the File is set to null.

TarHeader

For a list of all members of this type, see TarEntry Members.


```
public class TarEntry : ICloneable
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

Namespace: ISharpCode.SharpZipLib.Tar

Assembly: ISharpCode.SharpZipLib (in
See Also

TarEntry Members | IICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarEntry Members

**TarEntry overview**

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdjustEntryName</td>
<td>Convenience method that will modify an entry's name directly in place in an entry header buffer byte array.</td>
</tr>
<tr>
<td>CreateEntryFromFile</td>
<td>Construct an entry for a file. File is set to file, and the header is constructed from information from the file.</td>
</tr>
<tr>
<td>CreateTarEntry</td>
<td>Construct an entry with only a name. This allows the programmer to construct the entry's header &quot;by hand&quot;.</td>
</tr>
<tr>
<td>NameTarHeader</td>
<td>Fill in a TarHeader given only the entry's name.</td>
</tr>
</tbody>
</table>

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TarEntry</td>
<td>Overloaded. Initializes a new instance of the TarEntry class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Get this entry's file.</td>
</tr>
<tr>
<td>GroupId</td>
<td>Get/set this entry's group id.</td>
</tr>
<tr>
<td>GroupName</td>
<td>Get/set this entry's group name.</td>
</tr>
<tr>
<td>IsDirectory</td>
<td>Return true if this entry represents a directory, false otherwise</td>
</tr>
<tr>
<td>ModTime</td>
<td>Get/Set the modification time for this entry</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name</td>
<td>Get/Set this entry's name.</td>
</tr>
<tr>
<td>Size</td>
<td>Get/set this entry's recorded file size.</td>
</tr>
<tr>
<td>TarHeader</td>
<td>Get this entry's header.</td>
</tr>
<tr>
<td>UserId</td>
<td>Get/set this entry's user id.</td>
</tr>
<tr>
<td>UserName</td>
<td>Get/set this entry's user name.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Clone this tar entry.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determine if the two entries are equal. Equality is determined by the header names being equal.</td>
</tr>
<tr>
<td>GetDirectoryEntries</td>
<td>Get entries for all files present in this entries directory. If this entry doesn't represent a directory zero entries are returned.</td>
</tr>
<tr>
<td>GetFileTarHeader</td>
<td>Fill in a TarHeader with information from a File.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Must be overridden when you override Equals.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>IsDescendent</td>
<td>Determine if the given entry is a descendant of this entry. Descendancy is determined by the name of the descendant starting with this entry's name.</td>
</tr>
<tr>
<td>SetIds</td>
<td>Convenience method to set this entry's group and user ids.</td>
</tr>
<tr>
<td>SetNames</td>
<td>Convenience method to set this entry's group and user names.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td>the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>WriteEntryHeader</strong></td>
<td>Write an entry's header information to a header buffer.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

| **Finalize** (inherited from **Object**) | Allows an **Object** to attempt to free resources and perform other cleanup operations before the **Object** is reclaimed by garbage collection. |
| **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

### See Also

[TarEntry Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarEntry Constructor**

Construct an entry from an archive's header bytes. File is set to null.

**Overload List**

Construct a TarEntry using the *header* provided

```csharp
public TarEntry(TarHeader);
```

Construct an entry from an archive's header bytes. File is set to null.

```csharp
public TarEntry(byte[]);
```

**See Also**

TarEntry Class  |  ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarEntry Constructor (Byte[])**

Construct an entry from an archive's header bytes. File is set to null.

```csharp
public TarEntry(byte[] headerBuffer);
```

**Parameters**

- `headerBuffer`  
  The header bytes from a tar archive entry.

**See Also**

- [TarEntry Class](#)  
- [ICSharpCode.SharpZipLib.Tar Namespace](#)  
- [TarEntry Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
**TarEntry Constructor (TarHeader)**

Construct a TarEntry using the *header* provided

```csharp
public TarEntry(TarHeader header);
```

**Parameters**

*header*

Header details for entry

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#) | [TarEntry Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
TarEntry Properties

The properties of the **TarEntry** class are listed below. For a complete list of **TarEntry** class members, see the **TarEntry Members** topic.

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>File</code></td>
<td>Get this entry's file.</td>
</tr>
<tr>
<td><code>GroupId</code></td>
<td>Get/set this entry's group id.</td>
</tr>
<tr>
<td><code>GroupName</code></td>
<td>Get/set this entry's group name.</td>
</tr>
<tr>
<td><code>IsDirectory</code></td>
<td>Return true if this entry represents a directory, false otherwise</td>
</tr>
<tr>
<td><code>ModTime</code></td>
<td>Get/Set the modification time for this entry</td>
</tr>
<tr>
<td><code>Name</code></td>
<td>Get/Set this entry's name.</td>
</tr>
<tr>
<td><code>Size</code></td>
<td>Get/set this entry's recorded file size.</td>
</tr>
<tr>
<td><code>TarHeader</code></td>
<td>Get this entry's header.</td>
</tr>
<tr>
<td><code>UserId</code></td>
<td>Get/set this entry's user id.</td>
</tr>
<tr>
<td><code>UserName</code></td>
<td>Get/set this entry's user name.</td>
</tr>
</tbody>
</table>

**See Also**

[TarEntry Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarEntry.File Property**

Get this entry's file.

```csharp
public string File {get;}
```

See Also

[TarEntry Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarEntry.GroupId Property**

Get/set this entry's group id.

```csharp
public int GroupId {get; set;}
```

See Also

[TarEntry Class] [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarEntry.GroupName Property

Get/set this entry's group name.

```csharp
public string GroupName {get; set;}
```

See Also

TarEntry Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarEntry.IsDirectory Property

Return true if this entry represents a directory, false otherwise

```csharp
public bool IsDirectory {get;}
```

See Also

TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarEntry.ModTime Property

Get/Set the modification time for this entry

```csharp
public System.DateTime ModTime {get; set;}
```

See Also

[TarEntry Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarEntry.Name Property**

Get/Set this entry's name.

```csharp
public string Name {get; set;}
```

See Also

[TarEntry Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
**TarEntry.Size Property**

Get/set this entry's recorded file size.

```
public long Size {get; set;}
```

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarEntry.TarHeader Property**

Get this entry's header.

```csharp
public TarHeader TarHeader {get;}
```

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarEntry.UserId Property**

Get/set this entry's user id.

```csharp
public int UserId {get; set;}
```

See Also

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarEntry UserName Property

Get/set this entry's user name.

```csharp
public string UserName {get; set;}
```

See Also

TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
The methods of the **TarEntry** class are listed below. For a complete list of **TarEntry** class members, see the **TarEntry Members** topic.

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AdjustEntryName</strong></td>
<td>Convenience method that will modify an entry's name directly in place in an entry header buffer byte array.</td>
</tr>
<tr>
<td><strong>CreateEntryFromFile</strong></td>
<td>Construct an entry for a file. File is set to file, and the header is constructed from information from the file.</td>
</tr>
<tr>
<td><strong>CreateTarEntry</strong></td>
<td>Construct an entry with only a <em>name</em>. This allows the programmer to construct the entry's header &quot;by hand&quot;.</td>
</tr>
<tr>
<td><strong>NameTarHeader</strong></td>
<td>Fill in a TarHeader given only the entry's name.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clone</strong></td>
<td>Clone this tar entry.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determine if the two entries are equal. Equality is determined by the header names being equal.</td>
</tr>
<tr>
<td><strong>GetDirectoryEntries</strong></td>
<td>Get entries for all files present in this entries directory. If this entry doesn't represent a directory zero entries are returned.</td>
</tr>
<tr>
<td><strong>GetFileTarHeader</strong></td>
<td>Fill in a TarHeader with information from a File.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Must be overridden when you override Equals.</td>
</tr>
</tbody>
</table>
**GetType (inherited from Object)**
Gets the **Type** of the current instance.

**IsDescendent**
Determine if the given entry is a descendant of this entry. Descendancy is determined by the name of the descendant starting with this entry's name.

**SetIds**
Convenience method to set this entry's group and user ids.

**SetNames**
Convenience method to set this entry's group and user names.

**ToString (inherited from Object)**
Returns a **String** that represents the current **Object**.

**WriteEntryHeader**
Write an entry's header information to a header buffer.

### Protected Instance Methods

**Finalize (inherited from Object)**
Allows an **Object** to attempt to free resources and perform other cleanup operations before the **Object** is reclaimed by garbage collection.

**MemberwiseClone (inherited from Object)**
Creates a shallow copy of the current **Object**.

### See Also

TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
TarEntry.AdjustEntryName Method

Convenience method that will modify an entry's name directly in place in an entry header buffer byte array.

```csharp
public static void AdjustEntryName(
    byte[] buffer,
    string newName
);
```

Parameters

- **buffer**
  The buffer containing the entry header to modify.

- **newName**
  The new name to place into the header buffer.

See Also

- [TarEntry Class](#)
- [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarEntry.Clone Method**

Clone this tar entry.

```csharp
public object Clone();
```

**Return Value**

Returns a clone of this entry.

**Implements**

ICloneable.Clone

**See Also**

TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
Construct an entry for a file. File is set to file, and the header is constructed from information from the file.

```csharp
public static TarEntry CreateEntryFromFile(
    string fileName
);
```

**Parameters**

*fileName*

The file that the entry represents.

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarEntry.CreateTarEntry Method**

Construct an entry with only a *name*. This allows the programmer to construct the entry's header "by hand".

```csharp
public static TarEntry CreateTarEntry(
    string name
);
```

See Also

TarEntry Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarEntry.Equals Method**

Determine if the two entries are equal. Equality is determined by the header names being equal.

```csharp
public override bool Equals(object obj);
```

**Return Value**

True if the entries are equal.

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarEntry.GetDirectoryEntries Method**

Get entries for all files present in this entries directory. If this entry does not represent a directory zero entries are returned.

```csharp
public TarEntry[] GetDirectoryEntries();
```

**Return Value**

An array of TarEntry's for this entry's children.

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarEntry.GetFileTarHeader Method

Fill in a TarHeader with information from a File.

```csharp
public void GetFileTarHeader(
    TarHeader header,
    string file
);
```

Parameters

- `header`  
The TarHeader to fill in.

- `file`  
The file from which to get the header information.

See Also

- TarEntry Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarEntry.GetHashCode Method**

Must be overridden when you override Equals.

```csharp
public override int GetHashCode();
```

See Also

[TarEntry Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarEntry.IsDescendent Method**

Determine if the given entry is a descendant of this entry. Descendancy is determined by the name of the descendant starting with this entry's name.

```csharp
public bool IsDescendent(TarEntry toTest);
```

**Parameters**

`toTest`  
Entry to be checked as a descendent of this.

**Return Value**

True if entry is a descendant of this.

**See Also**

TarEntry Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarEntry.NameTarHeader Method**

Fill in a TarHeader given only the entry's name.

```csharp
public static void NameTarHeader(
    TarHeader header,
    string name
);
```

**Parameters**

- **header**
  - The TarHeader to fill in.

- **name**
  - The tar entry name.

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
TarEntry.SetIds Method

Convenience method to set this entry's group and user ids.

```csharp
public void SetIds(
    int userId,
    int groupId
);
```

Parameters

- **userId**
  This entry's new user id.

- **groupId**
  This entry's new group id.

See Also

- TarEntry Class
- ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarEntry.SetNames Method

Convenience method to set this entry's group and user names.

```csharp
public void SetNames(
    string userName,
    string groupName
);
```

Parameters

- **userName**
  This entry's new user name.

- **groupName**
  This entry's new group name.

See Also

- TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarEntry.WriteEntryHeader Method**

Write an entry's header information to a header buffer.

```csharp
public void WriteEntryHeader(byte[] outBuffer);
```

**Parameters**

* `outBuffer`  
  The tar entry header buffer to fill in.

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
**TarException Class**

TarExceptions are used for exceptions specific to tar classes and code.

For a list of all members of this type, see *TarException Members*.

```csharp
System.Object System.Exception System.ApplicationException
ICSharpCode.SharpZipLib.SharpZipBaseException
ICSharpCode.SharpZipLib.Tar.TarException
ICSharpCode.SharpZipLib.Tar.InvalidHeaderException
```

**public class TarException :**

```
SharpZipBaseException
```

**Thread Safety**

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

- **Namespace:** [ICSharpCode.SharpZipLib.Tar](https://msdn.microsoft.com)
- **Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

**See Also**

[TarException Members](https://msdn.microsoft.com) | [ICSharpCode.SharpZipLib.Tar Namespace](https://msdn.microsoft.com)
ICSharpCode SharpZipLib Class Library
# TarException Members

## TarException overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TarException</strong></td>
<td>Overloaded. Initializes a new instance of the TarException class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HelpLink</strong></td>
<td>(inherited from Exception) Gets or sets a link to the help file associated with this exception.</td>
</tr>
<tr>
<td><strong>InnerException</strong></td>
<td>(inherited from Exception) Gets the Exception instance that caused the current exception.</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>(inherited from Exception) Gets a message that describes the current exception.</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>(inherited from Exception) Gets or sets the name of the application or the object that causes the error.</td>
</tr>
<tr>
<td><strong>StackTrace</strong></td>
<td>(inherited from Exception) Gets a string representation of the frames on the call stack at the time the current exception was thrown.</td>
</tr>
<tr>
<td><strong>TargetSite</strong></td>
<td>(inherited from Exception) Gets the method that throws the current exception.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>(inherited from Object) Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetBaseException</strong></td>
<td>(inherited from Exception) When overridden in a derived class, returns the Exception that</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetObjectData</strong> (inherited from Exception)</td>
<td>When overridden in a derived class, sets the <strong>SerializationInfo</strong> with information about the exception.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Exception)</td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TarException</strong></td>
<td>Overloaded. Initializes a new instance of the TarException class.</td>
</tr>
</tbody>
</table>

### Protected Instance Properties

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HResult</strong> (inherited from Exception)</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>
See Also

TarException Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarException Constructor**

Deserialization constructor

**Overload List**

Initialises a new instance of the TarException class.

```
public TarException();
```

Deserialization constructor

```
protected TarException(SerializationInfo, StreamingContext);
```

Initialises a new instance of the TarException class with a specified message.

```
public TarException(string);
```

```
public TarException(string, Exception);
```

**See Also**

[TarException Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
Deserialization constructor

```csharp
protected TarException(
    SerializationInfo info,
    StreamingContext context
);
```

**Parameters**

- `info`  
  `SerializationInfo` for this constructor

- `context`  
  `StreamingContext` for this constructor

**See Also**

- TarException Class | ISharpCode.SharpZipLib.Tar Namespace | TarException Constructor Overload List
ICSharpCode SharpZipLib Class Library
Initialises a new instance of the TarException class.

```java
public TarException();
```

See Also

- TarException Class
- ISharpCode.SharpZipLib.Tar Namespace
- TarException Constructor Overload List
ICSharpCode SharpZipLib Class Library
Initialises a new instance of the TarException class with a specified message.

```csharp
public TarException(string message);
```

**Parameters**

*message*

The message that describes the error.

**See Also**

TarException Class | ISharpCode.SharpZipLib.Tar Namespace | TarException Constructor Overload List
ICSharpCode SharpZipLib Class Library
public TarException(
    string message,
    Exception exception
);

Parameters

message
   A message describing the error.

exception
   The exception that is the cause of the current exception.

See Also

TarException Class | ISharpCode.SharpZipLib.Tar Namespace | TarException Constructor Overload List
ICSharpCode SharpZipLib Class Library
**TarHeader Class**

This class encapsulates the Tar Entry Header used in Tar Archives. The class also holds a number of tar constants, used mostly in headers.

For a list of all members of this type, see TarHeader Members.

**System.Object**  ISharpCode.SharpZipLib.Tar.TarHeader

```csharp
public class TarHeader : ICloneable
```

**Thread Safety**

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** ISharpCode.SharpZipLib.Tar

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

**See Also**

TarHeader Members | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
## TarHeader Members

### TarHeader overview

### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKSUMLEN</td>
<td>The length of the checksum field in a header buffer.</td>
</tr>
<tr>
<td>CHKSUMOFS</td>
<td>Offset of checksum in a header buffer.</td>
</tr>
<tr>
<td>DEVLEN</td>
<td>The length of the devices field in a header buffer.</td>
</tr>
<tr>
<td>GIDLEN</td>
<td>The length of the group id field in a header buffer.</td>
</tr>
<tr>
<td>GNAMELEN</td>
<td>The length of the group name field in a header buffer.</td>
</tr>
<tr>
<td>GNU_TMAGIC</td>
<td>The magic tag representing an old GNU tar archive where version is included</td>
</tr>
<tr>
<td></td>
<td>in magic and overwrites it.</td>
</tr>
<tr>
<td>LF_ACL</td>
<td>Solaris access control list file type</td>
</tr>
<tr>
<td>LF_BLK</td>
<td>Block device file type.</td>
</tr>
<tr>
<td>LF_CHR</td>
<td>Character device file type.</td>
</tr>
<tr>
<td>LF_CONTIG</td>
<td>Contiguous file type.</td>
</tr>
<tr>
<td>LF_DIR</td>
<td>Directory file type.</td>
</tr>
<tr>
<td>LF_EXTATTR</td>
<td>Solaris Extended Attribute File</td>
</tr>
<tr>
<td>LF_FIFO</td>
<td>FIFO (pipe) file type.</td>
</tr>
<tr>
<td>LF_GHDR</td>
<td>Posix.1 2001 global extended header</td>
</tr>
<tr>
<td>LF_GNU_DUMPDIR</td>
<td>GNU dir dump file type This is a dir entry that contains the names of files</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LF_GNU_LONGLINK</td>
<td>Identifies the next file on the tape as having a long link name</td>
</tr>
<tr>
<td>LF_GNU_LONGNAME</td>
<td>Identifies the next file on the tape as having a long name</td>
</tr>
<tr>
<td>LF_GNU_MULTIVOL</td>
<td>Continuation of a file that began on another volume</td>
</tr>
<tr>
<td>LF_GNU_NAMES</td>
<td>For storing filenames that don't fit in the main header (old GNU)</td>
</tr>
<tr>
<td>LF_GNU_SPARSE</td>
<td>GNU Sparse file</td>
</tr>
<tr>
<td>LF_GNU_VOLHDR</td>
<td>GNU Tape/volume header ignore on extraction</td>
</tr>
<tr>
<td>LF_LINK</td>
<td>Link file type.</td>
</tr>
<tr>
<td>LF_META</td>
<td>Inode (metadata only) no file content</td>
</tr>
<tr>
<td>LF_NORMAL</td>
<td>Normal file type.</td>
</tr>
<tr>
<td>LF_OLDNORM</td>
<td>The &quot;old way&quot; of indicating a normal file.</td>
</tr>
<tr>
<td>LF_SYMLINK</td>
<td>Symbolic link file type.</td>
</tr>
<tr>
<td>LF_XHDR</td>
<td>Posix.1 2001 extended header</td>
</tr>
<tr>
<td>MAGICLEN</td>
<td>The length of the magic field in a header buffer.</td>
</tr>
<tr>
<td>MODELEN</td>
<td>The length of the mode field in a header buffer.</td>
</tr>
<tr>
<td>MODTIMELEN</td>
<td>The length of the modification time field in a header buffer.</td>
</tr>
<tr>
<td>NAMELEN</td>
<td>The length of the name field in a header buffer.</td>
</tr>
<tr>
<td>SIZELEN</td>
<td>The length of the size field in a header buffer.</td>
</tr>
<tr>
<td><strong>TMAGIC</strong></td>
<td>The magic tag representing a POSIX tar archive. (includes trailing NULL)</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>UIDLEN</strong></td>
<td>The length of the user id field in a header buffer.</td>
</tr>
<tr>
<td><strong>UNAMELEN</strong></td>
<td>The length of the user name field in a header buffer.</td>
</tr>
<tr>
<td><strong>VERSIONLEN</strong></td>
<td>The length of the version field in a header buffer.</td>
</tr>
</tbody>
</table>

### Public Static Methods

<table>
<thead>
<tr>
<th><strong>GetAsciiBytes</strong></th>
<th>Add a string to a buffer as a collection of ascii bytes.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetLongOctalBytes</strong></td>
<td>Put an octal representation of a value into a buffer</td>
</tr>
<tr>
<td><strong>GetNameBytes</strong></td>
<td>Overloaded. Add <em>name</em> to the buffer as a collection of bytes.</td>
</tr>
<tr>
<td><strong>GetOctalBytes</strong></td>
<td>Put an octal representation of a value into a buffer</td>
</tr>
<tr>
<td><strong>ParseName</strong></td>
<td>Parse a name from a header buffer.</td>
</tr>
<tr>
<td><strong>ParseOctal</strong></td>
<td>Parse an octal string from a header buffer.</td>
</tr>
</tbody>
</table>

### Public Instance Constructors

| **TarHeader Constructor** | Initialise a default TarHeader instance                                |

### Public Instance Properties

<p>| <strong>Checksum</strong>           | Get the entry's checksum. This is only valid/updated after writing or reading an entry. |</p>
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DevMajor</strong></td>
<td>Get/set the entry's major device number.</td>
</tr>
<tr>
<td><strong>DevMinor</strong></td>
<td>Get/set the entry's minor device number.</td>
</tr>
<tr>
<td><strong>GroupId</strong></td>
<td>Get/set the entry's group id.</td>
</tr>
<tr>
<td><strong>GroupName</strong></td>
<td>Get/set the entry's group name.</td>
</tr>
<tr>
<td><strong>IsChecksumValid</strong></td>
<td>Get value of true if the header checksum is valid, false otherwise.</td>
</tr>
<tr>
<td><strong>LinkName</strong></td>
<td>The entry's link name.</td>
</tr>
<tr>
<td><strong>Magic</strong></td>
<td>Get/set the entry's magic tag.</td>
</tr>
<tr>
<td><strong>Mode</strong></td>
<td>Get/set the entry's Unix style permission mode.</td>
</tr>
<tr>
<td><strong>ModTime</strong></td>
<td>Get/set the entry's modification time.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Get/set the name for this tar entry.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Get/set the entry's size.</td>
</tr>
<tr>
<td><strong>TypeFlag</strong></td>
<td>Get/set the entry's type flag.</td>
</tr>
<tr>
<td><strong>UserId</strong></td>
<td>The entry's user id.</td>
</tr>
<tr>
<td><strong>UserName</strong></td>
<td>The entry's user name.</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>The entry's version.</td>
</tr>
</tbody>
</table>

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clone</strong></td>
<td>Clone a TAR header.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines if this instance is equal to the specified object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Get a hash code for the current object.</td>
</tr>
<tr>
<td><strong>GetName</strong></td>
<td><strong>Obsolete.</strong> Get the name of this</td>
</tr>
</tbody>
</table>
**Protected Instance Methods**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

**See Also**

TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader Constructor

Initialise a default TarHeader instance

```csharp
public TarHeader();
```

See Also

TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarHeader Fields**

The fields of the **TarHeader** class are listed below. For a complete list of **TarHeader** class members, see the [TarHeader Members](#) topic.

### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CHKSUMLEN</code></td>
<td>The length of the checksum field in a header buffer.</td>
</tr>
<tr>
<td><code>CHKSUMOFS</code></td>
<td>Offset of checksum in a header buffer.</td>
</tr>
<tr>
<td><code>DEVLEN</code></td>
<td>The length of the devices field in a header buffer.</td>
</tr>
<tr>
<td><code>GIDLEN</code></td>
<td>The length of the group id field in a header buffer.</td>
</tr>
<tr>
<td><code>GNAMELEN</code></td>
<td>The length of the group name field in a header buffer.</td>
</tr>
<tr>
<td><code>GNU_TMAGIC</code></td>
<td>The magic tag representing an old GNU tar archive where version is included in magic and overwrites it</td>
</tr>
<tr>
<td><code>LF_ACL</code></td>
<td>Solaris access control list file type</td>
</tr>
<tr>
<td><code>LF_BLK</code></td>
<td>Block device file type.</td>
</tr>
<tr>
<td><code>LF_CHR</code></td>
<td>Character device file type.</td>
</tr>
<tr>
<td><code>LF_CONTIG</code></td>
<td>Contiguous file type.</td>
</tr>
<tr>
<td><code>LF_DIR</code></td>
<td>Directory file type.</td>
</tr>
<tr>
<td><code>LF_EXTATTR</code></td>
<td>Solaris Extended Attribute File</td>
</tr>
<tr>
<td><code>LF_FIFO</code></td>
<td>FIFO (pipe) file type.</td>
</tr>
<tr>
<td><code>LF_GHDR</code></td>
<td>Posix.1 2001 global extended header</td>
</tr>
<tr>
<td><code>LF_GNU_DUMPDIR</code></td>
<td>GNU dir dump file type. This is a dir entry that contains the</td>
</tr>
</tbody>
</table>
names of files that were in the dir at the time the dump was made

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{LF_GNU_LONGLINK}$</td>
<td>Identifies the next file on the tape as having a long link name</td>
</tr>
<tr>
<td>$\text{LF_GNU_LONGNAME}$</td>
<td>Identifies the next file on the tape as having a long name</td>
</tr>
<tr>
<td>$\text{LF_GNU_MULTIVOL}$</td>
<td>Continuation of a file that began on another volume</td>
</tr>
<tr>
<td>$\text{LF_GNU_NAMES}$</td>
<td>For storing filenames that don't fit in the main header (old GNU)</td>
</tr>
<tr>
<td>$\text{LF_GNU_SPARSE}$</td>
<td>GNU Sparse file</td>
</tr>
<tr>
<td>$\text{LF_GNU_VOLHDR}$</td>
<td>GNU Tape/volume header ignore on extraction</td>
</tr>
<tr>
<td>$\text{LF_LINK}$</td>
<td>Link file type.</td>
</tr>
<tr>
<td>$\text{LF_META}$</td>
<td>Inode (metadata only) no file content</td>
</tr>
<tr>
<td>$\text{LF_NORMAL}$</td>
<td>Normal file type.</td>
</tr>
<tr>
<td>$\text{LF_OLDNORM}$</td>
<td>The &quot;old way&quot; of indicating a normal file.</td>
</tr>
<tr>
<td>$\text{LF_SYMLINK}$</td>
<td>Symbolic link file type.</td>
</tr>
<tr>
<td>$\text{LF_XHDR}$</td>
<td>Posix.1 2001 extended header</td>
</tr>
<tr>
<td>$\text{MAGICLEN}$</td>
<td>The length of the magic field in a header buffer.</td>
</tr>
<tr>
<td>$\text{MODELEN}$</td>
<td>The length of the mode field in a header buffer.</td>
</tr>
<tr>
<td>$\text{MODTIMELEN}$</td>
<td>The length of the modification time field in a header buffer.</td>
</tr>
<tr>
<td>$\text{NAMELEN}$</td>
<td>The length of the name field in a header buffer.</td>
</tr>
<tr>
<td>$\text{SIZELEN}$</td>
<td>The length of the size field in a header buffer.</td>
</tr>
</tbody>
</table>
header buffer.

<table>
<thead>
<tr>
<th>$TMAGIC</th>
<th>The magic tag representing a POSIX tar archive. (includes trailing NULL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$UIDLEN</td>
<td>The length of the user id field in a header buffer.</td>
</tr>
<tr>
<td>$UNAMELEN</td>
<td>The length of the user name field in a header buffer.</td>
</tr>
<tr>
<td>$VERSIONLEN</td>
<td>The length of the version field in a header buffer.</td>
</tr>
</tbody>
</table>

See Also

TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarHeader.CHKSUMLEN Field**

The length of the checksum field in a header buffer.

```java
public const int CHKSUMLEN = 8;
```

**See Also**

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarHeader.CHKSUMOFS Field**

Offset of checksum in a header buffer.

```csharp
public const int CHKSUMOFS = 148;
```

**See Also**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.DEVLEN Field

The length of the devices field in a header buffer.

```
public const int DEVLEN = 8;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader.GIDLEN Field

The length of the group id field in a header buffer.

```csharp
public const int GIDLEN = 8;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarHeader.GNAMELEN Field**

The length of the group name field in a header buffer.

```csharp
public const int GNAMELEN = 32;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.GNU_TMAGIC Field

The magic tag representing an old GNU tar archive where version is included in magic and overwrites it

public const string GNU_TMAGIC = "ustar ";

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
Solaris access control list file type

```csharp
public const byte LF_ACL = 65;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader.LF_BLK Field

Block device file type.

```csharp
public const byte LF_BLK = 52;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarHeader.LF_CHR Field**

Character device file type.

```
public const byte LF_CHR = 51;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.LF_CONTIG Field

Contiguous file type.

```
public const byte LF_CONTIG = 55;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCodeSharpZipLibClassLibrary
TarHeader.LF_DIR Field

Directory file type.

```java
public const byte LF_DIR = 53;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
Solaris Extended Attribute File

```csharp
public const byte LF_EXTATTR = 69;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader.LF_FIFO Field

FIFO (pipe) file type.

```csharp
public const byte LF_FIFO = 54;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarHeader.LF_GHDR Field**

Posix.1 2001 global extended header

```
public const byte LF_GHDR = 103;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
GNU dir dump file type This is a dir entry that contains the names of files that were in the dir at the time the dump was made

```csharp
public const byte LF_GNU_DUMPDIR = 68;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
Identifies the next file on the tape as having a long link name

```csharp
public const byte LF_GNU_LONGLINK = 75;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarHeader.LF_GNU_LONGNAME Field**

Identifies the next file on the tape as having a long name

```csharp
public const byte LF_GNU_LONGNAME = 76;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarHeader.LF_GNU_MULTIVOL Field

Continuation of a file that began on another volume

```csharp
public const byte LF_GNU_MULTIVOL = 77;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.LF_GNU_NAMES Field

For storing filenames that don't fit in the main header (old GNU)

```csharp
public const byte LF_GNU_NAMES = 78;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader.LF_GNU_SPARSE Field

GNU Sparse file

```csharp
public const byte LF_GNU_SPARSE = 83;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
GNU Tape/volume header ignore on extraction

```
public const byte LF_GNU_VOLHDR = 86;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarHeader.LF_LINK Field**

Link file type.

```csharp
public const byte LF_LINK = 49;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.LF_META Field

Inode (metadata only) no file content

```java
public const byte LF_META = 73;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarHeader.LF_NORMAL Field

Normal file type.

```csharp
public const byte LF_NORMAL = 48;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarHeader.LF_OLDNORM Field

The "old way" of indicating a normal file.

```java
public const byte LF_OLDNORM = 0;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarHeader.LF_SYMLINK Field**

Symbolic link file type.

```csharp
public const byte LF_SYMLINK = 50;
```

See Also

[TarHeader Class]([#1](#)) | [ICSharpCode.SharpZipLib.Tar Namespace]([#2](#))
ICSharpCode SharpZipLib Class Library
TarHeader.LF_XHDR Field

Posix.1 2001 extended header

```
public const byte LF_XHDR = 120;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader.MAGICLEN Field

The length of the magic field in a header buffer.

```csharp
public const int MAGICLEN = 6;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader.MODELEN Field

The length of the mode field in a header buffer.

```csharp
public const int MODELEN = 8;
```

See Also

[TarHeader Class]  |  [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarHeader.MODTIMELEN Field

The length of the modification time field in a header buffer.

```csharp
public const int MODTIMELEN = 12;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
TarHeader.NAMELEN Field

The length of the name field in a header buffer.

```
public const int NAMELEN = 100;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
TarHeader.SIZELEN Field

The length of the size field in a header buffer.

```
public const int SIZELEN = 12;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
The magic tag representing a POSIX tar archive. (includes trailing NULL)

```csharp
public const string TMAGIC = "ustar ";
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
TarHeader.UIDLEN Field

The length of the user id field in a header buffer.

```csharp
public const int UIDLEN = 8;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarHeader.UNAMELEN Field

The length of the user name field in a header buffer.

```csharp
public const int UNAMELEN = 32;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarHeader.VERSIONLEN Field**

The length of the version field in a header buffer.

```csharp
public const int VERSIONLEN = 2;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
TarHeader Properties

The properties of the **TarHeader** class are listed below. For a complete list of **TarHeader** class members, see the **TarHeader Members** topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Checksum</strong></td>
<td>Get the entry's checksum. This is only valid/updated after writing or reading an entry.</td>
</tr>
<tr>
<td><strong>DevMajor</strong></td>
<td>Get/set the entry's major device number.</td>
</tr>
<tr>
<td><strong>DevMinor</strong></td>
<td>Get/set the entry's minor device number.</td>
</tr>
<tr>
<td><strong>GroupId</strong></td>
<td>Get/set the entry's group id.</td>
</tr>
<tr>
<td><strong>GroupName</strong></td>
<td>Get/set the entry's group name.</td>
</tr>
<tr>
<td><strong>IsChecksumValid</strong></td>
<td>Get value of true if the header checksum is valid, false otherwise.</td>
</tr>
<tr>
<td><strong>LinkName</strong></td>
<td>The entry's link name.</td>
</tr>
<tr>
<td><strong>Magic</strong></td>
<td>Get/set the entry's magic tag.</td>
</tr>
<tr>
<td><strong>Mode</strong></td>
<td>Get/set the entry's Unix style permission mode.</td>
</tr>
<tr>
<td><strong>ModTime</strong></td>
<td>Get/set the entry's modification time.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Get/set the name for this tar entry.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Get/set the entry's size.</td>
</tr>
<tr>
<td><strong>TypeFlag</strong></td>
<td>Get/set the entry's type flag.</td>
</tr>
<tr>
<td><strong>UserId</strong></td>
<td>The entry's user id.</td>
</tr>
<tr>
<td><strong>UserName</strong></td>
<td>The entry's user name.</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>The entry's version.</td>
</tr>
</tbody>
</table>

**See Also**

- TarHeader Class
- ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarHeader.Checksum Property**

Get the entry's checksum. This is only valid/updated after writing or reading an entry.

```csharp
public int Checksum {get;}
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
TarHeader.DevMajor Property

Get/set the entry's major device number.

```csharp
public int DevMajor {get; set;}
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
Get/set the entry's minor device number.

```csharp
public int DevMinor {get; set;}
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
**TarHeader.GroupId Property**

Get/set the entry's group id.

```csharp
public int GroupId {get; set;}
```

**Remarks**

This is only directly relevant to linux/unix systems. The default value is zero.

**See Also**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarHeader.GroupName Property**

Get/set the entry's group name.

```csharp
public string GroupName {get; set;}
```

**Remarks**

This is only directly relevant to unix systems.

**See Also**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.IsChecksumValid Property

Get value of true if the header checksum is valid, false otherwise.

```csharp
public bool IsChecksumValid {get;}
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader.LinkName Property

The entry's link name.

```csharp
public string LinkName {get; set;}
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td>Thrown when attempting to set LinkName to null.</td>
</tr>
</tbody>
</table>

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarHeader.Magic Property**

Get/set the entry's magic tag.

```csharp
public string Magic {get; set;}
```

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentNullException</code></td>
<td>Thrown when attempting to set Magic to null.</td>
</tr>
</tbody>
</table>

**See Also**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.Mode Property

Get/set the entry's Unix style permission mode.

```
public int Mode {get; set;}
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader.ModTime Property

Get/set the entry's modification time.

```csharp
public System.DateTime ModTime {get; set;}
```

Remarks

The modification time is only accurate to within a second.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>Thrown when setting the date time to less than 1/1/1970.</td>
</tr>
</tbody>
</table>

See Also

TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
TarHeader.Name Property

Get/set the name for this tar entry.

```csharp
public string Name {get; set;}
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentException</td>
<td>Thrown when attempting to set the property to null.</td>
</tr>
</tbody>
</table>

See Also

TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarHeader.Size Property**

Get/set the entry's size.

```csharp
public long Size {get; set;}
```

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentOutOfRangeException</code></td>
<td>Thrown when setting the size to less than zero.</td>
</tr>
</tbody>
</table>

**See Also**

- [TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.TypeFlag Property

Get/set the entry's type flag.

```csharp
public byte TypeFlag {get; set;}
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.UserId Property

The entry's user id.

```csharp
public int UserId {get; set;}
```

Remarks

This is only directly relevant to unix systems. The default is zero.

See Also

TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarHeader.UserName Property

The entry's user name.

```csharp
public string UserName {get; set;}
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarHeader.Version Property**

The entry's version.

```csharp
public string Version {get; set;}
```

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentNullException</code></td>
<td>Thrown when attempting to set Version to null.</td>
</tr>
</tbody>
</table>

**See Also**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarHeader Methods**

The methods of the *TarHeader* class are listed below. For a complete list of *TarHeader* class members, see the *TarHeader Members* topic.

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetAsciiBytes</code></td>
<td>Add a string to a buffer as a collection of ascii bytes.</td>
</tr>
<tr>
<td><code>GetLongOctalBytes</code></td>
<td>Put an octal representation of a value into a buffer</td>
</tr>
<tr>
<td><code>GetNameBytes</code></td>
<td>Overloaded. Add <em>name</em> to the buffer as a collection of bytes</td>
</tr>
<tr>
<td><code>GetOctalBytes</code></td>
<td>Put an octal representation of a value into a buffer</td>
</tr>
<tr>
<td><code>ParseName</code></td>
<td>Parse a name from a header buffer.</td>
</tr>
<tr>
<td><code>ParseOctal</code></td>
<td>Parse an octal string from a header buffer.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Clone</code></td>
<td>Clone a TAR header.</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>Determines if this instance is equal to the specified object.</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Get a hash code for the current object.</td>
</tr>
<tr>
<td><code>GetName</code></td>
<td><strong>Obsolete.</strong> Get the name of this entry.</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <em>Type</em> of the current instance.</td>
</tr>
<tr>
<td><code>ParseBuffer</code></td>
<td>Parse TarHeader information from a header buffer.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <a href="https://docs.microsoft.com/en-us/dotnet/api/system.string">String</a> that represents the current <a href="https://docs.microsoft.com/en-us/dotnet/api/system.object">Object</a>.</td>
</tr>
<tr>
<td><strong>WriteHeader</strong></td>
<td>'Write' header information to buffer provided, updating the <a href="https://docs.microsoft.com/en-us/dotnet/api/system.datetime">check sum</a>.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

| **Finalize** (inherited from Object) | Allows an [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object) to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. |
| **MemberwiseClone** (inherited from Object) | Creates a shallow copy of the current [Object](https://docs.microsoft.com/en-us/dotnet/api/system.object). |

**See Also**

[TarHeader Class](https://docs.icsharpcode.net/SharpZipLib/Tar/TarHeader.html) | [ICSharpCode.SharpZipLib.Tar Namespace](https://docs.icsharpcode.net/SharpZipLib/Tar.html)
ICSharpCode SharpZipLib Class Library
**TarHeader.Clone Method**

Clone a TAR header.

```csharp
public object Clone();
```

**Implements**

*ICloneable.Clone*

**See Also**

*TarHeader Class* | *ICSharpCode.SharpZipLib.Tar Namespace*
ICSharpCode SharpZipLib Class Library
## TarHeader.Equals Method

Determines if this instance is equal to the specified object.

```csharp
public override bool Equals(
    object obj
);
```

### Parameters

**obj**

The object to compare with.

### Return Value

true if the objects are equal, false otherwise.

### See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.GetAsciiBytes Method

Add a string to a buffer as a collection of ascii bytes.

```csharp
public static int GetAsciiBytes(
    string toAdd,
    int nameOffset,
    byte[] buffer,
    int bufferOffset,
    int length
);
```

Parameters

- **toAdd**
  The string to add

- **nameOffset**
  The offset of the first character to add.

- **buffer**
  The buffer to add to.

- **bufferOffset**
  The offset to start adding at.

- **length**
  The number of ascii characters to add.

Return Value

The next free index in the buffer.

See Also

- TarHeader Class  |  ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarHeader.GetHashCode Method**

Get a hash code for the current object.

```csharp
public override int GetHashCode();
```

**Return Value**

A hash code for the current object.

**See Also**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
Put an octal representation of a value into a buffer

```csharp
public static int GetLongOctalBytes(
    long value,
    byte[] buffer,
    int offset,
    int length
);
```

**Parameters**

- `value`
  - Value to be convert to octal

- `buffer`
  - The buffer to update

- `offset`
  - The offset into the buffer to store the value

- `length`
  - The length of the octal string

**Return Value**

- Index of next byte

**See Also**

- [TarHeader Class](#)
- [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**NOTE:** This method is now obsolete.

Use the Name property instead

Get the name of this entry.

```csharp
public string GetName();
```

**Return Value**

The entry's name.

**See Also**

[ TarHeader Class ](index.html)  |  [ICSharpCode.SharpZipLib.Tar Namespace](index.html)
ICSharpCode SharpZipLib Class Library
TarHeader.GetNameBytes Method

Add an entry name to the buffer

Overload List

Add an entry name to the buffer

```csharp
public static int GetNameBytes(string, byte[], int, int);
```

Add name to the buffer as a collection of bytes

```csharp
public static int GetNameBytes(string, int, byte[], int, int);
```

Add an entry name to the buffer

```csharp
public static int GetNameBytes(StringBuilder, byte[], int, int);
```

Add name to the buffer as a collection of bytes

```csharp
public static int GetNameBytes(StringBuilder, int, byte[], int, int);
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
## TarHeader.GetNameBytes Method (String, Byte[], Int32, Int32)

Add an entry name to the buffer

```csharp
public static int GetNameBytes(
    string name,
    byte[] buffer,
    int offset,
    int length
);
```

### Parameters

- **name**
  - The name to add

- **buffer**
  - The buffer to add to

- **offset**
  - The offset into the buffer from which to start adding

- **length**
  - The number of header bytes to add

### Return Value

The index of the next free byte in the buffer

### See Also

- TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace | TarHeader.GetNameBytes Overload List
Add *name* to the buffer as a collection of bytes

```csharp
public static int GetNameBytes(
    string name,
    int nameOffset,
    byte[] buffer,
    int bufferOffset,
    int length
);
```

**Parameters**

- **name**
  The name to add

- **nameOffset**
  The offset of the first character

- **buffer**
  The buffer to add to

- **bufferOffset**
  The index of the first byte to add

- **length**
  The number of characters/bytes to add

**Return Value**

The next free index in the `buf`

**See Also**

- [TarHeader Class](#)
- [ICSharpCode.SharpZipLib.Tar Namespace](#)
- [TarHeader.GetNameBytes Overload List](#)
ICSharpCode SharpZipLib Class Library
TarHeader.GetNameBytes Method (StringBuilder, Byte[], Int32, Int32)

Add an entry name to the buffer

```csharp
public static int GetNameBytes(
    StringBuilder name,
    byte[] buffer,
    int offset,
    int length
);
```

Parameters

- **name**
  The name to add

- **buffer**
  The buffer to add to

- **offset**
  The offset into the buffer from which to start adding

- **length**
  The number of header bytes to add

Return Value

The index of the next free byte in the buffer

See Also

- [TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#) | [TarHeader.GetNameBytes Overload List](#)
ICSharpCode SharpZipLib Class Library
TarHeader.GetNameBytes Method (StringBuilder, Int32, Byte[], Int32, Int32)

Add name to the buffer as a collection of bytes

```csharp
public static int GetNameBytes(
    StringBuilder name,
    int nameOffset,
    byte[] buffer,
    int bufferOffset,
    int length
);
```

Parameters

- `name` : The name to add
- `nameOffset` : The offset of the first character
- `buffer` : The buffer to add to
- `bufferOffset` : The index of the first byte to add
- `length` : The number of characters/bytes to add

Return Value

- The next free index in the buffer

See Also

- TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace | TarHeader.GetNameBytes Overload List
Put an octal representation of a value into a buffer

```csharp
public static int GetOctalBytes(
    long value,
    byte[] buffer,
    int offset,
    int length
);
```

### Parameters

- **value**
  
  the value to be converted to octal

- **buffer**
  
  buffer to store the octal string

- **offset**
  
  The offset into the buffer where the value starts

- **length**
  
  The length of the octal string to create

### Return Value

The offset of the character next byte after the octal string

### See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarHeader.ParseBuffer Method**

Parse TarHeader information from a header buffer.

```csharp
public void ParseBuffer(byte[] header);
```

**Parameters**

*header*

The tar entry header buffer to get information from.

**See Also**

[ TarHeader Class ](TarHeaderClass) | [ ICSharpCode.SharpZipLib.Tar Namespace ](ICSharpCode.SharpZipLib.Tar)
ICSharpCode SharpZipLib Class Library
**TarHeader.ParseName Method**

Parse a name from a header buffer.

```csharp
public static StringBuilder ParseName(
    byte[] header,
    int offset,
    int length
);
```

**Parameters**

- **header**
  - The header buffer from which to parse.

- **offset**
  - The offset into the buffer from which to parse.

- **length**
  - The number of header bytes to parse.

**Return Value**

The name parsed.

**See Also**

- [TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarHeader.ParseOctal Method

Parse an octal string from a header buffer.

```csharp
public static long ParseOctal(
    byte[] header,
    int offset,
    int length
);
```

Parameters

- **header**
  The header buffer from which to parse.

- **offset**
  The offset into the buffer from which to parse.

- **length**
  The number of header bytes to parse.

Return Value

The long equivalent of the octal string.

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
'Write' header information to buffer provided, updating the check sum.

```csharp
public void WriteHeader(byte[] outBuffer);
```

Parameters

- `outBuffer` output buffer for header information

See Also

- TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarInputStream Class

The TarInputStream reads a UNIX tar archive as an InputStream. Methods are provided to position at each successive entry in the archive, and the read each entry as a normal input stream using read().

For a list of all members of this type, see TarInputStream Members.

System.Object    System.MarshalByRefObject
    System.IO.Stream

| public class TarInputStream : Stream |

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

TarInputStream Members | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
### TarInputStream Members

#### TarInputStream overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TarInputStream</td>
<td>Overloaded. Initializes a new instance of the TarInputStream class.</td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>Get the available data that can be read from the current entry in the archive. This does not indicate how much data is left in the entire archive, only in the current entry.</td>
</tr>
<tr>
<td>CanRead</td>
<td>Gets a value indicating whether the current stream supports reading.</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Gets a value indicating whether the current stream supports seeking. This property always returns false.</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Gets a value indicating if the stream supports writing. This property always returns false.</td>
</tr>
<tr>
<td>IsMarkSupported</td>
<td>Return a value of true if marking is supported; false otherwise.</td>
</tr>
<tr>
<td>Length</td>
<td>The length in bytes of the stream</td>
</tr>
<tr>
<td>Position</td>
<td>Gets or sets the position within the stream. Setting the Position is not supported and throws a NotSupportedException.</td>
</tr>
<tr>
<td>RecordSize</td>
<td>Get the record size being used by this stream's TarBuffer.</td>
</tr>
</tbody>
</table>
## Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from Stream)</td>
<td>Begins an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes this stream. Calls the TarBuffer's close() method. The underlying stream is closed by the TarBuffer.</td>
</tr>
<tr>
<td><strong>CopyEntryContents</strong></td>
<td>Copies the contents of the current tar archive entry directly into an output stream.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the baseInputStream.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetNextEntry</strong></td>
<td>Get the next entry in this tar }</td>
</tr>
</tbody>
</table>
archive. This will skip over any remaining data in the current entry, if there is one, and place the input stream at the header of the next entry, and read the header and instantiate a new TarEntry from the header bytes and return that entry. If there are no more entries in the archive, null will be returned to indicate that the end of the archive has been reached.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetRecordSize</td>
<td>Obsolete. Get the record size being used by this stream's TarBuffer.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>InitializeLifetimeService (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td>Mark</td>
<td>Since we do not support marking just yet, we do nothing.</td>
</tr>
<tr>
<td>Read</td>
<td>Reads bytes from the current tar archive entry. This method is aware of the boundaries of the current entry in the archive and will deal with them appropriately.</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Reads a byte from the current tar archive entry. This method simply calls Read(byte[], int, int).</td>
</tr>
<tr>
<td>Reset</td>
<td>Since we do not support marking just yet, we do nothing.</td>
</tr>
<tr>
<td>Seek</td>
<td>Set the streams position. This operation is not supported and</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SetEntryFactory</strong></td>
<td>Set the entry factory for this instance.</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Sets the length of the stream This operation is not supported and will throw a NotSupportedException</td>
</tr>
<tr>
<td><strong>Skip</strong></td>
<td>Skip bytes in the input buffer. This skips bytes in the current entry's data, not the entire archive, and will stop at the end of the current entry's data if the number to skip extends beyond that point.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>(inherited from Object) Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes a block of bytes to this stream using data from a buffer. This operation is not supported and will throw a NotSupportedException</td>
</tr>
<tr>
<td><strong>.WriteByte</strong></td>
<td>Writes a byte to the current position in the file stream. This operation is not supported and will throw a NotSupportedException</td>
</tr>
</tbody>
</table>

**Protected Instance Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>buffer</strong></td>
<td>Working buffer</td>
</tr>
<tr>
<td><strong>entryFactory</strong></td>
<td>Factory used to create TarEntry or descendant class instance</td>
</tr>
<tr>
<td><strong>entryOffset</strong></td>
<td>Number of bytes read for this entry so far</td>
</tr>
<tr>
<td>entrySize</td>
<td>Size of this entry as recorded in header</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>hasHitEOF</td>
<td>Flag set when last block has been read</td>
</tr>
<tr>
<td>readBuffer</td>
<td>Buffer used with calls to Read()</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>CreateWaitHandle (inherited from Stream)</th>
<th>Allocates a WaitHandle object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

### See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarInputStream Constructor**

Construct a TarInputStream with default block factor

**Overload List**

Construct a TarInputStream with default block factor

```java
public TarInputStream(Stream);
```

Construct a TarInputStream with user specified block factor

```java
public TarInputStream(Stream,int);
```

**See Also**

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarInputStream Constructor (Stream)

Construct a TarInputStream with default block factor

```csharp
public TarInputStream(
    Stream inputStream
);
```

Parameters

`inputStream`
stream to source data from

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace | TarInputStream Constructor Overload List
ICSharpCode SharpZipLib Class Library
Construct a TarInputStream with user specified block factor

```csharp
public TarInputStream(
    Stream inputStream,
    int blockFactor
);
```

**Parameters**

- `inputStream`  
  stream to source data from

- `blockFactor`  
  block factor to apply to archive

**See Also**

- TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace | TarInputStream Constructor Overload List
## TarInputStream Fields

The fields of the **TarInputStream** class are listed below. For a complete list of **TarInputStream** class members, see the [TarInputStream Members](#) topic.

### Protected Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>buffer</code></td>
<td>Working buffer</td>
</tr>
<tr>
<td><code>entryFactory</code></td>
<td>Factory used to create TarEntry or descendant class instance</td>
</tr>
<tr>
<td><code>entryOffset</code></td>
<td>Number of bytes read for this entry so far</td>
</tr>
<tr>
<td><code>entrySize</code></td>
<td>Size of this entry as recorded in header</td>
</tr>
<tr>
<td><code>hasHitEOF</code></td>
<td>Flag set when last block has been read</td>
</tr>
<tr>
<td><code>readBuffer</code></td>
<td>Buffer used with calls to <code>Read()</code></td>
</tr>
</tbody>
</table>

### See Also

[TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
 TarInputStream.buffer Field

Working buffer

protected TarBuffer buffer;

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarInputStream.entryFactory Field

Factory used to create TarEntry or descendant class instance

```java
protected IEntryFactory entryFactory;
```

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
**ICSharpCode SharpZipLib Class Library**
TarInputStream.entryOffset Field

Number of bytes read for this entry so far

```java
protected long entryOffset;
```

See Also

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarInputStream.entrySize Field

Size of this entry as recorded in header

```java
protected long entrySize;
```

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarInputStream.hasHitEOF Field

Flag set when last block has been read

protected bool hasHitEOF;

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarInputStream.readBuffer Field

Buffer used with calls to

Read()

protected byte[] readBuffer;

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
# TarInputStream Properties

The properties of the `TarInputStream` class are listed below. For a complete list of `TarInputStream` class members, see the `TarInputStream Members` topic.

## Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>Get the available data that can be read from the current entry in the archive. This does not indicate how much data is left in the entire archive, only in the current entry. This value is determined from the entry's size header field and the amount of data already read from the current entry.</td>
</tr>
<tr>
<td>CanRead</td>
<td>Gets a value indicating whether the current stream supports reading.</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Gets a value indicating whether the current stream supports seeking. This property always returns false.</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Gets a value indicating if the stream supports writing. This property always returns false.</td>
</tr>
<tr>
<td>IsMarkSupported</td>
<td>Return a value of true if marking is supported; false otherwise.</td>
</tr>
<tr>
<td>Length</td>
<td>The length in bytes of the stream</td>
</tr>
<tr>
<td>Position</td>
<td>Gets or sets the position within the stream. Setting the Position is not supported and throws a NotSupportedException.</td>
</tr>
<tr>
<td>RecordSize</td>
<td>Get the record size being used by this stream's TarBuffer.</td>
</tr>
</tbody>
</table>

## See Also

- [TarInputStream Class](#)
- [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarInputStream.Available Property

Get the available data that can be read from the current entry in the archive. This does not indicate how much data is left in the entire archive, only in the current entry. This value is determined from the entry's size header field and the amount of data already read from the current entry.

```
public long Available {get;}
```

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarInputStream.CanRead Property

Gets a value indicating whether the current stream supports reading

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

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarInputStream.CanSeek Property**

Gets a value indicating whether the current stream supports seeking. This property always returns false.

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

See Also

[TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarInputStream.CanWrite Property

Gets a value indicating if the stream supports writing. This property always returns false.

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

See Also

TarInputStream Class | ISharpCode.SharpZipLib.Tar Namespace
TarInputStream.IsMarkSupported Property

Return a value of true if marking is supported; false otherwise.

```csharp
public bool IsMarkSupported {get;}
```

Remarks

Currently marking is not supported, the return value is always false.

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
**TarInputStream.Length Property**

The length in bytes of the stream

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

See Also

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarInputStream.Position Property**

Gets or sets the position within the stream. Setting the Position is not supported and throws a NotSupportedException

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

### Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any attempt to set position</td>
</tr>
</tbody>
</table>

### See Also

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarInputStream.RecordSize Property

Get the record size being used by this stream's TarBuffer.

```csharp
public int RecordSize {get;}
```

See Also

[TarInputStream Class] [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
The methods of the **TarInputStream** class are listed below. For a complete list of **TarInputStream** class members, see the **TarInputStream Members** topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from Stream)</td>
<td>Begins an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes this stream. Calls the TarBuffer's close() method. The underlying stream is closed by the TarBuffer.</td>
</tr>
<tr>
<td><strong>CopyEntryContents</strong></td>
<td>Copies the contents of the current tar archive entry directly into an output stream.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the baseInputStream</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data.</td>
</tr>
</tbody>
</table>
structures like a hash table.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetNextEntry</strong></td>
<td>Get the next entry in this tar archive. This will skip over any remaining data in the current entry, if there is one, and place the input stream at the header of the next entry, and read the header and instantiate a new TarEntry from the header bytes and return that entry. If there are no more entries in the archive, null will be returned to indicate that the end of the archive has been reached.</td>
</tr>
<tr>
<td><strong>GetRecordSize</strong></td>
<td>Obsolete. Get the record size being used by this stream's TarBuffer.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Mark</strong></td>
<td>Since we do not support marking just yet, we do nothing.</td>
</tr>
<tr>
<td><strong>ReadStream</strong></td>
<td>Reads bytes from the current tar archive entry. This method is aware of the boundaries of the current entry in the archive and will deal with them appropriately.</td>
</tr>
<tr>
<td><strong>ReadByte</strong></td>
<td>Reads a byte from the current tar archive entry. This method simply calls Read(byte[], int,</td>
</tr>
</tbody>
</table>


Since we do not support marking just yet, we do nothing.

Set the streams position. This operation is not supported and will throw a NotSupportedException.

Set the entry factory for this instance.

Sets the length of the stream. This operation is not supported and will throw a NotSupportedException.

Skip bytes in the input buffer. This skips bytes in the current entry's data, not the entire archive, and will stop at the end of the current entry's data if the number to skip extends beyond that point.

Returns a `String` that represents the current `Object`.

Writes a block of bytes to this stream using data from a buffer. This operation is not supported and will throw a NotSupportedException.

Writes a byte to the current position in the file stream. This operation is not supported and will throw a NotSupportedException.
| **CreateWaitHandle** (inherited from **Stream**) | Allocates a **WaitHandle** object. |
| **Finalize** (inherited from **Object**) | Allows an **Object** to attempt to free resources and perform other cleanup operations before the **Object** is reclaimed by garbage collection. |
| **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

[- TarInputStream Class](#) | [- ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSSharpCode SharpZipLib Class Library
Closes this stream. Calls the TarBuffer's close() method. The underlying stream is closed by the TarBuffer.

```csharp
public override void Close();
```

See Also

TarInputStream Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarInputStream.CopyEntryContents Method

Copies the contents of the current tar archive entry directly into an output stream.

```csharp
public void CopyEntryContents(
    Stream outputStream
);
```

Parameters

`outputStream`

The OutputStream into which to write the entry's data.

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarInputStream.Flush Method

Flushes the baseInputStream

```csharp
public override void Flush();
```

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
Get the next entry in this tar archive. This will skip over any remaining data in the current entry, if there is one, and place the input stream at the header of the next entry, and read the header and instantiate a new TarEntry from the header bytes and return that entry. If there are no more entries in the archive, null will be returned to indicate that the end of the archive has been reached.

```csharp
public TarEntry GetNextEntry();
```

Return Value

The next TarEntry in the archive, or null.

See Also

TarInputStream Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
NOTE: This method is now obsolete.
Use RecordSize property instead

Get the record size being used by this stream's TarBuffer.

```csharp
public int GetRecordSize();
```

Return Value

TarBuffer record size.

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
**ICSharpCode SharpZipLib Class Library**
Since we do not support marking just yet, we do nothing.

```csharp
public void Mark(int markLimit);
```

**Parameters**

*markLimit*

The limit to mark.

**See Also**

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
Reads bytes from the current tar archive entry. This method is aware of the boundaries of the current entry in the archive and will deal with them appropriately.

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

**Parameters**

- `buffer`  
  The buffer into which to place bytes read.

- `offset`  
  The offset at which to place bytes read.

- `count`  
  The number of bytes to read.

**Return Value**

The number of bytes read, or 0 at end of stream/EOF.

**See Also**

- [TarInputStream Class](#)  
- ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarInputStream.ReadByte Method**

Reads a byte from the current tar archive entry. This method simply calls Read(byte[], int, int).

```
public override int ReadByte();
```

See Also

TarInputStream Class | ISharpCode.SharpZipLib.Tar Namespace
TarInputStream.Reset Method

Since we do not support marking just yet, we do nothing.

```csharp
public void Reset();
```

See Also

[TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
## TarInputStream.Seek Method

Set the streams position. This operation is not supported and will throw a NotSupportedException

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

### Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

### See Also

[TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
### TarInputStream.SetEntryFactory Method

Set the entry factory for this instance.

```csharp
public void SetEntryFactory(IEntryFactory factory);
```

#### Parameters

- **factory**
  The factory for creating new entries

#### See Also

- [TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarInputStream.SetLength Method**

Sets the length of the stream. This operation is not supported and will throw a NotSupportedException.

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

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

**See Also**

- TarInputStream Class
- ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarInputStream.Skip Method

Skip bytes in the input buffer. This skips bytes in the current entry's data, not the entire archive, and will stop at the end of the current entry's data if the number to skip extends beyond that point.

```csharp
public void Skip(
    long skipCount
);
```

Parameters

- **skipCount**
  - The number of bytes to skip.

See Also

- TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarInputStream.Write Method**

Writes a block of bytes to this stream using data from a buffer. This operation is not supported and will throw a NotSupportedException

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

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

**See Also**

[TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarInputStream.WriteByte Method

Writes a byte to the current position in the file stream. This operation is not supported and will throw a NotSupportedException

```csharp
public override void WriteByte(
    byte value
);
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

- TarInputStream Class
- ICSharpCode.SharpZipLib.Tar Namespace
TarInputStream.EntryFactoryAdapter Class

Standard entry factory class creating instances of the class TarEntry.

For a list of all members of this type, see TarInputStream.EntryFactoryAdapter Members.

System.Object


```csharp
public class TarInputStream.EntryFactoryAdapter : IEntryFactory
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

**Namespace:** ISharpCode.SharpZipLib.Tar

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

See Also

ICSharpCode SharpZipLib Class Library
## TarInputStream.EntryFactoryAdapter Members

### TarInputStream.EntryFactoryAdapter overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateEntry</td>
<td>Overloaded. Create and entry based on details in <code>headerBuf</code></td>
</tr>
<tr>
<td>CreateEntryFromFile</td>
<td>Create a tar entry with details obtained from <code>fileName</code></td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

#### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize (inherited from Object)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>
See Also

ICSharpCode SharpZipLib Class Library
TarInputStream.EntryFactoryAdapter Constructor

Initializes a new instance of the TarInputStream.EntryFactoryAdapter class.

```java
public TarInputStream.EntryFactoryAdapter();
```

See Also

ICSharpCode SharpZipLib Class Library
The methods of the `TarInputStream.EntryFactoryAdapter` class are listed below. For a complete list of `TarInputStream.EntryFactoryAdapter` class members, see the `TarInputStream.EntryFactoryAdapter Members` topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateEntry</code></td>
<td>Overloaded. Create and entry based on details in <code>headerBuf</code></td>
</tr>
<tr>
<td><code>CreateEntryFromFile</code></td>
<td>Create a tar entry with details obtained from <code>fileName</code></td>
</tr>
<tr>
<td><code>Equals</code> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>GetHashCode</code> (inherited from <code>Object</code>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><code>GetType</code> (inherited from <code>Object</code>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>ToString</code> (inherited from <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

### See Also
ICSharpCode SharpZipLib Class Library
Create and entry based on details in headerBuf

Overload List

Create and entry based on details in headerBuf

```csharp
public TarEntry CreateEntry(byte[]);
```
Create a TarEntry based on named

```csharp
public TarEntry CreateEntry(string);
```

See Also

ICSharpCode SharpZipLib Class Library
Create and entry based on details in $headerBuf$

```csharp
public TarEntry CreateEntry(byte[] headerBuf);
```

Implements

IEntryFactory.CreateEntry

See Also

ICSharpCode SharpZipLib Class Library
Create a TarEntry based on named

```csharp
public TarEntry CreateEntry(string name);
```

Implements

IEntryFactory.CreateEntry

See Also

Create a tar entry with details obtained from `fileName`

```csharp
public TarEntry CreateEntryFromFile(
    string fileName
);
```

**Implements**

`IEntryFactory.CreateEntryFromFile`

**See Also**

ICSharpCode SharpZipLib Class Library
**TarInputStream.IEntryFactory Interface**

This interface is provided, along with the method `SetEntryFactory`, to allow the programmer to have their own `TarEntry` subclass instantiated for the entries return from `GetNextEntry`.

For a list of all members of this type, see `TarInputStream.IEntryFactory Members`.

```
public interface TarInputStream.IEntryFactory
```

**Types that implement TarInputStream.IEntryFactory**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TarInputStream.EntryFactoryAdapter</code></td>
<td>Standard entry factory class creating instances of the class TarEntry</td>
</tr>
</tbody>
</table>

**Requirements**

- **Namespace**: `ICSharpCode.SharpZipLib.Tar`

**See Also**

- `TarInputStream.IEntryFactory Members`
- `ICSharpCode.SharpZipLib.Tar Namespace`
ICSharpCode SharpZipLib Class Library
TarInputStream.IEntryFactory Members

TarInputStream.IEntryFactory overview

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateEntry</td>
<td>Overloaded. Create a tar entry based on the header information passed</td>
</tr>
<tr>
<td>CreateEntryFromFile</td>
<td>Create an instance based on an actual file</td>
</tr>
</tbody>
</table>

See Also

TarInputStream.IEntryFactory Interface | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarInputStream.IEntryFactory Methods

The methods of the `TarInputStream.IEntryFactory` interface are listed below. For a complete list of `TarInputStream.IEntryFactory` interface members, see the `TarInputStream.IEntryFactory Members` topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateEntry</code></td>
<td>Overloaded. Create a tar entry based on the header information passed</td>
</tr>
<tr>
<td><code>CreateEntryFromFile</code></td>
<td>Create an instance based on an actual file</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
TarInputStream.IEntryFactory.CreateEntry Method

Create a tar entry based on the header information passed

Overload List

Create a tar entry based on the header information passed

TarEntry CreateEntry(byte[]);

Create an entry based on name alone

TarEntry CreateEntry(string);

See Also

TarInputStream.IEntryFactory Interface | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
Create a tar entry based on the header information passed

```csharp
TarEntry CreateEntry(byte[] headerBuf);
```

### Parameters

*headerBuf*  
Buffer containing header information to base entry on

### Return Value

Created TarEntry or descendant class

### See Also

ICSharpCode SharpZipLib Class Library
TarInputStream.IEntryFactory.CreateEntry Method (String)

Create an entry based on name alone

```csharp
TarEntry CreateEntry(string name);
```

Parameters

- **name**
  
  Name of the new EntryPointNotFoundException to create

Return Value

- created TarEntry or descendant class

See Also

- TarInputStream.IEntryFactory Interface
- ICSharpCode.SharpZipLib.Tar Namespace
- TarInputStream.IEntryFactory.CreateEntry Overload List
ICSharpCode SharpZipLib Class Library
TarInputStream.IEntryFactory.CreateEntryFromFile Method

Create an instance based on an actual file

```csharp
TarEntry CreateEntryFromFile(
    string fileName
);
```

Parameters

- `fileName`  
  Name of file to represent in the entry

Return Value

- Created TarEntry or descendant class

See Also

- TarInputStream.IEntryFactory Interface | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
The TarOutputStream writes a UNIX tar archive as an OutputStream. Methods are provided to put entries, and then write their contents by writing to this stream using write().

For a list of all members of this type, see TarOutputStream Members.

System.Object   System.MarshalByRefObject
                 System.IO.Stream

```public class TarOutputStream : Stream```

**Thread Safety**

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** ICSharpCode.SharpZipLib.Tar

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

**See Also**

TarOutputStream Members | ICSharpCode.SharpZipLib.Tar Namespace
## TarOutputStream Members

### TarOutputStream overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TarOutputStream</strong></td>
<td>Overloaded. Initializes a new instance of the TarOutputStream class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CanRead</strong></td>
<td>true if the stream supports reading; otherwise, false.</td>
</tr>
<tr>
<td><strong>CanSeek</strong></td>
<td>true if the stream supports seeking; otherwise, false.</td>
</tr>
<tr>
<td><strong>CanWrite</strong></td>
<td>true if stream supports writing; otherwise, false.</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>length of stream in bytes</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>gets or sets the position within the current stream.</td>
</tr>
<tr>
<td><strong>RecordSize</strong></td>
<td>Get the record size being used by this stream’s TarBuffer.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from Stream)**</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from Stream)**</td>
<td>Begins an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Ends the TAR archive and closes the underlying OutputStream.</td>
</tr>
<tr>
<td><strong>CloseEntry</strong></td>
<td>Close an entry. This method MUST be called for all file entries that contain data. The</td>
</tr>
</tbody>
</table>
reason is that we must buffer data written to the stream in order to satisfy the buffer's block based writes. Thus, there may be data fragments still being assembled that must be written to the output stream before this entry is closed and the next entry written.

<table>
<thead>
<tr>
<th>Method/Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateObjRef (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td>EndRead (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td>EndWrite (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>Finish</td>
<td>Ends the TAR archive without closing the underlying OutputStream. The result is that the EOF block of nulls is written.</td>
</tr>
<tr>
<td>Flush</td>
<td>All buffered data is written to destination.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetLifetimeService (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetRecordSize</code></td>
<td>Obsolete. Get the record size being used by this stream's TarBuffer.</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>InitializeLifetimeService</code></td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><code>PutNextEntry</code></td>
<td>Put an entry on the output stream. This writes the entry's header and positions the output stream for writing the contents of the entry. Once this method is called, the stream is ready for calls to write() to write the entry's contents. Once the contents are written, closeEntry() <strong>MUST</strong> be called to ensure that all buffered data is completely written to the output stream.</td>
</tr>
<tr>
<td><code>Read</code></td>
<td>read bytes from the current stream and advance the position within the stream by the number of bytes read.</td>
</tr>
<tr>
<td><code>ReadByte</code></td>
<td>Read a byte from the stream and advance the position within the stream by one byte or returns -1 if at the end of the stream.</td>
</tr>
<tr>
<td><code>Seek</code></td>
<td>set the position within the current stream</td>
</tr>
<tr>
<td><code>SetLength</code></td>
<td>set the length of the current stream</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Returns a <code>String</code> that represents</td>
</tr>
<tr>
<td><strong>Object)</strong></td>
<td>the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes bytes to the current tar archive entry. This method is aware of the current entry and will throw an exception if you attempt to write bytes past the length specified for the current entry. The method is also (painfully) aware of the record buffering required by TarBuffer, and manages buffers that are not a multiple of recordsize in length, including assembling records from small buffers.</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Writes a byte to the current tar archive entry. This method simply calls Write(byte[], int, int).</td>
</tr>
</tbody>
</table>

**Protected Instance Fields**

| **assemblyBuffer** | 'Assembly' buffer used to assemble data before writing |
| **blockBuffer** | single block working buffer |
| **buffer** | TarBuffer used to provide correct blocking factor |
| **currSize** | Size for the current entry |
| **outputStream** | the destination stream for the archive contents |

**Protected Instance Methods**

| **CreateWaitHandle** (inherited from **Stream**) | Allocates a **WaitHandle** object. |
| **Finalize** (inherited from **Object**) | Allows an **Object** to attempt to free resources and perform other cleanup operations before |
the **Object** is reclaimed by garbage collection.

| **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarOutputStream Constructor**

Construct TarOutputStream using default block factor

**Overload List**

Construct TarOutputStream using default block factor

```java
public TarOutputStream(Stream);
```

Construct TarOutputStream with user specified block factor

```java
public TarOutputStream(Stream,int);
```

**See Also**

[TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
Construct TarOutputStream using default block factor

```csharp
public TarOutputStream(
    Stream outputStream
);
```

**Parameters**

`outputStream`
stream to write to

**See Also**

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
| TarOutputStream Constructor Overload List
ICSharpCode SharpZipLib Class Library
TarOutputStream Constructor (Stream, Int32)

Construct TarOutputStream with user specified block factor

```csharp
public TarOutputStream(
    Stream outputStream,
    int blockFactor
);
```

Parameters

- `outputStream`: stream to write to
- `blockFactor`: blocking factor

See Also

- TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
- TarOutputStream Constructor Overload List
ICSharpCode SharpZipLib Class Library
TarOutputStream Fields

The fields of the **TarOutputStream** class are listed below. For a complete list of **TarOutputStream** class members, see the **TarOutputStream Members** topic.

**Protected Instance Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assemblyBuffer</td>
<td>'Assembly' buffer used to assemble data before writing</td>
</tr>
<tr>
<td>blockBuffer</td>
<td>single block working buffer</td>
</tr>
<tr>
<td>buffer</td>
<td>TarBuffer used to provide correct blocking factor</td>
</tr>
<tr>
<td>currSize</td>
<td>Size for the current entry</td>
</tr>
<tr>
<td>outputStream</td>
<td>the destination stream for the archive contents</td>
</tr>
</tbody>
</table>

**See Also**

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarOutputStream.assemblyBuffer Field

'Assembly' buffer used to assemble data before writing

protected byte[] assemblyBuffer;

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarOutputStream.blockBuffer Field

single block working buffer

protected byte[] blockBuffer;

See Also

TarOutputStream Class | ISharpCode.SharpZipLib.Tar Namespace
| ICSsharpCode SharpZipLib Class Library |
TarOutputStream.buffer Field

TarBuffer used to provide correct blocking factor

protected TarBuffer buffer;

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarOutputStream.currSize Field

Size for the current entry

```java
protected long currSize;
```

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarOutputStream.outputStream Field

the destination stream for the archive contents

```csharp
protected Stream outputStream;
```

See Also

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
# TarOutputStream Properties

The properties of the `TarOutputStream` class are listed below. For a complete list of `TarOutputStream` class members, see the [TarOutputStream Members](#) topic.

## Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CanRead</strong></td>
<td>true if the stream supports reading; otherwise, false.</td>
</tr>
<tr>
<td><strong>CanSeek</strong></td>
<td>true if the stream supports seeking; otherwise, false.</td>
</tr>
<tr>
<td><strong>CanWrite</strong></td>
<td>true if stream supports writing; otherwise, false.</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>length of stream in bytes</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>gets or sets the position within the current stream.</td>
</tr>
<tr>
<td><strong>RecordSize</strong></td>
<td>Get the record size being used by this stream's TarBuffer.</td>
</tr>
</tbody>
</table>

See Also

[TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarOutputStream.CanRead Property

true if the stream supports reading; otherwise, false.

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

See Also

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
**TarOutputStream.CanSeek Property**

true if the stream supports seeking; otherwise, false.

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

See Also

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarOutputStream.CanWrite Property

true if stream supports writing; otherwise, false.

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

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
length of stream in bytes

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

See Also

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarOutputStream.Position Property

gets or sets the position within the current stream.

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

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarOutputStream.RecordSize Property

Get the record size being used by this stream's TarBuffer.

```csharp
public int RecordSize {get;}
```

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarOutputStream Methods

The methods of the **TarOutputStream** class are listed below. For a complete list of **TarOutputStream** class members, see the [TarOutputStream Members](#) topic.

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from Stream)</td>
<td>Begins an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Ends the TAR archive and closes the underlying OutputStream.</td>
</tr>
<tr>
<td><strong>CloseEntry</strong></td>
<td>Close an entry. This method MUST be called for all file entries that contain data. The reason is that we must buffer data written to the stream in order to satisfy the buffer's block based writes. Thus, there may be data fragments still being assembled that must be written to the output stream before this entry is closed and the next entry written.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Ends the TAR archive without closing the underlying OutputStream. The result is that the EOF block of nulls is written.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>All buffered data is written to destination.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from <strong>MarshalByRefObject</strong>)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetRecordSize</strong></td>
<td><strong>Obsolete.</strong> Get the record size being used by this stream's TarBuffer.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from <strong>MarshalByRefObject</strong>)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>PutNextEntry</strong></td>
<td>Put an entry on the output stream. This writes the entry's header and positions the output stream for writing the contents of the entry. Once this method is called, the stream is ready for calls to write() to write the entry's contents. Once the contents are written, closeEntry() <strong>MUST</strong> be called to ensure that all buffered data is</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Read</code></td>
<td>read bytes from the current stream and advance the position within the stream by the number of bytes read.</td>
</tr>
<tr>
<td><code>ReadByte</code></td>
<td>Read a byte from the stream and advance the position within the stream by one byte or returns -1 if at the end of the stream.</td>
</tr>
<tr>
<td><code>Seek</code></td>
<td>set the position within the current stream</td>
</tr>
<tr>
<td><code>SetLength</code></td>
<td>set the length of the current stream</td>
</tr>
<tr>
<td><code>ToString</code> (inherited from Object)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>Write</code></td>
<td>Writes bytes to the current tar archive entry. This method is aware of the current entry and will throw an exception if you attempt to write bytes past the length specified for the current entry. The method is also (painfully) aware of the record buffering required by TarBuffer, and manages buffers that are not a multiple of recordsize in length, including assembling records from small buffers.</td>
</tr>
<tr>
<td><code>WriteByte</code></td>
<td>Writes a byte to the current tar archive entry. This method simply calls <code>Write(byte[], int, int)</code>.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateWaitHandle</strong> (inherited from Stream)</td>
<td>Allocates a <a href="#">WaitHandle</a> object.</td>
</tr>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <a href="#">Object</a> to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <a href="#">Object</a>.</td>
</tr>
</tbody>
</table>

See Also

[TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
**TarOutputStream.Close Method**

Ends the TAR archive and closes the underlying OutputStream.

```csharp
public override void Close();
```

**Remarks**

This means that Finish() is called followed by calling the TarBuffer's Close().

**See Also**

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSsharpCode SharpZipLib Class Library
Close an entry. This method MUST be called for all file entries that contain data. The reason is that we must buffer data written to the stream in order to satisfy the buffer's block based writes. Thus, there may be data fragments still being assembled that must be written to the output stream before this entry is closed and the next entry written.

```csharp
public void CloseEntry();
```

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
Ends the TAR archive without closing the underlying OutputStream. The result is that the EOF block of nulls is written.

```java
public void Finish();
```

See Also

[TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarOutputStream.Flush Method

All buffered data is written to destination

```csharp
public override void Flush();
```

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
The TarBuffer record size.

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
Put an entry on the output stream. This writes the entry’s header and positions the output stream for writing the contents of the entry. Once this method is called, the stream is ready for calls to write() to write the entry's contents. Once the contents are written, closeEntry() MUST be called to ensure that all buffered data is completely written to the output stream.

```
public void PutNextEntry(
    TarEntry entry
);
```

**Parameters**

- **entry**
  The TarEntry to be written to the archive.

**See Also**

- TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarOutputStream.Read Method

read bytes from the current stream and advance the position within the stream by the number of bytes read.

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

Return Value

The total number of bytes read, or zero if at the end of the stream

See Also

[TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
ICSharpCode SharpZipLib Class Library
TarOutputStream.ReadByte Method

Read a byte from the stream and advance the position within the stream by one byte or returns -1 if at the end of the stream.

```csharp
public override int ReadByte();
```

Return Value

The byte value or -1 if at end of stream

See Also

TarOutputStream Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
**TarOutputStream.Seek Method**

set the position within the current stream

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

See Also

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
ICSharpCode SharpZipLib Class Library
TarOutputStream.SetLength Method

set the length of the current stream

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

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library
TarOutputStream.Write Method

Writes bytes to the current tar archive entry. This method is aware of the current entry and will throw an exception if you attempt to write bytes past the length specified for the current entry. The method is also (painfully) aware of the record buffering required by TarBuffer, and manages buffers that are not a multiple of recordsize in length, including assembling records from small buffers.

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

Parameters

- `buffer`  
The buffer to write to the archive.
- `offset`  
The offset in the buffer from which to get bytes.
- `count`  
The number of bytes to write.

See Also

[ TarOutputStream Class ](index.html) | [ ICSharpCode.SharpZipLib.Tar Namespace ](index.html)
ICSharpCode SharpZipLib Class Library
**TarOutputStream.WriteByte Method**

Writes a byte to the current tar archive entry. This method simply calls Write(byte[], int, int).

```csharp
public override void WriteByte( byte value );
```

**Parameters**

- **value**
  - The byte to be written.

**See Also**

TarOutputStream Class | ISharpCode.SharpZipLib.Tar Namespace
ICSharpCode SharpZipLib Class Library

Namespace hierarchy

Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseArchiveStorage</td>
<td>An abstract IArchiveStorage suitable for extension by inheritance.</td>
</tr>
<tr>
<td>DiskArchiveStorage</td>
<td>An IArchiveStorage implementation suitable for hard disks.</td>
</tr>
<tr>
<td>FastZip</td>
<td>FastZip provides facilities for creating and extracting zip files. Only relative paths are supported.</td>
</tr>
<tr>
<td>FastZipEvents</td>
<td>FastZipEvents supports all events applicable to FastZip operations.</td>
</tr>
<tr>
<td>KeysRequiredEventArgs</td>
<td>Arguments used with KeysRequiredEvent</td>
</tr>
<tr>
<td>MemoryArchiveStorage</td>
<td>An IArchiveStorage implementation suitable for in memory streams.</td>
</tr>
<tr>
<td>TestStatus</td>
<td>Status returned returned by ZipTestResultHandler during testing.</td>
</tr>
<tr>
<td>ZipConstants</td>
<td>This class contains constants used for Zip format files</td>
</tr>
<tr>
<td>ZipEntry</td>
<td>This class represents an entry in a zip archive. This can be a file or a directory ZipFile and ZipInputStream will give you instances of this class as</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>information about the members in an archive. ZipOutputStream uses an instance of this class when creating an entry in a Zip file.</td>
<td></td>
</tr>
<tr>
<td>Author of the original java version : Jochen Hoenicke</td>
<td></td>
</tr>
<tr>
<td>ZipException</td>
<td>Represents exception conditions specific to Zip archive handling</td>
</tr>
<tr>
<td>ZipExtraData</td>
<td>A class to handle the extra data field for Zip entries</td>
</tr>
<tr>
<td>ZipFile</td>
<td>This class represents a Zip archive. You can ask for the contained entries, or get an input stream for a file entry. The entry is automatically decompressed. You can also update the archive adding or deleting entries. This class is thread safe for input: You can open input streams for arbitrary entries in different threads.</td>
</tr>
<tr>
<td>Author of the original java version : Jochen Hoenicke</td>
<td></td>
</tr>
<tr>
<td>ZipInputStream</td>
<td>This is anInflaterInputStream that reads the files baseInputStream an zip archive one after another. It has a special method to get the zip entry of the next file. The zip entry contains information about the file name size, compressed size, Crc, etc. It includes support for Stored and Deflated</td>
</tr>
</tbody>
</table>
Entries.

Author of the original java version : Jochen Hoenicke

<table>
<thead>
<tr>
<th><strong>ZipNameTransform</strong></th>
<th>ZipNameTransform transforms names as per the Zip file naming convention.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ZipOutputStream</strong></td>
<td>This is a DeflaterOutputStream that writes the files into a zip archive one after another. It has a special method to start a new zip entry. The zip entries contains information about the file name size, compressed size, CRC, etc. It includes support for Stored and Deflated entries. This class is not thread safe.</td>
</tr>
<tr>
<td></td>
<td>Author of the original java version : Jochen Hoenicke</td>
</tr>
</tbody>
</table>

**Interfaces**

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IArchiveStorage</strong></td>
<td>Defines facilities for data storage when updating Zip Archives.</td>
</tr>
<tr>
<td><strong>IDynamicDataSource</strong></td>
<td>Represents a source of data that dynamically provide multiple data sources based on the parameters passed.</td>
</tr>
<tr>
<td><strong>IEntryFactory</strong></td>
<td>Defines factory methods for creating new ZipEntry values.</td>
</tr>
<tr>
<td><strong>IStaticDataSource</strong></td>
<td>Provides a static way to obtain a source of data for an entry.</td>
</tr>
</tbody>
</table>
### Delegates

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastZip.ConfirmOverwriteDelegate</td>
<td>Delegate called when confirming overwriting of files.</td>
</tr>
<tr>
<td>ZipTestResultHandler</td>
<td>Delegate invoked during testing if supplied indicating current progress and status.</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompressionMethod</td>
<td>The kind of compression used for an entry in an archive</td>
</tr>
<tr>
<td>EncryptionAlgorithm</td>
<td>Identifies the encryption algorithm used for an entry</td>
</tr>
<tr>
<td>FastZip.Overwrite</td>
<td>Defines the desired handling when overwriting files during extraction.</td>
</tr>
<tr>
<td>FileUpdateMode</td>
<td>The possible ways of applying updates to an archive.</td>
</tr>
<tr>
<td>GeneralBitFlags</td>
<td>Defines the contents of the general bit flags field for an archive entry.</td>
</tr>
<tr>
<td>HostSystemID</td>
<td>Defines known values for the HostSystemID property.</td>
</tr>
<tr>
<td>TestOperation</td>
<td>The operation in progress reported by a ZipTestResultHandler during testing.</td>
</tr>
<tr>
<td><strong>TestStrategy</strong></td>
<td>The strategy to apply to testing.</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td><strong>UseZip64</strong></td>
<td>Determines how entries are tested to see if they should use Zip64 extensions or not.</td>
</tr>
</tbody>
</table>
ICSharpCode SharpZipLib Class Library
BaseArchiveStorage Class

An abstract `IArchiveStorage` suitable for extension by inheritance.

For a list of all members of this type, see `BaseArchiveStorage Members`.

`System.Object`


```
public abstract class BaseArchiveStorage : IArchiveStorage
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

Requirements

**Namespace:** ISharpCode.SharpZipLib.Zip

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

See Also

BaseArchiveStorage Members | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
# BaseArchiveStorage Members

## BaseArchiveStorage overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BaseArchiveStorage Constructor</strong></td>
<td>Initializes a new instance of the <code>BaseArchiveStorage</code> class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UpdateMode</strong></td>
<td>Gets the update mode applicable.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConvertTemporaryToFinal</strong></td>
<td>Converts the temporary Stream to its final form.</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes this instance.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <code>Object</code>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetTemporaryOutput</strong></td>
<td>Gets a temporary output Stream.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <code>Object</code>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><strong>MakeTemporaryCopy</strong></td>
<td>Make a temporary copy of a Stream.</td>
</tr>
<tr>
<td><strong>OpenForDirectUpdate</strong></td>
<td>Return a stream suitable for performing direct updates on the original source.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>
### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗ <strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>✗ <strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### See Also

ICSharpCode SharpZipLib Class Library
BaseArchiveStorage Constructor

Initializes a new instance of the BaseArchiveStorage class.

```csharp
public BaseArchiveStorage(
    FileUpdateMode updateMode
);
```

Parameters

`updateMode`

The update mode.

See Also

BaseArchiveStorage Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
The properties of the `BaseArchiveStorage` class are listed below. For a complete list of `BaseArchiveStorage` class members, see the `BaseArchiveStorage Members` topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdateMode</td>
<td>Gets the update mode applicable.</td>
</tr>
</tbody>
</table>

See Also

### BaseArchiveStorage.UpdateMode Property

Gets the update mode applicable.

```csharp
public FileUpdateMode UpdateMode {get;}
```

#### Property Value

The update mode.

#### Implements

IArchiveStorage.UpdateMode

#### See Also

ICSharpCode SharpZipLib Class Library
BaseArchiveStorage Methods

The methods of the **BaseArchiveStorage** class are listed below. For a complete list of **BaseArchiveStorage** class members, see the **BaseArchiveStorage Members** topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConvertTemporaryToFinal</strong></td>
<td>Converts the temporary Stream to its final form.</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes this instance.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetTemporaryOutput</strong></td>
<td>Gets a temporary output Stream.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>MakeTemporaryCopy</strong></td>
<td>Make a temporary copy of a Stream.</td>
</tr>
<tr>
<td><strong>OpenForDirectUpdate</strong></td>
<td>Return a stream suitable for performing direct updates on the original source.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
</tbody>
</table>
MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object.

See Also

BaseArchiveStorage Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
BaseArchiveStorage(ConvertTemporaryToFinal Method)

Converts the temporary Stream to its final form.

```java
public abstract Stream ConvertTemporaryToFinal();
```

Return Value

Returns a Stream that can be used to read the final storage for the archive.

Implements

IArchiveStorage(ConvertTemporaryToFinal)

See Also

BaseArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace | GetTemporaryOutput
ICSharpCode SharpZipLib Class Library
BaseArchiveStorage.Dispose Method

Disposes this instance.

```csharp
public abstract void Dispose();
```

Implements

IArchiveStorage.Dispose

See Also

BaseArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
BaseArchiveStorage.GetTemporaryOutput Method

Gets a temporary output Stream

```csharp
public abstract Stream GetTemporaryOutput();
```

Return Value

Returns the temporary output stream.

Implements

IArchiveStorage.GetTemporaryOutput

See Also

BaseArchiveStorage Class | ICSharpCode.SharpZipLib.Zip Namespace | ConvertTemporaryToFinal
Make a temporary copy of a Stream.

```csharp
public abstract Stream MakeTemporaryCopy(Stream stream);
```

**Parameters**

*stream*

The Stream to make a copy of.

**Return Value**

Returns a temporary output Stream that is a copy of the input.

**Implements**

IArchiveStorage.MakeTemporaryCopy

**See Also**

BaseArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
Return a stream suitable for performing direct updates on the original source.

```csharp
public abstract Stream OpenForDirectUpdate(Stream stream);
```

**Return Value**

Returns a stream suitable for direct updating.

**Implements**

`IArchiveStorage.OpenForDirectUpdate`

**See Also**

`BaseArchiveStorage Class` | `ICSharpCode.SharpZipLib.Zip Namespace`
ICSharpCode SharpZipLib Class Library
CompressionMethod Enumeration

The kind of compression used for an entry in an archive

```csharp
public enum CompressionMethod
```

Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored</td>
<td>A direct copy of the file contents is held in the archive</td>
</tr>
<tr>
<td>Deflated</td>
<td>Common Zip compression method using a sliding dictionary of up to 32KB and secondary compression from Huffman/Shannon-Fano trees</td>
</tr>
<tr>
<td>Deflate64</td>
<td>An extension to deflate with a 64KB window. Not supported by #Zip currently</td>
</tr>
<tr>
<td>BZip2</td>
<td>Not supported by #Zip currently</td>
</tr>
<tr>
<td>WinZipAES</td>
<td>WinZip special for AES encryption, Not supported by #Zip</td>
</tr>
</tbody>
</table>

Requirements


See Also

ICSharpCode SharpZipLib Class Library
DiskArchiveStorage Class

An `IArchiveStorage` implementation suitable for hard disks.

For a list of all members of this type, see `DiskArchiveStorage Members`.

```
```

```
public class DiskArchiveStorage : BaseArchiveStorage
```

**Thread Safety**

Public static (`Shared` in Visual Basic) members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** `ICSharpCode.SharpZipLib.Zip`

**Assembly:** `ICSharpCode.SharpZipLib` (in `ICSharpCode.SharpZipLib.dll`)

**See Also**

ICSharpCode SharpZipLib Class Library
### DiskArchiveStorage Members

**DiskArchiveStorage overview**

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiskArchiveStorage</td>
<td>Overloaded. Initializes a new instance of the DiskArchiveStorage class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdateMode</td>
<td>Gets the update mode applicable.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertTemporaryToFinal</td>
<td>Converts a temporary Stream to its final form.</td>
</tr>
<tr>
<td>Dispose</td>
<td>Disposes this instance.</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetTemporaryOutput</td>
<td>Gets a temporary output Stream for performing updates on.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>MakeTemporaryCopy</td>
<td>Make a temporary copy of a stream.</td>
</tr>
<tr>
<td>OpenForDirectUpdate</td>
<td>Return a stream suitable for performing direct updates on the original source.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th><strong>Finalize</strong> (inherited from Object)</th>
<th>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
DiskArchiveStorage Constructor

Initializes a new instance of the `DiskArchiveStorage` class.

**Overload List**

Initializes a new instance of the `DiskArchiveStorage` class.

```csharp
public DiskArchiveStorage(ZipFile);
```

Initializes a new instance of the `DiskArchiveStorage` class.

```csharp
public DiskArchiveStorage(ZipFile, FileUpdateMode);
```

**See Also**

`DiskArchiveStorage Class` | `ICSharpCode.SharpZipLib.Zip Namespace`
ICSharpCode SharpZipLib Class Library
DiskArchiveStorage Constructor (ZipFile, FileUpdateMode)

Initializes a new instance of the DiskArchiveStorage class.

```java
public DiskArchiveStorage(
    ZipFile file,
    FileUpdateMode updateMode
);
```

Parameters

- `file`
  - The file.
- `updateMode`
  - The update mode.

See Also

- [DiskArchiveStorage Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [DiskArchiveStorage Constructor Overload List](#)
DiskArchiveStorage Constructor (ZipFile)

Initializes a new instance of the DiskArchiveStorage class.

```csharp
public DiskArchiveStorage(
    ZipFile file
);
```

Parameters

- `file`
  - The file.

See Also

ICSharpCode SharpZipLib Class Library
## DiskArchiveStorage Methods

The methods of the **DiskArchiveStorage** class are listed below. For a complete list of **DiskArchiveStorage** class members, see the [DiskArchiveStorage Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConvertTemporaryToFinal</strong></td>
<td>Converts a temporary Stream to its final form.</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes this instance.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetTemporaryOutput</strong></td>
<td>Gets a temporary output Stream for performing updates on.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>MakeTemporaryCopy</strong></td>
<td>Make a temporary copy of a stream.</td>
</tr>
<tr>
<td><strong>OpenForDirectUpdate</strong></td>
<td>Return a stream suitable for performing direct updates on the original source.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by</td>
</tr>
</tbody>
</table>
garbage collection.

| 🍂 **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

See Also

- DiskArchiveStorage Class
ICSharpCode SharpZipLib Class Library
DiskArchiveStorage.ConvertTemporaryToFinal Method

Converts a temporary Stream to its final form.

```csharp
public override Stream ConvertTemporaryToFinal();
```

Return Value

Returns a Stream that can be used to read the final storage for the archive.

Implements

IArchiveStorage.ConvertTemporaryToFinal

See Also

DiskArchiveStorage Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
DiskArchiveStorage.Dispose Method

Disposes this instance.

```csharp
public override void Dispose();
```

Implements

IArchiveStorage.Dispose

See Also

DiskArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
DiskArchiveStorage.GetTemporaryOutput Method

Gets a temporary output Stream for performing updates on.

```csharp
public override Stream GetTemporaryOutput();
```

Return Value

Returns the temporary output stream.

Implements

IArchiveStorage.GetTemporaryOutput

See Also

DiskArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
DiskArchiveStorage.MakeTemporaryCopy Method

Make a temporary copy of a stream.

```csharp
public override Stream MakeTemporaryCopy(Stream stream);
```

Parameters

`stream`
- The `Stream` to copy.

Return Value

Returns a temporary output `Stream` that is a copy of the input.

Implements

IArchiveStorage.MakeTemporaryCopy

See Also

DiskArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
DiskArchiveStorage.OpenForDirectUpdate Method

Return a stream suitable for performing direct updates on the original source.

```csharp
public override Stream OpenForDirectUpdate(Stream current);
```

Return Value

Returns a stream suitable for direct updating.

Implements

IArchiveStorage.OpenForDirectUpdate

See Also

DiskArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
**EncryptionAlgorithm Enumeration**

Identifies the encryption algorithm used for an entry

```java
public enum EncryptionAlgorithm
```

**Members**

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>No encryption has been used.</td>
</tr>
<tr>
<td>PkzipClassic</td>
<td>Encrypted using PKZIP 2.0 or 'classic' encryption.</td>
</tr>
<tr>
<td>Des</td>
<td>DES encryption has been used.</td>
</tr>
<tr>
<td>RC2</td>
<td>RCS encryption has been used for encryption.</td>
</tr>
<tr>
<td>TripleDes168</td>
<td>Triple DES encryption with 168 bit keys has been used for this entry.</td>
</tr>
<tr>
<td>TripleDes112</td>
<td>Triple DES with 112 bit keys has been used for this entry.</td>
</tr>
<tr>
<td>Aes128</td>
<td>AES 128 has been used for encryption.</td>
</tr>
<tr>
<td>Aes192</td>
<td>AES 192 has been used for encryption.</td>
</tr>
<tr>
<td>Aes256</td>
<td>AES 256 has been used for encryption.</td>
</tr>
<tr>
<td>RC2Corrected</td>
<td>RC2 corrected has been used for encryption.</td>
</tr>
<tr>
<td>Blowfish</td>
<td>Blowfish has been used for encryption.</td>
</tr>
<tr>
<td>Twofish</td>
<td>Twofish has been used for encryption.</td>
</tr>
<tr>
<td>RC4</td>
<td>RCS has been used for</td>
</tr>
</tbody>
</table>
An unknown algorithm has been used for encryption.

Requirements


See Also

ICSHarpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
FastZip Class

FastZip provides facilities for creating and extracting zip files. Only relative paths are supported.

For a list of all members of this type, see FastZip Members.


public class FastZip

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

## FastZip Members

### FastZip overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastZip</td>
<td>Overloaded. Initializes a new instance of the FastZip class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateEmptyDirectories</td>
<td>Get/set a value indicating whether empty directories should be created.</td>
</tr>
<tr>
<td>EntryFactory</td>
<td>Get or set the IEntryFactory active when creating Zip files.</td>
</tr>
<tr>
<td>NameTransform</td>
<td>Get or set the INameTransform active when creating Zip files.</td>
</tr>
<tr>
<td>Password</td>
<td>Get / set the password value.</td>
</tr>
<tr>
<td>RestoreDateTimeOnExtract</td>
<td>Get/set a value indicating whether file dates and times should be restored when extracting files from an archive.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateZip</td>
<td>Overloaded. Create a zip file.</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>ExtractZip</td>
<td>Overloaded. Extract the contents of a zip file.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

| **Finalize** (inherited from **Object**) | Allows an **Object** to attempt to free resources and perform other cleanup operations before the **Object** is reclaimed by garbage collection. |
| **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

[FastZip Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
FastZip Constructor

Initialise a default instance of FastZip.

Overload List

Initialise a default instance of FastZip.

```csharp
public FastZip();
```

Initialise a new instance of FastZip

```csharp
public FastZip(FastZipEvents);
```

See Also

ICSharpCode SharpZipLib Class Library
FastZip Constructor ()

Initialise a default instance of FastZip.

```csharp
public FastZip();
```

See Also

ICSharpCode SharpZipLib Class Library
FastZip Constructor (FastZipEvents)

Initialise a new instance of FastZip

```csharp
public FastZip(FastZipEvents events);
```

Parameters

*events*

The *events* to use during operations.

See Also

ICSharpCode SharpZipLib Class Library
FastZip Properties

The properties of the FastZip class are listed below. For a complete list of FastZip class members, see the FastZip Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateEmptyDirectories</td>
<td>Get/set a value indicating whether empty directories should be created.</td>
</tr>
<tr>
<td>EntryFactory</td>
<td>Get or set the IEntryFactory active when creating Zip files.</td>
</tr>
<tr>
<td>NameTransform</td>
<td>Get or set the INameTransform active when creating Zip files.</td>
</tr>
<tr>
<td>Password</td>
<td>Get / set the password value.</td>
</tr>
<tr>
<td>RestoreDateTimeOnExtract</td>
<td>Get/set a value indicating whether file dates and times should be restored when extracting files from an archive.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
FastZip.CreateEmptyDirectories Property

Get/set a value indicating whether empty directories should be created.

```csharp
public bool CreateEmptyDirectories {get; set;}
```

See Also

ICSharpCode SharpZipLib Class Library
FastZip.EntryFactory Property

Get or set the IEntryFactory active when creating Zip files.

```csharp
public IEntryFactory EntryFactory {get; set;}
```

See Also

ICSharpCode SharpZipLib Class Library
FastZip.NameTransform Property

Get or set the **INameTransform** active when creating Zip files.

```csharp
public ISharpCode.SharpZipLib.Core.INameTransform
```

See Also

ICSharpCode SharpZipLib Class Library
FastZip.Password Property

Get / set the password value.

```csharp
public string Password {get; set;}
```

See Also

ICSharpCode SharpZipLib Class Library
**FastZip.RestoreDateTimeOnExtract Property**

Get/set a value indicating whether file dates and times should be restored when extracting files from an archive.

```csharp
public bool RestoreDateTimeOnExtract {get; set;}
```

**Remarks**
The default value is false.

**See Also**
- [FastZip Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
### FastZip Methods

The methods of the **FastZip** class are listed below. For a complete list of **FastZip** class members, see the **FastZip Members** topic.

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✷ <strong>CreateZip</strong></td>
<td>Overloaded. Create a zip file.</td>
</tr>
<tr>
<td>✷ <strong>Equals</strong> <em>(inherited from Object)</em></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>✷ <strong>ExtractZip</strong></td>
<td>Overloaded. Extract the contents of a zip file.</td>
</tr>
<tr>
<td>✷ <strong>GetHashCode</strong> <em>(inherited from Object)</em></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>✷ <strong>GetType</strong> <em>(inherited from Object)</em></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>✷ <strong>ToString</strong> <em>(inherited from Object)</em></td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

#### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✷ <strong>Finalize</strong> <em>(inherited from Object)</em></td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>✷ <strong>MemberwiseClone</strong> <em>(inherited from Object)</em></td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

#### See Also

ICSharpCode SharpZipLib Class Library
FastZip.CreateZip Method

Create a zip archive sending output to the *outputStream* passed.

**Overload List**

Create a zip archive sending output to the *outputStream* passed.

```csharp
public void CreateZip(Stream,string,bool,string,string);
```

Create a zip file/archive.

```csharp
public void CreateZip(string,string,bool,string);
```

Create a zip file.

```csharp
public void CreateZip(string,string,bool,string,string);
```

**See Also**

ICSharpCode SharpZipLib Class Library
Create a zip archive sending output to the `outputStream` passed.

```csharp
public void CreateZip(
    Stream outputStream,
    string sourceDirectory,
    bool recurse,
    string fileFilter,
    string directoryFilter
);
```

### Parameters

- **outputStream**
  The stream to write archive data to.

- **sourceDirectory**
  The directory to source files from.

- **recurse**
  True to recurse directories, false for no recursion.

- **fileFilter**
  The file filter to apply.

- **directoryFilter**
  The directory filter to apply.

### See Also

ICSharpCode SharpZipLib Class Library
FastZip.CreateZip Method (String, String, Boolean, String)

Create a zip file/archive.

```csharp
public void CreateZip(
    string zipFileName,
    string sourceDirectory,
    bool recurse,
    string fileFilter
);
```

Parameters

- `zipFileName`  
The name of the zip file to create.

- `sourceDirectory`  
The directory to obtain files and directories from.

- `recurse`  
  True to recurse directories, false for no recursion.

- `fileFilter`  
The file filter to apply.

See Also

ICSharpCode SharpZipLib Class Library
Create a zip file.

```csharp
public void CreateZip(
    string zipFileName,
    string sourceDirectory,
    bool recurse,
    string fileFilter,
    string directoryFilter
);
```

### Parameters

- **zipFileName**
  - The name of the zip file to create.

- **sourceDirectory**
  - The directory to source files from.

- **recurse**
  - True to recurse directories, false for no recursion.

- **fileFilter**
  - The file filter to apply.

- **directoryFilter**
  - The directory filter to apply.

### See Also

- [FastZip Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [FastZip.CreateZip Overload List](#)
ICSharpCode SharpZipLib Class Library
FastZip.ExtractZip Method

Extract the contents of a zip file.

Overload List

Extract the contents of a zip file.

public void ExtractZip(string,string,Overwrite,ConfirmOverwriteDelegate)

Extract the contents of a zip file.

public void ExtractZip(string,string,string);

See Also

ICSharpCode SharpZipLib Class Library
FastZip.ExtractZip Method (String, String, Overwrite, ConfirmOverwriteDelegate, String, String, Boolean)

Extract the contents of a zip file.

```csharp
public void ExtractZip(
    string zipFileName,
    string targetDirectory,
    Overwrite overwrite,
    ConfirmOverwriteDelegate confirmDelegate,
    string fileFilter,
    string directoryFilter,
    bool restoreDateTime
);
```

Parameters

- `zipFileName`  
  The zip file to extract from.

- `targetDirectory`  
  The directory to save extracted information in.

- `overwrite`  
  The style of overwriting to apply.

- `confirmDelegate`  
  A delegate to invoke when confirming overwriting.

- `fileFilter`  
  A filter to apply to files.

- `directoryFilter`  
  A filter to apply to directories.

- `restoreDateTime`  
  Flag indicating whether to restore the date and time for extracted files.

See Also

FastZip.ExtractZip Overload List
ICSharpCode SharpZipLib Class Library
FastZip.ExtractZip Method (String, String, String)

Extract the contents of a zip file.

```csharp
public void ExtractZip(
    string zipFileName,
    string targetDirectory,
    string fileFilter
);
```

Parameters

- `zipFileName`
  The zip file to extract from.

- `targetDirectory`
  The directory to save extracted information in.

- `fileFilter`
  A filter to apply to files.

See Also

- [FastZip Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [FastZip.ExtractZip Overload List](#)
FastZip.ConfirmOverwriteDelegate Delegate

Delegate called when confirming overwriting of files.

```csharp
public delegate bool FastZip.ConfirmOverwriteDelegate(
    string fileName);
```

Requirements


**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

See Also

ICSharpCode SharpZipLib Class Library
FastZip.Overwrite Enumeration

Defines the desired handling when overwriting files during extraction.

```java
public enum FastZip.Overwrite
```

Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt</td>
<td>Prompt the user to confirm overwriting</td>
</tr>
<tr>
<td>Never</td>
<td>Never overwrite files.</td>
</tr>
<tr>
<td>Always</td>
<td>Always overwrite files.</td>
</tr>
</tbody>
</table>

Requirements


See Also

FastZipEvents Class

FastZipEvents supports all events applicable to FastZip operations. For a list of all members of this type, see FastZipEvents Members.


```
public class FastZipEvents
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

**Namespace:** ISharpCode.SharpZipLib.Zip

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

See Also

ICSharpCode SharpZipLib Class Library
# FastZipEvents Members

## FastZipEvents overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FastZipEvents Constructor</strong></td>
<td>Initializes a new instance of the FastZipEvents class.</td>
</tr>
</tbody>
</table>

### Public Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DirectoryFailure</strong></td>
<td>Delegate to invoke when processing directory failures.</td>
</tr>
<tr>
<td><strong>FileFailure</strong></td>
<td>Delegate to invoke when processing file failures.</td>
</tr>
<tr>
<td><strong>ProcessDirectory</strong></td>
<td>Delegate to invoke when processing directories.</td>
</tr>
<tr>
<td><strong>ProcessFile</strong></td>
<td>Delegate to invoke when processing files.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>OnDirectoryFailure</strong></td>
<td>Raise the directory failure event.</td>
</tr>
<tr>
<td><strong>OnFileFailure</strong></td>
<td>Raises the file failure delegate.</td>
</tr>
<tr>
<td><strong>OnProcessDirectory</strong></td>
<td>Fires the process directory delegate.</td>
</tr>
<tr>
<td><strong>OnProcessFile</strong></td>
<td>Raises the Process File</td>
</tr>
</tbody>
</table>
**delegate.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delegate.<strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>delegate.<strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>[<strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
FastZipEvents Constructor

Initializes a new instance of the FastZipEvents class.

```java
public FastZipEvents();
```

See Also

ICSharpCode SharpZipLib Class Library
FastZipEvents Fields

The fields of the FastZipEvents class are listed below. For a complete list of FastZipEvents class members, see the FastZipEvents Members topic.

Public Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectoryFailure</td>
<td>Delegate to invoke when processing directory failures.</td>
</tr>
<tr>
<td>FileFailure</td>
<td>Delegate to invoke when processing file failures.</td>
</tr>
<tr>
<td>ProcessDirectory</td>
<td>Delegate to invoke when processing directories.</td>
</tr>
<tr>
<td>ProcessFile</td>
<td>Delegate to invoke when processing files.</td>
</tr>
</tbody>
</table>

See Also

FastZipEvents.DirectoryFailure Field

Delegate to invoke when processing directory failures.

```csharp
public DirectoryFailureDelegate DirectoryFailure;
```

See Also

ICSharpCode SharpZipLib Class Library
FastZipEvents.FileFailure Field

Delegate to invoke when processing file failures.

```csharp
public FileFailureDelegate FileFailure;
```

See Also

ICSharpCode SharpZipLib Class Library
FastZipEvents.ProcessDirectory Field

Delegate to invoke when processing directories.

```csharp
```

See Also

ICSharpCode SharpZipLib Class Library
FastZipEvents.ProcessFile Field

Delegate to invoke when processing files.

```csharp
public ProcessFileDelegate ProcessFile;
```

See Also

ICSharpCode SharpZipLib Class Library
The methods of the `FastZipEvents` class are listed below. For a complete list of `FastZipEvents` class members, see the `FastZipEvents Members` topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>OnDirectoryFailure</code></td>
<td>Raise the directory failure event.</td>
</tr>
<tr>
<td><code>OnFileFailure</code></td>
<td>Raises the file failure delegate.</td>
</tr>
<tr>
<td><code>OnProcessDirectory</code></td>
<td>Fires the process directory delegate.</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code></td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>
See Also

ICSharpCode SharpZipLib Class Library
Raise the directory failure event.

```csharp
public bool OnDirectoryFailure(
    string directory,
    Exception e
);
```

**Parameters**

- `directory`  
  The directory causing the failure.

- `e`  
  The exception for this event.

**Return Value**

A boolean indicating if execution should continue or not.

**See Also**

ICSharpCode SharpZipLib Class Library
FastZipEvents.OnFileFailure Method

Raises the file failure delegate.

```csharp
public bool OnFileFailure(
    string file,
    Exception e
);
```

**Parameters**

- **file**
  - The file causing the failure.

- **e**
  - The exception for this failure.

**Return Value**

A boolean indicating if execution should continue or not.

**See Also**

[FastZipEvents Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
Fires the `process directory` delegate.

```csharp
public void OnProcessDirectory(
    string directory,
    bool hasMatchingFiles
);
```

**Parameters**

- `directory`
  The directory being processed.

- `hasMatchingFiles`
  Flag indicating if directory has matching files as determined by the current filter.

**See Also**

ICSharpCode SharpZipLib Class Library
FastZipEvents.OnProcessFile Method

Raises the Process File delegate.

```csharp
class FastZipEvents
{
    public bool OnProcessFile(string file);
}
```

Parameters

- **file**: The file being processed.

Return Value

A boolean indicating if execution should continue or not.

See Also

ICSharpCode SharpZipLib Class Library
The possible ways of applying updates to an archive.

**public enum FileUpdateMode**

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safe</td>
<td>Perform all updates on temporary files ensuring that the original file is saved.</td>
</tr>
<tr>
<td>Direct</td>
<td>Update the archive directly, which is faster but less safe.</td>
</tr>
</tbody>
</table>

**Requirements**

**Namespace:** IICSharpCode.SharpZipLib.Zip

**Assembly:** IICSharpCode.SharpZipLib (in IICSharpCode.SharpZipLib.dll)

**See Also**

ICSharpCode SharpZipLib Class Library
GeneralBitFlags Enumeration

Defines the contents of the general bit flags field for an archive entry.
This enumeration has a FlagsAttribute attribute that allows a bitwise combination of its member values.

```java
public enum GeneralBitFlags {
    Encrypted(1),
    Method(6),
    Descriptor(8),
    ReservedPKware4(16),
    Patched(32),
    StrongEncryption(64),
    Unused7(128),
    Unused8(256);
}
```

Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypted</td>
<td>Bit 0 if set indicates that the file is encrypted</td>
<td>1</td>
</tr>
<tr>
<td>Method</td>
<td>Bits 1 and 2 - Two bits defining the compression method (only for Method 6 Imploding and 8,9 Deflating)</td>
<td>6</td>
</tr>
<tr>
<td>Descriptor</td>
<td>Bit 3 if set indicates a trailing data descriptor is appended to the entry data</td>
<td>8</td>
</tr>
<tr>
<td>ReservedPKware4</td>
<td>Bit 4 is reserved for use with method 8 for enhanced deflation</td>
<td>16</td>
</tr>
<tr>
<td>Patched</td>
<td>Bit 5 if set indicates the file contains Pkzip compressed patched data. Requires version 2.7 or greater.</td>
<td>32</td>
</tr>
<tr>
<td>StrongEncryption</td>
<td>Bit 6 if set strong encryption has been used for this entry.</td>
<td>64</td>
</tr>
<tr>
<td>Unused7</td>
<td>Bit 7 is currently unused</td>
<td>128</td>
</tr>
<tr>
<td>Unused8</td>
<td>Bit 8 is currently unused</td>
<td>256</td>
</tr>
<tr>
<td>Unused9</td>
<td>Bit 9 is currently unused</td>
<td>512</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Unused10</td>
<td>Bit 10 is currently unused</td>
<td>1024</td>
</tr>
<tr>
<td>UnicodeText</td>
<td>Bit 11 if set indicates the filename and comment fields for this file must be encoded using UTF-8.</td>
<td>2048</td>
</tr>
<tr>
<td>EnhancedCompress</td>
<td>Bit 12 is documented as being reserved by PKware for enhanced compression.</td>
<td>4096</td>
</tr>
<tr>
<td>HeaderMasked</td>
<td>Bit 13 if set indicates that values in the local header are masked to hide their actual values, and the central directory is encrypted.</td>
<td>8192</td>
</tr>
<tr>
<td>ReservedPkware14</td>
<td>Bit 14 is documented as being reserved for use by PKware</td>
<td>16384</td>
</tr>
<tr>
<td>ReservedPkware15</td>
<td>Bit 15 is documented as being reserved for use by PKware</td>
<td>32768</td>
</tr>
</tbody>
</table>

Requirements

See Also
**HostSystemID Enumeration**

Defines known values for the **HostSystemID** property.

```java
public enum HostSystemID
```

**Members**

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Msdos</td>
<td>Host system = MSDOS</td>
</tr>
<tr>
<td>Amiga</td>
<td>Host system = Amiga</td>
</tr>
<tr>
<td>OpenVms</td>
<td>Host system = Open VMS</td>
</tr>
<tr>
<td>Unix</td>
<td>Host system = Unix</td>
</tr>
<tr>
<td>VMCms</td>
<td>Host system = VMCms</td>
</tr>
<tr>
<td>AtariST</td>
<td>Host system = Atari ST</td>
</tr>
<tr>
<td>OS2</td>
<td>Host system = OS2</td>
</tr>
<tr>
<td>Macintosh</td>
<td>Host system = Macintosh</td>
</tr>
<tr>
<td>ZSystem</td>
<td>Host system = ZSystem</td>
</tr>
<tr>
<td>Cpm</td>
<td>Host system = Cpm</td>
</tr>
<tr>
<td>WindowsNT</td>
<td>Host system = Windows NT</td>
</tr>
<tr>
<td>MVS</td>
<td>Host system = MVS</td>
</tr>
<tr>
<td>Vse</td>
<td>Host system = VSE</td>
</tr>
<tr>
<td>AcornRisc</td>
<td>Host system = Acorn RISC</td>
</tr>
<tr>
<td>Vfat</td>
<td>Host system = VFAT</td>
</tr>
<tr>
<td>AlternateMvs</td>
<td>Host system = Alternate MVS</td>
</tr>
<tr>
<td>BeOS</td>
<td>Host system = BEOS</td>
</tr>
<tr>
<td>Tandem</td>
<td>Host system = Tandem</td>
</tr>
<tr>
<td>OS400</td>
<td>Host system = OS400</td>
</tr>
<tr>
<td>OSX</td>
<td>Host system = OSX</td>
</tr>
</tbody>
</table>
WinZipAES

Requirements


See Also

ICSharpCode SharpZipLib Class Library
IArchiveStorage Interface

Defines facilities for data storage when updating Zip Archives.
For a list of all members of this type, see IArchiveStorage Members.

```
public interface IArchiveStorage
```

Types that implement IArchiveStorage

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseArchiveStorage</td>
<td>An abstract IArchiveStorage suitable for extension by inheritance.</td>
</tr>
<tr>
<td>DiskArchiveStorage</td>
<td>An IArchiveStorage implementation suitable for hard disks.</td>
</tr>
<tr>
<td>MemoryArchiveStorage</td>
<td>An IArchiveStorage implementation suitable for in memory streams.</td>
</tr>
</tbody>
</table>

Requirements

- **Namespace:** ISharpCode.SharpZipLib.Zip
- **Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

See Also

- IArchiveStorage Members | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
IArchiveStorage Members

IArchiveStorage overview

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UpdateMode</strong></td>
<td>Get the FileUpdateMode to apply during updates.</td>
</tr>
</tbody>
</table>

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConvertTemporaryToFinal</strong></td>
<td>Convert a temporary output stream to a final stream.</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Dispose of this instance.</td>
</tr>
<tr>
<td><strong>GetTemporaryOutput</strong></td>
<td>Get an empty Stream that can be used for temporary output.</td>
</tr>
<tr>
<td><strong>MakeTemporaryCopy</strong></td>
<td>Make a temporary copy of the original stream.</td>
</tr>
<tr>
<td><strong>OpenForDirectUpdate</strong></td>
<td>Return a stream suitable for performing direct updates on the original source.</td>
</tr>
</tbody>
</table>

See Also

IArchiveStorage Interface | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
The properties of the `IArchiveStorage` interface are listed below. For a complete list of `IArchiveStorage` interface members, see the `IArchiveStorage Members` topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>UpdateMode</code></td>
<td>Get the <code>FileUpdateMode</code> to apply during updates.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
Get the `FileUpdateMode` to apply during updates.

```csharp
public FileUpdateMode UpdateMode { get; }
```

See Also

- `IArchiveStorage Interface`
ICSharpCode SharpZipLib Class Library
The methods of the `IArchiveStorage` interface are listed below. For a complete list of `IArchiveStorage` interface members, see the `IArchiveStorage Members` topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertTemporaryToFinal</td>
<td>Convert a temporary output stream to a final stream.</td>
</tr>
<tr>
<td>Dispose</td>
<td>Dispose of this instance.</td>
</tr>
<tr>
<td>GetTemporaryOutput</td>
<td>Get an empty <code>Stream</code> that can be used for temporary output.</td>
</tr>
<tr>
<td>MakeTemporaryCopy</td>
<td>Make a temporary copy of the original stream.</td>
</tr>
<tr>
<td>OpenForDirectUpdate</td>
<td>Return a stream suitable for performing direct updates on the original source.</td>
</tr>
</tbody>
</table>

### See Also

IArchiveStorage(ConvertTemporaryToFinal Method

Convert a temporary output stream to a final stream.

```
Stream ConvertTemporaryToFinal();
```

Return Value

The resulting final Stream

See Also

IArchiveStorage Interface | ISharpCode.SharpZipLib.Zip Namespace | GetTemporaryOutput
ICSharpCode SharpZipLib Class Library
Dispose of this instance.

```csharp
void Dispose();
```

See Also

Get an empty Stream that can be used for temporary output.

```csharp
Stream GetTemporaryOutput();
```

Return Value

Returns a temporary output Stream

See Also

IArchiveStorage Interface | ISharpCode.SharpZipLib.Zip Namespace | ConvertTemporaryToFinal
IArchiveStorage.MakeTemporaryCopy Method

Make a temporary copy of the original stream.

```csharp
Stream MakeTemporaryCopy(
    Stream stream
);
```

Parameters

- `stream`  
  The `Stream` to copy.

Return Value

Returns a temporary output `Stream` that is a copy of the input.

See Also

ICSharpCode SharpZipLib Class Library
**IArchiveStorage.OpenForDirectUpdate Method**

Return a stream suitable for performing direct updates on the original source.

```csharp
Stream OpenForDirectUpdate(
    Stream stream
);
```

**Parameters**

- `stream`  
  The current stream.

**Return Value**

Returns a stream suitable for direct updating.

**Remarks**

This may be the current stream passed.

**See Also**

- [IArchiveStorage Interface](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
IDynamicDataSource Interface

Represents a source of data that dynamically provide multiple data sources based on the parameters passed.

For a list of all members of this type, see IDynamicDataSource Members.

```
public interface IDynamicDataSource
```

Requirements


See Also

IDynamicDataSource Members | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
**IDynamicDataSource Members**

**IDynamicDataSource overview**

**Public Instance Methods**

| **GetSource** | Get a data source. |

**See Also**

ICSharpCode SharpZipLib Class Library
IDynamicDataSource Methods

The methods of the IDynamicDataSource interface are listed below. For a complete list of IDynamicDataSource interface members, see the IDynamicDataSource Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetSource</strong></td>
<td>Get a data source.</td>
</tr>
</tbody>
</table>

See Also

[IDataBaseSource Interface](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
Get a data source.

```csharp
Stream GetSource(
    ZipEntry entry,
    string name
);
```

### Parameters

**entry**
- The `ZipEntry` to get a source for.

**name**
- The name for data if known.

### Return Value
- Returns a `Stream` to use for compression input.

### See Also
ICSharpCode SharpZipLib Class Library
IEntryFactory Interface

Defines factory methods for creating new ZipEntry values.
For a list of all members of this type, see IEntryFactory Members.

public interface IEntryFactory

Requirements


See Also

IEntryFactory Members | ICSharpCode.SharpZipLib.Zip Namespace
## IEntryFactory Members

### IEntryFactory overview

#### Public Instance Properties

| NameTransform | Get/set the INameTransform applicable. |

#### Public Instance Methods

| MakeDirectoryEntry | Create a ZipEntry for a directory given its name |
| MakeFileEntry | Create a ZipEntry for a file given its name |

### See Also

ICSharpCode SharpZipLib Class Library
The properties of the **IEntryFactory** interface are listed below. For a complete list of **IEntryFactory** interface members, see the **IEntryFactory Members** topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NameTransform</strong></td>
<td>Get/set the <strong>INameTransform</strong> applicable.</td>
</tr>
</tbody>
</table>

### See Also

- **IEntryFactory Interface**  |  **ICSharpCode.SharpZipLib.Zip Namespace**
ICSharpCode SharpZipLib Class Library
IEntryFactory.NameTransform Property

Get/set the INameTransform applicable.

See Also

ICSharpCode SharpZipLib Class Library
IEntryFactory Methods

The methods of the IEntryFactory interface are listed below. For a complete list of IEntryFactory interface members, see the IEntryFactory Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MakeDirectoryEntry</td>
<td>Create a ZipEntry for a directory given its name</td>
</tr>
<tr>
<td>MakeFileEntry</td>
<td>Create a ZipEntry for a file given its name</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
Create a **ZipEntry** for a directory given its name

```csharp
ZipEntry MakeDirectoryEntry(
    string directoryName
);
```

**Parameters**

*directoryName*

The name of the directory to create an entry for.

**Return Value**

**See Also**

ICSharpCode SharpZipLib Class Library
Create a `ZipEntry` for a file given its name

```csharp
ZipEntry MakeFileEntry(string fileName);
```

Parameters

`fileName`
The name of the file to create an entry for.

Return Value

See Also

IStaticDataSource Interface

Provides a static way to obtain a source of data for an entry.
For a list of all members of this type, see IStaticDataSource Members.

```
public interface IStaticDataSource
```

Remarks
The

Requirements


See Also
IStaticDataSource Members | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
IStaticDataSource Members

IStaticDataSource overview

Public Instance Methods

| GetSource | Get a data source. |

See Also

IStaticDataSource Interface | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
The methods of the `IStaticDataSource` interface are listed below. For a complete list of `IStaticDataSource` interface members, see the `IStaticDataSource Members` topic.

### Public Instance Methods

| GetSource | Get a data source. |

### See Also

- `IStaticDataSource Interface` | `ICSharpCode.SharpZipLib.Zip Namespace`
ICSharpCode SharpZipLib Class Library
IStaticDataSource.GetSource Method

Get a data source.

```csharp
Stream GetSource();
```

Return Value

Returns a `Stream` to use for compression input.

See Also

- `IStaticDataSource Interface`
Arguments used with KeysRequiredEvent

For a list of all members of this type, see KeysRequiredEventArgs Members.

System.Object System.EventArgs

public class KeysRequiredEventArgs : EventArgs

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

## KeysRequiredEventArgs Members

### KeysRequiredEventArgs overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KeysRequiredEventArgs</strong> Overloaded. Initializes a new instance of the KeysRequiredEventArgs class.</td>
<td></td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fileName</strong></td>
<td>Get the name of the file for which keys are required.</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>Get/set the key value</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

#### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited)</td>
<td>Creates a shallow copy of the</td>
</tr>
</tbody>
</table>
See Also

KeysRequiredEventArgs Constructor

Initialise a new instance of *KeysRequiredEventArgs*

**Overload List**

Initialise a new instance of *KeysRequiredEventArgs*

public KeysRequiredEventArgs(string);

Initialise a new instance of *KeysRequiredEventArgs*

public KeysRequiredEventArgs(string, byte[]);

See Also

ICSharpCode SharpZipLib Class Library
KeysRequiredEventArgs Constructor (String)

Initialise a new instance of `KeysRequiredEventArgs`:

```csharp
public KeysRequiredEventArgs(
    string name
);
```

Parameters

`name`  
The name of the file for which keys are required.

See Also

ICSharpCode SharpZipLib Class Library
Initialise a new instance of `KeysRequiredEventArgs`:

```csharp
public KeysRequiredEventArgs(
    string name,
    byte[] keyValue
);
```

**Parameters**

- `name`
  The name of the file for which keys are required.

- `keyValue`
  The current key value.

**See Also**

- [KeysRequiredEventArgs Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [KeysRequiredEventArgs Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
The properties of the **KeysRequiredEventArgs** class are listed below. For a complete list of **KeysRequiredEventArgs** class members, see the [KeysRequiredEventArgs Members](#) topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FileName</strong></td>
<td>Get the name of the file for which keys are required.</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>Get/set the key value</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
KeysRequiredEventArgs.FileName Property

Get the name of the file for which keys are required.

```csharp
public string FileName {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
**KeysRequiredEventArgs.Key Property**

Get/set the key value

```csharp
public byte[] Key {get; set;}
```

See Also

ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage Class

An **IArchiveStorage** implementation suitable for in memory streams.

For a list of all members of this type, see **MemoryArchiveStorage Members**.


```csharp
public class MemoryArchiveStorage : BaseArchiveStorage
```

**Thread Safety**

Public static (**Shared** in Visual Basic) members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** **ICSharpCode.SharpZipLib.Zip**

**Assembly:** **ICSharpCode.SharpZipLib (in IICSharpCode.SharpZipLib.dll)**

**See Also**

**MemoryArchiveStorage Members** | **ICSharpCode.SharpZipLib.Zip Namespace**
ICSharpCode SharpZipLib Class Library
# MemoryArchiveStorage Members

## MemoryArchiveStorage overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemoryArchiveStorage</td>
<td>Overloaded. Initializes a new instance of the MemoryArchiveStorage class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FinalStream</td>
<td>Get the stream returned by ConvertTemporaryToFinal if this was in fact called.</td>
</tr>
<tr>
<td>UpdateMode</td>
<td>Gets the update mode applicable. (inherited from BaseArchiveStorage)</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertTemporaryToFinal</td>
<td>Converts the temporary Stream to its final form.</td>
</tr>
<tr>
<td>Dispose</td>
<td>Disposes this instance.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object. (inherited from Object)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. (inherited from Object)</td>
</tr>
<tr>
<td>GetTemporaryOutput</td>
<td>Gets the temporary output Stream.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (inherited from Object)</td>
</tr>
<tr>
<td>MakeTemporaryCopy</td>
<td>Make a temporary copy of the original stream.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>OpenForDirectUpdate</strong></td>
<td>Return a stream suitable for performing direct updates on the original source.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage Constructor

Initializes a new instance of the MemoryArchiveStorage class.

Overload List

Initializes a new instance of the MemoryArchiveStorage class.

public MemoryArchiveStorage();

Initializes a new instance of the MemoryArchiveStorage class.

public MemoryArchiveStorage(FileUpdateMode);

See Also

MemoryArchiveStorage Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage Constructor ()

Initializes a new instance of the MemoryArchiveStorage class.

public MemoryArchiveStorage();

See Also

MemoryArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace | MemoryArchiveStorage Constructor Overload List
ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage Constructor (FileUpdateMode)

Initializes a new instance of the MemoryArchiveStorage class.

```csharp
public MemoryArchiveStorage(
    FileUpdateMode updateMode
);
```

Parameters

- `updateMode` The FileUpdateMode to use

Remarks

This constructor is for testing as memory streams don't really require safe mode.

See Also

- MemoryArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace | MemoryArchiveStorage Constructor Overload List
ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage Properties

The properties of the MemoryArchiveStorage class are listed below. For a complete list of MemoryArchiveStorage class members, see the MemoryArchiveStorage Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FinalStream</td>
<td>Get the stream returned by ConvertTemporaryToFinal if this was in fact called.</td>
</tr>
<tr>
<td>UpdateMode (inherited from BaseArchiveStorage)</td>
<td>Gets the update mode applicable.</td>
</tr>
</tbody>
</table>

See Also

MemoryArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage.FinalStream Property

Get the stream returned by ConvertTemporaryToFinal if this was in fact called.

```csharp
public System.IO.MemoryStream FinalStream {get;}
```

See Also

MemoryArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
### MemoryArchiveStorage Methods

The methods of the **MemoryArchiveStorage** class are listed below. For a complete list of **MemoryArchiveStorage** class members, see the [MemoryArchiveStorage Members](#) topic.

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConvertTemporaryToFinal</strong></td>
<td>Converts the temporary <strong>Stream</strong> to its final form.</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes this instance.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetTemporaryOutput</strong></td>
<td>Gets the temporary output <strong>Stream</strong></td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>MakeTemporaryCopy</strong></td>
<td>Make a temporary copy of the original stream.</td>
</tr>
<tr>
<td><strong>OpenForDirectUpdate</strong></td>
<td>Return a stream suitable for performing direct updates on the original source.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

#### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by</td>
</tr>
</tbody>
</table>
garbage collection.

| 🌸 MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object. |

See Also

MemoryArchiveStorage Class | ICSHarpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage.ConvertTemporaryToFinal Method

Converts the temporary Stream to its final form.

```csharp
public override Stream ConvertTemporaryToFinal();
```

Return Value

Returns a Stream that can be used to read the final storage for the archive.

Implements

IArchiveStorage.ConvertTemporaryToFinal

See Also

MemoryArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage.Dispose Method

Disposes this instance.

```csharp
public override void Dispose();
```

Implements

IArchiveStorage.Dispose

See Also

MemoryArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
MemoryArchiveStorage.GetTemporaryOutput Method

Gets the temporary output Stream

```csharp
public override Stream GetTemporaryOutput();
```

Return Value

Returns the temporary output stream.

Implements

IArchiveStorage.GetTemporaryOutput

See Also

MemoryArchiveStorage Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage.MakeTemporaryCopy Method

Make a temporary copy of the original stream.

```csharp
public override Stream MakeTemporaryCopy(Stream stream);
```

Parameters

- **stream**
  The `Stream` to copy.

Return Value

Returns a temporary output `Stream` that is a copy of the input.

Implements

- `IArchiveStorage.MakeTemporaryCopy`

See Also

- `MemoryArchiveStorage Class` | `ICSharpCode.SharpZipLib.Zip Namespace`
ICSharpCode SharpZipLib Class Library
MemoryArchiveStorage.OpenForDirectUpdate Method

Return a stream suitable for performing direct updates on the original source.

```csharp
public override Stream OpenForDirectUpdate(Stream stream);
```

Return Value

Returns a stream suitable for direct updating.

Implements

IArchiveStorage.OpenForDirectUpdate

See Also

MemoryArchiveStorage Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
TestOperation Enumeration

The operation in progress reported by a `ZipTestResultHandler` during testing.

```java
public enum TestOperation
```

Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialising</td>
<td>Setting up testing.</td>
</tr>
<tr>
<td>EntryHeader</td>
<td>Testing an individual entries header</td>
</tr>
<tr>
<td>EntryData</td>
<td>Testing an individual entries data</td>
</tr>
<tr>
<td>EntryComplete</td>
<td>Testing an individual entry has completed.</td>
</tr>
<tr>
<td>MiscellaneousTests</td>
<td>Running miscellaneous tests</td>
</tr>
<tr>
<td>Complete</td>
<td>Testing is complete</td>
</tr>
</tbody>
</table>

Requirements


See Also

ICSharpCode SharpZipLib Class Library
TestStatus Class

Status returned returned by ZipTestResultHandler during testing. For a list of all members of this type, see TestStatus Members.


public class TestStatus

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

TestStatus Members | ISharpCode.SharpZipLib.Zip Namespace | TestArchive
ICSharpCode SharpZipLib Class Library
## TestStatus Members

### TestStatus overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestStatus Constructor</td>
<td>Initialise a new instance of TestStatus</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesTested</td>
<td>Get the number of bytes tested so far for the current entry.</td>
</tr>
<tr>
<td>Entry</td>
<td>Get the current/last entry tested.</td>
</tr>
<tr>
<td>EntryValid</td>
<td>Get a value indicating whether the last entry test was valid.</td>
</tr>
<tr>
<td>ErrorCount</td>
<td>Get the number of errors detected so far.</td>
</tr>
<tr>
<td>File</td>
<td>Get the ZipFile this status is applicable to.</td>
</tr>
<tr>
<td>Operation</td>
<td>Get the current TestOperation in progress.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals (Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode (Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString (Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>
Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

- [TestStatus Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#) | [TestArchive](#)
ICSharpCode SharpZipLib Class Library
TestStatus Constructor

Initialise a new instance of TestStatus

```java
public TestStatus(
    ZipFile file
);
```

Parameters

*file*

The ZipFile this status applies to.

See Also

TestStatus Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
The properties of the `TestStatus` class are listed below. For a complete list of `TestStatus` class members, see the `TestStatus Members` topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BytesTested</code></td>
<td>Get the number of bytes tested so far for the current entry.</td>
</tr>
<tr>
<td><code>Entry</code></td>
<td>Get the current/last entry tested.</td>
</tr>
<tr>
<td><code>EntryValid</code></td>
<td>Get a value indicating whether the last entry test was valid.</td>
</tr>
<tr>
<td><code>ErrorCount</code></td>
<td>Get the number of errors detected so far.</td>
</tr>
<tr>
<td><code>File</code></td>
<td>Get the <code>ZipFile</code> this status is applicable to.</td>
</tr>
<tr>
<td><code>Operation</code></td>
<td>Get the current <code>TestOperation</code> in progress.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
TestStatus.BytesTested Property

Get the number of bytes tested so far for the current entry.

```csharp
public long BytesTested {get;}
```

See Also

[TestStatus Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
TestStatus.Entry Property

Get the current/last entry tested.

```csharp
public ZipEntry Entry {get;}
```

See Also

[TestStatus Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
Get a value indicating whether the last entry test was valid.

```csharp
public bool EntryValid {get;}
```

See Also

- TestStatus Class
**TestStatus>ErrorCount Property**

Get the number of errors detected so far.

```csharp
public int ErrorCount {get;}
```

See Also

[TestStatus Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
TestStatus.File Property

Get the ZipFile this status is applicable to.

```csharp
public ZipFile File {get;}
```

See Also

TestStatus Class | ICSharpCode.SharpZipLib.Zip Namespace
**TestStatus.Operation Property**

Get the current `TestOperation` in progress.

```csharp
public TestOperation Operation {get;}
```

See Also

- [TestStatus Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
**TestStrategy Enumeration**

The strategy to apply to testing.

```csharp
public enum TestStrategy
```

**Members**

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FindFirstError</td>
<td>Find the first error only.</td>
</tr>
<tr>
<td>FindAllErrors</td>
<td>Find all possible errors.</td>
</tr>
</tbody>
</table>

**Requirements**

- **Namespace:** [ICSharpCode.SharpZipLib.Zip](https://icsharpcode.net/SharpZipLib/)
- **Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

**See Also**

[ICSharpCode.SharpZipLib.Zip Namespace](https://icsharpcode.net/SharpZipLib/)

ICSharpCode SharpZipLib Class Library
UseZip64 Enumeration

Determines how entries are tested to see if they should use Zip64 extensions or not.

```java
public enum UseZip64
```

Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Zip64 will not be forced on entries during processing.</td>
</tr>
<tr>
<td>On</td>
<td>Zip64 should always be used.</td>
</tr>
<tr>
<td>Dynamic</td>
<td>#ZipLib will determine use based on entry values when added to archive.</td>
</tr>
</tbody>
</table>

Requirements


See Also

ICSharpCode SharpZipLib Class Library
ZipConstants Class

This class contains constants used for Zip format files
For a list of all members of this type, see ZipConstants Members.


public sealed class ZipConstants

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

### ZipConstants Members

#### ZipConstants overview

**Public Static Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArchiveExtraDataSignature</code></td>
<td>Signature for archive extra data signature (were headers are encrypted).</td>
</tr>
<tr>
<td><code>CENDIGITALSIG</code></td>
<td><strong>Obsolete.</strong> Central header digital signature</td>
</tr>
<tr>
<td><code>CENHDR</code></td>
<td><strong>Obsolete.</strong> Size of central header entry</td>
</tr>
<tr>
<td><code>CENSIG</code></td>
<td><strong>Obsolete.</strong> Signature for central header</td>
</tr>
<tr>
<td><code>CENSIG64</code></td>
<td><strong>Obsolete.</strong> Signature for Zip64 central file header</td>
</tr>
<tr>
<td><code>CentralHeaderBaseSize</code></td>
<td>Size of central header entry (excluding variable fields)</td>
</tr>
<tr>
<td><code>CentralHeaderDigitalSignature</code></td>
<td>Central header digital signature</td>
</tr>
<tr>
<td><code>CentralHeaderSignature</code></td>
<td>Signature for central header</td>
</tr>
<tr>
<td><code>CRYPTO_HEADER_SIZE</code></td>
<td><strong>Obsolete.</strong> Size of cryptographic header stored before entry data</td>
</tr>
<tr>
<td><code>CryptoHeaderSize</code></td>
<td>Size of 'classic' cryptographic header stored before any entry data</td>
</tr>
<tr>
<td><code>DataDescriptorSignature</code></td>
<td>Signature for data descriptor</td>
</tr>
<tr>
<td><code>DataDescriptorSize</code></td>
<td>Size of data descriptor</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENDHDR</td>
<td>Obsolete. Size of end of central record (excluding variable fields)</td>
</tr>
<tr>
<td>EndOfCentralDirectorySignature</td>
<td>End of central directory record signature</td>
</tr>
<tr>
<td>EndOfCentralRecordBaseSize</td>
<td>Size of end of central record (excluding variable fields)</td>
</tr>
<tr>
<td>ENDSIG</td>
<td>Obsolete. End of central directory record signature</td>
</tr>
<tr>
<td>EXTHDR</td>
<td>Obsolete. Size of data descriptor</td>
</tr>
<tr>
<td>EXTSIG</td>
<td>Obsolete. Signature for data descriptor</td>
</tr>
<tr>
<td>LocalHeaderBaseSize</td>
<td>Size of local entry header (excluding variable length fields at end)</td>
</tr>
<tr>
<td>LocalHeaderSignature</td>
<td>Signature for local entry header</td>
</tr>
<tr>
<td>LOCHDR</td>
<td>Obsolete. Size of local entry header (excluding variable length fields at end)</td>
</tr>
<tr>
<td>LOCSIG</td>
<td>Obsolete. Signature for local entry header</td>
</tr>
<tr>
<td>SPANNINGSIG</td>
<td>Obsolete. Signature for spanning entry</td>
</tr>
<tr>
<td>SpanningSignature</td>
<td>Signature for spanning entry</td>
</tr>
<tr>
<td>SpanningTempSignature</td>
<td>Signature for temporary spanning entry</td>
</tr>
<tr>
<td>SPANTEMPSIG</td>
<td>Obsolete. Signature for</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>❗️ <strong>VERSION</strong></td>
<td>❗️ <strong>VERSION</strong></td>
</tr>
<tr>
<td>❗️ <strong>VERSION</strong></td>
<td>❗️ <strong>VERSION</strong></td>
</tr>
<tr>
<td>❗️ <strong>VERSION</strong></td>
<td>❗️ <strong>VERSION</strong></td>
</tr>
<tr>
<td>❗️ <strong>VersionMadeBy</strong></td>
<td>❗️ <strong>VersionMadeBy</strong></td>
</tr>
<tr>
<td>❗️ <strong>VersionStrongEncryption</strong></td>
<td>❗️ <strong>VersionStrongEncryption</strong></td>
</tr>
<tr>
<td>❗️ <strong>VersionZip64</strong></td>
<td>❗️ <strong>VersionZip64</strong></td>
</tr>
<tr>
<td>❗️ <strong>Zip64CentralDirLocatorSignature</strong></td>
<td>❗️ <strong>Zip64CentralDirLocatorSignature</strong></td>
</tr>
<tr>
<td>❗️ <strong>Zip64CentralFileHeaderSignature</strong></td>
<td>❗️ <strong>Zip64CentralFileHeaderSignature</strong></td>
</tr>
<tr>
<td>❗️ <strong>Zip64DataDescriptorSize</strong></td>
<td>❗️ <strong>Zip64DataDescriptorSize</strong></td>
</tr>
</tbody>
</table>

**Public Static Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>❗️ <strong>DefaultCodePage</strong></td>
<td>❗️ <strong>DefaultCodePage</strong></td>
</tr>
</tbody>
</table>
choice for European users, however be careful about compatibility.

Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ConvertToArray</td>
<td>Overloaded. Convert a string to a byte array</td>
</tr>
<tr>
<td>$ConvertToString</td>
<td>Overloaded. Convert a portion of a byte array to a string.</td>
</tr>
<tr>
<td>$ConvertToStringExt</td>
<td>Overloaded. Convert a byte array to string</td>
</tr>
</tbody>
</table>

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>$GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>$GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>$ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

See Also

- ZipConstants Class
ICSharpCode SharpZipLib Class Library
### ZipConstants Fields

The fields of the **ZipConstants** class are listed below. For a complete list of **ZipConstants** class members, see the [ZipConstants Members](#) topic.

#### Public Static Fields

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArchiveExtraDataSignature</code></td>
<td>Signature for archive extra data signature (were headers are encrypted).</td>
</tr>
<tr>
<td><code>CENDIGITALSIG</code></td>
<td><strong>Obsolete.</strong> Central header digital signature</td>
</tr>
<tr>
<td><code>CENHDR</code></td>
<td><strong>Obsolete.</strong> Size of central header entry</td>
</tr>
<tr>
<td><code>CENSIG</code></td>
<td><strong>Obsolete.</strong> Signature for central header</td>
</tr>
<tr>
<td><code>CENSIG64</code></td>
<td><strong>Obsolete.</strong> Signature for Zip64 central file header</td>
</tr>
<tr>
<td><code>CentralHeaderBaseSize</code></td>
<td>Size of central header entry (excluding variable fields)</td>
</tr>
<tr>
<td><code>CentralHeaderDigitalSignature</code></td>
<td>Central header digital signature</td>
</tr>
<tr>
<td><code>CentralHeaderSignature</code></td>
<td>Signature for central header</td>
</tr>
<tr>
<td><code>CRYPTO_HEADER_SIZE</code></td>
<td><strong>Obsolete.</strong> Size of cryptographic header stored before entry data</td>
</tr>
<tr>
<td><code>CryptoHeaderSize</code></td>
<td>Size of 'classic' cryptographic header stored before any entry data</td>
</tr>
<tr>
<td><code>DataDescriptorSignature</code></td>
<td>Signature for data descriptor</td>
</tr>
<tr>
<td><strong>s</strong> Datadescsimple</td>
<td><strong>s</strong> Endhdr</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>s</strong> Datadesc</td>
<td><strong>s</strong> Endhdr</td>
</tr>
<tr>
<td>Size of data descriptor</td>
<td>Obsolete. Size of end of central record (excluding variable fields)</td>
</tr>
<tr>
<td>❍  SPANTEMPSIG</td>
<td><strong>Obsolete.</strong> Signature for temporary spanning entry</td>
</tr>
<tr>
<td>❍  VERSION_MADE_BY</td>
<td><strong>Obsolete.</strong> The version made by field for entries in the central header when created by this library</td>
</tr>
<tr>
<td>❍  VERSION_STRONG_ENCODPTION</td>
<td><strong>Obsolete.</strong> The minimum version required to support strong encryption</td>
</tr>
<tr>
<td>❍  VersionMadeBy</td>
<td>The version made by field for entries in the central header when created by this library</td>
</tr>
<tr>
<td>❍  VersionStrongEncryption</td>
<td>The minimum version required to support strong encryption</td>
</tr>
<tr>
<td>❍  VersionZip64</td>
<td>The version required for Zip64 extensions</td>
</tr>
<tr>
<td>❍  Zip64CentralDirLocatorSignature</td>
<td>Signature for Zip64 central directory locator</td>
</tr>
<tr>
<td>❍  Zip64CentralFileHeaderSignature</td>
<td>Signature for Zip64 central file header</td>
</tr>
<tr>
<td>❍  Zip64DataDescriptorSize</td>
<td>Size of Zip64 data descriptor</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
ZipConstants.ArchiveExtraDataSignature Field

Signature for archive extra data signature (were headers are encrypted).

```csharp
public const int ArchiveExtraDataSignature = 117853008;
```

See Also

ICSharpCode SharpZipLib Class Library
NOTE: This field is now obsolete. Use CentralHeaderDigitalSignature instead

Central header digital signature

```
public const int CENDIGITALSIG = 84233040;
```

See Also

[ZipConstants Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
**ZipConstants.CENHDR Field**

**NOTE:** This field is now obsolete.
Use CentralHeaderBaseSize instead

---

Size of central header entry

```csharp
public const int CENHDR = 46;
```

See Also

ICSharpCode SharpZipLib Class Library
NOTE: This field is now obsolete.
Use CentralHeaderSignature instead

Signature for central header

```csharp
public const int CENSIG = 33639248;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.CENSIG64 Field

NOTE: This field is now obsolete.
Use Zip64CentralFileHeaderSignature instead

Signature for Zip64 central file header

```csharp
public const int CENSIG64 = 101075792;
```

See Also

[ZipConstants Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
ZipConstants.CentralHeaderBaseSize Field

Size of central header entry (excluding variable fields)

```csharp
public const int CentralHeaderBaseSize = 46;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.CentralHeaderDigitalSignature Field

Central header digital signature

```csharp
public const int CentralHeaderDigitalSignature = 84233040;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.CentralHeaderSignature Field

Signature for central header

```java
public const int CentralHeaderSignature = 33639248;
```

See Also

ICSharpCode SharpZipLib Class Library
NOTE: This field is now obsolete.
Use CryptoHeaderSize instead

Size of cryptographic header stored before entry data

```csharp
public const int CRYPTO_HEADER_SIZE = 12;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.CryptoHeaderSize Field

Size of 'classic' cryptographic header stored before any entry data

```csharp
public const int CryptoHeaderSize = 12;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.DataDescriptorSignature Field

Signature for data descriptor

```csharp
public const int DataDescriptorSignature = 134695760;
```

Remarks

This is only used where the length, Crc, or compressed size isn't known when the entry is created and the output stream doesn't support seeking. The local entry cannot be 'patched' with the correct values in this case so the values are recorded after the data prefixed by this header, as well as in the central directory.

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.DataDescriptorSize Field

Size of data descriptor

```csharp
public const int DataDescriptorSize = 16;
```

See Also

**ZipConstants.ENDHDR Field**

**NOTE:** This field is now obsolete.

Use `EndOfCentralRecordBaseSize` instead

---

Size of end of central record (excluding variable fields)

```csharp
public const int ENDHDR = 22;
```

**See Also**

ICSharpCode SharpZipLib Class Library
ZipConstants.EndOfCentralDirectorySignature Field

End of central directory record signature

```java
public const int EndOfCentralDirectorySignature = 101010256;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.EndOfCentralRecordBaseSize Field

Size of end of central record (excluding variable fields)

```csharp
public const int EndOfCentralRecordBaseSize = 22;
```

See Also

[ZipConstants Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
## ZipConstants.ENDSIG Field

**NOTE:** This field is now obsolete.

Use EndOfCentralDirectorySignature instead

---

End of central directory record signature

```csharp
public const int ENDSIG = 101010256;
```

See Also

ICSharpCode SharpZipLib Class Library
NOTE: This field is now obsolete.
Use DataDescriptorSize instead

Size of data descriptor

```csharp
public const int EXTHDR = 16;
```

See Also

ICSharpCode SharpZipLib Class Library
**ZipConstants.EXTSIG Field**

**NOTE:** This field is now obsolete.

Use DataDescriptorSignature instead

---

**Signature for data descriptor**

```csharp
public const int EXTSIG = 134695760;
```

**Remarks**

This is only used where the length, Crc, or compressed size isn't known when the entry is created and the output stream doesn't support seeking. The local entry cannot be 'patched' with the correct values in this case so the values are recorded after the data prefixed by this header, as well as in the central directory.

**See Also**

ZipConstants.LocalHeaderBaseSize Field

Size of local entry header (excluding variable length fields at end)

```csharp
public const int LocalHeaderBaseSize = 30;
```

See Also

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipConstants.LocalHeaderSignature Field

Signature for local entry header

```csharp
public const int LocalHeaderSignature = 67324752;
```

See Also

ICSharpCode SharpZipLib Class Library
**ZipConstants.LOCHDR Field**

**NOTE:** This field is now obsolete.

Use LocalHeaderBaseSize instead

---

Size of local entry header (excluding variable length fields at end)

```csharp
public const int LOCHDR = 30;
```

**See Also**

[ZipConstants Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
**NOTE:** This field is now obsolete.

Use LocalHeaderSignature instead

---

Signature for local entry header

```csharp
public const int LOCSIG = 67324752;
```

**See Also**

ICSharpCode SharpZipLib Class Library
NOTE: This field is now obsolete.
Use SpanningSignature instead

Signature for spanning entry

```csharp
public const int SPANNINGSIG = 134695760;
```

See Also

ICSharpCode SharpZipLib Class Library
**ZipConstants.SpanningSignature Field**

Signature for spanning entry

```csharp
public const int SpanningSignature = 134695760;
```

See Also

[ZipConstants Class](javascript:) | [ICSharpCode.SharpZipLib.Zip Namespace](javascript:):
ICSharpCode SharpZipLib Class Library
ZipConstants.SpanningTempSignature Field

Signature for temporary spanning entry

```csharp
public const int SpanningTempSignature = 808471376;
```

See Also

ICSharpCode SharpZipLib Class Library
NOTE: This field is now obsolete.
Use SpanningTempSignature instead

Signature for temporary spanning entry

```csharp
public const int SPANTEMPSIG = 808471376;
```

See Also

- [ZipConstants Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
**ZipConstants.VERSION_MADE_BY Field**

**NOTE:** This field is now obsolete.

Use VersionMadeBy instead

The version made by field for entries in the central header when created by this library

```csharp
public const int VERSION_MADE_BY = 45;
```

**Remarks**

This is also the Zip version for the library when comparing against the version required to extract for an entry. See `ZipInputStream.CanDecompressEntry`.

**See Also**

ICSharpCode SharpZipLib Class Library
ZipConstants.VERSION_STRONG_ENCRYPTION Field

NOTE: This field is now obsolete.
Use VersionStrongEncryption instead

The minimum version required to support strong encryption

```csharp
public const int VERSION_STRONG_ENCRYPTION = 50;
```

See Also

[ZipConstants Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
ZipConstants.VersionMadeBy Field

The version made by field for entries in the central header when created by this library

```csharp
public const int VersionMadeBy = 45;
```

Remarks

This is also the Zip version for the library when comparing against the version required to extract for an entry. See `CanDecompressEntry`.

See Also

- ZipConstants Class
ICSharpCode SharpZipLib Class Library
ZipConstants.VersionStrongEncryption Field

The minimum version required to support strong encryption

```csharp
public const int VersionStrongEncryption = 50;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.VersionZip64 Field

The version required for Zip64 extensions

public const int VersionZip64 = 45;

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.Zip64CentralDirLocatorSignature Field

Signature for Zip64 central directory locator

```csharp
public const int Zip64CentralDirLocatorSignature = 117853008;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.Zip64CentralFileHeaderSignature Field

Signature for Zip64 central file header

```csharp
public const int Zip64CentralFileHeaderSignature = 101075792;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.Zip64DataDescriptorSize Field

Size of Zip64 data descriptor

```csharp
public const int Zip64DataDescriptorSize = 24;
```

See Also

ICSharpCode SharpZipLib Class Library
The properties of the **ZipConstants** class are listed below. For a complete list of **ZipConstants** class members, see the [ZipConstants Members](#) topic.

### Public Static Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DefaultCodePage</strong></td>
<td>Default encoding used for string conversion. 0 gives the default system Ansi code page. Don't use unicode encodings if you want to be Zip compatible! Using the default code page isn't the full solution necessarily there are many variable factors, codepage 850 is often a good choice for European users, however be careful about compatability.</td>
</tr>
</tbody>
</table>

### See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.DefaultCodePage Property

Default encoding used for string conversion. 0 gives the default system Ansi code page. Don't use unicode encodings if you want to be Zip compatible! Using the default code page isn't the full solution necessarily there are many variable factors, codepage 850 is often a good choice for European users, however be careful about compatibility.

```csharp
public static int DefaultCodePage {get; set;}
```

See Also

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
### ZipConstants Methods

The methods of the **ZipConstants** class are listed below. For a complete list of **ZipConstants** class members, see the **ZipConstants Members** topic.

#### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✯ <strong>ConvertToArray</strong></td>
<td>Overloaded. Convert a string to a byte array</td>
</tr>
<tr>
<td>✯ <strong>ConvertToString</strong></td>
<td>Overloaded. Convert a portion of a byte array to a string.</td>
</tr>
<tr>
<td>✯ <strong>ConvertToStringExt</strong></td>
<td>Overloaded. Convert a byte array to string</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✯ <strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>✯ <strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>✯ <strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>✯ <strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

[ZipConstants Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
ZipConstants.ConvertToArray Method

Convert a string to a byte array

Overload List

Convert a string to a byte array

\[
\text{public static byte[] ConvertToArray(int,string);}\
\]

Convert a string to a byte array

\[
\text{public static byte[] ConvertToArray(string);}\
\]

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.ConvertToArray Method (Int32, String)

Convert a string to a byte array

```csharp
public static byte[] ConvertToArray(
    int flags,
    string str
);
```

Parameters

- *flags*
  The applicable general purpose bits flags

- *str*
  String to convert to an array

Return Value

Converted array

See Also

- ZipConstants Class
- ZipConstants.ConvertToArray Overload List
ICSharpCode SharpZipLib Class Library
ZipConstants.ConvertToArray Method (String)

Convert a string to a byte array

```csharp
public static byte[] ConvertToArray(
    string str
);
```

Parameters

- `str`  
  String to convert to an array

Return Value

Converted array

See Also

ICSharpCode SharpZipLib Class Library
ZipConstants.ConvertToString Method

Convert a byte array to string

Overload List

Convert a byte array to string

    public static string ConvertToString(byte[]);

Convert a portion of a byte array to a string.

    public static string ConvertToString(byte[], int);

See Also

Convert a byte array to string

```csharp
public static string ConvertToString(byte[] data);
```

**Parameters**

`data`  
Byte array to convert

**Return Value**

`data` converted to a string

**See Also**

ICSharpCode SharpZipLib Class Library
ZipConstants.ConvertToString Method (Byte[], Int32)

Convert a portion of a byte array to a string.

```csharp
public static string ConvertToString(byte[] data,
                                      int count);
```

**Parameters**

- `data`  
  Data to convert to string

- `count`  
  Number of bytes to convert starting from index 0

**Return Value**

- `data[0]..data[length - 1]` converted to a string

**See Also**

ZipConstants.ConvertToStringExt Method

Convert a byte array to string

**Overload List**

Convert a byte array to string

```csharp
public static string ConvertToStringExt(int, byte[]);
```

Convert a byte array to string

```csharp
public static string ConvertToStringExt(int, byte[], int);
```

**See Also**

ICSharpCode SharpZipLib Class Library
ZipConstants.ConvertToStringExt Method (Int32, Byte[])  

Convert a byte array to string

```csharp
public static string ConvertToStringExt(int flags, byte[] data);
```

**Parameters**

- `flags`  
  The applicable general purpose bits flags

- `data`  
  Byte array to convert

**Return Value**

`data` converted to a string

**See Also**

ZipConstants.ConvertToStringExt Method (Int32, Byte[], Int32)

Convert a byte array to string

```csharp
public static string ConvertToStringExt(
    int flags,
    byte[] data,
    int count
);
```

Parameters

- **flags**
  The applicable general purpose bits flags
- **data**
  Byte array to convert
- **count**
  The number of bytes to convert.

Return Value

- **data** converted to a string

See Also

- [ZipConstants Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [ZipConstants.ConvertToStringExt Overload List](#)
ICSharpCode SharpZipLib Class Library
ZipEntry Class

This class represents an entry in a zip archive. This can be a file or a directory. ZipFile and ZipInputStream will give you instances of this class as information about the members in an archive. ZipOutputStream uses an instance of this class when creating an entry in a Zip file.

Author of the original java version: Jochen Hoenicke

For a list of all members of this type, see ZipEntry Members.

System.Object


```csharp
public class ZipEntry : ICloneable
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

ZipEntry Members | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
### ZipEntry Members

#### ZipEntry overview

Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CleanName</strong></td>
<td>Cleans a name making it conform to Zip file conventions. Devices names ('c:') and UNC share names ('\server\share') are removed and forward slashes ('') are converted to back slashes ('/'). Names are made relative by trimming leading slashes which is compatible with the ZIP naming convention.</td>
</tr>
<tr>
<td><strong>IsCompressionMethodSupported</strong></td>
<td>Overloaded. Test a <a href="#">compression method</a> to see if this library supports extracting data compressed with that method</td>
</tr>
</tbody>
</table>

Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ZipEntry</strong></td>
<td>Overloaded. Initializes a new instance of the ZipEntry class.</td>
</tr>
</tbody>
</table>

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CanDecompress</strong></td>
<td>Get a value indicating wether this entry can be decompressed by the library.</td>
</tr>
<tr>
<td><strong>CentralHeaderRequiresZip64</strong></td>
<td>Get a value indicating wether the central directory entry requires Zip64 extensions to be stored.</td>
</tr>
<tr>
<td><strong>Comment</strong></td>
<td>Gets/Sets the entry comment.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CompressedSize</td>
<td>Gets/Sets the size of the compressed data.</td>
</tr>
<tr>
<td>CompressionMethod</td>
<td>Gets/Sets the compression method. Only Deflated and Stored are supported.</td>
</tr>
<tr>
<td>Crc</td>
<td>Gets/Sets the crc of the uncompressed data.</td>
</tr>
<tr>
<td>DateTime</td>
<td>Gets/Sets the time of last modification of the entry.</td>
</tr>
<tr>
<td>DosTime</td>
<td>Get/Set DosTime</td>
</tr>
<tr>
<td>ExternalFileAttributes</td>
<td>Get/Set external file attributes as an integer. The values of this are operating system dependant see HostSystem for details</td>
</tr>
<tr>
<td>ExtraData</td>
<td>Gets/Sets the extra data.</td>
</tr>
<tr>
<td>Flags</td>
<td>Get/Set general purpose bit flag for entry</td>
</tr>
<tr>
<td>HasCrc</td>
<td>Get a value indicating wether the entry has a CRC value available.</td>
</tr>
<tr>
<td>HostSystem</td>
<td>Gets the compatibility information for the external file attribute</td>
</tr>
<tr>
<td></td>
<td>If the external file attributes are compatible with MS-DOS and can be read by PKZIP for DOS version 2.04g then this value will be zero. Otherwise the value will be non-zero and identify the host system on which the attributes are compatible.</td>
</tr>
<tr>
<td>IsCrypted</td>
<td>Get/Set flag indicating if entry is encrypted. A simple helper routine to aid interpretation of</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Flags</strong></td>
<td></td>
</tr>
<tr>
<td>IsDirectory</td>
<td>Gets a value indicating if the entry is a directory. However.</td>
</tr>
<tr>
<td>IsFile</td>
<td>Get a value of true if the entry appears to be a file; false otherwise</td>
</tr>
<tr>
<td>IsUnicodeText</td>
<td>Get / set a flag indicating whether entry name and comment text are encoded in Unicode UTF8</td>
</tr>
<tr>
<td>LocalHeaderRequiresZip64</td>
<td>Gets a value indicating if the entry requires Zip64 extensions to store the full entry values.</td>
</tr>
<tr>
<td>Name</td>
<td>Returns the entry name. The path components in the entry should always be separated by slashes (<code>/</code>). Dos device names like C: should also be removed. See the <code>ZipNameTransform</code> class, or <code>CleanName</code></td>
</tr>
<tr>
<td>Offset</td>
<td>Get / set offset for use in central header</td>
</tr>
<tr>
<td>Size</td>
<td>Gets / Sets the size of the uncompressed data.</td>
</tr>
<tr>
<td>Version</td>
<td>Get minimum Zip feature version required to extract this entry</td>
</tr>
<tr>
<td>VersionMadeBy</td>
<td>Get the version made by for this entry or zero if unknown. The value / 10 indicates the major version number, and the value mod 10 is the minor version number</td>
</tr>
<tr>
<td>ZipFileIndex</td>
<td>Get / Set index of this entry in Zip file</td>
</tr>
</tbody>
</table>
### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clone</strong></td>
<td>Creates a copy of this zip entry.</td>
</tr>
<tr>
<td><strong>Equals</strong> <em>(inherited from Object)</em></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>ForceZip64</strong></td>
<td>Force this entry to be recorded using Zip64 extensions.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> <em>(inherited from Object)</em></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> <em>(inherited from Object)</em></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>IsCompressionMethodSupported</strong></td>
<td>Overloaded. Test entry to see if data can be extracted.</td>
</tr>
<tr>
<td><strong>IsZip64Forced</strong></td>
<td>Get a value indicating whether Zip64 extensions were forced.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Gets the string representation of this ZipEntry.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> <em>(inherited from Object)</em></td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> <em>(inherited from Object)</em></td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### See Also

- **ZipEntry Class**
- **ICSharpCode.SharpZipLib.Zip Namespace**
ICSharpCode SharpZipLib Class Library
ZipEntry Constructor

Creates a zip entry with the given name.

Overload List

**Obsolete.** Creates a deep copy of the given zip entry.

```csharp
public ZipEntry(ZipEntry);
```

Creates a zip entry with the given name.

```csharp
public ZipEntry(string);
```

See Also

ICSharpCode SharpZipLib Class Library
ZipEntry Constructor (String)

Creates a zip entry with the given name.

```java
public ZipEntry(
    string name
);
```

Parameters

`name`
The name for this entry. Can include directory components. The convention for names is 'unix' style paths with relative names only. There are with no device names and path elements are separated by '/' characters.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>The name passed is null</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
NOTE: This constructor is now obsolete.
Use Clone instead

Creates a deep copy of the given zip entry.

```
public ZipEntry(
    ZipEntry entry
);
```

Parameters

- `entry`
  The entry to copy.

See Also

**ZipEntry Properties**

The properties of the **ZipEntry** class are listed below. For a complete list of **ZipEntry** class members, see the [ZipEntry Members](#) topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanDecompress</td>
<td>Get a value indicating whether this entry can be decompressed by the library.</td>
</tr>
<tr>
<td>CentralHeaderRequiresZip64</td>
<td>Get a value indicating whether the central directory entry requires Zip64 extensions to be stored.</td>
</tr>
<tr>
<td>Comment</td>
<td>Gets/Sets the entry comment.</td>
</tr>
<tr>
<td>CompressedSize</td>
<td>Gets/Sets the size of the compressed data.</td>
</tr>
<tr>
<td>CompressionMethod</td>
<td>Gets/Sets the compression method. Only Deflated and Stored are supported.</td>
</tr>
<tr>
<td>Crc</td>
<td>Gets/Sets the crc of the uncompressed data.</td>
</tr>
<tr>
<td>DateTime</td>
<td>Gets/Sets the time of last modification of the entry.</td>
</tr>
<tr>
<td>DosTime</td>
<td>Get/Set DosTime</td>
</tr>
<tr>
<td>ExternalFileAttributes</td>
<td>Get/Set external file attributes as an integer. The values of this are operating system dependant see <a href="#">HostSystem</a> for details</td>
</tr>
<tr>
<td>ExtraData</td>
<td>Gets/Sets the extra data.</td>
</tr>
<tr>
<td>Flags</td>
<td>Get/Set general purpose bit flag for entry</td>
</tr>
<tr>
<td>HasCrc</td>
<td>Get a value indicating whether</td>
</tr>
<tr>
<td><strong>Entry</strong></td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>the entry has a CRC value available.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>HostSystem</strong></td>
<td>Gets the compatibility information for the external file attribute. If the external file attributes are compatible with MS-DOS and can be read by PKZIP for DOS version 2.04g then this value will be zero. Otherwise the value will be non-zero and identify the host system on which the attributes are compatible.</td>
</tr>
<tr>
<td><strong>IsCrypted</strong></td>
<td>Get/Set flag indicating if entry is encrypted. A simple helper routine to aid interpretation of flags.</td>
</tr>
<tr>
<td><strong>IsDirectory</strong></td>
<td>Gets a value indicating if the entry is a directory. However.</td>
</tr>
<tr>
<td><strong>IsFile</strong></td>
<td>Get a value of true if the entry appears to be a file; false otherwise.</td>
</tr>
<tr>
<td><strong>IsUnicodeText</strong></td>
<td>Get / set a flag indicating wether entry name and comment text are encoded in Unicode UTF8.</td>
</tr>
<tr>
<td><strong>LocalHeaderRequiresZip64</strong></td>
<td>Gets a value indicating if the entry requires Zip64 extensions to store the full entry values.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Returns the entry name. The path components in the entry should always separated by slashes (<code>'/</code>). Dos device names like C: should also be removed. See the ZipNameTransform class, or CleanName.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Offset</td>
<td>Get/set offset for use in central header</td>
</tr>
<tr>
<td>Size</td>
<td>Gets/Sets the size of the uncompressed data.</td>
</tr>
<tr>
<td>Version</td>
<td>Get minimum Zip feature version required to extract this entry</td>
</tr>
<tr>
<td>VersionMadeBy</td>
<td>Get the version made by for this entry or zero if unknown. The value / 10 indicates the major version number, and the value mod 10 is the minor version number</td>
</tr>
<tr>
<td>ZipFileIndex</td>
<td>Get/Set index of this entry in Zip file</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipEntry.CanDecompress Property

Get a value indicating whether this entry can be decompressed by the library.

```csharp
public bool CanDecompress {get;}
```

See Also

ZipEntry.CentralHeaderRequiresZip64 Property

Get a value indicating whether the central directory entry requires Zip64 extensions to be stored.

```csharp
public bool CentralHeaderRequiresZip64 {get;]
```

See Also

- ZipEntry Class
ICSharpCode SharpZipLib Class Library
ZipEntry.Comment Property

Gets/Sets the entry comment.

```csharp
public string Comment {get; set;}
```

Remarks

A comment is only available for entries when read via the `ZipFile` class. The `ZipInputStream` class doesn't have the comment data available.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentOutOfRangeException</code></td>
<td>If comment is longer than 0xffff.</td>
</tr>
</tbody>
</table>

See Also

`ZipEntry Class` | `ICSharpCode.SharpZipLib.Zip Namespace`
ICSharpCode SharpZipLib Class Library
ZipEntry.CompressedSize Property

Gets/Sets the size of the compressed data.

```csharp
public long CompressedSize {get; set;}
```

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipEntry.CompressionMethod Property

Gets/Sets the compression method. Only Deflated and Stored are supported.

```csharp
public CompressionMethod CompressionMethod {get; set;}
```

See Also

- ZipEntry Class
ICSharpCode SharpZipLib Class Library
ZipEntry.Crc Property

Gets/Sets the crc of the uncompressed data.

```csharp
public long Crc {get; set;}
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentOutOfRangeException</code></td>
<td>Crc is not in the range 0..0xffffffffL</td>
</tr>
</tbody>
</table>

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
ZipEntry.DateTime Property

Gets/Sets the time of last modification of the entry.

```csharp
public System.DateTime DateTime {get; set;
```

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
ZipEntry.DosTime Property

Get/Set DosTime

```csharp
public long DosTime {get; set;}
```

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipEntry.ExternalFileAttributes Property

Get/Set external file attributes as an integer. The values of this are operating system dependant see HostSystem for details.

```csharp
public int ExternalFileAttributes {get; set;}
```

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
**ICSharpCode SharpZipLib Class Library**
ZipEntry.ExtraData Property

Gets/Sets the extra data.

```csharp
public byte[] ExtraData {get; set;}
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>Extra data is longer than 64KB (0xffff) bytes.</td>
</tr>
</tbody>
</table>

See Also

ZipEntry.Flags Property

Get/Set general purpose bit flag for entry

```csharp
public int Flags {get; set;}
```

Remarks

General purpose bit flag
Bit 0: If set, indicates the file is encrypted
Bit 1-2 Only used for compression type 6 Imploding, and 8, 9
deflating
Imploding:
Bit 1 if set indicates an 8K sliding dictionary was used. If clear a 4k
dictionary was used
Bit 2 if set indicates 3 Shannon-Fanno trees were used to encode
the sliding dictionary, 2 otherwise

Deflating:
Bit 2 Bit 1
0 0 Normal compression was used
0 1 Maximum compression was used
1 0 Fast compression was used
1 1 Super fast compression was used

Bit 3: If set, the fields crc-32, compressed size and uncompressed
size are were not able to be written during zip file creation The
correct values are held in a data descriptor immediately following the
compressed data.
Bit 4: Reserved for use by PKZIP for enhanced deflating
Bit 5: If set indicates the file contains compressed patch data
Bit 6: If set indicates strong encryption was used.
Bit 7-15: Unused or reserved

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipEntry.HasCrc Property

Get a value indicating whether the entry has a CRC value available.

```csharp
public bool HasCrc {get;}
```

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
Gets the compatability information for the external file attribute. If the external file attributes are compatible with MS-DOS and can be read by PKZIP for DOS version 2.04g then this value will be zero. Otherwise the value will be non-zero and identify the host system on which the attributes are compatible.

```csharp
public int HostSystem {get; set;}
```

Remarks

The values for this as defined in the Zip File format and by others are shown below. The values are somewhat misleading in some cases as they are not all used as shown. You should consult the relevant documentation to obtain up to date and correct information. The modified appnote by the infozip group is particularly helpful as it documents a lot of peculiarities. The document is however a little dated.

- 0 - MS-DOS and OS/2 (FAT / VFAT / FAT32 file systems)
- 1 - Amiga
- 2 - OpenVMS
- 3 - Unix
- 4 - VM/CMS
- 5 - Atari ST
- 6 - OS/2 HPFS
- 7 - Macintosh
- 8 - Z-System
- 9 - CP/M
- 10 - Windows NTFS
- 11 - MVS (OS/390 - Z/OS)
- 12 - VSE
- 13 - Acorn Risc
- 14 - VFAT
- 15 - Alternate
- 16 - BeOS
- 17 - Tandem
- 18 - OS/400
- 19 - OS/X (Darwin)
- 99 - WinZip AES
- remainder - unused

See Also

- ZipEntry Class
ICSharpCode SharpZipLib Class Library
ZipEntry.IsCrypted Property

Get/Set flag indicating if entry is encrypted. A simple helper routine to aid interpretation of flags.

```csharp
public bool IsCrypted {get; set;}
```

See Also

- [ZipEntry Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
## ZipEntry.IsDirectory Property

Gets a value indicating if the entry is a directory. However.

```csharp
public bool IsDirectory {get;}
```

**Remarks**

A directory is determined by an entry name with a trailing slash '/'.
The external file attributes can also indicate an entry is for a directory. Currently only dos/windows attributes are tested in this manner. The trailing slash convention should always be followed.

**See Also**

ICSharpCode SharpZipLib Class Library
ZipEntry.IsFile Property

Get a value of true if the entry appears to be a file; false otherwise

```csharp
public bool IsFile {get;}
```

Remarks

This only takes account of DOS/Windows attributes. Other operating systems are ignored. For Linux and others the result may be incorrect.

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipEntry.IsUnicodeText Property

Get / set a flag indicating whether entry name and comment text are encoded in Unicode UTF8

```csharp
public bool IsUnicodeText {get; set;}
```

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipEntry.LocalHeaderRequiresZip64 Property

Gets a value indicating if the entry requires Zip64 extensions to store the full entry values.

```csharp
public bool LocalHeaderRequiresZip64 {get;}
```

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipEntry.Name Property

Returns the entry name. The path components in the entry should always separated by slashes ('/'). Dos device names like C: should also be removed. See the `ZipNameTransform` class, or `CleanName`.

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

See Also

[ZipEntry Class]  |  [ICSharpCode.SharpZipLib.Zip Namespace]
**ZipEntry.Offset Property**

Get/set offset for use in central header

```csharp
public long Offset {get; set;}
```

See Also

[ZipEntry Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
**ZipEntry.Size Property**

Gets/Sets the size of the uncompressed data.

```
public long Size {get; set;}
```

See Also

[ZipEntry Class] [ICSharpCode.SharpZipLib.Zip Namespace]
### ZipEntry.Version Property

Get minimum Zip feature version required to extract this entry

```csharp
public int Version {get;}
```

### Remarks

Minimum features are defined as:

1.0 - Default value
1.1 - File is a volume label
2.0 - File is a folder/directory
2.0 - File is compressed using Deflate compression
2.0 - File is encrypted using traditional encryption
2.1 - File is compressed using Deflate64
2.5 - File is compressed using PKWARE DCL Implode
2.7 - File is a patch data set
4.5 - File uses Zip64 format extensions
4.6 - File is compressed using BZIP2 compression
5.0 - File is encrypted using DES
5.0 - File is encrypted using 3DES
5.0 - File is encrypted using original RC2 encryption
5.0 - File is encrypted using RC4 encryption
5.1 - File is encrypted using AES encryption
5.1 - File is encrypted using corrected RC2 encryption
5.1 - File is encrypted using corrected RC2-64 encryption
6.1 - File is encrypted using non-OAEP key wrapping
6.2 - Central directory encryption (not confirmed yet)
6.3 - File is compressed using LZMA
6.3 - File is compressed using PPMD+
6.3 - File is encrypted using Blowfish
6.3 - File is encrypted using Twofish

### See Also

ZipEntry.VersionMadeBy Property

Get the version made by for this entry or zero if unknown. The value / 10 indicates the major version number, and the value mod 10 is the minor version number.

```csharp
public int VersionMadeBy {get;}
```

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipEntry.ZipFileIndex Property

Get/Set index of this entry in Zip file

```csharp
public long ZipFileIndex {get; set;}
```

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
# ZipEntry Methods

The methods of the **ZipEntry** class are listed below. For a complete list of **ZipEntry** class members, see the [ZipEntry Members](#) topic.

## Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CleanName</strong></td>
<td>Cleans a name making it conform to Zip file conventions. Devices names ('c:') and UNC share names ('\server\share') are removed and forward slashes ('') are converted to back slashes ('/'). Names are made relative by trimming leading slashes which is compatible with the ZIP naming convention.</td>
</tr>
<tr>
<td><strong>IsCompressionMethodSupported</strong></td>
<td>Overloaded. Test a compression method to see if this library supports extracting data compressed with that method</td>
</tr>
</tbody>
</table>

## Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clone</strong></td>
<td>Creates a copy of this zip entry.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>ForceZip64</strong></td>
<td>Force this entry to be recorded using Zip64 extensions.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current object</td>
</tr>
</tbody>
</table>

[ZipEntry Members](#)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsCompressionMethodSupported</code></td>
<td>Overloaded. Test entry to see if data can be extracted.</td>
</tr>
<tr>
<td><code>IsZip64Forced</code></td>
<td>Get a value indicating whether Zip64 extensions were forced.</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Gets the string representation of this ZipEntry.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Finalize</code> (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code> (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
ZipEntry.CleanName Method

Cleans a name making it conform to Zip file conventions. Devices names ('c:') and UNC share names ('\server\share') are removed and forward slashes ('\') are converted to back slashes ('/'). Names are made relative by trimming leading slashes which is compatible with the ZIP naming convention.

```csharp
public static string CleanName(string name);
```

Parameters

- **name**
  - Name to clean

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipEntry.Clone Method

Creates a copy of this zip entry.

```csharp
public object Clone();
```

Implements

ICloneable.Clone

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
ZipEntry.ForceZip64 Method

Force this entry to be recorded using Zip64 extensions.

```csharp
public void ForceZip64();
```

See Also

ICSharpCode SharpZipLib Class Library
ZipEntry.IsCompressionMethodSupported Method

Test entry to see if data can be extracted.

Overload List

Test entry to see if data can be extracted.

```csharp
public bool IsCompressionMethodSupported();
```

Test a compression method to see if this library supports extracting data compressed with that method

```csharp
public static bool IsCompressionMethodSupported(CompressionMethod);
```

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
ZipEntry.IsCompressionMethodSupported Method ()

Test entry to see if data can be extracted.

```csharp
public bool IsCompressionMethodSupported();
```

Return Value

Returns true if data can be extracted for this entry; false otherwise.

See Also

- ZipEntry Class
- ZipEntry.IsCompressionMethodSupported Overload List
ICSharpCode SharpZipLib Class Library
Test a compression method to see if this library supports extracting data compressed with that method

```csharp
public static bool IsCompressionMethodSupported(CompressionMethod method);
```

**Parameters**

*method*

The compression method to test.

**Return Value**

Returns true if the compression method is supported; false otherwise

**See Also**

ZipEntry.IsZip64Forced Method

Get a value indicating whether Zip64 extensions were forced.

```csharp
public bool IsZip64Forced();
```

Return Value

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipEntry.ToString Method

Gets the string representation of this ZipEntry.

```csharp
public override string ToString();
```

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipException Class

Represents exception conditions specific to Zip archive handling.

For a list of all members of this type, see ZipException Members.

- System.Object
- System.Exception
- System.ApplicationException

```csharp
public class ZipException :
    SharpZipBaseException
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements

- **Namespace**: ICSharpCode.SharpZipLib.Zip

See Also

- ZipException Members
ICSharpCode SharpZipLib Class Library
### ZipException Members

#### ZipException overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipException</td>
<td>Overloaded. Initializes a new instance of the ZipException class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HelpLink</td>
<td>Gets or sets a link to the help file associated with this exception.</td>
</tr>
<tr>
<td>InnerException</td>
<td>Gets the Exception instance that caused the current exception.</td>
</tr>
<tr>
<td>Message</td>
<td>Gets a message that describes the current exception.</td>
</tr>
<tr>
<td>Source</td>
<td>Gets or sets the name of the application or the object that causes the error.</td>
</tr>
<tr>
<td>StackTrace</td>
<td>Gets a string representation of the frames on the call stack at the time the current exception was thrown.</td>
</tr>
<tr>
<td>TargetSite</td>
<td>Gets the method that throws the current exception.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetBaseException</td>
<td>When overridden in a derived class, returns the Exception that</td>
</tr>
</tbody>
</table>
is the root cause of one or more subsequent exceptions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetObjectData</strong> (inherited from Exception)</td>
<td>When overridden in a derived class, sets the SerializationInfo with information about the exception.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Exception)</td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ZipException</strong></td>
<td>Overloaded. Initializes a new instance of the ZipException class.</td>
</tr>
</tbody>
</table>

### Protected Instance Properties

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HResult</strong> (inherited from Exception)</td>
<td>Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>
See Also

ZipException Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipException Constructor

Deserialization constructor

Overload List

Initializes a new instance of the ZipException class.

- `public ZipException();` Deserialization constructor

  `protected ZipException(SerializationInfo, StreamingContext);`

  Initializes a new instance of the ZipException class with a specified error message.

  `public ZipException(string);`

  Initialise a new instance of ZipException.

  `public ZipException(string, Exception);`

See Also

  ZipException Class | ICSsharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipException Constructor (SerializationInfo, StreamingContext)

Deserialization constructor

```csharp
protected ZipException(
    SerializationInfo info,
    StreamingContext context
);
```

Parameters

- `info`  
  `SerializationInfo` for this constructor

- `context`  
  `StreamingContext` for this constructor

See Also

- [ZipException Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [ZipException Constructor Overload List](#)
ZipException Constructor ()

Initializes a new instance of the ZipException class.

```java
public ZipException();
```

See Also

- [ZipException Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [ZipException Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
### ZipException Constructor (String)

Initializes a new instance of the ZipException class with a specified error message.

```java
public ZipException(
    string message
);
```

#### Parameters

- **message**
  
  The error message that explains the reason for the exception.

#### See Also

- [ZipException Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [ZipException Constructor Overload List](#)
ZipException Constructor (String, Exception)

Initialise a new instance of ZipException.

```java
public ZipException(
    string message,
    Exception exception
);
```

Parameters

- `message`  
  A message describing the error.

- `exception`  
  The exception that is the cause of the current exception.

See Also

# ZipExtraData Class

A class to handle the extra data field for Zip entries

For a list of all members of this type, see [ZipExtraData Members](#).

```csharp
```

```csharp
public sealed class ZipExtraData : IDisposable
```

## Thread Safety

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

## Remarks

Extra data contains 0 or more values each prefixed by a header tag and length. They contain zero or more bytes of actual data. The data is held internally using a copy on write strategy. This is more efficient but means that for extra data created by passing in data can have the values modified by the caller in some circumstances.

## Requirements

**Namespace:** [ICSharpCode.SharpZipLib.Zip](#)

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

## See Also

ICSharpCode SharpZipLib Class Library
## ZipExtraData Members

### ZipExtraData overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipExtraData</td>
<td>Overloaded. Initializes a new instance of the ZipExtraData class.</td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentReadIndex</td>
<td>Get the index for the current read value.</td>
</tr>
<tr>
<td>Length</td>
<td>Gets the current extra data length.</td>
</tr>
<tr>
<td>UnreadCount</td>
<td>Get the number of bytes remaining to be read for the current value;</td>
</tr>
<tr>
<td>ValueLength</td>
<td>Get the length of the last value found by Find</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddData</td>
<td>Overloaded. Add a byte of data to the pending new entry.</td>
</tr>
<tr>
<td>AddEntry</td>
<td>Add a new entry to extra data</td>
</tr>
<tr>
<td>AddLeInt</td>
<td>Add an integer value in little endian order to the pending new entry.</td>
</tr>
<tr>
<td>AddLeLong</td>
<td>Add a long value in little endian order to the pending new entry.</td>
</tr>
<tr>
<td>AddLeShort</td>
<td>Add a short value in little endian order to the pending new entry.</td>
</tr>
<tr>
<td>AddNewEntry</td>
<td>Add entry data added since StartNewEntry using the ID</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><img src="clear_image.png" alt="Clear" /></td>
<td>Clear the stored data.</td>
</tr>
<tr>
<td><img src="delete_image.png" alt="Delete" /></td>
<td>Delete an extra data field.</td>
</tr>
<tr>
<td><img src="dispose_image.png" alt="Dispose" /></td>
<td>Dispose of this instance.</td>
</tr>
<tr>
<td><img src="equals_image.png" alt="Equals" /> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><img src="find_image.png" alt="Find" /></td>
<td>Find an extra data value</td>
</tr>
<tr>
<td><img src="getentrydata_image.png" alt="GetEntryData" /></td>
<td>Get the raw extra data value</td>
</tr>
<tr>
<td><img src="gethashcode_image.png" alt="GetHashCode" /> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><img src="getstreamfortag_image.png" alt="GetStreamForTag" /></td>
<td>Get a read-only <strong>Stream</strong> for the associated tag.</td>
</tr>
<tr>
<td><img src="gettype_image.png" alt="GetType" /> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><img src="readbyte_image.png" alt="ReadByte" /></td>
<td>Read a byte from an extra data</td>
</tr>
<tr>
<td><img src="readint_image.png" alt="ReadInt" /></td>
<td>Read an integer in little endian form from the last <strong>found</strong> data value.</td>
</tr>
<tr>
<td><img src="readlong_image.png" alt="ReadLong" /></td>
<td>Read a long in little endian form from the last <strong>found</strong> data value.</td>
</tr>
<tr>
<td><img src="readshort_image.png" alt="ReadShort" /></td>
<td>Read a short value in little endian form from the last <strong>found</strong> data value.</td>
</tr>
<tr>
<td><img src="skip_image.png" alt="Skip" /></td>
<td>Skip data during reading.</td>
</tr>
<tr>
<td><img src="startnewentry_image.png" alt="StartNewEntry" /></td>
<td>Start adding a new entry</td>
</tr>
<tr>
<td><img src="tostring_image.png" alt="ToString" /> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>
See Also

**ZipExtraData Constructor**

Initialise a default instance.

**Overload List**

Initialise a default instance.

public `ZipExtraData()`;

Initialise with known extra data.

public `ZipExtraData(byte[])`;

**See Also**

ICSharpCode SharpZipLib Class Library
ZipExtraData Constructor ()

Initialise a default instance.

```csharp
public ZipExtraData();
```

See Also

ICSharpCode SharpZipLib Class Library
ZipExtraData Constructor (Byte[])

Initialise with known extra data.

```csharp
public ZipExtraData(
    byte[] data
);
```

Parameters

- `data`  
  The extra data.

See Also

- [ZipExtraData Class](#)  
- [ICSharpCode.SharpZipLib.Zip Namespace](#)  
- [ZipExtraData Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
ZipExtraData Properties

The properties of the **ZipExtraData** class are listed below. For a complete list of **ZipExtraData** class members, see the **ZipExtraData Members** topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentReadIndex</td>
<td>Get the index for the current read value.</td>
</tr>
<tr>
<td>Length</td>
<td>Gets the current extra data length.</td>
</tr>
<tr>
<td>UnreadCount</td>
<td>Get the number of bytes remaining to be read for the current value;</td>
</tr>
<tr>
<td>ValueLength</td>
<td>Get the length of the last value found by Find.</td>
</tr>
</tbody>
</table>

See Also

[ZipExtraData Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ZipExtraData.CurrentReadIndex Property

Get the index for the current read value.

```
public int CurrentReadIndex {get;}
```

Remarks

This is only valid if `Find` has previously returned true. Initially it will be the index of the first byte of actual data. Its is updated after calls to `ReadInt`, `ReadShort` and `ReadLong`.

See Also

ICSharpCode SharpZipLib Class Library
**ZipExtraData.Length Property**

Gets the current extra data length.

```csharp
public int Length {get;}
```

See Also

[ZipExtraData Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
**ZipExtraData.UnreadCount Property**

Get the number of bytes remaining to be read for the current value;

```csharp
public int UnreadCount {get;}
```

**See Also**

ICSharpCode SharpZipLib Class Library
ZipExtraData.ValueLength Property

Get the length of the last value found by Find

```csharp
public int ValueLength {get;}
```

Remarks

This is only value if Find has previously returned true.

See Also

ICSharpCode SharpZipLib Class Library
## ZipExtraData Methods

The methods of the **ZipExtraData** class are listed below. For a complete list of **ZipExtraData** class members, see the [ZipExtraData Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddData</strong></td>
<td>Overloaded. Add a byte of data to the pending new entry.</td>
</tr>
<tr>
<td><strong>AddEntry</strong></td>
<td>Add a new entry to extra data</td>
</tr>
<tr>
<td><strong>AddLeInt</strong></td>
<td>Add an integer value in little endian order to the pending new entry.</td>
</tr>
<tr>
<td><strong>AddLeLong</strong></td>
<td>Add a long value in little endian order to the pending new entry.</td>
</tr>
<tr>
<td><strong>AddLeShort</strong></td>
<td>Add a short value in little endian order to the pending new entry.</td>
</tr>
<tr>
<td><strong>AddNewEntry</strong></td>
<td>Add entry data added since <strong>StartNewEntry</strong> using the ID passed.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Clear the stored data.</td>
</tr>
<tr>
<td><strong>Delete</strong></td>
<td>Delete an extra data field.</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Dispose of this instance.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Find</strong></td>
<td>Find an extra data value</td>
</tr>
<tr>
<td><strong>GetEntryData</strong></td>
<td>Get the raw extra data value</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetStreamForTag</code></td>
<td>Get a read-only <code>Stream</code> for the associated tag.</td>
</tr>
<tr>
<td><code>GetType</code> (inherited from <code>Object</code>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>ReadByte</code></td>
<td>Read a byte from an extra data</td>
</tr>
<tr>
<td><code>ReadInt</code></td>
<td>Read an integer in little endian form from the last <code>found</code> data value.</td>
</tr>
<tr>
<td><code>ReadLong</code></td>
<td>Read a long in little endian form from the last <code>found</code> data value.</td>
</tr>
<tr>
<td><code>ReadShort</code></td>
<td>Read a short value in little endian form from the last <code>found</code> data value.</td>
</tr>
<tr>
<td><code>Skip</code></td>
<td>Skip data during reading.</td>
</tr>
<tr>
<td><code>StartNewEntry</code></td>
<td>Start adding a new entry.</td>
</tr>
<tr>
<td><code>ToString</code> (inherited from <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

See Also

*ZipExtraData Class* | *ICSharpCode.SharpZipLib.Zip Namespace*
ICSharpCode SharpZipLib Class Library
Add a byte of data to the pending new entry.

**Overload List**

Add a byte of data to the pending new entry.

`public void AddData(byte);`

Add data to a pending new entry.

`public void AddData(byte[]);`

See Also

ICSharpCode SharpZipLib Class Library
ZipExtraData.AddData Method (Byte)

Add a byte of data to the pending new entry.

```csharp
public void AddData(
    byte data
);
```

Parameters

`data`

The byte to add.

See Also

- ZipExtraData Class
- ZipExtraData.AddData Overload List
- StartNewEntry
ICSharpCode SharpZipLib Class Library
Add data to a pending new entry.

```csharp
public void AddData(byte[] data);
```

**Parameters**

*data*

The data to add.

**See Also**

ICSharpCode SharpZipLib Class Library
ZipExtraData.AddEntry Method

Add a new entry to extra data

```csharp
public void AddEntry(
    int headerID,
    byte[] fieldData
);
```

Parameters

- **headerID**
  The ID for this entry.

- **fieldData**
  The data to add.

Remarks

If the ID already exists its contents are replaced.

See Also

- [ZipExtraData Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
ZipExtraData.AddLeInt Method

Add an integer value in little endian order to the pending new entry.

```csharp
public void AddLeInt(int toAdd);
```

Parameters

toAdd

The data to add.

See Also

ICSharpCode SharpZipLib Class Library
ZipExtraData.AddLeLong Method

Add a long value in little endian order to the pending new entry.

```csharp
public void AddLeLong(
    long toAdd
);
```

Parameters

`toAdd`

The data to add.

See Also

Add a short value in little endian order to the pending new entry.

```csharp
public void AddLeShort(int toAdd);
```

Parameters

`toAdd`
- The data to add.

See Also

- [ZipExtraData Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [StartNewEntry](#)
ICSharpCode SharpZipLib Class Library
ZipExtraData.AddNewEntry Method

Add entry data added since StartNewEntry using the ID passed.

```csharp
public void AddNewEntry(
    int headerID
);
```

Parameters

`headerID`

The identifier to use for this entry.

See Also

ICSharpCode SharpZipLib Class Library
ZipExtraData.Clear Method

Clear the stored data.

```csharp
public void Clear();
```

See Also

ICSharpCode SharpZipLib Class Library
ZipExtraData.Delete Method

Delete an extra data field.

```csharp
public bool Delete(int headerID);
```

**Parameters**

*headerID*  
The identifier of the field to delete.

**Return Value**

Returns true if the field was found and deleted.

**See Also**

ICSharpCode SharpZipLib Class Library
ZipExtraData.Dispose Method

Dispose of this instance.

```csharp
public void Dispose();
```

Implements

- `IDisposable.Dispose`

See Also

- `ZipExtraData Class`
ZipExtraData.Find Method

Find an extra data value

```csharp
public bool Find(
    int headerID
);
```

Parameters

**headerID**

The identifier for the value to find.

Return Value

Returns true if the value was found; false otherwise.

See Also

ICSharpCode SharpZipLib Class Library
**ZipExtraData.GetEntryData Method**

Get the raw extra data value

```csharp
public byte[] GetEntryData();
```

**Return Value**

Returns the raw byte[] extra data this instance represents.

**See Also**

ICSharpCode SharpZipLib Class Library
ZipExtraData.GetStreamForTag Method

Get a read-only Stream for the associated tag.

```csharp
public Stream GetStreamForTag(
    int tag
);
```

Parameters

tag
The tag to locate data for.

Return Value

Returns a Stream containing tag data or null if no tag was found.

See Also

ICSharpCode SharpZipLib Class Library
ZipExtraData.ReadByte Method

Read a byte from an extra data

```csharp
public int ReadByte();
```

Return Value

The byte value read or -1 if the end of data has been reached.

See Also

**ZipExtraData.ReadInt Method**

Read an integer in little endian form from the last found data value.

```csharp
public int ReadInt();
```

**Return Value**

Returns the integer read.

**See Also**

[ZipExtraData Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ZipExtraData.ReadLong Method

Read a long in little endian form from the last found data value

```csharp
public long ReadLong();
```

Return Value

Returns the long value read.

See Also

ICSharpCode SharpZipLib Class Library
ZipExtraData.ReadShort Method

Read a short value in little endian form from the last found data value.

```csharp
public int ReadShort();
```

Return Value

Returns the short value read.

See Also

[ZipExtraData Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
ZipExtraData.Skip Method

Skip data during reading.

```java
public void Skip(int amount);
```

Parameters

- `amount`  
The number of bytes to skip.

See Also

ICSharpCode SharpZipLib Class Library
ZipExtraData.StartNewEntry Method

Start adding a new entry.

```csharp
public void StartNewEntry();
```

Remarks

Add data using `AddData`, `AddLeShort`, `AddLeInt`, or `AddLeLong`. The new entry is completed and actually added by calling `AddNewEntry`.

See Also

ICSharpCode SharpZipLib Class Library
ZipFile Class

This class represents a Zip archive. You can ask for the contained entries, or get an input stream for a file entry. The entry is automatically decompressed. You can also update the archive adding or deleting entries. This class is thread safe for input: You can open input streams for arbitrary entries in different threads.

Author of the original java version : Jochen Hoenicke
For a list of all members of this type, see ZipFile Members.

System.Object

public class ZipFile : IEnumerable, IDisposable

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Example

using System;
using System.Text;
using System.Collections;
using System.IO;


class MainClass
{
    static public void Main(string[] args)
    {
        using (ZipFile zFile = new ZipFile(args[0]))
            Console.WriteLine("Listing of : " + zFil
            Console.WriteLine("");
Console.WriteLine("Raw Size Size");
Console.WriteLine("-------- -------- ---");
foreach (ZipEntry e in zFile) {
    if (e.IsFile) {
        DateTime d = e.DateTime;
        Console.WriteLine("{0, -10}{1, -10}{2}		{3}			{4}",
                        e.Size, e.CompressedSize, 
                        d.ToString("dd-MM-yy"), d.ToString("HH:mm"),
                        e.Name);
    }
}

Requirements


See Also

ICSharpCode SharpZipLib Class Library
**ZipFile Members**

**ZipFile overview**

**Public Static Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Overloaded. Create a new ZipFile whose data will be stored in a file.</td>
</tr>
</tbody>
</table>

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipFile</td>
<td>Overloaded. Initializes a new instance of the ZipFile class.</td>
</tr>
</tbody>
</table>

**Public Instance Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeysRequired</td>
<td>Event handler for handling encryption keys.</td>
</tr>
</tbody>
</table>

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BufferSize</td>
<td>Get /set the buffer size to be used when updating this zip file.</td>
</tr>
<tr>
<td>Count</td>
<td>Get the number of entries contained in this ZipFile.</td>
</tr>
<tr>
<td>EntryByIndex</td>
<td>Indexer property for ZipEntries</td>
</tr>
<tr>
<td>IsEmbeddedArchive</td>
<td>Get a value indicating wether this archive is embedded in another file or not.</td>
</tr>
<tr>
<td>isNewArchive</td>
<td>Get a value indicating that this archive is a new one.</td>
</tr>
<tr>
<td>IsStreamOwner</td>
<td>Get/set a flag indicating if the underlying stream is owned by the ZipFile instance. If the flag is true then the stream will be closed when Close is called.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>IsUpdating</code></td>
<td>Get a value indicating an update has been started.</td>
</tr>
<tr>
<td><code>Name</code></td>
<td>Gets the name of this zip file.</td>
</tr>
<tr>
<td><code>NameTransform</code></td>
<td>Get / set the <code>INameTransform</code> to apply to names when updating.</td>
</tr>
<tr>
<td><code>Password</code></td>
<td>Password to be used for encrypting/decrypting files.</td>
</tr>
<tr>
<td><code>Size</code></td>
<td><strong>Obsolete.</strong> Gets the number of entries in this zip file.</td>
</tr>
<tr>
<td><code>UseZip64</code></td>
<td>Get / set a value indicating how Zip64 Extension usage is determined when adding entries.</td>
</tr>
<tr>
<td><code>ZipFileComment</code></td>
<td>Gets the comment for the zip file.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AbortUpdate</code></td>
<td>Abort updating leaving the archive unchanged.</td>
</tr>
<tr>
<td><code>Add</code></td>
<td>Overloaded. Add a new entry to the archive.</td>
</tr>
<tr>
<td><code>AddDirectory</code></td>
<td>Add a directory entry to the archive.</td>
</tr>
<tr>
<td><code>BeginUpdate</code></td>
<td>Overloaded. Begin updating this <code>ZipFile</code> archive.</td>
</tr>
<tr>
<td><code>Close</code></td>
<td>Closes the ZipFile. If the stream is <strong>owned</strong> then this also closes the underlying input stream. Once closed, no further instance methods should be called.</td>
</tr>
<tr>
<td><code>CommitUpdate</code></td>
<td>Commit current updates, updating this archive.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Delete</td>
<td>Overloaded. Delete an entry by name</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified <a href="#">Object</a> is equal to the current <a href="#">Object</a>.</td>
</tr>
<tr>
<td>FindEntry</td>
<td>Return the index of the entry with a matching name</td>
</tr>
<tr>
<td>GetEntry</td>
<td>Searches for a zip entry in this archive with the given name. String comparisons are case insensitive</td>
</tr>
<tr>
<td>GetEnumerator</td>
<td>Returns an enumerator for the Zip entries in this Zip file.</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetInputStream</td>
<td>Overloaded. Creates an input stream reading the given zip entry as uncompressed data. Normally zip entry should be an entry returned by GetEntry().</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the <a href="#">Type</a> of the current instance.</td>
</tr>
<tr>
<td>SetComment</td>
<td>Set the file comment to be recorded when the current update is committed.</td>
</tr>
<tr>
<td>TestArchive</td>
<td>Overloaded. Test an archive for integrity/validity</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a <a href="#">String</a> that represents the current <a href="#">Object</a>.</td>
</tr>
</tbody>
</table>

Protected Instance Methods
| **Dispose** | Releases the unmanaged resources used by the this instance and optionally releases the managed resources. |
| **Finalize** | Finalize this instance. |
| **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

ICSharpCode SharpZipLib Class Library
ZipFile Constructor

Opens a Zip file with the given name for reading.

Overload List

Opens a Zip file reading the given FileStream.

public ZipFile(FileStream);

Opens a Zip file reading the given Stream.

public ZipFile(Stream);

Opens a Zip file with the given name for reading.

public ZipFile(string);

See Also

ZipFile Constructor (String)

Opens a Zip file with the given name for reading.

```java
public ZipFile(
    string name
);
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>An i/o error occurs</td>
</tr>
<tr>
<td>ZipException</td>
<td>The file doesn't contain a valid zip archive.</td>
</tr>
</tbody>
</table>

See Also

**ZipFile Constructor (FileStream)**

Opens a Zip file reading the given `FileStream`.

```
public ZipFile(
    FileStream file
);
```

### Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>An i/o error occurs.</td>
</tr>
<tr>
<td>ZipException</td>
<td>The file doesn't contain a valid zip archive.</td>
</tr>
</tbody>
</table>

### See Also

ICSharpCode SharpZipLib Class Library
ZipFile Constructor (Stream)

Opens a Zip file reading the given Stream.

```java
public ZipFile(
    Stream stream
);
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>An i/o error occurs</td>
</tr>
<tr>
<td>ZipException</td>
<td>The file doesn't contain a valid zip archive. The stream provided cannot seek</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipFile Fields

The fields of the ZipFile class are listed below. For a complete list of ZipFile class members, see the ZipFile Members topic.

Public Instance Fields

| KeysRequired | Event handler for handling encryption keys. |

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.KeysRequired Field

Event handler for handling encryption keys.

```csharp
public KeysRequiredEventHandler KeysRequired;
```

See Also

ICSharpCode SharpZipLib Class Library
ZipFile Properties

The properties of the **ZipFile** class are listed below. For a complete list of **ZipFile** class members, see the [ZipFile Members](#) topic.

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BufferSize</td>
<td>Get /set the buffer size to be used when updating this zip file.</td>
</tr>
<tr>
<td>Count</td>
<td>Get the number of entries contained in this ZipFile.</td>
</tr>
<tr>
<td>EntryByIndex</td>
<td>Indexer property for ZipEntries.</td>
</tr>
<tr>
<td>IsEmbeddedArchive</td>
<td>Get a value indicating whether this archive is embedded in another file or not.</td>
</tr>
<tr>
<td>isNewArchive</td>
<td>Get a value indicating that this archive is a new one.</td>
</tr>
<tr>
<td>IsStreamOwner</td>
<td>Get /set a flag indicating if the underlying stream is owned by the ZipFile instance. If the flag is true then the stream will be closed when <strong>Close</strong> is called.</td>
</tr>
<tr>
<td>IsUpdating</td>
<td>Get a value indicating an update has been started.</td>
</tr>
<tr>
<td>Name</td>
<td>Gets the name of this zip file.</td>
</tr>
<tr>
<td>NameTransform</td>
<td>Get / set the <strong>INameTransform</strong> to apply to names when updating.</td>
</tr>
<tr>
<td>Password</td>
<td>Password to be used for encrypting/decrypting files.</td>
</tr>
<tr>
<td>Size</td>
<td><strong>Obsolete.</strong> Gets the number of entries in this zip file.</td>
</tr>
<tr>
<td>UseZip64</td>
<td>Get / set a value indicating how Zip64 Extension usage is determined when adding</td>
</tr>
</tbody>
</table>
ZipFileComment

| ZipFileComment | Gets the comment for the zip file. |

See Also

ZipFile.BufferSize Property

Get /set the buffer size to be used when updating this zip file.

```csharp
public int BufferSize {get; set;}
```

See Also

[ZipFile Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
ZipFile.Count Property

Get the number of entries contained in this ZipFile.

```csharp
public long Count {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.EntryByIndex Property

Indexer property for ZipEntries

```csharp
public ZipEntry this[int index] { get; }
```

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.IsEmbeddedArchive Property

Get a value indicating wether this archive is embedded in another file or not.

```csharp
public bool IsEmbeddedArchive {get;}
```

See Also

- ZipFile Class
ICSharpCode SharpZipLib Class Library
ZipFile.IsNewArchive Property

Get a value indicating that this archive is a new one.

```csharp
public bool IsNewArchive {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
**ZipFile.IsStreamOwner Property**

Get/set a flag indicating if the underlying stream is owned by the ZipFile instance. If the flag is true then the stream will be closed when **Close** is called.

```csharp
public bool IsStreamOwner {get; set;}
```

**Remarks**

The default value is true in all cases.

**See Also**

ICSharpCode SharpZipLib Class Library
ZipFile.IsUpdating Property

Get a value indicating an update has been started.

```csharp
public bool IsUpdating {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.Name Property

Gets the name of this zip file.

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

See Also

ZipFile.NameTransform Property

Get / set the INameTransform to apply to names when updating.

public ISharpCode.SharpZipLib.Core.INameTransform

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.Password Property

Password to be used for encrypting/decrypting files.

```csharp
public string Password {set;}
```

Remarks

Set to null if no password is required.

See Also

ICSharpCode SharpZipLib Class Library
**ZipFile.Size Property**

*NOTE: This property is now obsolete.*

*Use the Count property instead*

---

Gets the number of entries in this zip file.

```csharp
public int Size {get;}
```

### Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidOperationException</code></td>
<td>The Zip file has been closed.</td>
</tr>
</tbody>
</table>

### See Also

ICSharpCode SharpZipLib Class Library
ZipFile.UseZip64 Property

Get / set a value indicating how Zip64 Extension usage is determined when adding entries.

```csharp
public UseZip64 UseZip64 {get; set;}
```

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.ZipFileComment Property

Gets the comment for the zip file.

```csharp
public string ZipFileComment {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
### ZipFile Methods

The methods of the `ZipFile` class are listed below. For a complete list of `ZipFile` class members, see the [ZipFile Members](#) topic.

#### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Create" /></td>
<td>Overloaded. Create a new <code>ZipFile</code> whose data will be stored in a file.</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="AbortUpdate" /></td>
<td>Abort updating leaving the archive unchanged.</td>
</tr>
<tr>
<td><img src="#" alt="Add" /></td>
<td>Overloaded. Add a new entry to the archive.</td>
</tr>
<tr>
<td><img src="#" alt="AddDirectory" /></td>
<td>Add a directory entry to the archive.</td>
</tr>
<tr>
<td><img src="#" alt="BeginUpdate" /></td>
<td>Overloaded. Begin updating this <code>ZipFile</code> archive.</td>
</tr>
<tr>
<td><img src="#" alt="Close" /></td>
<td>Closes the ZipFile. If the stream is owned then this also closes the underlying input stream. Once closed, no further instance methods should be called.</td>
</tr>
<tr>
<td><img src="#" alt="CommitUpdate" /></td>
<td>Commit current updates, updating this archive.</td>
</tr>
<tr>
<td><img src="#" alt="Delete" /></td>
<td>Overloaded. Delete an entry by name</td>
</tr>
<tr>
<td><img src="#" alt="Equals" /> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><img src="#" alt="FindEntry" /></td>
<td>Return the index of the entry with a matching name</td>
</tr>
<tr>
<td><strong>GetEntry</strong></td>
<td>Searches for a zip entry in this archive with the given name. String comparisons are case insensitive.</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetEnumerator</strong></td>
<td>Returns an enumerator for the Zip entries in this Zip file.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>(inherited from Object) Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetInputStream</strong></td>
<td>Overloaded. Creates an input stream reading the given zip entry as uncompressed data. Normally zip entry should be an entry returned by GetEntry().</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>(inherited from Object) Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>SetComment</strong></td>
<td>Set the file comment to be recorded when the current update is committed.</td>
</tr>
<tr>
<td><strong>TestArchive</strong></td>
<td>Overloaded. Test an archive for integrity/validity.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>(inherited from Object) Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th><strong>Dispose</strong></th>
<th>Releases the unmanaged resources used by the this instance and optionally releases the managed resources.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Finalize this instance.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>(inherited from Object) Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>
See Also

ICSharpCode SharpZipLib Class Library
ZipFile.AbortUpdate Method

Abort updating leaving the archive unchanged.

```csharp
public void AbortUpdate();
```

See Also

- [ZipFile Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [BeginUpdate](#)
- [CommitUpdate](#)
ICSharpCode SharpZipLib Class Library
ZipFile.Add Method

Add a file entry with data.

Overload List

Add a file entry with data.

```
public void Add(IStaticDataSource,string);
```
Add a file entry with data.

```
public void Add(IStaticDataSource,string,CompressionMethod);
```
Add a file entry with data.

```
public void Add(IStaticDataSource,string,CompressionMethod,bool);
```
Add a ZipEntry that contains no data.

```
public void Add(ZipEntry);
```
Add a file to the archive.

```
public void Add(string);
```
Add a new entry to the archive.

```
public void Add(string,CompressionMethod);
```
Add a new entry to the archive.

```
public void Add(string,CompressionMethod,bool);
```

See Also

ICSharpCode SharpZipLib Class Library
Add a file entry with data.

```csharp
public void Add(IStaticDataSource dataSource, string entryName);
```

**Parameters**

*dataSource*

The source of the data for this entry.

*entryName*

The name to give to the entry.

**See Also**

ZipFile.Add Method (IStaticDataSource, String, CompressionMethod)

Add a file entry with data.

```csharp
public void Add(
    IStaticDataSource dataSource,
    string entryName,
    CompressionMethod compressionMethod
);
```

Parameters

- **dataSource**
  The source of the data for this entry.

- **entryName**
  The name to give to the entry.

- **compressionMethod**
  The compression method to use.

See Also

ICSharpCode SharpZipLib Class Library
Add a file entry with data.

```csharp
public void Add(
    IStaticDataSource dataSource,
    string entryName,
    CompressionMethod compressionMethod,
    bool useUnicodeText
);
```

**Parameters**

*dataSource*

   The source of the data for this entry.

*entryName*

   The name to give to the entry.

*compressionMethod*

   The compression method to use.

*useUnicodeText*

   Ensure Unicode text is used for name and comments for this entry.

**See Also**

ICSharpCode SharpZipLib Class Library
Add a `ZipEntry` that contains no data.

```csharp
public void Add(ZipEntry entry);
```

**Parameters**

- `entry`  
  The entry to add.

**Remarks**

This can be used to add directories, volume labels, or empty file entries.

**See Also**

ICSharpCode SharpZipLib Class Library
ZipFile.Add Method (String)

Add a file to the archive.

```csharp
public void Add(
    string fileName
);
```

Parameters

(fileName)

The name of the file to add.

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.Add Method (String, CompressionMethod)

Add a new entry to the archive.

```csharp
public void Add(
    string fileName,
    CompressionMethod compressionMethod
);
```

Parameters

- `fileName` : The name of the file to add.

- `compressionMethod` : The compression method to use.

See Also

- ZipFile Class
- ZipFile.Add Overload List
ZipFile.Add Method (String, CompressionMethod, Boolean)

Add a new entry to the archive.

```csharp
public void Add(
    string fileName,
    CompressionMethod compressionMethod,
    bool useUnicodeText
);
```

Parameters

- **fileName**
  - The name of the file to add.

- **compressionMethod**
  - The compression method to use.

- **useUnicodeText**
  - Ensure Unicode text is used for name and comment for this entry.

See Also

ZipFile.AddDirectory Method

Add a directory entry to the archive.

```csharp
public void AddDirectory(
    string directoryName
);
```

Parameters

`directoryName`  
The directory to add.

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.BeginUpdate Method

Begin updating this ZipFile archive.

Overload List

Begin updating this ZipFile archive.

public void BeginUpdate();

Begin updating to this ZipFile archive.

public void BeginUpdate(IArchiveStorage);

Begin updating this ZipFile archive.

public void BeginUpdate(IArchiveStorage, IDynamicDataSource);

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.BeginUpdate Method ()

Begin updating this ZipFile archive.

```csharp
public void BeginUpdate();
```

See Also

- ZipFile Class
- ZipFile.BeginUpdate Overload List
- CommitUpdate
- AbortUpdate
ZipFile.BeginUpdate Method (IArchiveStorage)

Begin updating to this ZipFile archive.

```csharp
public void BeginUpdate(IArchiveStorage archiveStorage);
```

Parameters

- `archiveStorage` The storage to use during the update.

See Also

ICSharpCode SharpZipLib Class Library
Begin updating this ZipFile archive.

```csharp
public void BeginUpdate(
    IArchiveStorage archiveStorage,
    IDynamicDataSource dataSource
);
```

**Parameters**

*archiveStorage*

The archive storage for use during the update.

*dataSource*

The data source to utilise during updating.

**See Also**

ICSharpCode SharpZipLib Class Library
ZipFile.Close Method

Closes the ZipFile. If the stream is owned then this also closes the underlying input stream. Once closed, no further instance methods should be called.

```csharp
public void Close();
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>An i/o error occurs.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.CommitUpdate Method

Commit current updates, updating this archive.

```csharp
public void CommitUpdate();
```

See Also

- ZipFile Class
- BeginUpdate
- AbortUpdate
ICSharpCode SharpZipLib Class Library
**ZipFile.Create Method**

Create a new `ZipFile` whose data will be stored on a stream.

**Overload List**

Create a new `ZipFile` whose data will be stored on a stream.

```csharp
public static ZipFile Create(Stream);
```

Create a new `ZipFile` whose data will be stored in a file.

```csharp
public static ZipFile Create(string);
```

**See Also**

ICSharpCode SharpZipLib Class Library
ZipFile.Create Method (Stream)

Create a new ZipFile whose data will be stored on a stream.

```csharp
public static ZipFile Create(Stream outStream);
```

Parameters

- `outStream`
  The stream providing data storage.

Return Value

Returns the newly created ZipFile

See Also

ZipFile.Create Method (String)

Create a new ZipFile whose data will be stored in a file.

```java
public static ZipFile Create(
    string fileName
);
```

Parameters

*fileName*

The name of the archive to create.

Return Value

Returns the newly created ZipFile

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.Delete Method

Delete a `ZipEntry` from the archive.

**Overload List**

Delete a `ZipEntry` from the archive.

```csharp
public void Delete(ZipEntry);
```

Delete an entry by name

```csharp
public bool Delete(string);
```

**See Also**


ICSharpCode SharpZipLib Class Library
Delete a `ZipEntry` from the archive.

```csharp
public void Delete(
    ZipEntry entry
);
```

**Parameters**

`entry`
The entry to delete.

**See Also**

- [ZipFile Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [ZipFile.Delete Overload List](#)
ICSharpCode SharpZipLib Class Library
ZipFile.Delete Method (String)

Delete an entry by name

```csharp
public bool Delete(
    string fileName
);
```

Parameters

fileName
The filename to delete

Return Value
True if the entry was found and deleted; false otherwise.

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.Dispose Method

Releases the unmanaged resources used by the this instance and optionally releases the managed resources.

```csharp
protected virtual void Dispose(
    bool disposing
);
```

Parameters

- `disposing`  
  true to release both managed and unmanaged resources; false to release only unmanaged resources.

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.Finalize Method

Finalize this instance.

```csharp
protected override void Finalize();
```

See Also

[ZipFile Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
**ZipFile.FindEntry Method**

Return the index of the entry with a matching name

```csharp
public int FindEntry(
    string name,
    bool ignoreCase
);
```

**Parameters**

- `name`  
  Entry name to find

- `ignoreCase`  
  If true the comparison is case insensitive

**Return Value**

The index position of the matching entry or -1 if not found

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>The Zip file has been closed.</td>
</tr>
</tbody>
</table>

**See Also**

- [ZipFile Class](#)  
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
ZipFile.GetEntry Method

Searches for a zip entry in this archive with the given name. String comparisons are case insensitive

```csharp
public ZipEntry GetEntry(
    string name
);
```

Parameters

`name`  
The name to find. May contain directory components separated by slashes (`'/'`).

Return Value

A clone of the zip entry, or null if no entry with that name exists.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>The Zip file has been closed.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.GetEnumerator Method

Returns an enumerator for the Zip entries in this Zip file.

```csharp
public IEnumerator GetEnumerator();
```

Implements

IEquatable<ZipEntry>

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>The Zip file has been closed.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.GetInputStream Method

Creates an input stream reading the given zip entry as uncompressed data. Normally zip entry should be an entry returned by GetEntry().

Overload List

Creates an input stream reading the given zip entry as uncompressed data. Normally zip entry should be an entry returned by GetEntry().

```
public Stream GetInputStream(ZipEntry);
```

Creates an input stream reading a zip entry

```
public Stream GetInputStream(long);
```

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.GetInputStream Method (ZipEntry)

Creates an input stream reading the given zip entry as uncompressed data. Normally zip entry should be an entry returned by GetEntry().

```csharp
public Stream GetInputStream(ZipEntry entry);
```

Return Value
the input stream.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>The ZipFile has already been closed</td>
</tr>
<tr>
<td>ZipException</td>
<td>The compression method for the entry is unknown</td>
</tr>
<tr>
<td>IndexOutOfRangeException</td>
<td>The entry is not found in the ZipFile</td>
</tr>
</tbody>
</table>

See Also
ICSharpCode SharpZipLib Class Library
ZipFile.GetInputStream Method (Int64)

Creates an input stream reading a zip entry

```csharp
public Stream GetInputStream(
    long entryIndex
);
```

Parameters

`entryIndex`

The index of the entry to obtain an input stream for.

Return Value

An input stream.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidOperationException</code></td>
<td>The ZipFile has already been closed</td>
</tr>
<tr>
<td><code>ZipException</code></td>
<td>The compression method for the entry is unknown</td>
</tr>
<tr>
<td><code>IndexOutOfRangeException</code></td>
<td>The entry is not found in the ZipFile</td>
</tr>
</tbody>
</table>

See Also

- `ZipFile Class`
- `ZipFile.GetInputStream Overload List`
ICSharpCode SharpZipLib Class Library
ZipFile.SetComment Method

Set the file comment to be recorded when the current update is committed.

```csharp
public void SetComment(
    string comment
);
```

Parameters

`comment`

The comment to record.

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.TestArchive Method

Test an archive for integrity/validity

Overload List

Test an archive for integrity/validity

```csharp
public bool TestArchive(bool);
```

Test an archive for integrity/validity

```csharp
public bool TestArchive(bool, TestStrategy, ZipTestResultHandler);
```

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.TestArchive Method (Boolean)

Test an archive for integrity/validity

```csharp
public bool TestArchive(
    bool testData
);
```

Parameters

testData
    Perform low level data Crc check

Return Value

true if all tests pass, false otherwise

Remarks

Testing will terminate on the first error found.

See Also

ICSharpCode SharpZipLib Class Library
ZipFile.TestArchive Method (Boolean, TestStrategy, ZipTestResultHandler)

Test an archive for integrity/validity

public bool TestArchive(
    bool testData,
    TestStrategy strategy,
    ZipTestResultHandler resultHandler
);

Parameters

testData
    Perform low level data Crc check

strategy
    The TestStrategy to apply.

resultHandler
    The ZipTestResultHandler handler to call during testing.

Return Value
    true if all tests pass, false otherwise

See Also

ICSharpCode SharpZipLib Class Library
Delegate for handling keys/password setting during compression/decompression.

```csharp
public delegate void ZipFile.KeysRequiredEventHandler(
    object sender,
    KeysRequiredEventArgs e
);
```

Requirements

**Namespace:** ICSharpCode.SharpZipLib.Zip

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

See Also

ICSharpCode SharpZipLib Class Library
ZipInputStream Class

This is an InflaterInputStream that reads the files baseInputStream an zip archive one after another. It has a special method to get the zip entry of the next file. The zip entry contains information about the file name size, compressed size, Crc, etc. It includes support for Stored and Deflated entries.

Author of the original java version: Jochen Hoenicke

For a list of all members of this type, see ZipInputStream Members.

System.Object
  System.MarshalByRefObject
  System.IO.Stream

public class ZipInputStream : InflaterInputStream

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Example

This sample shows how to read a zip file

[C#]
using System;
using System.Text;
using System.IO;


class MainClass
{
    public static void Main(string[] args)
```csharp
using (ZipInputStream s = new ZipInputStream(File.OpenRead(args[0])))
{
    ZipEntry theEntry;
    while ((theEntry = s.GetNextEntry()) != null)
    {
        int size = 2048;
        byte[] data = new byte[2048];

        Console.Write("Show contents (y/n) ?
        if (Console.ReadLine() == "y") {
            while (true) {
                size = s.Read(data, 0, data.
                if (size > 0) {
                    Console.Write(new ASCIIEncoding().GetString(data, 0, size));
                } else {
                    break;
                }
            }
        }
    }
}
```

Requirements

**Namespace:** [ICSharpCode.SharpZipLib.Zip](https://icsharpcode.net/SharpZipLib/)

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

See Also

### ZipInputStream Members

#### ZipInputStream overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th><strong>ZipInputStream Constructor</strong></th>
<th>Creates a new Zip input stream, for reading a zip archive.</th>
</tr>
</thead>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th><strong>Available</strong></th>
<th>Returns 1 if there is an entry available Otherwise returns 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CanDecompressEntry</strong></td>
<td>Gets a value indicating if there is a current entry and it can be decompressed</td>
</tr>
<tr>
<td><strong>CanRead</strong> (inherited from InflaterInputStream)</td>
<td>Gets a value indicating whether the current stream supports reading</td>
</tr>
<tr>
<td><strong>CanSeek</strong> (inherited from InflaterInputStream)</td>
<td>Gets a value of false indicating seeking is not supported for this stream.</td>
</tr>
<tr>
<td><strong>CanWrite</strong> (inherited from InflaterInputStream)</td>
<td>Gets a value of false indicating that this stream is not writeable.</td>
</tr>
<tr>
<td><strong>IsStreamOwner</strong> (inherited from InflaterInputStream)</td>
<td>Get/set flag indicating ownership of underlying stream. When the flag is true Close will close the underlying stream also.</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>Returns the current size that can be read from the current entry if available</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Optional password used for encryption when non-null</td>
</tr>
<tr>
<td><strong>Position</strong> (inherited from InflaterInputStream)</td>
<td>The current position within the stream. Throws a</td>
</tr>
</tbody>
</table>
NotSupportedException when attempting to set the position

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from InflaterInputStream)</td>
<td>Entry point to begin an asynchronous write. Always throws a NotSupportedException.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the zip input stream</td>
</tr>
<tr>
<td><strong>CloseEntry</strong></td>
<td>Closes the current zip entry and moves to the next one.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Flush</strong> (inherited from InflaterInputStream)</td>
<td>Flushes the baseInputStream</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetNextEntry</td>
<td>Advances to the next entry in the archive</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>InitializeLifetimeService (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td>Read</td>
<td>Read a block of bytes from the stream.</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Reads a byte from the current zip entry.</td>
</tr>
<tr>
<td>Seek (inherited from InflaterInputStream)</td>
<td>Sets the position within the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td>SetLength (inherited from InflaterInputStream)</td>
<td>Set the length of the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td>Skip (inherited from InflaterInputStream)</td>
<td>Skip specified number of bytes of uncompressed data</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td>Write (inherited from InflaterInputStream)</td>
<td>Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException</td>
</tr>
<tr>
<td>WriteByte (inherited from InflaterInputStream)</td>
<td>Writes one byte to the current stream and advances the current position Always throws a NotSupportedException</td>
</tr>
</tbody>
</table>

**Protected Instance Fields**
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>baseInputStream</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Base stream the inflater reads from.</td>
</tr>
<tr>
<td><code>csize</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>The compressed size</td>
</tr>
<tr>
<td><code>inf</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Decompressor for this stream</td>
</tr>
<tr>
<td><code>inputBuffer</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Input buffer for this stream.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateWaitHandle</code> (inherited from <code>Stream</code>)</td>
<td>Allocates a <code>WaitHandle</code> object.</td>
</tr>
<tr>
<td><code>Fill</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Fills the buffer with more data to decompress.</td>
</tr>
<tr>
<td><code>Finalize</code> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>StopDecrypting</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Clear any cryptographic state.</td>
</tr>
</tbody>
</table>

### See Also

ZipInputStream Constructor

Creates a new Zip input stream, for reading a zip archive.

```java
public ZipInputStream(
    Stream baseInputStream
);
```

See Also

ICSharpCode SharpZipLib Class Library
# ZipInputStream Properties

The properties of the `ZipInputStream` class are listed below. For a complete list of `ZipInputStream` class members, see the [ZipInputStream Members](#) topic.

## Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Available</strong></td>
<td>Returns 1 if there is an entry available. Otherwise returns 0.</td>
</tr>
<tr>
<td><strong>CanDecompressEntry</strong></td>
<td>Gets a value indicating if there is a current entry and it can be decompressed.</td>
</tr>
<tr>
<td><strong>CanRead</strong> (inherited from <code>InflaterInputStream</code>)</td>
<td>Gets a value indicating whether the current stream supports reading.</td>
</tr>
<tr>
<td><strong>CanSeek</strong> (inherited from <code>InflaterInputStream</code>)</td>
<td>Gets a value of false indicating seeking is not supported for this stream.</td>
</tr>
<tr>
<td><strong>CanWrite</strong> (inherited from <code>InflaterInputStream</code>)</td>
<td>Gets a value of false indicating that this stream is not writeable.</td>
</tr>
<tr>
<td><strong>IsStreamOwner</strong> (inherited from <code>InflaterInputStream</code>)</td>
<td>Get/set flag indicating ownership of underlying stream. When the flag is true, <code>Close</code> will close the underlying stream also.</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>Returns the current size that can be read from the current entry if available.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Optional password used for encryption when non-null.</td>
</tr>
<tr>
<td><strong>Position</strong> (inherited from <code>InflaterInputStream</code>)</td>
<td>The current position within the stream. Throws a <code>NotSupportedException</code> when attempting to set the position.</td>
</tr>
</tbody>
</table>
See Also

ICSharpCode SharpZipLib Class Library
ZipInputStream.Available Property

Returns 1 if there is an entry available. Otherwise returns 0.

```csharp
public override int Available {get;}
```

See Also

ZipInputStream.CanDecompressEntry Property

Gets a value indicating if there is a current entry and it can be decompressed

```csharp
public bool CanDecompressEntry {get;}
```

Remarks

The entry can only be decompressed if the library supports the zip features required to extract it. See the ZipEntry Version property for more details.

See Also

ICSharpCode SharpZipLib Class Library
ZipInputStream.Length Property

Returns the current size that can be read from the current entry if available

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipException</td>
<td>Thrown if the entry size is not known.</td>
</tr>
<tr>
<td>InvalidOperationException</td>
<td>Thrown if no entry is currently available.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipInputStream>Password Property

Optional password used for encryption when non-null

```
public string Password {get; set;}
```

See Also

ICSharpCode SharpZipLib Class Library
The methods of the `ZipInputStream` class are listed below. For a complete list of `ZipInputStream` class members, see the `ZipInputStream Members` topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from <code>Stream</code>)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from <code>InflaterInputStream</code>)</td>
<td>Entry point to begin an asynchronous write. Always throws a <code>NotSupportedException</code>.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the zip input stream</td>
</tr>
<tr>
<td><strong>CloseEntry</strong></td>
<td>Closes the current zip entry and moves to the next one.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from <code>MarshalByRefObject</code>)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from <code>Stream</code>)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from <code>Stream</code>)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>Flush</strong> (inherited from <code>InflaterInputStream</code>)</td>
<td>Flushes the baseInputStream</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <code>Object</code>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
</tbody>
</table>

**ZipInputStream Methods**
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetNextEntry</strong></td>
<td>Advances to the next entry in the archive</td>
</tr>
<tr>
<td><strong>GetTypeInfo</strong> (inherited from MarshalByRefObject)</td>
<td>Gets the TypeInfo of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Read a block of bytes from the stream.</td>
</tr>
<tr>
<td><strong>ReadByte</strong></td>
<td>Reads a byte from the current zip entry.</td>
</tr>
<tr>
<td><strong>Seek</strong> (inherited from InflaterInputStream)</td>
<td>Sets the position within the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td><strong>SetLength</strong> (inherited from InflaterInputStream)</td>
<td>Set the length of the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td><strong>Skip</strong> (inherited from InflaterInputStream)</td>
<td>Skip specified number of bytes of uncompressed data</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Write</strong> (inherited from InflaterInputStream)</td>
<td>Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException</td>
</tr>
<tr>
<td><strong>WriteByte</strong> (inherited from InflaterInputStream)</td>
<td>Writes one byte to the current stream and advances the current position Always throws a NotSupportedException</td>
</tr>
</tbody>
</table>
### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateWaitHandle</code> (inherited from <code>Stream</code>)</td>
<td>Allocates a <code>WaitHandle</code> object.</td>
</tr>
<tr>
<td><code>Fill</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Fills the buffer with more data to decompress.</td>
</tr>
<tr>
<td><code>Finalize</code> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>StopDecrypting</code> (inherited from <code>InflaterInputStream</code>)</td>
<td>Clear any cryptographic state.</td>
</tr>
</tbody>
</table>

### See Also

ICSharpCode SharpZipLib Class Library
ZipInputStream.Close Method

Closes the zip input stream

```csharp
public override void Close();
```

See Also

- [ZipInputStream Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
**ZipInputStream.CloseEntry Method**

Closes the current zip entry and moves to the next one.

```csharp
public void CloseEntry();
```

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidOperationException</code></td>
<td>The stream is closed</td>
</tr>
<tr>
<td><code>ZipException</code></td>
<td>The Zip stream ends early</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
ZipInputStream.GetNextEntry Method

Advances to the next entry in the archive

```java
public ZipEntry GetNextEntry();
```

Return Value

The next entry in the archive or null if there are no more entries.

Remarks

If the previous entry is still open CloseEntry is called.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>Input stream is closed</td>
</tr>
<tr>
<td>ZipException</td>
<td>Password is not set, password is invalid, compression method is invalid, version required to extract is not supported</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipInputStream.Read Method

Read a block of bytes from the stream.

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

Parameters

- **buffer**
  - The destination for the bytes.
- **offset**
  - The index to start storing data.
- **count**
  - The number of bytes to attempt to read.

Return Value

Returns the number of bytes read.

Remarks

Zero bytes read means end of stream.

See Also

ICSharpCode SharpZipLib Class Library
ZipInputStream.ReadByte Method

Reads a byte from the current zip entry.

```csharp
public override int ReadByte();
```

Return Value

The byte or -1 if end of stream is reached.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The deflated stream is corrupted.</td>
</tr>
<tr>
<td></td>
<td>An i/o error occurred.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipNameTransform Class

ZipNameTransform transforms names as per the Zip file naming convention.

For a list of all members of this type, see ZipNameTransform Members.

System.Object

```
public class ZipNameTransform : INameTransform
```

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Remarks

The use of absolute names is supported although its use is not valid according to Zip naming conventions, and should not be used if maximum compatibility is desired.

Requirements


See Also

ZipNameTransform Members | ICSHarpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipNameTransform Members

ZipNameTransform overview

Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsValidName</td>
<td>Overloaded. Test a name to see if it is a valid name for a zip entry.</td>
</tr>
</tbody>
</table>

Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipNameTransform</td>
<td>Overloaded. Initializes a new instance of the ZipNameTransform class.</td>
</tr>
</tbody>
</table>

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrimPrefix</td>
<td>Get/set the path prefix to be trimmed from paths if present.</td>
</tr>
</tbody>
</table>

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>(inherited from Object) Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>(inherited from Object) Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType</td>
<td>(inherited from Object) Gets the Type of the current instance.</td>
</tr>
<tr>
<td>ToString</td>
<td>(inherited from Object) Returns a String that represents the current Object.</td>
</tr>
<tr>
<td>TransformDirectory</td>
<td>Transform a directory name according to the Zip file naming conventions.</td>
</tr>
<tr>
<td>TransformFile</td>
<td>Transform a file name according</td>
</tr>
</tbody>
</table>
to the Zip file naming conventions.

Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✤ <strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>✤ <strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

[ZipNameTransform Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
| ICSharpCode SharpZipLib Class Library |
ZipNameTransform Constructor

Initialize a new instance of ZipNameTransform

Overload List

Initialize a new instance of ZipNameTransform

public ZipNameTransform();

Initialize a new instance of ZipNameTransform

public ZipNameTransform(string);

See Also

ZipNameTransform Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipNameTransform Constructor ()

Initialize a new instance of ZipNameTransform

```csharp
public ZipNameTransform();
```

See Also

[ZipNameTransform Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#) | [ZipNameTransform Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
ZipNameTransform Constructor (String)

Initialize a new instance of `ZipNameTransform`

```csharp
public ZipNameTransform(
    string trimPrefix
);
```

Parameters

`trimPrefix`
The string to trim from front of paths if found.

See Also

[ZipNameTransform Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#) | [ZipNameTransform Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
The properties of the **ZipNameTransform** class are listed below. For a complete list of **ZipNameTransform** class members, see the **ZipNameTransform Members** topic.

### Public Instance Properties

| **TrimPrefix** | Get/set the path prefix to be trimmed from paths if present. |

### See Also

- [ZipNameTransform Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
ZipNameTransform.TrimPrefix Property

Get/set the path prefix to be trimmed from paths if present.

```csharp
public string TrimPrefix {get; set;}
```

See Also

ZipNameTransform Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
ZipNameTransform Methods

The methods of the **ZipNameTransform** class are listed below. For a complete list of **ZipNameTransform** class members, see the [ZipNameTransform Members] topic.

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsValidName</td>
<td>Overloaded. Test a name to see if it is a valid name for a zip entry.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>TransformDirectory</td>
<td>Transform a directory name according to the Zip file naming conventions.</td>
</tr>
<tr>
<td>TransformFile</td>
<td>Transform a file name according to the Zip file naming conventions.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by</td>
</tr>
<tr>
<td>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>

**See Also**

[ZipNameTransform Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
ZipNameTransform.IsValidName Method

Test a name to see if it is a valid name for a zip entry.

Overload List

Test a name to see if it is a valid name for a zip entry.

public static bool IsValidName(string);

Test a name to see if it is a valid name for a zip entry.

public static bool IsValidName(string,bool);

See Also

ZipNameTransform Class | ISharpCode.SharpZipLib.Zip Namespace
ZipNameTransform.IsValidName Method (String)

Test a name to see if it is a valid name for a zip entry.

```csharp
public static bool IsValidName(
    string name
);
```

Parameters

- `name`  
  The name to test.

Return Value

Returns true if the name is a valid zip name; false otherwise.

Remarks

Zip path names are actually in unix format, and should only contain relative paths if a path is present. This means that the path stored should not contain a drive or device letter, or a leading slash. All slashes should forward slashes '/'. An empty name is valid where the input comes from standard input. A null name is not considered valid.

See Also

- ZipNameTransform Class  |  ISharpCode.SharpZipLib.Zip Namespace  |  ZipNameTransform.IsValidName Overload List
ICSharpCode SharpZipLib Class Library
ZipNameTransform.IsValidName Method (String, Boolean)

Test a name to see if it is a valid name for a zip entry.

```csharp
public static bool IsValidName(string name, bool relaxed);
```

**Parameters**

- `name`  
  The name to test.

- `relaxed`  
  If true checking is relaxed about windows file names and absolute paths.

**Return Value**

Returns true if the name is a valid zip name; false otherwise.

**Remarks**

Zip path names are actually in Unix format, and should only contain relative paths. This means that any path stored should not contain a drive or device letter, or a leading slash. All slashes should forward slashes `/`. An empty name is valid for a file where the input comes from standard input. A null name is not considered valid.

**See Also**

**ZipNameTransform.TransformDirectory Method**

Transform a directory name according to the Zip file naming conventions.

```csharp
public string TransformDirectory(
    string name
);
```

**Parameters**

- `name`  
  The directory name to transform.

**Return Value**

The transformed name.

**Implements**

- `INameTransform.TransformDirectory`

**See Also**

- [ZipNameTransform Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
ICSharpCode SharpZipLib Class Library
ZipNameTransform.TransformFile Method

Transform a file name according to the Zip file naming conventions.

```csharp
public string TransformFile(string name);
```

Parameters

*name*

The file name to transform.

Return Value

The transformed name.

Implements

INameTransform.TransformFile

See Also

ZipNameTransform Class | ISharpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library
**ZipOutputStream Class**

This is a DeflaterOutputStream that writes the files into a zip archive one after another. It has a special method to start a new zip entry. The zip entries contains information about the file name size, compressed size, CRC, etc. It includes support for Stored and Deflated entries. This class is not thread safe.

Author of the original java version : Jochen Hoenicke

For a list of all members of this type, see [ZipOutputStream Members](#).

**System.Object**

**System.MarshalByRefObject**

**System.IO.Stream**


```csharp
public class ZipOutputStream : DeflaterOutputStream
```

**Thread Safety**

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are *not* guaranteed to be thread-safe.

**Example**

This sample shows how to create a zip file

```csharp
using System;
using System.IO;
using ISharpCode.SharpZipLib.Core;

class MainClass
{
    public static void Main(string[] args)
```csharp
string[] filenames = Directory.GetFiles(args[0]);
byte[] buffer = new byte[4096];

using (ZipOutputStream s = new ZipOutputStream(File.Create(args[1])))
{
    s.SetLevel(9); // 0 - store only to 9 -

    foreach (string file in filenames)
    {
        ZipEntry entry = new ZipEntry(file);
        s.PutNextEntry(entry);

        using (FileStream fs = File.OpenRead(file))
        {
            StreamUtils.Copy(fs, s, buffer);
        }
    }
}
```

**Requirements**

**Namespace:** [ICSharpCode.SharpZipLib.Zip](https://icsharpcode.net/ncr/SharpZipLib/Zip)

**Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

**See Also**

ICSharpCode SharpZipLib Class Library
### ZipOutputStream Members

#### ZipOutputStream overview

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>ZipOutputStream Constructor</th>
<th>Creates a new Zip output stream, writing a zip archive.</th>
</tr>
</thead>
</table>

**Public Instance Properties**

<table>
<thead>
<tr>
<th>CanPatchEntries (inherited from DeflaterOutputStream)</th>
<th>Allows client to determine if an entry can be patched after its added</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead (inherited from DeflaterOutputStream)</td>
<td>Gets value indicating stream can be read from</td>
</tr>
<tr>
<td>CanSeek (inherited from DeflaterOutputStream)</td>
<td>Gets a value indicating if seeking is supported for this stream This property always returns false</td>
</tr>
<tr>
<td>CanWrite (inherited from DeflaterOutputStream)</td>
<td>Get value indicating if this stream supports writing</td>
</tr>
<tr>
<td>IsFinished</td>
<td>Gets boolean indicating central header has been added for this archive... No further entries can be added once this has been done.</td>
</tr>
<tr>
<td>IsStreamOwner (inherited from DeflaterOutputStream)</td>
<td>Get/set flag indicating ownership of the underlying stream. When the flag is true Close will close the underlying stream also.</td>
</tr>
<tr>
<td>Length (inherited from DeflaterOutputStream)</td>
<td>Get current length of stream</td>
</tr>
<tr>
<td>Password (inherited from DeflaterOutputStream)</td>
<td>Get/set the password used for encryption.</td>
</tr>
</tbody>
</table>
### Position (inherited from DeflaterOutputStream)
Gets the current position within the stream.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous reads are not supported a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous writes aren't supported, a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>Close</strong> (inherited from DeflaterOutputStream)</td>
<td>Calls Finish and closes the underlying stream when IsStreamOwner is true.</td>
</tr>
<tr>
<td><strong>CloseEntry</strong></td>
<td>Closes the current entry, updating header and footer information as required</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Finishes the stream. This will write the central directory at the end of the zip file and flush the stream.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Flush</strong> (inherited from DeflaterOutputStream)**</td>
<td>Flushes the stream by calling flush() on the deflater and then on the underlying stream. This ensures that all bytes are flushed.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLevel</strong></td>
<td>Get the current deflate compression level</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)**</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from MarshalByRefObject)**</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>PutNextEntry</strong></td>
<td>Starts a new Zip entry. It automatically closes the previous entry if present. All entry elements bar name are optional, but must be correct if present. If the compression method is stored and the output is not patchable the compression for that entry is automatically changed to deflate level 0</td>
</tr>
<tr>
<td><strong>Read</strong> (inherited from DeflaterOutputStream)**</td>
<td>Read a block of bytes from stream</td>
</tr>
<tr>
<td><strong>ReadByte</strong> (inherited from DeflaterOutputStream)**</td>
<td>Read a byte from stream advancing position by one</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Seek (inherited from DeflaterOutputStream)</td>
<td>Sets the current position of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td>SetComment</td>
<td>Set the zip file comment.</td>
</tr>
<tr>
<td>SetLength (inherited from DeflaterOutputStream)</td>
<td>Sets the length of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td>SetLevel</td>
<td>Sets default compression level. The new level will be activated immediately.</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td>Write</td>
<td>Writes the given buffer to the current entry.</td>
</tr>
<tr>
<td>WriteByte (inherited from DeflaterOutputStream)</td>
<td>Writes a single byte to the compressed output stream.</td>
</tr>
</tbody>
</table>

Protected Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseOutputStream (inherited from DeflaterOutputStream)</td>
<td>Base stream the deflater depends on.</td>
</tr>
<tr>
<td>def (inherited from DeflaterOutputStream)</td>
<td>The deflater which is used to deflate the stream.</td>
</tr>
</tbody>
</table>

Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateWaitHandle (inherited from Stream)</td>
<td>Allocates a WaitHandle object.</td>
</tr>
<tr>
<td>Deflate (inherited from DeflaterOutputStream)</td>
<td>Deflates everything in the input buffers. This will call def.deflate() until all bytes from the input buffers are processed.</td>
</tr>
<tr>
<td>EncryptBlock (inherited from)</td>
<td>Encrypt a block of data</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>EncryptByte (inherited from DeflaterOutputStream)</td>
<td>Encrypt a single byte</td>
</tr>
<tr>
<td>Finalize (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>InitializePassword (inherited from DeflaterOutputStream)</td>
<td>Initializes encryption keys based on given password</td>
</tr>
<tr>
<td>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>UpdateKeys (inherited from DeflaterOutputStream)</td>
<td>Update encryption keys</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipOutputStream Constructor

Creates a new Zip output stream, writing a zip archive.

```java
public ZipOutputStream(
    Stream baseOutputStream
);
```

Parameters

*baseOutputStream*

The output stream to which the archive contents are written.

See Also

ICSharpCode SharpZipLib Class Library
The properties of the **ZipOutputStream** class are listed below. For a complete list of **ZipOutputStream** class members, see the [ZipOutputStream Members](#) topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CanPatchEntries</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Allows client to determine if an entry can be patched after its added</td>
</tr>
<tr>
<td><strong>CanRead</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Gets value indicating stream can be read from</td>
</tr>
<tr>
<td><strong>CanSeek</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Gets a value indicating if seeking is supported for this stream. This property always returns false</td>
</tr>
<tr>
<td><strong>CanWrite</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Get value indicating if this stream supports writing</td>
</tr>
<tr>
<td><strong>IsFinished</strong></td>
<td>Gets boolean indicating central header has been added for this archive... No further entries can be added once this has been done.</td>
</tr>
<tr>
<td><strong>IsStreamOwner</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Get/set flag indicating ownership of the underlying stream. When the flag is true <strong>Close</strong> will close the underlying stream also.</td>
</tr>
<tr>
<td><strong>Length</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Get current length of stream</td>
</tr>
<tr>
<td><strong>Password</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Get/set the password used for encryption.</td>
</tr>
<tr>
<td><strong>Position</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Gets the current position within the stream.</td>
</tr>
</tbody>
</table>
See Also

ICSharpCode SharpZipLib Class Library
ZipOutputStream.IsFinished Property

Gets boolean indicating central header has been added for this archive... No further entries can be added once this has been done.

```csharp
public bool IsFinished {get;}
```

See Also

[ZipOutputStream Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
ICSharpCode SharpZipLib Class Library
ZipOutputStream Methods

The methods of the **ZipOutputStream** class are listed below. For a complete list of **ZipOutputStream** class members, see the [ZipOutputStream Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous reads are not supported a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous writes are not supported, a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>Close</strong> (inherited from DeflaterOutputStream)</td>
<td>Calls <strong>Finish</strong> and closes the underlying stream when <strong>IsStreamOwner</strong> is true.</td>
</tr>
<tr>
<td><strong>CloseEntry</strong></td>
<td>Closes the current entry, updating header and footer information as required</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Finishes the stream. This will</td>
</tr>
</tbody>
</table>
write the central directory at the end of the zip file and flush the stream.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flush</strong> (inherited from DeflaterOutputStream)</td>
<td>Flushes the stream by calling flush() on the deflater and then on the underlying stream. This ensures that all bytes are flushed.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLevel</strong></td>
<td>Get the current deflate compression level</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>PutNextEntry</strong></td>
<td>Starts a new Zip entry. It automatically closes the previous entry if present. All entry elements bar name are optional, but must be correct if present. If the compression method is stored and the output is not patchable the compression for that entry is automatically changed to deflate level 0</td>
</tr>
<tr>
<td><strong>Read</strong> (inherited from DeflaterOutputStream)</td>
<td>Read a block of bytes from stream</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>ReadStream</strong> (inherited from DeflaterOutputStream)</td>
<td>Read a byte from stream advancing position by one</td>
</tr>
<tr>
<td><strong>Seek</strong> (inherited from DeflaterOutputStream)</td>
<td>Sets the current position of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><strong>SetComment</strong></td>
<td>Set the zip file comment.</td>
</tr>
<tr>
<td><strong>SetLength</strong> (inherited from DeflaterOutputStream)</td>
<td>Sets the length of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><strong>SetLevel</strong></td>
<td>Sets default compression level. The new level will be activated immediately.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes the given buffer to the current entry.</td>
</tr>
<tr>
<td><strong>WriteByte</strong> (inherited from DeflaterOutputStream)</td>
<td>Writes a single byte to the compressed output stream.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateWaitHandle</strong> (inherited from Stream)</td>
<td>Allocates a WaitHandle object.</td>
</tr>
</tbody>
</table>
| **Deflate** (inherited from DeflaterOutputStream) | Deflates everything in the input buffers. This will call

```java
def.deflate()
```
until all bytes from the input buffers are processed. |
<p>| <strong>EncryptBlock</strong> (inherited from DeflaterOutputStream) | Encrypt a block of data |
| <strong>EncryptByte</strong> (inherited from DeflaterOutputStream) | Encrypt a single byte |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>InitializePassword</strong> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>Initializes encryption keys based on given password.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>UpdateKeys</strong> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>Update encryption keys.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
ZipOutputStream.CloseEntry Method

Closes the current entry, updating header and footer information as required

```csharp
public void CloseEntry();
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>An I/O error occurs.</td>
</tr>
<tr>
<td>InvalidOperationException</td>
<td>No entry is active.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
**ZipOutputStream.Finish Method**

Finishes the stream. This will write the central directory at the end of the zip file and flush the stream.

```csharp
public override void Finish();
```

**Remarks**

This is automatically called when the stream is closed.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOException</td>
<td>An I/O error occurs.</td>
</tr>
<tr>
<td>ZipException</td>
<td>Comment exceeds the maximum length</td>
</tr>
<tr>
<td></td>
<td>Entry name exceeds the maximum length</td>
</tr>
</tbody>
</table>

**See Also**

ZipOutputStream.GetLevel Method

Get the current deflate compression level

```csharp
public int GetLevel();
```

Return Value

The current compression level

See Also

ICSharpCode SharpZipLib Class Library
ZipOutputStream.PutNextEntry Method

Starts a new Zip entry. It automatically closes the previous entry if present. All entry elements bar name are optional, but must be correct if present. If the compression method is stored and the output is not patchable the compression for that entry is automatically changed to deflate level 0

```csharp
public void PutNextEntry(
    ZipEntry entry
);
```

Parameters

- `entry` the entry.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td>if entry passed is null.</td>
</tr>
<tr>
<td>IOException</td>
<td>if an I/O error occured.</td>
</tr>
<tr>
<td>InvalidOperationException</td>
<td>if stream was finished</td>
</tr>
</tbody>
</table>
| ZipException                 | Too many entries in the Zip file Entry name is too long
|                               | Finish has already been called                |

See Also

ICSharpCode SharpZipLib Class Library
Set the zip file comment.

```csharp
class ZipOutputStream
{
    public void SetComment(
        string comment
    );
}
```

Parameters

- **comment**
  - The comment string

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Encoding of comment is longer than 0xffff bytes.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipOutputStream.SetLevel Method

Sets default compression level. The new level will be activated immediately.

```csharp
public void SetLevel(int level);
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>Level specified is not supported.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipOutputStream.Write Method

Writes the given buffer to the current entry.

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipException</td>
<td>Archive size is invalid</td>
</tr>
<tr>
<td>InvalidOperationException</td>
<td>No entry is active.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
ZipTestResultHandler Delegate

Delegate invoked during testing if supplied indicating current progress and status.

```csharp
public delegate void ZipTestResultHandler(
    TestStatus status,
    string message
);
```

Remarks
If the message is non-null an error has occurred. If the message is null the operation as found in status has started.

Requirements


See Also

ICSHarpCode.SharpZipLib.Zip Namespace
ICSharpCode SharpZipLib Class Library

## Namespace hierarchy

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deflater</strong></td>
<td>This is the Deflater class. The deflater class compresses input with the deflate algorithm described in RFC 1951. It has several compression levels and three different strategies described below. This class is <em>not</em> thread safe. This is inherent in the API, due to the split of deflate and setInput. author of the original java version : Jochen Hoenicke</td>
</tr>
<tr>
<td><strong>DeflaterConstants</strong></td>
<td>This class contains constants used for deflation.</td>
</tr>
<tr>
<td><strong>DeflaterEngine</strong></td>
<td>Low level compression engine for deflate algorithm which uses a 32K sliding window with secondary compression from Huffman/Shannon-Fano codes.</td>
</tr>
<tr>
<td><strong>DeflaterHuffman</strong></td>
<td>This is the DeflaterHuffman class. This class is <em>not</em> thread safe. This is inherent in the API, due to the split of Deflate and SetInput. author of the original java version : Jochen Hoenicke</td>
</tr>
<tr>
<td><strong>DeflaterPending</strong></td>
<td>This class stores the pending output of the Deflater. author of the original java version : Jochen Hoenicke</td>
</tr>
<tr>
<td><strong>Inflator</strong></td>
<td>Inflater is used to decompress data that has been compressed according to the &quot;deflate&quot; standard described in rfc1951. By default Zlib (rfc1950) headers and footers are expected in the input. You can use constructor</td>
</tr>
</tbody>
</table>
public Inflater(bool noHeader)

passing true if there is no Zlib header information The usage is as following. First you have to set some input with 

SetInput()

, then Inflate() it. If inflate doesn't inflate any bytes there may be three reasons:

- IsNeedingInput() returns true because the input buffer is empty. You have to provide more input with 

  SetInput()

. NOTE: IsNeedingInput() also returns true when, the stream is finished.

- IsNeedingDictionary() returns true, you have to provide a preset dictionary with 

  SetDictionary()

. 

- IsFinished returns true, the inflater has finished.

Once the first output byte is produced, a dictionary will not be needed at a later stage. author of the original java version : John Leuner, Jochen Hoenicke

<table>
<thead>
<tr>
<th>InflaterHuffmanTree</th>
<th>Huffman tree used for inflation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PendingBuffer</td>
<td>This class is general purpose class for writing data to a buffer. It allows you to write bits as well as bytes Based on</td>
</tr>
</tbody>
</table>
Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflateStrategy</td>
<td>Strategies for deflater</td>
</tr>
</tbody>
</table>
ICSharpCode SharpZipLib Class Library
# Deflater Class

This is the Deflater class. The deflater class compresses input with the deflate algorithm described in RFC 1951. It has several compression levels and three different strategies described below. This class is *not* thread safe. This is inherent in the API, due to the split of deflate and setInput. author of the original java version: Jochen Hoenicke

For a list of all members of this type, see [Deflater Members](#).

```System.Object
```

## Thread Safety

Public static (*Shared* in Visual Basic) members of this type are safe for multithreaded operations. Instance members are *not* guaranteed to be thread-safe.

## Requirements

- **Namespace:** [ICSharpCode.SharpZipLib.Zip.Compression](#)
- **Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

## See Also

ICSharpCode SharpZipLib Class Library
### Deflater Members

#### Deflater overview

#### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEST_COMPRESSION</td>
<td>The best and slowest compression level. This tries to find very long and distant string repetitions.</td>
</tr>
<tr>
<td>BEST_SPEED</td>
<td>The worst but fastest compression level.</td>
</tr>
<tr>
<td>DEFAULT_COMPRESSION</td>
<td>The default compression level.</td>
</tr>
<tr>
<td>DEFLATED</td>
<td>The compression method. This is the only method supported so far. There is no need to use this constant at all.</td>
</tr>
<tr>
<td>NO_COMPRESSION</td>
<td>This level won't compress at all but output uncompressed blocks.</td>
</tr>
</tbody>
</table>

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deflater</td>
<td>Overloaded. Initializes a new instance of the Deflater class.</td>
</tr>
</tbody>
</table>

#### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adler</td>
<td>Gets the current adler checksum of the data that was processed so far.</td>
</tr>
<tr>
<td>IsFinished</td>
<td>Returns true if the stream was finished and no more output bytes are available.</td>
</tr>
<tr>
<td>IsNeedingInput</td>
<td>Returns true, if the input buffer is empty. You should then call setInput(). NOTE: This method</td>
</tr>
</tbody>
</table>
can also return true when the stream was finished.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TotalIn</strong></td>
<td>Gets the number of input bytes processed so far.</td>
</tr>
<tr>
<td><strong>TotalOut</strong></td>
<td>Gets the number of output bytes so far.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deflate</strong></td>
<td>Overloaded. Deflates the current input block with to the given array.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Finishes the deflater with the current input block. It is an error to give</td>
</tr>
<tr>
<td></td>
<td>more input after this method was called. This method must be called to</td>
</tr>
<tr>
<td></td>
<td>force all bytes to be flushed.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the current input block. Further calls to deflate() will produce</td>
</tr>
<tr>
<td></td>
<td>enough output to inflate everything in the current input block. This is not</td>
</tr>
<tr>
<td></td>
<td>part of Sun's JDK so I have made it package private. It is used by</td>
</tr>
<tr>
<td></td>
<td>DeflaterOutputStream to implement flush().</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in</td>
</tr>
<tr>
<td></td>
<td>hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLevel</strong></td>
<td>Get current compression level</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets the deflater. The deflater acts afterwards as if it was just</td>
</tr>
<tr>
<td></td>
<td>created with the same compression level and strategy as it had before.</td>
</tr>
</tbody>
</table>
| **SetDictionary** | Overloaded. Sets the dictionary which should be used in the deflate process. This call is equivalent to  
|                 | `setDictionary(dict, 0, dict.Length)` |
| **SetInput**    | Overloaded. Sets the data which should be compressed next. This should be only called when `needsInput` indicates that more input is needed. If you call `setInput` when `needsInput()` returns false, the previous input that is still pending will be thrown away. The given byte array should not be changed, before `needsInput()` returns true again. This call is equivalent to  
|                 | `setInput(input, 0, input.length)` |
| **SetLevel**    | Sets the compression level. There is no guarantee of the exact position of the change, but if you call this when `needsInput` is true the change of compression level will occur somewhere near before the end of the so far given input.  
| **SetStrategy** | Sets the compression strategy. Strategy is one of DEFAULT_STRATEGY, HUFFMAN_ONLY and FILTERED. For the exact position where the strategy is changed, the same as for `SetLevel()` applies.  
| **ToString**   | Returns a `String` that represents the current `Object`. |

**Protected Instance Methods**

| **Finalize** (inherited from `Object`) | Allows an `Object` to attempt to free resources and perform |
other cleanup operations before the **Object** is reclaimed by garbage collection.

| 🍃 **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

### Deflater Constructor

Creates a new deflater with default compression level.

### Overload List

Creates a new deflater with default compression level.

```csharp
public Deflater();
```

Creates a new deflater with given compression level.

```csharp
public Deflater(int);
```

Creates a new deflater with given compression level.

```csharp
public Deflater(int,bool);
```

### See Also

ICSharpCode SharpZipLib Class Library
Deflater Constructor ()

Creates a new deflater with default compression level.

public Deflater();

See Also

ICSharpCode SharpZipLib Class Library
**Deflater Constructor (Int32)**

Creates a new deflater with given compression level.

```java
public Deflater(
    int level
);
```

**Parameters**

*level*  
the compression level, a value between NO_COMPRESSION and BEST_COMPRESSION, or DEFAULT_COMPRESSION.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>if lvl is out of range.</td>
</tr>
</tbody>
</table>

**See Also**

Deflater Constructor (Int32, Boolean)

Creates a new deflater with given compression level.

```csharp
public Deflater(
    int level,
    bool noZlibHeaderOrFooter
);
```

Parameters

- `level` the compression level, a value between NO_COMPRESSION and BEST_COMPRESSION.

- `noZlibHeaderOrFooter` true, if we should suppress the Zlib/RFC1950 header at the beginning and the adler checksum at the end of the output. This is useful for the GZIP/PKZIP formats.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>if lvl is out of range.</td>
</tr>
</tbody>
</table>

See Also

### Deflater Fields

The fields of the **Deflater** class are listed below. For a complete list of **Deflater** class members, see the **Deflater Members** topic.

#### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BEST_COMPRESSION</strong></td>
<td>The best and slowest compression level. This tries to find very long and distant string repetitions.</td>
</tr>
<tr>
<td><strong>BEST_SPEED</strong></td>
<td>The worst but fastest compression level.</td>
</tr>
<tr>
<td><strong>DEFAULT_COMPRESSION</strong></td>
<td>The default compression level.</td>
</tr>
<tr>
<td><strong>DEFLATED</strong></td>
<td>The compression method. This is the only method supported so far. There is no need to use this constant at all.</td>
</tr>
<tr>
<td><strong>NO_COMPRESSION</strong></td>
<td>This level won't compress at all but output uncompressed blocks.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
Deflater.BEST_COMPRESSION Field

The best and slowest compression level. This tries to find very long and distant string repetitions.

```csharp
public const int BEST_COMPRESSION = 9;
```

See Also

- Deflater Class
ICSharpCode SharpZipLib Class Library
Deflater.BEST_SPEED Field

The worst but fastest compression level.

```
public const int BEST_SPEED = 1;
```

See Also

- [Deflater Class](#)
ICSharpCode SharpZipLib Class Library
Deflater.DEFAULT_COMPRESSION Field

The default compression level.

```csharp
public const int DEFAULT_COMPRESSION = -1;
```

See Also

ICSharpCode SharpZipLib Class Library
The compression method. This is the only method supported so far. There is no need to use this constant at all.

```csharp
public const int DEFLATED = 8;
```

See Also

ICSharpCode SharpZipLib Class Library
Deflater.NO_COMPRESSION Field

This level won't compress at all but output uncompressed blocks.

```public const int NO_COMPRESSION = 0;```

See Also

# Deflater Properties

The properties of the **Deflater** class are listed below. For a complete list of **Deflater** class members, see the [Deflater Members](#) topic.

## Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adler</strong></td>
<td>Gets the current adler checksum of the data that was processed so far.</td>
</tr>
<tr>
<td><strong>IsFinished</strong></td>
<td>Returns true if the stream was finished and no more output bytes are available.</td>
</tr>
<tr>
<td><strong>IsNeedingInput</strong></td>
<td>Returns true, if the input buffer is empty. You should then call setInput(). NOTE: This method can also return true when the stream was finished.</td>
</tr>
<tr>
<td><strong>TotalIn</strong></td>
<td>Gets the number of input bytes processed so far.</td>
</tr>
<tr>
<td><strong>TotalOut</strong></td>
<td>Gets the number of output bytes so far.</td>
</tr>
</tbody>
</table>

## See Also

ICSharpCode SharpZipLib Class Library
**Deflater.Adler Property**

Gets the current adler checksum of the data that was processed so far.

```csharp
public int Adler {get;}
```

**See Also**

ICSharpCode SharpZipLib Class Library
Deflater.IsFinished Property

Returns true if the stream was finished and no more output bytes are available.

```csharp
public bool IsFinished {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
Deflater.IsNeedingInput Property

Returns true, if the input buffer is empty. You should then call setInput(). NOTE: This method can also return true when the stream was finished.

```csharp
public bool IsNeedingInput {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
Deflater.TotalIn Property

Gets the number of input bytes processed so far.

```csharp
public int TotalIn {get;}
```

See Also

- [Deflater Class](#)
ICSharpCode SharpZipLib Class Library
Deflater.TotalOut Property

Gets the number of output bytes so far.

```csharp
public long TotalOut {get;}
```

See Also

- Deflater Class
ICSharpCode SharpZipLib Class Library
## Deflater Methods

The methods of the **Deflater** class are listed below. For a complete list of **Deflater** class members, see the **Deflater Members** topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deflate</strong></td>
<td>Overloaded. Deflates the current input block with to the given array.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Finishes the deflater with the current input block. It is an error to give</td>
</tr>
<tr>
<td></td>
<td>more input after this method was called. This method must be called to force</td>
</tr>
<tr>
<td></td>
<td>all bytes to be flushed.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the current input block. Further calls to deflate() will produce</td>
</tr>
<tr>
<td></td>
<td>enough output to inflate everything in the current input block. This is not</td>
</tr>
<tr>
<td></td>
<td>part of Sun's JDK so I have made it package private. It is used by</td>
</tr>
<tr>
<td></td>
<td>DeflaterOutputStream to implement flush().</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing</td>
</tr>
<tr>
<td>(inherited from</td>
<td>algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GetLevel</strong></td>
<td>Get current compression level</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>(inherited from</td>
<td></td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets the deflater. The deflater acts afterwards as if it was just created</td>
</tr>
<tr>
<td></td>
<td>with the same compression level and strategy as it had before.</td>
</tr>
<tr>
<td><strong>SetDictionary</strong></td>
<td>Overloaded. Sets the dictionary which should be used in the deflate process.</td>
</tr>
<tr>
<td></td>
<td>This call is equivalent to</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SetInput</strong></td>
<td>Overloaded. Sets the data which should be compressed next. This should be only called when needsInput indicates that more input is needed. If you call setInput when needsInput() returns false, the previous input that is still pending will be thrown away. The given byte array should not be changed, before needsInput() returns true again. This call is equivalent to setInput(input, 0, input.length).</td>
</tr>
<tr>
<td><strong>SetLevel</strong></td>
<td>Sets the compression level. There is no guarantee of the exact position of the change, but if you call this when needsInput is true the change of compression level will occur somewhere near before the end of the so far given input.</td>
</tr>
<tr>
<td><strong>SetStrategy</strong></td>
<td>Sets the compression strategy. Strategy is one of DEFAULT_STRATEGY, HUFFMAN_ONLY and FILTERED. For the exact position where the strategy is changed, the same as for SetLevel() applies.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Example Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
<td></td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
**Deflater.Deflate Method**

Deflates the current input block with to the given array.

**Overload List**

Deflates the current input block with to the given array.

```java
public int Deflate(byte[]);
```

Deflates the current input block to the given array.

```java
public int Deflate(byte[],int,int);
```

**See Also**

ICSharpCode SharpZipLib Class Library
Deflates the current input block with the given array.

```csharp
public int Deflate(byte[] output);
```

**Parameters**

- `output`
  The buffer where compressed data is stored

**Return Value**

The number of compressed bytes added to the output, or 0 if either `IsNeedingInput()` or `IsFinished` returns true or length is zero.

**See Also**

- [Deflater Class](#)
- [Deflater.Deflate Overload List](#)
ICSharpCode SharpZipLib Class Library
Deflates the current input block to the given array.

```csharp
public int Deflate(
    byte[] output,
    int offset,
    int length
);
```

**Parameters**

- `output` Buffer to store the compressed data.
- `offset` Offset into the output array.
- `length` The maximum number of bytes that may be stored.

**Return Value**

The number of compressed bytes added to the output, or 0 if either `needsInput()` or `finished()` returns true or `length` is zero.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidOperationException</code></td>
<td>If Finish() was previously called.</td>
</tr>
<tr>
<td><code>ArgumentOutOfRangeException</code></td>
<td>If offset or length don't match the array length.</td>
</tr>
</tbody>
</table>

**See Also**

- [Deflater Class](#)
- [Deflater.Deflate Overload List](#)
Deflater.Finish Method

Finishes the deflater with the current input block. It is an error to give more input after this method was called. This method must be called to force all bytes to be flushed.

```csharp
public void Finish();
```

See Also

ICSharpCode SharpZipLib Class Library
Deflater.Flush Method

Flushes the current input block. Further calls to deflate() will produce enough output to inflate everything in the current input block. This is not part of Sun's JDK so I have made it package private. It is used by DeflaterOutputStream to implement flush().

```java
public void Flush();
```

See Also

ICSharpCode SharpZipLib Class Library
Deflater.GetLevel Method

Get current compression level

```csharp
public int GetLevel();
```

Return Value

Returns the current compression level

See Also

ICSharpCode SharpZipLib Class Library
**Deflater.Reset Method**

Resets the deflater. The deflater acts afterwards as if it was just created with the same compression level and strategy as it had before.

```csharp
public void Reset();
```

**See Also**

ICSharpCode SharpZipLib Class Library
**Deflater.SetDictionary Method**

Sets the dictionary which should be used in the deflate process. This call is equivalent to

```csharp
setDictionary(dict, 0, dict.Length)
```

**Overload List**

Sets the dictionary which should be used in the deflate process. This call is equivalent to

```csharp
setDictionary(dict, 0, dict.Length)
```

```csharp
public void SetDictionary(byte[]);
```

Sets the dictionary which should be used in the deflate process. The dictionary is a byte array containing strings that are likely to occur in the data which should be compressed. The dictionary is not stored in the compressed output, only a checksum. To decompress the output you need to supply the same dictionary again.

```csharp
public void SetDictionary(byte[],int,int);
```

**See Also**

*Deflater Class* | *ICSharpCode.SharpZipLib.Zip.Compression Namespace*
ICSharpCode SharpZipLib Class Library
Sets the dictionary which should be used in the deflate process. This call is equivalent to

```csharp
setDictionary(dict, 0, dict.Length)
```

```csharp
public void SetDictionary(byte[] dictionary);
```

**Parameters**

- `dictionary` the dictionary.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidOperationException</code></td>
<td>if SetInput () or Deflate () were already called or another dictionary was already set.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
Deflater.SetDictionary Method (Byte[], Int32, Int32)

Sets the dictionary which should be used in the deflate process. The dictionary is a byte array containing strings that are likely to occur in the data which should be compressed. The dictionary is not stored in the compressed output, only a checksum. To decompress the output you need to supply the same dictionary again.

```csharp
public void SetDictionary(
    byte[] dictionary,
    int index,
    int count
);
```

**Parameters**

*dictionary*
- The dictionary data

*index*
- The index where dictionary information commences.

*count*
- The number of bytes in the dictionary.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>If SetInput () or Deflate() were already called or another dictionary was already set.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
Deflater.SetInput Method

Sets the data which should be compressed next. This should be only called when needsInput indicates that more input is needed. If you call setInputStream when needsInput() returns false, the previous input that is still pending will be thrown away. The given byte array should not be changed, before needsInput() returns true again. This call is equivalent to

```java
setInput(input, 0, input.length)
```

.  

Overload List

Sets the data which should be compressed next. This should be only called when needsInput indicates that more input is needed. If you call setInputStream when needsInput() returns false, the previous input that is still pending will be thrown away. The given byte array should not be changed, before needsInput() returns true again. This call is equivalent to

```java
setInput(input, 0, input.length)
```

.  

```java
public void SetInput(byte[]);
```

Sets the data which should be compressed next. This should be only called when needsInput indicates that more input is needed. The given byte array should not be changed, before needsInput() returns true again.

```java
public void SetInput(byte[],int,int);
```

See Also

ICSharpCode SharpZipLib Class Library
Deflater.SetInput Method (Byte[])  

Sets the data which should be compressed next. This should be only called when needsInput indicates that more input is needed. If you call setInput when needsInput() returns false, the previous input that is still pending will be thrown away. The given byte array should not be changed, before needsInput() returns true again. This call is equivalent to

```java
setInput(input, 0, input.length)
```

```java
public void setInput(byte[] input);
```

Parameters

- **input**  
  the buffer containing the input data.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>if the buffer was finished() or ended().</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
Sets the data which should be compressed next. This should be only called when needsInput indicates that more input is needed. The given byte array should not be changed, before needsInput() returns true again.

```csharp
public void SetInput(
    byte[] input,
    int offset,
    int count
);
```

### Parameters

- **input**
  the buffer containing the input data.

- **offset**
  the start of the data.

- **count**
  the number of data bytes of input.

### Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidOperationException</code></td>
<td>if the buffer was Finish()ed or if previous input is still pending.</td>
</tr>
</tbody>
</table>

### See Also

- [Deflater Class](#)
- [Deflater.SetInput Overload List](#)
ICSharpCode SharpZipLib Class Library
**Deflater.SetLevel Method**

Sets the compression level. There is no guarantee of the exact position of the change, but if you call this when needsInput is true the change of compression level will occur somewhere near before the end of the so far given input.

```csharp
public void SetLevel(
    int level
);
```

**Parameters**

- `level` the new compression level.

**See Also**

ICSharpCode SharpZipLib Class Library
Sets the compression strategy. Strategy is one of DEFAULT_STRATEGY, HUFFMAN_ONLY and FILTERED. For the exact position where the strategy is changed, the same as for SetLevel() applies.

```csharp
public void SetStrategy(
    DeflateStrategy strategy
);
```

Parameters

- **strategy**
  The new compression strategy.

See Also

ICSharpCode SharpZipLib Class Library
This class contains constants used for deflation.

For a list of all members of this type, see DeflaterConstants Members.

System.Object


public class DeflaterConstants

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

ICSharpCode SharpZipLib Class Library
**DeflaterConstants Members**

**DeflaterConstants overview**

### Public Static Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>COMPR_FUNC</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>DEBUGGING</code></td>
<td>Set to true to enable debugging</td>
</tr>
<tr>
<td><code>DEFAULT_MEM_LEVEL</code></td>
<td>Sets internal buffer sizes for Huffman encoding</td>
</tr>
<tr>
<td><code>DEFLATE_FAST</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>DEFLATE_SLOW</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>DEFLATE_STORED</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>DYN_TREES</code></td>
<td>Identifies dynamic tree in Zip file</td>
</tr>
<tr>
<td><code>GOOD_LENGTH</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>HASH_BITS</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>HASH_MASK</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>HASH_SHIFT</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>HASH_SIZE</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MAX_BLOCK_SIZE</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MAX_CHAIN</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MAX_DIST</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>MAX_LAZY</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>MAX_MATCH</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>MAX_WBITS</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>MIN_LOOKAHEAD</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>MIN_MATCH</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>NICE_LENGTH</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>PENDING_BUF_SIZE</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>PRESET_DICT</td>
<td>Header flag indicating a preset dictionary for deflation</td>
</tr>
<tr>
<td>STATIC_TREES</td>
<td>Identifies static tree in Zip file</td>
</tr>
<tr>
<td>STORED_BLOCK</td>
<td>Written to Zip file to identify a stored block</td>
</tr>
<tr>
<td>WMASK</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>WSIZE</td>
<td>Internal compression engine constant</td>
</tr>
</tbody>
</table>

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflaterConstants Constructor</td>
<td>Initializes a new instance of the DeflaterConstants class.</td>
</tr>
</tbody>
</table>

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### See Also

- [DeflaterConstants Class](#)
ICSharpCode SharpZipLib Class Library
DeflaterConstants Constructor

Initializes a new instance of the DeflaterConstants class.

public DeflaterConstants();

See Also

ICSharpCode SharpZipLib Class Library
**DeflaterConstants Fields**

The fields of the **DeflaterConstants** class are listed below. For a complete list of **DeflaterConstants** class members, see the **DeflaterConstants Members** topic.

**Public Static Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>COMPR_FUNC</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>DEBUGGING</code></td>
<td>Set to true to enable debugging</td>
</tr>
<tr>
<td><code>DEFAULT_MEM_LEVEL</code></td>
<td>Sets internal buffer sizes for Huffman encoding</td>
</tr>
<tr>
<td><code>DEFLATE_FAST</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>DEFLATE_SLOW</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>DEFLATE_STORED</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>DYN_TREES</code></td>
<td>Identifies dynamic tree in Zip file</td>
</tr>
<tr>
<td><code>GOOD_LENGTH</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>HASH_BITS</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>HASH_MASK</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>HASH_SHIFT</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>HASH_SIZE</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MAX_BLOCK_SIZE</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MAX_CHAIN</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>Macro</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>MAX_DIST</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MAX_LAZY</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MAX_MATCH</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MAX_WBITS</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MIN_LOOKAHEAD</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>MIN_MATCH</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>NICE_LENGTH</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>PENDING_BUF_SIZE</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>PRESET_DICT</code></td>
<td>Header flag indicating a preset dictionary for deflation</td>
</tr>
<tr>
<td><code>STATIC_TREES</code></td>
<td>Identifies static tree in Zip file</td>
</tr>
<tr>
<td><code>STORED_BLOCK</code></td>
<td>Written to Zip file to identify a stored block</td>
</tr>
<tr>
<td><code>WMASK</code></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><code>FSIZE</code></td>
<td>Internal compression engine constant</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
DeflaterConstants.COMPR_FUNC Field

Internal compression engine constant

```java
public static int[] COMPR_FUNC;
```

See Also

ICSharpCode SharpZipLib Class Library
Set to true to enable debugging

```csharp
public const bool DEBUGGING = False;
```

See Also

- [DeflaterConstants Class](#)
ICSharpCode SharpZipLib Class Library
DeflaterConstants.DEFAULT_MEM_LEVEL Field

Sets internal buffer sizes for Huffman encoding

```csharp
public const int DEFAULT_MEM_LEVEL = 8;
```

See Also

- [DeflaterConstants Class](#)
ICSharpCode SharpZipLib Class Library
**DeflaterConstants.DEFLATE_FAST Field**

Internal compression engine constant

```
public const int DEFLATE_FAST = 1;
```

See Also

ICSharpCode SharpZipLib Class Library
DeflaterConstants.DEFLATE_SLOW Field

Internal compression engine constant

```csharp
public const int DEFLATE_SLOW = 2;
```

See Also

[DeflaterConstants Class](#)
ICSharpCode SharpZipLib Class Library
DeflaterConstants.DEFLATE_STORED Field

Internal compression engine constant

```
public const int DEFLATE_STORED = 0;
```

See Also

DeflaterConstants Class
ICSharpCode SharpZipLib Class Library
**DeflaterConstants.DYN_TREES Field**

Identifies dynamic tree in Zip file

```
public const int DYN_TREES = 2;
```

See Also

- [DeflaterConstants Class](#)
ICSharpCode SharpZipLib Class Library
DeflaterConstants.GOOD_LENGTH Field

Internal compression engine constant

```
public static int[] GOOD_LENGTH;
```

See Also

- DeflaterConstants Class
ICSharpCode SharpZipLib Class Library
**DeflaterConstants.HASH_BITS Field**

Internal compression engine constant

```csharp
public const int HASH_BITS = 15;
```

**See Also**

- [DeflaterConstants Class](#)
**DeflaterConstants.HASH_MASK Field**

Internal compression engine constant

```csharp
public const int HASH_MASK = 32767;
```

See Also

ICSharpCode SharpZipLib Class Library
DeflaterConstants.HASH_SHIFT Field

Internal compression engine constant

```csharp
public const int HASH_SHIFT = 5;
```

See Also

- [DeflaterConstants Class](#)
ICSharpCode SharpZipLib Class Library
**DeflaterConstants.HASH_SIZE Field**

Internal compression engine constant

```csharp
public const int HASH_SIZE = 32768;
```

See Also

- [DeflaterConstants Class](#)
ICSharpCode SharpZipLib Class Library
DeflaterConstants.MAX_BLOCK_SIZE Field

Internal compression engine constant

public static int MAX_BLOCK_SIZE;

See Also

DeflaterConstants Class
ICSharpCode SharpZipLib Class Library
DeflaterConstants.MAX_CHAIN Field

Internal compression engine constant

```java
public static int[] MAX_CHAIN;
```

See Also

ICSharpCode SharpZipLib Class Library
DeflaterConstants.MAX_DIST Field

Internal compression engine constant

```csharp
public const int MAX_DIST = 32506;
```

See Also

- DeflaterConstants Class
ICSharpCode SharpZipLib Class Library
**DeflaterConstants.MAX_LAZY Field**

Internal compression engine constant

```
public static int[] MAX_LAZY;
```

See Also

DeflaterConstants.MAX_MATCH Field

Internal compression engine constant

```csharp
public const int MAX_MATCH = 258;
```

See Also

- DeflaterConstants Class
**ICSharpCode SharpZipLib Class Library**
DeflaterConstants.MAX_WBITS Field

Internal compression engine constant

```
public const int MAX_WBITS = 15;
```

See Also

DeflaterConstants Class
ICSharpCode SharpZipLib Class Library
Internal compression engine constant

```
public const int MIN_LOOKAHEAD = 262;
```

See Also

- DeflaterConstants Class
DeflaterConstants.MIN_MATCH Field

Internal compression engine constant

```csharp
public const int MIN_MATCH = 3;
```

See Also

- DeflaterConstants Class
DeflaterConstants.NICE_LENGTH Field

Internal compression engine constant

```
public static int[] NICE_LENGTH;
```

See Also

- DeflaterConstants Class
ICSharpCode SharpZipLib Class Library
DeflaterConstants.PENDING_BUF_SIZE Field

Internal compression engine constant

```
public const int PENDING_BUF_SIZE = 65536;
```

See Also

DeflaterConstants Class
ICSharpCode SharpZipLib Class Library
**DeflaterConstants.PRESET_DICT Field**

Header flag indicating a preset dictionary for deflation

```
public const int PRESET_DICT = 32;
```

See Also

- [DeflaterConstants Class](#)
DeflaterConstants.STATIC_TREES Field

Identifies static tree in Zip file

```java
public const int STATIC_TREES = 1;
```

See Also

ICSharpCode SharpZipLib Class Library
**DeflaterConstants.STORED_BLOCK Field**

Written to Zip file to identify a stored block

```csharp
public const int STORED_BLOCK = 0;
```

See Also

- [DeflaterConstants Class](#)
ICSharpCode SharpZipLib Class Library
DeflaterConstants.WMASK Field

Internal compression engine constant

```csharp
public const int WMASK = 32767;
```

See Also

- DeflaterConstants Class
DeflaterConstants.WSIZE Field

Internal compression engine constant

```
public const int WSIZE = 32768;
```

See Also

DeflaterEngine Class

Low level compression engine for deflate algorithm which uses a 32K sliding window with secondary compression from Huffman/Shannon-Fano codes.

For a list of all members of this type, see DeflaterEngine Members.

System.Object

public class DeflaterEngine : DeflaterConstants

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

ICSharpCode SharpZipLib Class Library
## DeflaterEngine Members

### DeflaterEngine overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflaterEngine Constructor</td>
<td>Construct instance with pending buffer</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adler</td>
<td>Get current value of Adler checksum</td>
</tr>
<tr>
<td>Strategy</td>
<td>Get/set the deflate strategy</td>
</tr>
<tr>
<td>TotalIn</td>
<td>Total data processed</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deflate</td>
<td>Deflate drives actual compression of data</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>FillWindow</td>
<td>Fill the window</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>NeedsInput</td>
<td>Return true if input is needed via SetInput</td>
</tr>
<tr>
<td>Reset</td>
<td>Reset internal state</td>
</tr>
<tr>
<td>ResetAdler</td>
<td>Reset Adler checksum</td>
</tr>
<tr>
<td>SetDictionary</td>
<td>Set compression dictionary</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>SetInput</strong></td>
<td>Sets input data to be deflated. Should only be called when <code>NeedsInput()</code> returns true.</td>
</tr>
<tr>
<td><strong>SetLevel</strong></td>
<td>Set the deflate level (0-9)</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

**See Also**

- [DeflaterEngine Class](https://icsharpcode.net/SharpZipLib/Zip.Compression)
ICSharpCode SharpZipLib Class Library
**DeflaterEngine Constructor**

Construct instance with pending buffer

```csharp
public DeflaterEngine(
    DeflaterPending pending
);
```

**Parameters**

*pending*

Pending buffer to use

**See Also**

ICSharpCode SharpZipLib Class Library
DeflaterEngine Properties

The properties of the DeflaterEngine class are listed below. For a complete list of DeflaterEngine class members, see the DeflaterEngine Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adler</td>
<td>Get current value of Adler checksum</td>
</tr>
<tr>
<td>Strategy</td>
<td>Get/set the deflate strategy</td>
</tr>
<tr>
<td>TotalIn</td>
<td>Total data processed</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
Get current value of Adler checksum

```csharp
public int Adler {get;}
```

See Also

- [DeflaterEngine Class](#)
ICSharpCode SharpZipLib Class Library
DeflaterEngine.Strategy Property

Get/set the **deflate strategy**

    public DeflateStrategy Strategy {get; set;}

See Also

ICSharpCode SharpZipLib Class Library
DeflaterEngine.TotalIn Property

Total data processed

```csharp
public int TotalIn {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
# DeflaterEngine Methods

The methods of the **DeflaterEngine** class are listed below. For a complete list of **DeflaterEngine** class members, see the [DeflaterEngine Members](#) topic.

## Public Instance Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deflate</strong></td>
<td>Deflate drives actual compression of data</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>FillWindow</strong></td>
<td>Fill the window</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>NeedsInput</strong></td>
<td>Return true if input is needed via <strong>SetInput</strong></td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Reset internal state</td>
</tr>
<tr>
<td><strong>ResetAdler</strong></td>
<td>Reset Adler checksum</td>
</tr>
<tr>
<td><strong>SetDictionary</strong></td>
<td>Set compression dictionary</td>
</tr>
<tr>
<td><strong>SetInput</strong></td>
<td>Sets input data to be deflated. Should only be called when <strong>NeedsInput()</strong> returns true</td>
</tr>
<tr>
<td><strong>SetLevel</strong></td>
<td>Set the deflate level (0-9)</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>
## Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✨ <strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>✨ <strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### See Also

DeflaterEngine.Deflate Method

Deflate drives actual compression of data

```csharp
public bool Deflate(bool flush, bool finish);
```

Return Value

Returns true if progress has been made.

See Also

DeflaterEngine.FillWindow Method

Fill the window

```
public void FillWindow();
```

See Also

DeflaterEngine.NeedsInput Method

Return true if input is needed via SetInput

```csharp
public bool NeedsInput();
```

See Also

ICSharpCode SharpZipLib Class Library
DeflaterEngine.Reset Method

Reset internal state

```csharp
public void Reset();
```

See Also

ICSharpCode SharpZipLib Class Library
DeflaterEngine.ResetAdler Method

Reset Adler checksum

```csharp
public void ResetAdler();
```

See Also

ICSharpCode SharpZipLib Class Library
Set compression dictionary

```csharp
public void SetDictionary(
    byte[] buffer,
    int offset,
    int length
);
```

See Also

ICSharpCode SharpZipLib Class Library
DeflaterEngine.SetInput Method

Sets input data to be deflated. Should only be called when NeedsInput() returns true

```csharp
public void SetInput(byte[] buffer, int offset, int count);
```

Parameters

- **buffer**
  The buffer containing input data.

- **offset**
  The offset of the first byte of data.

- **count**
  The number of bytes of data to use as input.

See Also

ICSharpCode SharpZipLib Class Library
# DeflaterEngine.SetLevel Method

Set the deflate level (0-9)

```csharp
public void SetLevel(int level);
```

## Parameters

- **level**
  - The value to set the level to.

## See Also

- [DeflaterEngine Class](#)
ICSharpCode SharpZipLib Class Library
This is the DeflaterHuffman class. This class is *not* thread safe. This is inherent in the API, due to the split of Deflate and SetInput. author of the original java version : Jochen Hoenicke

For a list of all members of this type, see DeflaterHuffman Members.

System.Object


---

###public class DeflaterHuffman

####Thread Safety

Public static (*Shared in Visual Basic*) members of this type are safe for multithreaded operations. Instance members are *not* guaranteed to be thread-safe.

####Requirements

**Namespace:** ISharpCode.SharpZipLib.Zip.Compression

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

####See Also

ICSharpCode SharpZipLib Class Library
# DeflaterHuffman Members

## DeflaterHuffman overview

### Public Static Methods

| ❀ S BitReverse | Reverse the bits of a 16 bit value. |

### Public Instance Constructors

| ❀ DeflaterHuffman Constructor | Construct instance with pending buffer |

### Public Instance Fields

| ❀ pending | Pending buffer to use |

### Public Instance Methods

<p>| ❀ CompressBlock | Compress current buffer writing data to pending buffer |
| ❀ Equals (inherited from Object) | Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>. |
| ❀ FlushBlock | Flush block to output with compression |
| ❀ FlushStoredBlock | Flush block to output with no compression |
| ❀ GetHashCode (inherited from Object) | Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table. |
| ❀ GetType (inherited from Object) | Gets the <strong>Type</strong> of the current instance. |
| ❀ IsFull | Get value indicating if internal buffer is full |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reset</strong></td>
<td>Reset internal state</td>
</tr>
<tr>
<td><strong>SendAllTrees</strong></td>
<td>Write all trees to pending buffer</td>
</tr>
<tr>
<td><strong>TallyDist</strong></td>
<td>Add distance code and length to literal and distance trees</td>
</tr>
<tr>
<td><strong>TallyLit</strong></td>
<td>Add literal to buffer</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
DeflaterHuffman Constructor

Construct instance with pending buffer

```csharp
public DeflaterHuffman(DeflaterPending pending);
```

Parameters

`pending`

Pending buffer to use

See Also

ICSharpCode SharpZipLib Class Library
The fields of the **DeflaterHuffman** class are listed below. For a complete list of **DeflaterHuffman** class members, see the [DeflaterHuffman Members](#) topic.

### Public Instance Fields

| pending | Pending buffer to use |

### See Also

ICSharpCode SharpZipLib Class Library
DeflaterHuffman.pending Field

Pending buffer to use

```csharp
public DeflaterPending pending;
```

See Also

ICSharpCode SharpZipLib Class Library
### DeflaterHuffman Methods

The methods of the `DeflaterHuffman` class are listed below. For a complete list of `DeflaterHuffman` class members, see the [DeflaterHuffman Members](#) topic.

#### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitReverse</td>
<td>Reverse the bits of a 16 bit value.</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompressBlock</td>
<td>Compress current buffer writing data to pending buffer</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>FlushBlock</td>
<td>Flush block to output with compression</td>
</tr>
<tr>
<td>FlushStoredBlock</td>
<td>Flush block to output with no compression</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>IsFull</td>
<td>Get value indicating if internal buffer is full</td>
</tr>
<tr>
<td>Reset</td>
<td>Reset internal state</td>
</tr>
<tr>
<td>SendAllTrees</td>
<td>Write all trees to pending buffer</td>
</tr>
<tr>
<td>TallyDist</td>
<td>Add distance code and length to literal and distance trees</td>
</tr>
<tr>
<td>TallyLit</td>
<td>Add literal to buffer</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
Reverse the bits of a 16 bit value.

```csharp
public static short BitReverse(int toReverse);
```

**Parameters**

toReverse

Value to reverse bits

**Return Value**

Value with bits reversed

**See Also**

ICSharpCode SharpZipLib Class Library
DeflaterHuffman.CompressBlock Method

Compress current buffer writing data to pending buffer

```csharp
public void CompressBlock();
```

See Also

ICSharpCode SharpZipLib Class Library
DeflaterHuffman.FlushBlock Method

Flush block to output with compression

```csharp
public void FlushBlock(  
    byte[] stored,  
    int storedOffset,  
    int storedLength,  
    bool lastBlock
);
```

Parameters

- `stored`  
  Data to flush

- `storedOffset`  
  Index of first byte to flush

- `storedLength`  
  Count of bytes to flush

- `lastBlock`  
  True if this is the last block

See Also

ICSharpCode SharpZipLib Class Library
Flush block to output with no compression

```csharp
public void FlushStoredBlock(
    byte[] stored,
    int storedOffset,
    int storedLength,
    bool lastBlock
);
```

**Parameters**

- `stored`
  - Data to write

- `storedOffset`
  - Index of first byte to write

- `storedLength`
  - Count of bytes to write

- `lastBlock`
  - True if this is the last block

**See Also**

- [DeflaterHuffman Class](#)
ICSharpCode SharpZipLib Class Library
**DeflaterHuffman.IsFull Method**

Get value indicating if internal buffer is full

```csharp
public bool IsFull();
```

**Return Value**

true if buffer is full

**See Also**

ICSharpCode SharpZipLib Class Library
**DeflaterHuffman.Reset Method**

Reset internal state

```csharp
public void Reset();
```

See Also

- [DeflaterHuffman Class](#)
DeflaterHuffman.SendAllTrees Method

Write all trees to pending buffer

```csharp
public void SendAllTrees(int blTreeCodes);
```

See Also

DeflaterHuffman Class | ICSsharpCode.SharpZipLib.Zip.Compression Namespace
ICSharpCode SharpZipLib Class Library
DeflaterHuffman.TallyDist Method

Add distance code and length to literal and distance trees

```csharp
public bool TallyDist(
    int distance,
    int length
);
```

Parameters

*distance*
Distance code

*length*
Length

Return Value

Value indicating if internal buffer is full

See Also

ICSharpCode SharpZipLib Class Library
DeflaterHuffman.TallyLit Method

Add literal to buffer

```csharp
public bool TallyLit(int literal);
```

Parameters

- `literal` - Literal value to add to buffer.

Return Value

Value indicating internal buffer is full

See Also

ICSharpCode SharpZipLib Class Library
DeflaterPending Class

This class stores the pending output of the Deflater. author of the original java version : Jochen Hoenicke

For a list of all members of this type, see DeflaterPending Members.

System.Object


public class DeflaterPending : PendingBuffer

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

DeflaterPending Members |
ICSharpCode SharpZipLib Class Library
# DeflaterPending Members

## DeflaterPending overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflaterPending Constructor</td>
<td>Construct instance with default buffer size</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitCount</td>
<td>The number of bits written to the buffer</td>
</tr>
<tr>
<td>IsFlushed</td>
<td>Indicates if buffer has been flushed</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlignToByte</td>
<td>Align internal buffer on a byte boundary</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>Flush</td>
<td>Flushes the pending buffer into the given output array. If the output array is to small, only a partial flush is done.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>Reset</td>
<td>Clear internal state/buffers</td>
</tr>
<tr>
<td>ToByteArray</td>
<td>Convert internal buffer to byte array. Buffer is empty on</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>WriteBits</strong> (inherited from PendingBuffer)</td>
<td>Write bits to internal buffer</td>
</tr>
<tr>
<td><strong>WriteBlock</strong> (inherited from PendingBuffer)</td>
<td>Write a block of data to buffer</td>
</tr>
<tr>
<td><strong>WriteByte</strong> (inherited from PendingBuffer)</td>
<td>Write a byte to buffer</td>
</tr>
<tr>
<td><strong>WriteInt</strong> (inherited from PendingBuffer)</td>
<td>Write an integer LSB first</td>
</tr>
<tr>
<td><strong>WriteShort</strong> (inherited from PendingBuffer)</td>
<td>Write a short value to buffer LSB first</td>
</tr>
<tr>
<td><strong>WriteShortMSB</strong> (inherited from PendingBuffer)</td>
<td>Write a short value to internal buffer most significant byte first</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### See Also

ICSharpCode SharpZipLib Class Library
DeflaterPending Constructor

Construct instance with default buffer size

```java
public DeflaterPending();
```

See Also

- [DeflaterPending Class](#)
ICSharpCode SharpZipLib Class Library
## DeflateStrategy Enumeration

Strategies for deflater

```csharp
public enum DeflateStrategy
```

### Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>The default strategy</td>
</tr>
<tr>
<td>Filtered</td>
<td>This strategy will only allow longer string repetitions. It is useful for random data with a small character set.</td>
</tr>
<tr>
<td>HuffmanOnly</td>
<td>This strategy will not look for string repetitions at all. It only encodes with Huffman trees (which means, that more common characters get a smaller encoding).</td>
</tr>
</tbody>
</table>

### Requirements

**Namespace:** [ICSharpCode.SharpZipLib.Zip.Compression](https://icsharpcode.net/SharpZipLib/)

**Assembly:** `ICSharpCode.SharpZipLib.dll`

### See Also


ICSharpCode SharpZipLib Class Library
Inflater is used to decompress data that has been compressed according to the "deflate" standard described in rfc1951. By default Zlib (rfc1950) headers and footers are expected in the input. You can use constructor

```csharp
public Inflater(bool noHeader)
```

passing true if there is no Zlib header information. The usage is as following. First you have to set some input with

```csharp
SetInput()
```

, then Inflate() it. If inflate doesn't inflate any bytes there may be three reasons:

- IsNeedingInput() returns true because the input buffer is empty. You have to provide more input with

```csharp
SetInput()
```

. NOTE: IsNeedingInput() also returns true when, the stream is finished.

- IsNeedingDictionary() returns true, you have to provide a preset dictionary with

```csharp
SetDictionary()
```

.

- IsFinished returns true, the inflator has finished.

Once the first output byte is produced, a dictionary will not be needed at a later stage. author of the original java version : John Leuner, Jochen Hoenicke

For a list of all members of this type, see Inflater Members.

System.Object

```csharp
public class Inflater
```
Thread Safety

Public static (**Shared** in Visual Basic) members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

Requirements

**Namespace:** IČSharpCode.SharpZipLib.Zip.Compression

**Assembly:** IČSharpCode.SharpZipLib (in IČSharpCode.SharpZipLib.dll)

See Also


Namespace
ICSharpCode SharpZipLib Class Library
## Inflater Members

### Inflater overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Inflater</code></td>
<td>Overloaded. Initializes a new instance of the Inflater class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Adler</code></td>
<td>Gets the adler checksum. This is either the checksum of all uncompressed bytes returned by inflate(), or if needsDictionary() returns true (and thus no output was yet produced) this is the adler checksum of the expected dictionary.</td>
</tr>
<tr>
<td><code>IsFinished</code></td>
<td>Returns true, if the inflater has finished. This means, that no input is needed and no output can be produced.</td>
</tr>
<tr>
<td><code>IsNeedingDictionary</code></td>
<td>Returns true, if a preset dictionary is needed to inflate the input.</td>
</tr>
<tr>
<td><code>IsNeedingInput</code></td>
<td>Returns true, if the input buffer is empty. You should then call setInput(). NOTE: This method also returns true when the stream is finished.</td>
</tr>
<tr>
<td><code>RemainingInput</code></td>
<td>Gets the number of unprocessed input bytes. Useful, if the end of the stream is reached and you want to further process the bytes after the deflate stream.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>TotalIn</strong></td>
<td>Gets the total number of processed compressed input bytes.</td>
</tr>
<tr>
<td><strong>TotalOut</strong></td>
<td>Gets the total number of output bytes returned by Inflate().</td>
</tr>
</tbody>
</table>

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>Inflate</strong></td>
<td>Overloaded. Inflates the compressed stream to the output buffer. If this returns 0, you should check, whether IsNeedingDictionary(), IsNeedingInput() or IsFinished() returns true, to determine why no further output is produced.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets the inflater so that a new stream can be decompressed. All pending input and output will be discarded.</td>
</tr>
<tr>
<td><strong>SetDictionary</strong></td>
<td>Overloaded. Sets the preset dictionary. This should only be called, if needsDictionary() returns true and it should set the same dictionary, that was used for deflating. The getAdler() function returns the checksum.</td>
</tr>
</tbody>
</table>
of the dictionary needed.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![SetInput](inherited from Object)</td>
<td>Overloaded. Sets the input. This should only be called, if needsInput() returns true.</td>
</tr>
<tr>
<td>![ToString](inherited from Object)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

Protected Instance Methods

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Finalize](inherited from Object)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>![MemberwiseClone](inherited from Object)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

See Also

- [Inflater Class](inherited from Object)
- [ICSharpCode.SharpZipLib.Zip.Compression Namespace](inherited from Object)
ICSharpCode SharpZipLib Class Library
Inflater Constructor

Creates a new inflater or RFC1951 decompressor RFC1950/Zlib headers and footers will be expected in the input data

Overload List

Creates a new inflater or RFC1951 decompressor RFC1950/Zlib headers and footers will be expected in the input data

public Inflater();

Creates a new inflater.

public Inflater(bool);

See Also

ICSharpCode SharpZipLib Class Library
Inflater Constructor ()

Creates a new inflater or RFC1951 decompressor RFC1950/Zlib headers and footers will be expected in the input data

```plaintext
public Inflater();
```

See Also

- Inflater Class
- Inflater Constructor Overload List
Inflater Constructor (Boolean)

Creates a new inflater.

```csharp
public Inflater(
    bool noHeader
);
```

Parameters

`noHeader`  
True if no RFC1950/Zlib header and footer fields are expected in the input data. This is used for GZIPed/Zipped input. For compatibility with Sun JDK you should provide one byte of input more than needed in this case.

See Also

ICSharpCode SharpZipLib Class Library
### Inflater Properties

The properties of the **Inflater** class are listed below. For a complete list of **Inflater** class members, see the **Inflater Members** topic.

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adler</strong></td>
<td>Gets the adler checksum. This is either the checksum of all uncompressed bytes returned by <code>inflate()</code>, or if <code>needsDictionary()</code> returns true (and thus no output was yet produced) this is the adler checksum of the expected dictionary.</td>
</tr>
<tr>
<td><strong>IsFinished</strong></td>
<td>Returns true, if the inflater has finished. This means, that no input is needed and no output can be produced.</td>
</tr>
<tr>
<td><strong>IsNeedingDictionary</strong></td>
<td>Returns true, if a preset dictionary is needed to inflate the input.</td>
</tr>
<tr>
<td><strong>IsNeedingInput</strong></td>
<td>Returns true, if the input buffer is empty. You should then call <code>setInput()</code>. NOTE: This method also returns true when the stream is finished.</td>
</tr>
<tr>
<td><strong>RemainingInput</strong></td>
<td>Gets the number of unprocessed input bytes. Useful, if the end of the stream is reached and you want to further process the bytes after the deflate stream.</td>
</tr>
<tr>
<td><strong>TotalIn</strong></td>
<td>Gets the total number of processed compressed input bytes.</td>
</tr>
<tr>
<td><strong>TotalOut</strong></td>
<td>Gets the total number of output bytes returned by Inflate().</td>
</tr>
</tbody>
</table>

See Also

**ICSharpCode SharpZipLib Class Library**
**Inflater.Adler Property**

Gets the adler checksum. This is either the checksum of all uncompressed bytes returned by inflate(), or if needsDictionary() returns true (and thus no output was yet produced) this is the adler checksum of the expected dictionary.

```csharp
public int Adler {get;}
```

**See Also**

ICSharpCode SharpZipLib Class Library
Inflater.IsFinished Property

Returns true, if the inflater has finished. This means, that no input is needed and no output can be produced.

```csharp
public bool IsFinished {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
Inflater.IsNeedingDictionary Property

Returns true, if a preset dictionary is needed to inflate the input.

```csharp
public bool IsNeedingDictionary {get;}
```

See Also

- Inflater Class
ICSharpCode SharpZipLib Class Library
Inflater.IsNeedingInput Property

Returns true, if the input buffer is empty. You should then call setInput(). NOTE: This method also returns true when the stream is finished.

```csharp
public bool IsNeedingInput {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
Gets the number of unprocessed input bytes. Useful, if the end of the stream is reached and you want to further process the bytes after the deflate stream.

```csharp
public int RemainingInput {get;}
```

See Also

- [Inflater Class](#)
ICSharpCode SharpZipLib Class Library
**Inflater.TotalIn Property**

Gets the total number of processed compressed input bytes.

```csharp
public int TotalIn {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
Inflater.TotalOut Property

Gets the total number of output bytes returned by Inflate().

```csharp
public int TotalOut {get;}
```

See Also

- Inflater Class
- Namespace
ICSharpCode SharpZipLib Class Library
The methods of the **Inflater** class are listed below. For a complete list of **Inflater** class members, see the [Inflater Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>Inflate</strong></td>
<td>Overloaded. Inflates the compressed stream to the output buffer. If this returns 0, you should check, whether IsNeedingDictionary(), IsNeedingInput() or IsFinished() returns true, to determine why no further output is produced.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets the inflater so that a new stream can be decompressed. All pending input and output will be discarded.</td>
</tr>
<tr>
<td><strong>SetDictionary</strong></td>
<td>Overloaded. Sets the preset dictionary. This should only be called, if needsDictionary() returns true and it should set the same dictionary, that was used for deflating. The getAdler() function returns the checksum of the dictionary needed.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SetInput</strong></td>
<td>Overloaded. Sets the input. This should only be called, if needsInput() returns true.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

**See Also**

- [Inflater Class](#)
ICSharpCode SharpZipLib Class Library
Inflates the compressed stream to the output buffer. If this returns 0, you should check, whether IsNeedingDictionary(), IsNeedingInput() or IsFinished() returns true, to determine why no further output is produced.

**Overload List**

Inflates the compressed stream to the output buffer. If this returns 0, you should check, whether IsNeedingDictionary(), IsNeedingInput() or IsFinished() returns true, to determine why no further output is produced.

```csharp
public int Inflate(byte[]):
```

Inflates the compressed stream to the output buffer. If this returns 0, you should check, whether needsDictionary(), needsInput() or finished() returns true, to determine why no further output is produced.

```csharp
public int Inflate(byte[], int,int):
```

**See Also**

ICSharpCode SharpZipLib Class Library
Inflaters.Inflate Method (Byte[])  

Inflates the compressed stream to the output buffer. If this returns 0, you should check, whether IsNeedingDictionary(), IsNeedingInput() or IsFinished() returns true, to determine why no further output is produced.

```csharp
public int Inflate(
    byte[] buffer
);
```

Parameters

- **buffer**
  - the output buffer.

Return Value

The number of bytes written to the buffer, 0 if no further output can be produced.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>if buffer has length 0.</td>
</tr>
<tr>
<td>FormatException</td>
<td>if deflated stream is invalid.</td>
</tr>
</tbody>
</table>

See Also

- Namespace      | Inflater.Inflate Overload List
Inflates the compressed stream to the output buffer. If this returns 0, you should check, whether needsDictionary(), needsInput() or finished() returns true, to determine why no further output is produced.

```csharp
public int Inflate(
    byte[] buffer,
    int offset,
    int count
);
```

**Parameters**

- **buffer**
  - the output buffer.

- **offset**
  - the offset in buffer where storing starts.

- **count**
  - the maximum number of bytes to output.

**Return Value**

- the number of bytes written to the buffer, 0 if no further output can be produced.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>if count is less than 0.</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>if the index and / or count are wrong.</td>
</tr>
<tr>
<td>FormatException</td>
<td>if deflated stream is invalid.</td>
</tr>
</tbody>
</table>

**See Also**

Namespace | Inflater.Inflate Overload List
ICSharpCode SharpZipLib Class Library
**Inflater.Reset Method**

Resets the inflater so that a new stream can be decompressed. All pending input and output will be discarded.

```csharp
public void Reset();
```

**See Also**

ICSharpCode SharpZipLib Class Library
Inflater.SetDictionary Method

Sets the preset dictionary. This should only be called, if needsDictionary() returns true and it should set the same dictionary, that was used for deflating. The getAdler() function returns the checksum of the dictionary needed.

Overload List

Sets the preset dictionary. This should only be called, if needsDictionary() returns true and it should set the same dictionary, that was used for deflating. The getAdler() function returns the checksum of the dictionary needed.

public void SetDictionary(byte[]);

Sets the preset dictionary. This should only be called, if needsDictionary() returns true and it should set the same dictionary, that was used for deflating. The getAdler() function returns the checksum of the dictionary needed.

public void SetDictionary(byte[],int,int);

See Also

ICSharpCode SharpZipLib Class Library
Inflater.SetDictionary Method (Byte[])

Sets the preset dictionary. This should only be called, if needsDictionary() returns true and it should set the same dictionary, that was used for deflating. The getAdler() function returns the checksum of the dictionary needed.

```csharp
public void SetDictionary(
    byte[] buffer
);
```

Parameters

- `buffer`
  - The dictionary.

See Also

- Inflater Class
- Inflater.SetDictionary Overload List
ICSharpCode SharpZipLib Class Library
Inflater.SetDictionary Method (Byte[], Int32, Int32)

Sets the preset dictionary. This should only be called, if needsDictionary() returns true and it should set the same dictionary, that was used for deflating. The getAdler() function returns the checksum of the dictionary needed.

```csharp
public void SetDictionary(
    byte[] buffer,
    int index,
    int count
);
```

Parameters

- `buffer` The dictionary.
- `index` The index into buffer where the dictionary starts.
- `count` The number of bytes in the dictionary.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>No dictionary is needed.</td>
</tr>
<tr>
<td>SharpZipBaseException</td>
<td>The adler checksum for the buffer is invalid</td>
</tr>
</tbody>
</table>

See Also

- Inflater Class
- Inflater.SetDictionary Overload List
ICSharpCode SharpZipLib Class Library
Inflater.SetInput Method

Sets the input. This should only be called, if needsInput() returns true.

Overload List

Sets the input. This should only be called, if needsInput() returns true.

public void SetInput(byte[]);

Sets the input. This should only be called, if needsInput() returns true.

public void SetInput(byte[],int,int);

See Also

ICSharpCode SharpZipLib Class Library
**Inflater.SetInput Method (Byte[])**

Sets the input. This should only be called, if needsInput() returns true.

```csharp
public void SetInput(byte[] buffer);
```

**Parameters**

- `buffer`  
  The input.

**See Also**

ICSharpCode SharpZipLib Class Library
Inflater.SetInput Method (Byte[], Int32, Int32)

Sets the input. This should only be called, if needsInput() returns true.

```csharp
public void SetInput(
    byte[] buffer,
    int index,
    int count
);
```

Parameters

- **buffer**
  - The source of input data

- **index**
  - The index into buffer where the input starts.

- **count**
  - The number of bytes of input to use.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>No input is needed.</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>The index and/or count are wrong.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
Huffman tree used for inflation

For a list of all members of this type, see InflaterHuffmanTree Members.

System.Object

public class InflaterHuffmanTree

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

ICSharpCodeSharpZipLibClassLibrary
### InflaterHuffmanTree Members

#### InflaterHuffmanTree overview

### Public Static Fields

<table>
<thead>
<tr>
<th>✡️ defDistTree</th>
<th>Distance tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>✡️ defLitLenTree</td>
<td>Literal length tree</td>
</tr>
</tbody>
</table>

### Public Instance Constructors

<table>
<thead>
<tr>
<th>✡️ InflaterHuffmanTree Constructor</th>
<th>Constructs a Huffman tree from the array of code lengths.</th>
</tr>
</thead>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>✡️ Equals (inherited from Object)</th>
<th>Determines whether the specified Object is equal to the current Object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>✡️ GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>✡️ GetSymbol</td>
<td>Reads the next symbol from input. The symbol is encoded using the huffman tree.</td>
</tr>
<tr>
<td>✡️ GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>✡️ ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>✡️ Finalize (inherited from Object)</th>
<th>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

- **InflaterHuffmanTree Class**
ICSharpCode SharpZipLib Class Library
**InflaterHuffmanTree Constructor**

Constructs a Huffman tree from the array of code lengths.

```java
public InflaterHuffmanTree(
    byte[] codeLengths
);
```

**Parameters**

*codeLengths*
the array of code lengths

**See Also**

- [InflaterHuffmanTree Class](#)
ICSharpCode SharpZipLib Class Library
InflaterHuffmanTree Fields

The fields of the InflaterHuffmanTree class are listed below. For a complete list of InflaterHuffmanTree class members, see the InflaterHuffmanTree Members topic.

Public Static Fields

| $ defDistTree | Distance tree |
| $ defLitLenTree | Literal length tree |

See Also

InflaterHuffmanTree Class | ISharpCode.SharpZipLib.Zip.Compression Namespace
ICSharpCode SharpZipLib Class Library
InflaterHuffmanTree.defDistTree Field

Distance tree

`public static InflaterHuffmanTree defDistTree`

See Also

InflaterHuffmanTree Class | ISharpCode.SharpZipLib.Zip.Compression Namespace
ICSharpCode SharpZipLib Class Library
InflaterHuffmanTree.defLitLenTree Field

Literal length tree

public static InflaterHuffmanTree defLitLenTree

See Also

InflaterHuffmanTree Class | ICSHarpCode.SharpZipLib.Zip.Compression Namespace
InflaterHuffmanTree Methods

The methods of the InflaterHuffmanTree class are listed below. For a complete list of InflaterHuffmanTree class members, see the InflaterHuffmanTree Members topic.

Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>➡️ Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>➡️ GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>➡️ GetSymbol</td>
<td>Reads the next symbol from input. The symbol is encoded using the huffman tree.</td>
</tr>
<tr>
<td>➡️ GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>➡️ ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>➡️ Finalize (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>➡️ MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

See Also

InflaterHuffmanTree Class | ISharpCode.SharpZipLib.Zip.Compression Namespace
ICSharpCode SharpZipLib Class Library
InflaterHuffmanTree.GetSymbol Method

Reads the next symbol from input. The symbol is encoded using the huffman tree.

```csharp
public int GetSymbol(
    Stream Manipulator input);
```

Parameters

`input`
input the input source.

Return Value

do the next symbol, or -1 if not enough input is available.

See Also

InflaterHuffmanTree Class | ISharpCode.SharpZipLib.Zip.Compression Namespace
ICSharpCode SharpZipLib Class Library
PendingBuffer Class

This class is general purpose class for writing data to a buffer. It allows you to write bits as well as bytes Based on DeflaterPending.java author of the original java version : Jochen Hoenicke

For a list of all members of this type, see PendingBuffer Members.

**System.Object**


```
public class PendingBuffer
```

**Thread Safety**

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** IICSharpCode.SharpZipLib.Zip.Compression

**Assembly:** IICSharpCode.SharpZipLib (in IICSharpCode.SharpZipLib.dll)

**See Also**

# PendingBuffer Members

## PendingBuffer overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PendingBuffer</code></td>
<td>Overloaded. Initializes a new instance of the PendingBuffer class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BitCount</code></td>
<td>The number of bits written to the buffer</td>
</tr>
<tr>
<td><code>IsFlushed</code></td>
<td>Indicates if buffer has been flushed</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AlignToByte</code></td>
<td>Align internal buffer on a byte boundary</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>Flush</code></td>
<td>Flushes the pending buffer into the given output array. If the output array is too small, only a partial flush is done.</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>Reset</code></td>
<td>Clear internal state/buffers</td>
</tr>
<tr>
<td><code>ToByteArray</code></td>
<td>Convert internal buffer to byte array. Buffer is empty on</td>
</tr>
</tbody>
</table>
Completion

| **ToWrite** (inherited from Object) | Returns a String that represents the current Object. |
| **WriteBits** | Write bits to internal buffer |
| **WriteBlock** | Write a block of data to buffer |
| **WriteByte** | Write a byte to buffer |
| **WriteInt** | Write an integer LSB first |
| **WriteShort** | Write a short value to buffer LSB first |
| **WriteShortMSB** | Write a short value to internal buffer most significant byte first |

Protected Instance Methods

| **Finalize** (inherited from Object) | Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. |
| **MemberwiseClone** (inherited from Object) | Creates a shallow copy of the current Object. |

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer Constructor

construct instance using default buffer size of 4096

Overload List

construct instance using default buffer size of 4096

   public PendingBuffer();

construct instance using specified buffer size

   public PendingBuffer(int);

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer Constructor ()

construct instance using default buffer size of 4096

public PendingBuffer();

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer Constructor (Int32)

construct instance using specified buffer size

```csharp
public PendingBuffer(
    int bufferSize
);
```

Parameters

`bufferSize`
size to use for internal buffer

See Also

PendingBuffer Properties

The properties of the PendingBuffer class are listed below. For a complete list of PendingBuffer class members, see the PendingBuffer Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BitCount</strong></td>
<td>The number of bits written to the buffer</td>
</tr>
<tr>
<td><strong>IsFlushed</strong></td>
<td>Indicates if buffer has been flushed</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer.BitCount Property

The number of bits written to the buffer

```csharp
public int BitCount {get;}
```

See Also

PendingBuffer.IsFlushed Property

Indicates if buffer has been flushed

```csharp
public bool IsFlushed {get;}
```

See Also

ICSharpCode SharpZipLib Class Library
### PendingBuffer Methods

The methods of the **PendingBuffer** class are listed below. For a complete list of **PendingBuffer** class members, see the [PendingBuffer Members](#) topic.

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AlignToByte</strong></td>
<td>Align internal buffer on a byte boundary</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the pending buffer into the given output array. If the output array is to small, only a partial flush is done.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Clear internal state/buffers</td>
</tr>
<tr>
<td><strong>ToByteArray</strong></td>
<td>Convert internal buffer to byte array. Buffer is empty on completion.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>WriteBits</strong></td>
<td>Write bits to internal buffer</td>
</tr>
<tr>
<td><strong>WriteBlock</strong></td>
<td>Write a block of data to buffer</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Write a byte to buffer</td>
</tr>
<tr>
<td><strong>WriteInt</strong></td>
<td>write an integer LSB first</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>WriteShort</td>
<td>Write a short value to buffer LSB first</td>
</tr>
<tr>
<td>WriteShortMSB</td>
<td>Write a short value to internal buffer most significant byte first</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
PendingBuffer.AlignToByte Method

Align internal buffer on a byte boundary

public void AlignToByte();

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer.Flush Method

Flushes the pending buffer into the given output array. If the output array is to small, only a partial flush is done.

```csharp
public int Flush(byte[] output, int offset, int length);
```

Parameters

- **output**
  The output array.

- **offset**
  The offset into output array.

- **length**
  The maximum number of bytes to store.

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer.Reset Method

Clear internal state/buffers

```csharp
public void Reset();
```

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer.ToByteArray Method

Convert internal buffer to byte array. Buffer is empty on completion

```csharp
public byte[] ToByteArray();
```

Return Value

The internal buffer contents converted to a byte array.

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer.WriteBits Method

Write bits to internal buffer

```csharp
public void WriteBits(
    int b,
    int count
);
```

Parameters

- `b`
  - source of bits
- `count`
  - number of bits to write

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer.WriteBlock Method

Write a block of data to buffer

```csharp
public void WriteBlock(
    byte[] block,
    int offset,
    int length
);
```

Parameters

- `block`  
  data to write

- `offset`  
  offset of first byte to write

- `length`  
  number of bytes to write

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer.WriteByte Method

Write a byte to buffer

```csharp
public void WriteByte(int value);
```

Parameters

- `value`  
  The value to write

See Also

- PendingBuffer Class
ICSharpCode SharpZipLib Class Library
PendingBuffer.WriteInt Method

write an integer LSB first

```java
public void WriteInt(
    int value
);
```

Parameters

`value`

The value to write.

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer.WriteShort Method

Write a short value to buffer LSB first

```csharp
public void WriteShort(
    int value
);
```

Parameters

value

The value to write.

See Also

ICSharpCode SharpZipLib Class Library
PendingBuffer.WriteShortMSB Method

Write a short value to internal buffer most significant byte first

public void WriteShortMSB(int s);

Parameters

s
value to write

See Also

# Namespace hierarchy

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DeflaterOutputStream</strong></td>
<td>A special stream deflating or compressing the bytes that are written to it. It uses a Deflater to perform actual deflating. Authors of the original java version : Tom Tromey, Jochen Hoenicke</td>
</tr>
<tr>
<td><strong>InflaterInputBuffer</strong></td>
<td>An input buffer customised for use by <strong>InflaterInputStream</strong></td>
</tr>
<tr>
<td><strong>InflaterInputStream</strong></td>
<td>This filter stream is used to decompress data compressed using the &quot;deflate&quot; format. The &quot;deflate&quot; format is described in RFC 1951. This stream may form the basis for other decompression filters, such as the <strong>GZipInputStream</strong>. Author of the original java version : John Leuner.</td>
</tr>
<tr>
<td><strong>OutputWindow</strong></td>
<td>Contains the output from the Inflation process. We need to have a window so that we can refer backwards into the output stream to repeat stuff. Author of the original java version : John Leuner</td>
</tr>
<tr>
<td><strong>StreamManipulator</strong></td>
<td>This class allows us to retrieve a specified number of bits from...</td>
</tr>
</tbody>
</table>
the input buffer, as well as copy big byte blocks. It uses an int buffer to store up to 31 bits for direct manipulation. This guarantees that we can get at least 16 bits, but we only need at most 15, so this is all safe. There are some optimizations in this class, for example, you must never peek more than 8 bits more than needed, and you must first peek bits before you may drop them. This is not a general purpose class but optimized for the behaviour of the Inflater. authors of the original java version : John Leuner, Jochen Hoenicke
ICSharpCode SharpZipLib Class Library
A special stream deflating or compressing the bytes that are written to it. It uses a Deflater to perform actual deflating.
Authors of the original java version: Tom Tromey, Jochen Hoenicke

For a list of all members of this type, see DeflaterOutputStream Members.

System.Object
System.MarshalByRefObject
System.IO.Stream
ICSharpCode.SharpZipLib.GZip.GZipOutputStream

public class DeflaterOutputStream : Stream

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

DeflaterOutputStream Members |
# DeflaterOutputStream Members

## DeflaterOutputStream overview

### Public Instance Constructors

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="DeflaterOutputStream" /></td>
<td>Overloaded. Initializes a new instance of the DeflaterOutputStream class.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="CanPatchEntries" /></td>
<td>Allows client to determine if an entry can be patched after its added</td>
</tr>
<tr>
<td><img src="image" alt="CanRead" /></td>
<td>Gets value indicating stream can be read from</td>
</tr>
<tr>
<td><img src="image" alt="CanSeek" /></td>
<td>Gets a value indicating if seeking is supported for this stream. This property always returns false</td>
</tr>
<tr>
<td><img src="image" alt="CanWrite" /></td>
<td>Get value indicating if this stream supports writing</td>
</tr>
<tr>
<td><img src="image" alt="IsStreamOwner" /></td>
<td>Get/set flag indicating ownership of the underlying stream. When the flag is true, <strong>Close</strong> will close the underlying stream also.</td>
</tr>
<tr>
<td><img src="image" alt="Length" /></td>
<td>Get current length of stream</td>
</tr>
<tr>
<td><img src="image" alt="Password" /></td>
<td>Get/set the password used for encryption.</td>
</tr>
<tr>
<td><img src="image" alt="Position" /></td>
<td>Gets the current position within the stream.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="BeginRead" /></td>
<td>Asynchronous reads are not</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>Asynchronous writes aren't supported, a NotSupportedException is always thrown.</td>
</tr>
<tr>
<td>Close</td>
<td>Calls Finish and closes the underlying stream when IsStreamOwner is true.</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>(inherited from MarshalByRefObject) Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td>EndRead</td>
<td>(inherited from Stream) Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td>EndWrite</td>
<td>(inherited from Stream) Ends an asynchronous write operation.</td>
</tr>
<tr>
<td>Equals</td>
<td>(inherited from Object) Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>Finish</td>
<td>Finishes the stream by calling finish() on the deflater.</td>
</tr>
<tr>
<td>Flush</td>
<td>Flushes the stream by calling flush() on the deflater and then on the underlying stream. This ensures that all bytes are flushed.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>(inherited from Object) Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>(inherited) Retrieves the current lifetime</td>
</tr>
</tbody>
</table>
### MarshalByRefObject

<table>
<thead>
<tr>
<th>Method/Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Read a block of bytes from stream</td>
</tr>
<tr>
<td><strong>ReadByte</strong></td>
<td>Read a byte from stream advancing position by one</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>Sets the current position of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Sets the length of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes bytes from an array to the compressed stream.</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Writes a single byte to the compressed output stream.</td>
</tr>
</tbody>
</table>

### Protected Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>baseOutputStream</strong></td>
<td>Base stream the deflater depends on.</td>
</tr>
<tr>
<td><strong>def</strong></td>
<td>The deflater which is used to deflate the stream.</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method/Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateWaitHandle</strong> (inherited from Stream)</td>
<td>Allocates a WaitHandle object.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Deflate</strong></td>
<td>Deflates everything in the input buffers. This will call <code>def.deflate()</code> until all bytes from the input buffers are processed.</td>
</tr>
<tr>
<td><strong>EncryptBlock</strong></td>
<td>Encrypt a block of data</td>
</tr>
<tr>
<td><strong>EncryptByte</strong></td>
<td>Encrypt a single byte</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>InitializePassword</strong></td>
<td>Initializes encryption keys based on given password</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td><strong>UpdateKeys</strong></td>
<td>Update encryption keys</td>
</tr>
</tbody>
</table>

**See Also**

DeflaterOutputStream Class  |
ICSharpCode SharpZipLib Class Library
**DeflaterOutputStream Constructor**

Creates a new DeflaterOutputStream with a default Deflater and default buffer size.

**Overload List**

Creates a new DeflaterOutputStream with a default Deflater and default buffer size.

```java
public DeflaterOutputStream(Stream);
```

Creates a new DeflaterOutputStream with the given Deflater and default buffer size.

```java
public DeflaterOutputStream(Stream, Deflater);
```

Creates a new DeflaterOutputStream with the given Deflater and buffer size.

```java
public DeflaterOutputStream(Stream, Deflater, int);
```

**See Also**

[DeflaterOutputStream Class](#)  
DeflaterOutputStream Constructor (Stream)

Creates a new DeflaterOutputStream with a default Deflater and default buffer size.

```java
public DeflaterOutputStream(
    Stream baseOutputStream
);
```

Parameters

`baseOutputStream`  
the output stream where deflated output should be written.

See Also

ICSharpCode SharpZipLib Class Library
DeflaterOutputStream Constructor (Stream, Deflater)

Creates a new DeflaterOutputStream with the given Deflater and default buffer size.

```java
public DeflaterOutputStream(
    Stream baseOutputStream,
    Deflater deflater
);
```

Parameters

- `baseOutputStream`  
  the output stream where deflated output should be written.

- `deflater`  
  the underlying deflater.

See Also

- DeflaterOutputStream Class
- DeflaterOutputStream Constructor Overload List
ICSharpCode SharpZipLib Class Library
**DeflaterOutputStream Constructor (Stream, Deflater, Int32)**

Creates a new DeflaterOutputStream with the given Deflater and buffer size.

```java
public DeflaterOutputStream(
    Stream baseOutputStream,
    Deflater deflater,
    int bufferSize
);
```

**Parameters**

- `baseOutputStream`  
  The output stream where deflated output is written.

- `deflater`  
  The underlying deflater to use

- `bufferSize`  
  The buffer size to use when deflating

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentOutOfRangeException</code></td>
<td>bufsize is less than or equal to zero.</td>
</tr>
<tr>
<td><code>ArgumentException</code></td>
<td>baseOutputStream does not support writing.</td>
</tr>
<tr>
<td><code>ArgumentNullException</code></td>
<td>deflater instance is null</td>
</tr>
</tbody>
</table>

**See Also**

- [DeflaterOutputStream Class](#)  
- [DeflaterOutputStream Constructor Overload List](#)
ICSharpCode SharpZipLib Class Library
The fields of the **DeflaterOutputStream** class are listed below. For a complete list of **DeflaterOutputStream** class members, see the [DeflaterOutputStream Members](#) topic.

### Protected Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseOutputStream</td>
<td>Base stream the deflater depends on.</td>
</tr>
<tr>
<td>def</td>
<td>The deflater which is used to deflate the stream.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.baseOutputStream Field

Base stream the deflater depends on.

```java
protected Stream baseOutputStream;
```

See Also

- DeflaterOutputStream Class
ICSharpCode SharpZipLib Class Library
The deflater which is used to deflate the stream.

```java
protected Deflater def;
```

See Also

- DeflaterOutputStream Class
DeflaterOutputStream Properties

The properties of the DeflaterOutputStream class are listed below. For a complete list of DeflaterOutputStream class members, see the DeflaterOutputStream Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanPatchEntries</td>
<td>Allows client to determine if an entry can be patched after its added</td>
</tr>
<tr>
<td>CanRead</td>
<td>Gets value indicating stream can be read from</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Gets a value indicating if seeking is supported for this stream. This property always returns false.</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Get value indicating if this stream supports writing</td>
</tr>
<tr>
<td>IsStreamOwner</td>
<td>Get/set flag indicating ownership of the underlying stream. When the flag is true, Close will close the underlying stream also.</td>
</tr>
<tr>
<td>Length</td>
<td>Get current length of stream</td>
</tr>
<tr>
<td>Password</td>
<td>Get/set the password used for encryption.</td>
</tr>
<tr>
<td>Position</td>
<td>Gets the current position within the stream.</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.CanPatchEntries Property

Allows client to determine if an entry can be patched after its added

```csharp
public bool CanPatchEntries {get;}
```

See Also

**DeflaterOutputStream.CanRead Property**

Gets value indicating stream can be read from

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

**See Also**

[DeflaterOutputStream Class](#)
**DeflaterOutputStream.CanSeek Property**

Gets a value indicating if seeking is supported for this stream. This property always returns false.

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

See Also

- [DeflaterOutputStream Class](#)
DeflaterOutputStream.CanWrite Property

Get value indicating if this stream supports writing

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

See Also

ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.IsStreamOwner Property

Get/set flag indicating ownership of the underlying stream. When the flag is true Close will close the underlying stream also.

```csharp
public bool IsStreamOwner { get; set; }
```

See Also

ICSharpCode SharpZipLib Class Library
**DeflaterOutputStream.Length Property**

Get current length of stream

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

See Also

ICSharpCode SharpZipLib Class Library
**DeflaterOutputStream.Password Property**

Get/set the password used for encryption.

```csharp
public string Password {get; set;}
```

**Remarks**

When set to null or if the password is empty no encryption is performed

**See Also**

- DeflaterOutputStream Class
ICSharpCode SharpZipLib Class Library
**DeflaterOutputStream.Position Property**

Gets the current position within the stream.

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

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>NotSupportedException</code></td>
<td>Any attempt to set position</td>
</tr>
</tbody>
</table>

**See Also**

- [DeflaterOutputStream Class](#)
ICSharpCode SharpZipLib Class Library
The methods of the **DeflaterOutputStream** class are listed below. For a complete list of **DeflaterOutputStream** class members, see the [DeflaterOutputStream Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong></td>
<td>Asynchronous reads are not supported a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>BeginWrite</strong></td>
<td>Asynchronous writes are not supported, a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Calls <strong>Finish</strong> and closes the underlying stream when <strong>IsStreamOwner</strong> is true.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from <strong>MarshalByRefObject</strong>)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from <strong>Stream</strong>)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from <strong>Stream</strong>)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Finishes the stream by calling finish() on the deflater.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flashes the stream by calling flush() on the deflater and then</td>
</tr>
</tbody>
</table>
on the underlying stream. This ensures that all bytes are flushed.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong></td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong></td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Read a block of bytes from stream</td>
</tr>
<tr>
<td><strong>ReadByte</strong></td>
<td>Read a byte from stream advancing position by one</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>Sets the current position of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Sets the length of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes bytes from an array to the compressed stream.</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Writes a single byte to the compressed output stream.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**
| **CreateWaitHandle** (inherited from **Stream**) | Allocates a **WaitHandle** object. |
| **Deflate** | Deflates everything in the input buffers. This will call `def.deflate()` until all bytes from the input buffers are processed. |
| **EncryptBlock** | Encrypt a block of data |
| **EncryptByte** | Encrypt a single byte |
| **Finalize** (inherited from **Object**) | Allows an **Object** to attempt to free resources and perform other cleanup operations before the **Object** is reclaimed by garbage collection. |
| **InitializePassword** | Initializes encryption keys based on given password |
| **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |
| **UpdateKeys** | Update encryption keys |

**See Also**

- DeflaterOutputStream Class
ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.BeginRead Method

Asynchronous reads are not supported a NotSupportedException is always thrown

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

Parameters

- **buffer**
  The buffer to read into.

- **offset**
  The offset to start storing data at.

- **count**
  The number of bytes to read.

- **callback**
  The async callback to use.

- **state**
  The state to use.

Return Value

Returns an IAsyncResult

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

DeflaterOutputStream Class
ICSharpCode SharpZipLib Class Library
Asynchronous writes aren't supported, a NotSupportedException is always thrown.

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

### Parameters

- **buffer**
  - The buffer to write.

- **offset**
  - The offset to begin writing at.

- **count**
  - The number of bytes to write.

- **callback**
  - The `AsyncCallback` to use.

- **state**
  - The state object.

### Return Value

Returns an `IAsyncResult`.

### Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>NotSupportedException</code></td>
<td>Any access</td>
</tr>
</tbody>
</table>

### See Also

- `DeflaterOutputStream Class |`
ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.Close Method

Calls Finish and closes the underlying stream when IsStreamOwner is true.

```csharp
public override void Close();
```

See Also

ICSharpCode SharpZipLib Class Library
Deflates everything in the input buffers. This will call

```java
def deflate()
```

until all bytes from the input buffers are processed.

```java
protected void Deflate();
```

See Also

ICSharpCode SharpZipLib Class Library
Encrypt a block of data

```csharp
protected void EncryptBlock(
    byte[] buffer,
    int offset,
    int length
);
```

**Parameters**

- **buffer**
  Data to encrypt. NOTE the original contents of the buffer are lost

- **offset**
  Offset of first byte in buffer to encrypt

- **length**
  Number of bytes in buffer to encrypt

**See Also**

- [DeflaterOutputStream Class](#)
ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.EncryptByte Method

Encrypt a single byte

```csharp
protected byte EncryptByte();
```

Return Value

The encrypted value

See Also

ICSharpCode SharpZipLib Class Library
**DeflaterOutputStream.Finish Method**

Finishes the stream by calling `finish()` on the deflater.

```csharp
public virtual void Finish();
```

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SharpZipBaseException</code></td>
<td>Not all input is deflated</td>
</tr>
</tbody>
</table>

**See Also**

- [DeflaterOutputStream Class](#)
ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.Flush Method

Flushes the stream by calling flush() on the deflater and then on the underlying stream. This ensures that all bytes are flushed.

```csharp
public override void Flush();
```

See Also

ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.InitializePassword Method

Initializes encryption keys based on given password

protected void InitializePassword(string password);

See Also

ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.Read Method

Read a block of bytes from stream

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

- DeflaterOutputStream Class
ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.ReadByte Method

Read a byte from stream advancing position by one

```csharp
public override int ReadByte();
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.Seek Method

Sets the current position of this stream to the given value. Not supported by this class!

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

- DeflaterOutputStream Class
ICSharpCode SharpZipLib Class Library
Sets the length of this stream to the given value. Not supported by this class!

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

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.UpdateKeys Method

Update encryption keys

```csharp
protected void UpdateKeys(
    byte ch
);
```

See Also

ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.Write Method

 Writes bytes from an array to the compressed stream.

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

Parameters

- `buffer` The byte array
- `offset` The offset into the byte array where to start.
- `count` The number of bytes to write.

See Also

- DeflaterOutputStream Class
ICSharpCode SharpZipLib Class Library
DeflaterOutputStream.WriteByte Method

Writes a single byte to the compressed output stream.

```csharp
public override void WriteByte(byte value);
```

Parameters

`value`

The byte value.

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputBuffer Class

An input buffer customised for use by InflaterInputStream
For a list of all members of this type, see InflaterInputBuffer Members.

System.Object

public class InflaterInputBuffer

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Remarks

The buffer supports decryption of incoming data.

Requirements


See Also

ICSharpCode SharpZipLib Class Library
## InflaterInputBuffer Members

### InflaterInputBuffer overview

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overloaded. Initializes a new instance of the InflaterInputBuffer class.</td>
</tr>
</tbody>
</table>

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Available</strong></td>
<td>Get/set the number of bytes available</td>
</tr>
<tr>
<td><strong>ClearText</strong></td>
<td>Get the contents of the clear text buffer.</td>
</tr>
<tr>
<td><strong>ClearTextLength</strong></td>
<td>Get the number of useable bytes in ClearText</td>
</tr>
<tr>
<td><strong>CryptoTransform</strong></td>
<td>Get/set the ICryptoTransform to apply to any data.</td>
</tr>
<tr>
<td><strong>RawData</strong></td>
<td>Get the contents of the raw data buffer.</td>
</tr>
<tr>
<td><strong>RawLength</strong></td>
<td>Get the length of bytes bytes in the RawData</td>
</tr>
</tbody>
</table>

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Fill</strong></td>
<td>Fill the buffer from the underlying input stream.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>Get Type</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>Read Clear Text Buffer</strong></td>
<td>Read clear text data from the input stream.</td>
</tr>
<tr>
<td><strong>Read Le Byte</strong></td>
<td>Read a byte from the input stream.</td>
</tr>
<tr>
<td><strong>Read Le Int</strong></td>
<td>Read an int in little endian byte order.</td>
</tr>
<tr>
<td><strong>Read Le Long</strong></td>
<td>Read an int base InputStream little endian byte order.</td>
</tr>
<tr>
<td><strong>Read Le Short</strong></td>
<td>Read an unsigned short in little endian byte order.</td>
</tr>
<tr>
<td><strong>Read Raw Buffer</strong></td>
<td>Overloaded. Read a buffer directly from the input stream</td>
</tr>
<tr>
<td><strong>Set Inflater Input</strong></td>
<td>Call <strong>Set Input</strong> passing the current clear text buffer contents.</td>
</tr>
<tr>
<td><strong>To String</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

| **Finalize** (inherited from **Object**) | Allows an **Object** to attempt to free resources and perform other cleanup operations before the **Object** is reclaimed by garbage collection. |
| **Memberwise Clone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

- **Inflater Input Buffer Class**
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer Constructor

Initialise a new instance of InflaterInputBuffer with a default buffer size

Overload List

Initialise a new instance of InflaterInputBuffer with a default buffer size

    public InflaterInputBuffer(Stream);

Initialise a new instance of InflaterInputBuffer

    public InflaterInputBuffer(Stream,int);

See Also

ICSharpCode SharpZipLib Class Library
Initialise a new instance of InflaterInputBuffer with a default buffer size

```csharp
public InflaterInputBuffer(
    Stream stream
);
```

Parameters

*stream*

The stream to buffer.

See Also

ICSharpCode SharpZipLib Class Library
Initialise a new instance of `InflaterInputBuffer`

```csharp
public InflateInputBuffer(
    Stream stream,
    int bufferSize
);
```

**Parameters**

- `stream`
  - The stream to buffer.

- `bufferSize`
  - The size to use for the buffer

**Remarks**

A minimum buffer size of 1KB is permitted. Lower sizes are treated as 1KB.

**See Also**

- `InflaterInputBuffer Class`
- `InflaterInputBuffer Constructor Overload List`
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer Properties

The properties of the InflaterInputBuffer class are listed below. For a complete list of InflaterInputBuffer class members, see the InflaterInputBuffer Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>Get/set the number of bytes available</td>
</tr>
<tr>
<td>ClearText</td>
<td>Get the contents of the clear text buffer.</td>
</tr>
<tr>
<td>ClearTextLength</td>
<td>Get the number of useable bytes in ClearText</td>
</tr>
<tr>
<td>CryptoTransform</td>
<td>Get/set the ICryptoTransform to apply to any data.</td>
</tr>
<tr>
<td>RawData</td>
<td>Get the contents of the raw data buffer.</td>
</tr>
<tr>
<td>RawLength</td>
<td>Get the length of bytes bytes in the RawData</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.Available Property

Get/set the number of bytes available

```csharp
public int Available {get; set;}
```

See Also

InflaterInputBuffer Class
InflaterInputBuffer.ClearText Property

Get the contents of the clear text buffer.

```csharp
public byte[] ClearText {get;}
```

See Also

- InflaterInputBuffer Class
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.ClearTextLength Property

Get the number of useable bytes in ClearText

```csharp
public int ClearTextLength {get;}
```

See Also

InflaterInputBuffer Class |
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.CryptoTransform Property

Get/set the ICryptoTransform to apply to any data.

```csharp
```

Remarks

Set this value to null to have no transform applied.

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.RawData Property

Get the contents of the raw data buffer.

```csharp
public byte[] RawData {get;}
```

Remarks

This may contain encrypted data.

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.RawLength Property

Get the length of bytes bytes in the RawData

```csharp
public int RawLength {get;}
```

See Also

InflaterInputBuffer Class
ICSharpCode SharpZipLib Class Library
The methods of the **InflaterInputBuffer** class are listed below. For a complete list of **InflaterInputBuffer** class members, see the **InflaterInputBuffer Members** topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Fill</strong></td>
<td>Fill the buffer from the underlying input stream.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ReadClearTextBuffer</strong></td>
<td>Read clear text data from the input stream.</td>
</tr>
<tr>
<td><strong>ReadLeByte</strong></td>
<td>Read a byte from the input stream.</td>
</tr>
<tr>
<td><strong>ReadLeInt</strong></td>
<td>Read an int in little endian byte order.</td>
</tr>
<tr>
<td><strong>ReadLeLong</strong></td>
<td>Read an int baseInputStream little endian byte order.</td>
</tr>
<tr>
<td><strong>ReadLeShort</strong></td>
<td>Read an unsigned short in little endian byte order.</td>
</tr>
<tr>
<td><strong>ReadRawBuffer</strong></td>
<td>Overloaded. Read a buffer directly from the input stream.</td>
</tr>
<tr>
<td><strong>SetInflaterInput</strong></td>
<td>Call <strong>SetInput</strong> passing the current clear text buffer contents.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
</tbody>
</table>

### Protected Instance Methods

<table>
<thead>
<tr>
<th><strong>Finalize</strong> (inherited from Object)</th>
<th>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### See Also

- [InflaterInputBuffer Class](#)
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.Fill Method

Fill the buffer from the underlying input stream.

```csharp
public void Fill();
```

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.ReadClearTextBuffer Method

Read clear text data from the input stream.

```csharp
public int ReadClearTextBuffer(
    byte[] outBuffer,
    int offset,
    int length
);
```

Parameters

* `outBuffer`  
  The buffer to add data to.

* `offset`  
  The offset to start adding data at.

* `length`  
  The number of bytes to read.

Return Value

Returns the number of bytes actually read.

See Also

* InflaterInputBuffer Class  
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.ReadLeByte Method

Read a byte from the input stream.

```csharp
public int ReadLeByte();
```

Return Value

Returns the byte read.

See Also

- InflaterInputBuffer Class
InflaterInputBuffer.ReadLeInt Method

Read an int in little endian byte order.

public int ReadLeInt();

See Also

InflaterInputBuffer Class
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.ReadLeLong Method

Read an int baseInputStream little endian byte order.

```public long ReadLeLong();``` 

See Also

InflaterInputBuffer Class
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.ReadLeShort Method

Read an unsigned short in little endian byte order.

public int ReadLeShort();

See Also

InflaterInputBuffer Class |
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.ReadRawBuffer Method

Read a buffer directly from the input stream

Overload List

Read a buffer directly from the input stream

```csharp
public int ReadRawBuffer(byte[]);
```

Read a buffer directly from the input stream

```csharp
public int ReadRawBuffer(byte[], int, int);
```

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.ReadRawBuffer Method (Byte[])

Read a buffer directly from the input stream

```csharp
public int ReadRawBuffer(
    byte[] buffer
);
```

Parameters

- `buffer`  
  The buffer to fill

Return Value

Returns the number of bytes read.

See Also

- [InflaterInputBuffer Class](#)
- [InflaterInputBuffer.ReadRawBuffer Overload List](#)
InflaterInputBuffer.ReadRawBuffer Method (Byte[], Int32, Int32)

Read a buffer directly from the input stream

```csharp
public int ReadRawBuffer(
    byte[] outBuffer, 
    int offset, 
    int length
);
```

Parameters

- **outBuffer**
  - The buffer to read into

- **offset**
  - The offset to start reading data into.

- **length**
  - The number of bytes to read.

Return Value

Returns the number of bytes read.

See Also

- InflaterInputBuffer Class
- InflaterInputBuffer.ReadRawBuffer Overload List
ICSharpCode SharpZipLib Class Library
InflaterInputBuffer.SetInflaterInput Method

Call **SetInput** passing the current clear text buffer contents.

```csharp
public void SetInflaterInput(Inflater inflater);
```

**Parameters**

- **inflater**
  The inflater to set input for.

**See Also**

- InflaterInputBuffer Class
ICSharpCode SharpZipLib Class Library
InflaterInputStream Class

This filter stream is used to decompress data compressed using the "deflate" format. The "deflate" format is described in RFC 1951. This stream may form the basis for other decompression filters, such as the GZipInputStream. Author of the original java version: John Leuner.

For a list of all members of this type, see InflaterInputStream Members.

System.Object  System.MarshalByRefObject  System.IO.Stream  
ICSharpCode.SharpZipLib.GZip.GZipInputStream  

**public class InflaterInputStream : Stream**

**Thread Safety**

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are **not** guaranteed to be thread-safe.

**Requirements**

**Namespace:** ISharpCode.SharpZipLib.Zip.Compression.Streams

**Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

**See Also**

InflaterInputStream Members  |  
ICSharpCode SharpZipLib Class Library
**InflaterInputStream Members**

**InflaterInputStream overview**

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InflaterInputStream</code></td>
<td>Overloaded. Initializes a new instance of the InflaterInputStream class.</td>
</tr>
</tbody>
</table>

**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Available</code></td>
<td>Returns 0 once the end of the stream (EOF) has been reached. Otherwise returns 1.</td>
</tr>
<tr>
<td><code>CanRead</code></td>
<td>Gets a value indicating whether the current stream supports reading.</td>
</tr>
<tr>
<td><code>CanSeek</code></td>
<td>Gets a value of false indicating seeking is not supported for this stream.</td>
</tr>
<tr>
<td><code>CanWrite</code></td>
<td>Gets a value of false indicating that this stream is not writeable.</td>
</tr>
<tr>
<td><code>IsStreamOwner</code></td>
<td>Get/set flag indicating ownership of underlying stream. When the flag is true Close will close the underlying stream also.</td>
</tr>
<tr>
<td><code>Length</code></td>
<td>A value representing the length of the stream in bytes.</td>
</tr>
<tr>
<td><code>Position</code></td>
<td>The current position within the stream. Throws a NotSupportedException when attempting to set the position</td>
</tr>
</tbody>
</table>

**Public Instance Methods**
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong> (inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong></td>
<td>Entry point to begin an asynchronous write. Always throws a NotSupportedException.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the input stream. When IsStreamOwner is true the underlying stream is also closed.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the baseInputStream</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong></td>
<td>Obtains a lifetime service object</td>
</tr>
<tr>
<td><strong>(inherited from MarshalByRefObject)</strong></td>
<td><strong>to control the lifetime policy for this instance.</strong></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>▶ <strong>Read</strong></td>
<td>Reads decompressed data into the provided buffer byte array</td>
</tr>
<tr>
<td>▶ <strong>ReadByte</strong> (inherited from Stream)</td>
<td>Reads a byte from the stream and advances the position within the stream by one byte, or returns -1 if at the end of the stream.</td>
</tr>
<tr>
<td>▶ <strong>Seek</strong></td>
<td>Sets the position within the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td>▶ <strong>SetLength</strong></td>
<td>Set the length of the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td>▶ <strong>Skip</strong></td>
<td>Skip specified number of bytes of uncompressed data</td>
</tr>
<tr>
<td>▶ <strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td>▶ <strong>Write</strong></td>
<td>Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException</td>
</tr>
<tr>
<td>▶ <strong>WriteByte</strong></td>
<td>Writes one byte to the current stream and advances the current position Always throws a NotSupportedException</td>
</tr>
</tbody>
</table>

**Protected Instance Fields**

| ▶ **baseInputStream** | Base stream the inflater reads from. |
| ▶ **csize**           | The compressed size |
| **inf** | Decompressor for this stream |
| **inputBuffer** | **Input buffer** for this stream. |

Protected Instance Methods

| **CreateWaitHandle** (inherited from **Stream**) | Allocates a **WaitHandle** object. |
| **Fill** | Fills the buffer with more data to decompress. |
| **Finalize** (inherited from **Object**) | Allows an **Object** to attempt to free resources and perform other cleanup operations before the **Object** is reclaimed by garbage collection. |
| **MemberwiseClone** (inherited from **Object**) | Creates a shallow copy of the current **Object**. |
| **StopDecrypting** | Clear any cryptographic state. |

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputStream Constructor

Create an InflaterInputStream with the default decompressor and a default buffer size of 4KB.

Overload List

Create an InflaterInputStream with the default decompressor and a default buffer size of 4KB.

    public InflaterInputStream(Stream);

Create an InflaterInputStream with the specified decompressor and a default buffer size of 4KB.

    public InflaterInputStream(Stream,Inflater);

Create an InflaterInputStream with the specified decompressor and the specified buffer size.

    public InflaterInputStream(Stream,Inflater,int);

See Also

InflaterInputStream Constructor (Stream)

Create an InflaterInputStream with the default decompressor and a default buffer size of 4KB.

```java
public InflaterInputStream(
    Stream baseInputStream
);
```

Parameters

`baseInputStream`  
The InputStream to read bytes from

See Also

ICSharpCode SharpZipLib Class Library
Create an InflaterInputStream with the specified decompressor and a default buffer size of 4KB.

```java
public InflaterInputStream(
    Stream baseInputStream,
    Inflater inf
);
```

**Parameters**

- `baseInputStream`
  The source of input data

- `inf`
  The decompressor used to decompress data read from `baseInputStream`

**See Also**

Create an InflaterInputStream with the specified decompressor and the specified buffer size.

```java
public InflaterInputStream(
    Stream baseInputStream,
    Inflater inflater,
    int bufferSize
);
```

### Parameters

- **baseInputStream**
  - The InputStream to read bytes from

- **inflater**
  - The decompressor to use

- **bufferSize**
  - Size of the buffer to use

### See Also

- [InflaterInputStream Class](#)
- [InflaterInputStream Constructor Overload List](#)
The fields of the **InflaterInputStream** class are listed below. For a complete list of **InflaterInputStream** class members, see the [InflaterInputStream Members](#) topic.

### Protected Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>baseInputStream</code></td>
<td>Base stream the inflater reads from.</td>
</tr>
<tr>
<td><code>csize</code></td>
<td>The compressed size</td>
</tr>
<tr>
<td><code>inf</code></td>
<td>Decompressor for this stream</td>
</tr>
<tr>
<td><code>inputBuffer</code></td>
<td>Input buffer for this stream.</td>
</tr>
</tbody>
</table>

### See Also

ICSharpCode SharpZipLib Class Library
InflaterInputStream.baseInputStream Field

Base stream the inflater reads from.

protected Stream baseInputStream;

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputStream.csize Field

The compressed size

protected long csize;

See Also

InflaterInputStream Class
ICSharpCode SharpZipLib Class Library
Decompressor for this stream

```java
protected Inflater inf;
```

See Also

- InflaterInputStream Class
InflaterInputStream.inputBuffer Field

Input buffer for this stream.

```java
protected InflaterInputBuffer inputBuffer;
```

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputStream Properties

The properties of the InflaterInputStream class are listed below. For a complete list of InflaterInputStream class members, see the InflaterInputStream Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>Returns 0 once the end of the stream (EOF) has been reached. Otherwise returns 1.</td>
</tr>
<tr>
<td>CanRead</td>
<td>Gets a value indicating whether the current stream supports reading</td>
</tr>
<tr>
<td>CanSeek</td>
<td>Gets a value of false indicating seeking is not supported for this stream.</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Gets a value of false indicating that this stream is not writeable.</td>
</tr>
<tr>
<td>IsStreamOwner</td>
<td>Get/set flag indicating ownership of underlying stream. When the flag is true Close will close the underlying stream also.</td>
</tr>
<tr>
<td>Length</td>
<td>A value representing the length of the stream in bytes.</td>
</tr>
<tr>
<td>Position</td>
<td>The current position within the stream. Throws a NotSupportedException when attempting to set the position</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputStream.Available Property

Returns 0 once the end of the stream (EOF) has been reached. Otherwise returns 1.

```csharp
public virtual int Available {get;}
```

See Also

- InflaterInputStream Class
ICSharpCode SharpZipLib Class Library
InflaterInputStream.CanRead Property

Gets a value indicating whether the current stream supports reading

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

See Also

InflaterInputStream.CanSeek Property

Gets a value of false indicating seeking is not supported for this stream.

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

See Also

- InflaterInputStream Class
InflaterInputStream.CanWrite Property

Gets a value of false indicating that this stream is not writeable.

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

See Also

- InflaterInputStream Class
InflaterInputStream.IsStreamOwner Property

Get/set flag indicating ownership of underlying stream. When the flag is true Close will close the underlying stream also.

```csharp
public bool IsStreamOwner {get; set;}
```

Remarks
The default value is true.

See Also
InflaterInputStream Class |
ICSharpCode SharpZipLib Class Library
InflaterInputStream.Length Property

A value representing the length of the stream in bytes.

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

See Also

InflaterInputStream.Position Property

The current position within the stream. Throws a NotSupportedException when attempting to set the position

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Attempting to set the position</td>
</tr>
</tbody>
</table>

See Also

- InflaterInputStream Class
ICSharpCode SharpZipLib Class Library
## InflaterInputStream Methods

The methods of the **InflaterInputStream** class are listed below. For a complete list of **InflaterInputStream** class members, see the [InflaterInputStream Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong></td>
<td>(inherited from Stream) Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong></td>
<td>Entry point to begin an asynchronous write. Always throws a NotSupportedException.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the input stream. When <a href="#">IsStreamOwner</a> is true the underlying stream is also closed.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong></td>
<td>(inherited from MarshalByRefObject) Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><strong>EndRead</strong></td>
<td>(inherited from Stream) Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong></td>
<td>(inherited from Stream) Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>(inherited from Object) Determines whether the specified <a href="#">Object</a> is equal to the current <a href="#">Object</a>.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td></td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>(inherited from Object) Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from <strong>MarshalByRefObject</strong>)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from <strong>MarshalByRefObject</strong>)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Reads decompressed data into the provided buffer byte array</td>
</tr>
<tr>
<td><strong>ReadByte</strong> (inherited from <strong>Stream</strong>)</td>
<td>Reads a byte from the stream and advances the position within the stream by one byte, or returns -1 if at the end of the stream.</td>
</tr>
<tr>
<td><strong>Seek</strong></td>
<td>Sets the position within the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Set the length of the current stream Always throws a NotSupportedException</td>
</tr>
<tr>
<td><strong>Skip</strong></td>
<td>Skip specified number of bytes of uncompressed data</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong></td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Writes one byte to the current stream and advances the current position Always throws a NotSupportedException</td>
</tr>
</tbody>
</table>
## Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateWaitHandle</code> (inherited from <code>Stream</code>)</td>
<td>Allocates a <code>WaitHandle</code> object.</td>
</tr>
<tr>
<td><code>Fill</code></td>
<td>Fills the buffer with more data to decompress.</td>
</tr>
<tr>
<td><code>Finalize</code> (inherited from <code>Object</code>)</td>
<td>Allows an <code>Object</code> to attempt to free resources and perform other cleanup operations before the <code>Object</code> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>StopDecrypting</code></td>
<td>Clear any cryptographic state.</td>
</tr>
</tbody>
</table>

## See Also

- [InflaterInputStream Class](InflaterInputStream.Class)
ICSharpCode SharpZipLib Class Library
InflaterInputStream.BeginWrite Method

Entry point to begin an asynchronous write. Always throws a NotSupportedException.

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

Parameters

- **buffer**
  The buffer to write data from

- **offset**
  Offset of first byte to write

- **count**
  The maximum number of bytes to write

- **callback**
  The method to be called when the asynchronous write operation is completed

- **state**
  A user-provided object that distinguishes this particular asynchronous write request from other requests

Return Value

An `IAsyncResult` that references the asynchronous write

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also
InflaterInputStream Class  |
InflaterInputStream.Close Method

Closes the input stream. When IsStreamOwner is true the underlying stream is also closed.

```csharp
public override void Close();
```

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputStream.Fill Method

Fills the buffer with more data to decompress.

```csharp
protected void Fill();
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SharpZipBaseException</td>
<td>Stream ends early</td>
</tr>
</tbody>
</table>

See Also

- InflaterInputStream Class
ICSharpCode SharpZipLib Class Library
Flushes the baseInputStream

```csharp
public override void Flush();
```

See Also

- `InflaterInputStream Class`
ICSharpCode SharpZipLib Class Library
InflaterInputStream.Read Method

Reads decompressed data into the provided buffer byte array

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

Parameters

- `buffer`  
The array to read and decompress data into
- `offset`  
The offset indicating where the data should be placed
- `count`  
The number of bytes to decompress

Return Value

The number of bytes read. Zero signals the end of stream

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SharpZipBaseException</td>
<td>Inflater needs a dictionary</td>
</tr>
</tbody>
</table>

See Also

- InflaterInputStream Class
InflaterInputStream.Seek Method

Sets the position within the current stream Always throws a NotSupportedException

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
InflaterInputStream.SetLength Method

Set the length of the current stream Always throws a NotSupportedException

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

- InflaterInputStream Class
ICSharpCode SharpZipLib Class Library
InflaterInputStream.Skip Method

Skip specified number of bytes of uncompressed data

```csharp
public long Skip(
    long count
);
```

Parameters

- `count`  
  Number of bytes to skip

Return Value

The number of bytes skipped, zero if the end of stream has been reached

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentOutOfRangeException</code></td>
<td>Number of bytes to skip is less than zero</td>
</tr>
</tbody>
</table>

See Also

- InflaterInputStream Class
ICSharpCode SharpZipLib Class Library
InflaterInputStream.StopDecrypting Method

Clear any cryptographic state.

```csharp
protected void StopDecrypting();
```

See Also

- InflaterInputStream Class
InflaterInputStream.Write Method

Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

See Also

InflaterInputStream Class | ICSince 2.0.0 | ICSince 2.0.0 |
InflaterInputStream.WriteByte Method

Writes one byte to the current stream and advances the current position. Always throws a NotSupportedException.

```csharp
public override void WriteByte(byte value);
```

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotSupportedException</td>
<td>Any access</td>
</tr>
</tbody>
</table>

**See Also**

OutputWindow Class

Contains the output from the Inflation process. We need to have a window so that we can refer backwards into the output stream to repeat stuff.
Author of the original java version: John Leuner

For a list of all members of this type, see OutputWindow Members.

System.Object

public class OutputWindow

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

ICSharpCode SharpZipLib Class Library
# OutputWindow Members

## OutputWindow overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OutputWindow Constructor</strong></td>
<td>Initializes a new instance of the OutputWindow class.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CopyDict</strong></td>
<td>Copy dictionary to window</td>
</tr>
<tr>
<td><strong>CopyOutput</strong></td>
<td>Copy contents of window to output</td>
</tr>
<tr>
<td><strong>CopyStored</strong></td>
<td>Copy from input manipulator to internal window</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <strong>Object</strong>)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetAvailable</strong></td>
<td>Get bytes available for output in window</td>
</tr>
<tr>
<td><strong>GetFreeSpace</strong></td>
<td>Get remaining unfilled space in window</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <strong>Object</strong>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>Repeat</strong></td>
<td>Append a byte pattern already in the window itself</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Reset by clearing window so GetAvailable returns 0</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Object</strong>)</td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Write a byte to this output window</td>
</tr>
<tr>
<td>Protected Instance Methods</td>
<td></td>
</tr>
<tr>
<td><strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
OutputWindow Constructor

Initializes a new instance of the OutputWindow class.

```csharp
public OutputWindow();
```

See Also

## OutputWindow Methods

The methods of the **OutputWindow** class are listed below. For a complete list of **OutputWindow** class members, see the **OutputWindow Members** topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CopyDict</strong></td>
<td>Copy dictionary to window</td>
</tr>
<tr>
<td><strong>CopyOutput</strong></td>
<td>Copy contents of window to output</td>
</tr>
<tr>
<td><strong>CopyStored</strong></td>
<td>Copy from input manipulator to internal window</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong></td>
</tr>
<tr>
<td><strong>GetAvailable</strong></td>
<td>Get bytes available for output in window</td>
</tr>
<tr>
<td><strong>GetFreeSpace</strong></td>
<td>Get remaining unfilled space in window</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>Repeat</strong></td>
<td>Append a byte pattern already in the window itself</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Reset by clearing window so <strong>GetAvailable</strong> returns 0</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Write a byte to this output window</td>
</tr>
</tbody>
</table>
### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finalize (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

### See Also

- [OutputWindow Class](#)
OutputWindow.CopyDict Method

Copy dictionary to window

```csharp
public void CopyDict(
    byte[] dictionary,
    int offset,
    int length
);
```

Parameters

- `dictionary` source dictionary
- `offset` offset of start in source dictionary
- `length` length of dictionary

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidOperationException</code></td>
<td>If window isn't empty</td>
</tr>
</tbody>
</table>

See Also

- OutputWindow Class
ICSharpCode SharpZipLib Class Library
OutputWindow.CopyOutput Method

Copy contents of window to output

```csharp
public int CopyOutput(
    byte[] output,
    int offset,
    int len
);
```

Parameters

- **output**
  buffer to copy to

- **offset**
  offset to start at

- **len**
  number of bytes to count

Return Value

The number of bytes copied

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>If a window underflow occurs</td>
</tr>
</tbody>
</table>

See Also

- OutputWindow Class
ICSharpCode SharpZipLib Class Library
OutputWindow.CopyToInternalWindow Method

Copy from input manipulator to internal window

```csharp
public int CopyStored(
    StreamManipulator input,
    int length
);
```

Parameters

`input`
- source of data

`length`
- length of data to copy

Return Value

the number of bytes copied

See Also

- [OutputWindow Class](#)
ICSharpCode SharpZipLib Class Library
OutputWindow.GetAvailable Method

Get bytes available for output in window

```csharp
public int GetAvailable();
```

Return Value

Number of bytes filled

See Also

- [OutputWindow Class](#)
ICSharpCode SharpZipLib Class Library
OutputWindow.GetFreeSpace Method

Get remaining unfilled space in window

public int GetFreeSpace();

Return Value

Number of bytes left in window

See Also

OutputWindow Class
ICSharpCode SharpZipLib Class Library
OutputWindow.Repeat Method

Append a byte pattern already in the window itself

```csharp
public void Repeat(
    int length,
    int distance
);
```

Parameters

- `length`: length of pattern to copy
- `distance`: distance from end of window pattern occurs

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>If the repeated data overflows the window</td>
</tr>
</tbody>
</table>

See Also

- OutputWindow Class
ICSharpCode SharpZipLib Class Library
OutputWindow.Reset Method

Reset by clearing window so GetAvailable returns 0

```
public void Reset();
```

See Also

ICSharpCode SharpZipLib Class Library
**OutputWindow.Write Method**

Write a byte to this output window

```csharp
public void Write(
    int value
);
```

**Parameters**

- **value**
  - value to write

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>if window is full</td>
</tr>
</tbody>
</table>

**See Also**

- [OutputWindow Class](#)
ICSharpCode SharpZipLib Class Library
StreamManipulator Class

This class allows us to retrieve a specified number of bits from the input buffer, as well as copy big byte blocks. It uses an int buffer to store up to 31 bits for direct manipulation. This guarantees that we can get at least 16 bits, but we only need at most 15, so this is all safe. There are some optimizations in this class, for example, you must never peek more than 8 bits more than needed, and you must first peek bits before you may drop them. This is not a general purpose class but optimized for the behaviour of the Inflater. authors of the original java version : John Leuner, Jochen Hoenicke

For a list of all members of this type, see StreamManipulator Members.

System.Object


public class StreamManipulator

Thread Safety

Public static (Shared in Visual Basic) members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

Requirements


See Also

ICSharpCode SharpZipLib Class Library
StreamManipulator Members

StreamManipulator overview

Public Instance Constructors

<table>
<thead>
<tr>
<th>StreamManipulator Constructor</th>
<th>Constructs a default StreamManipulator with all buffers empty</th>
</tr>
</thead>
</table>

Public Instance Properties

<table>
<thead>
<tr>
<th>AvailableBits</th>
<th>Gets the number of bits available in the bit buffer. This must be only called when a previous PeekBits() returned -1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvailableBytes</td>
<td>Gets the number of bytes available.</td>
</tr>
<tr>
<td>IsNeedingInput</td>
<td>Returns true when SetInput can be called</td>
</tr>
</tbody>
</table>

Public Instance Methods

<table>
<thead>
<tr>
<th>CopyBytes</th>
<th>Copies bytes from input buffer to output buffer starting at output[offset]. You have to make sure, that the buffer is byte aligned. If not enough bytes are available, copies fewer bytes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DropBits</td>
<td>Drops the next n bits from the input. You should have called PeekBits with a bigger or equal n before, to make sure that enough bits are in the bit buffer.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetBits</td>
<td>Gets the next n bits and</td>
</tr>
</tbody>
</table>
increases input pointer. This is equivalent to **PeekBits** followed by **DropBits**, except for correct error handling.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>PeekBits</strong></td>
<td>Get the next sequence of bits but don't increase input pointer. bitCount must be less or equal 16 and if this call succeeds, you must drop at least n - 8 bits in the next call.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets state and empties internal buffers</td>
</tr>
<tr>
<td><strong>SetInput</strong></td>
<td>Add more input for consumption. Only call when <strong>IsNeedingInput</strong> returns true</td>
</tr>
<tr>
<td><strong>SkipToByteBoundary</strong></td>
<td>Skips to the next byte boundary.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an <strong>Object</strong> to attempt to free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>
See Also

StreamManipulator Class |
ICSharpCode SharpZipLib Class Library
StreamManipulator Constructor

Constructs a default StreamManipulator with all buffers empty

```csharp
public StreamManipulator();
```

See Also

ICSharpCode SharpZipLib Class Library
StreamManipulator Properties

The properties of the StreamManipulator class are listed below. For a complete list of StreamManipulator class members, see the StreamManipulator Members topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvailableBits</td>
<td>Gets the number of bits available in the bit buffer. This must be only called when a previous PeekBits() returned -1.</td>
</tr>
<tr>
<td>AvailableBytes</td>
<td>Gets the number of bytes available.</td>
</tr>
<tr>
<td>IsNeedingInput</td>
<td>Returns true when SetInput can be called</td>
</tr>
</tbody>
</table>

See Also

StreamManipulator.AvailableBits Property

 Gets the number of bits available in the bit buffer. This must be only called when a previous PeekBits() returned -1.

```csharp
public int AvailableBits {get;}
```

See Also

StreamManipulator.AvailableBytes Property

Gets the number of bytes available.

```csharp
public int AvailableBytes {get;}
```

See Also

- StreamManipulator Class
ICSharpCode SharpZipLib Class Library
StreamManipulator.IsNeedingInput Property

Returns true when SetInput can be called

```csharp
public bool IsNeedingInput {get;}
```

See Also

- StreamManipulator Class
ICSharpCode SharpZipLib Class Library
**StreamManipulator Methods**

The methods of the `StreamManipulator` class are listed below. For a complete list of `StreamManipulator` class members, see the [StreamManipulator Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CopyBytes</strong></td>
<td>Copies bytes from input buffer to output buffer starting at output[(\text{offset})]. You have to make sure, that the buffer is byte aligned. If not enough bytes are available, copies fewer bytes.</td>
</tr>
<tr>
<td><strong>DropBits</strong></td>
<td>Drops the next (n) bits from the input. You should have called <code>PeekBits</code> with a bigger or equal (n) before, to make sure that enough bits are in the bit buffer.</td>
</tr>
<tr>
<td><strong>Equals</strong> (inherited from <code>Object</code>)</td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>GetBits</strong></td>
<td>Gets the next (n) bits and increases input pointer. This is equivalent to <code>PeekBits</code> followed by <code>DropBits</code>, except for correct error handling.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> (inherited from <code>Object</code>)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from <code>Object</code>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><strong>PeekBits</strong></td>
<td>Get the next sequence of bits but don't increase input pointer. <code>bitCount</code> must be less or equal</td>
</tr>
</tbody>
</table>
16 and if this call succeeds, you must drop at least n - 8 bits in the next call.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reset</strong></td>
<td>Resets state and empties internal buffers</td>
</tr>
<tr>
<td><strong>SetInput</strong></td>
<td>Add more input for consumption. Only call when IsNeedingInput returns true</td>
</tr>
<tr>
<td><strong>SkipToByteBoundary</strong></td>
<td>Skips to the next byte boundary.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
<td>Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

**See Also**

ICSharpCode SharpZipLib Class Library
StreamManipulator.CopyBytes Method

Copies bytes from input buffer to output buffer starting at output[offset]. You have to make sure, that the buffer is byte aligned. If not enough bytes are available, copies fewer bytes.

```csharp
public int CopyBytes(
    byte[] output,
    int offset,
    int length
);
```

Parameters

- **output**
  - The buffer to copy bytes to.

- **offset**
  - The offset in the buffer at which copying starts

- **length**
  - The length to copy, 0 is allowed.

Return Value

The number of bytes copied, 0 if no bytes were available.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>Length is less than zero</td>
</tr>
<tr>
<td>InvalidOperationException</td>
<td>Bit buffer isn't byte aligned</td>
</tr>
</tbody>
</table>

See Also

ICSharpCode SharpZipLib Class Library
StreamManipulator.DropBits Method

Drops the next n bits from the input. You should have called PeekBits with a bigger or equal n before, to make sure that enough bits are in the bit buffer.

```csharp
public void DropBits(
    int bitCount
);
```

See Also

Get the next n bits and increases input pointer. This is equivalent to PeekBits followed by DropBits, except for correct error handling.

```csharp
public int GetBits(int bitCount);
```

**Parameters**

- **bitCount**
  The number of bits to retrieve.

**Return Value**

- the value of the bits, or -1 if not enough bits available.

**See Also**

- StreamManipulator Class
ICSharpCode SharpZipLib Class Library
Get the next sequence of bits but don't increase input pointer. bitCount must be less or equal 16 and if this call succeeds, you must drop at least n - 8 bits in the next call.

```csharp
public int PeekBits(int bitCount);
```

Parameters

- `bitCount` The number of bits to peek.

Return Value

- the value of the bits, or -1 if not enough bits available. */

See Also

ICSharpCode SharpZipLib Class Library
StreamManipulator.Reset Method

Resets state and empties internal buffers

```csharp
public void Reset();
```

See Also

ICSharpCode SharpZipLib Class Library
StreamManipulator.SetInput Method

Add more input for consumption. Only call when IsNeedingInput returns true

```csharp
public void SetInput(
    byte[] buffer,
    int offset,
    int count
);
```

Parameters

- `buffer`  
  data to be input

- `offset`  
  offset of first byte of input

- `count`  
  number of bytes of input to add.

See Also

- StreamManipulator Class
ICSharpCode SharpZipLib Class Library
StreamManipulator.SkipToByteBoundary Method

Skips to the next byte boundary.

```csharp
public void SkipToByteBoundary();
```

See Also

- [StreamManipulator Class](#)
ICSharpCode SharpZipLib Class Library
ICSharpCode.SharpZipLib Hierarchy

System.Object
System.Exception ---- System.Runtime.Serialization.ISerializable
System.ApplicationException
ICSharpCode.SharpZipLib.SharpZipBaseException

See Also
ICSharpCode.SharpZipLib Namespace
ICSharpCode SharpZipLib Class Library
ICSharpCode.SharpZipLib.BZip2 Hierarchy

System.Object
ICSharpCode.SharpZipLib.BZip2.BZip2
ICSharpCode.SharpZipLib.BZip2.BZip2Constants
System.Exception ---- System.Runtime.Serialization.ISerializable
System.ApplicationException
ICSharpCode.SharpZipLib.SharpZipBaseException
ICSharpCode.SharpZipLib.BZip2.BZip2Exception
System.MarshalByRefObject
System.IO.Stream ---- System.IDisposable
ICSharpCode.SharpZipLib.BZip2.BZip2InputStream
ICSharpCode.SharpZipLib.BZip2.BZip2OutputStream

See Also
ICSharpCode.SharpZipLib.BZip2 Namespace
ICSharpCode.SharpZipLib.Checksums Hierarchy

System.Object
ICSharpCode.SharpZipLib.Checksums.IChecksum
ICSharpCode.SharpZipLib.Checksums.IChecksum
ICSharpCode.SharpZipLib.Checksums.IChecksum
ICSharpCode.SharpZipLib.Checksums.IChecksum

See Also
ICSharpCode.SharpZipLib.Checksums Namespace
ICSharpCode SharpZipLib Class Library
### ISharpCode.SharpZipLib.Core Hierarchy

- `System.Object`
- `ICSharpCode.SharpZipLib.Core.INameTransform`
- `ICSharpCode.SharpZipLib.Core.IScanFilter`
- `ICSharpCode.SharpZipLib.Core.NameFilter` ----
- `ICSharpCode.SharpZipLib.Core.IScanFilter`
- `ICSharpCode.SharpZipLib.Core.IScanFilter`
- `ICSharpCode.SharpZipLib.Core.NameAndSizeFilter`
- `ICSharpCode.SharpZipLib.Core.ScanFailureEventArgs`
- `System.Delegate` ---- `System.ICloneable`, `System.Runtime.Serialization.ISerializable`
- `System.MulticastDelegate`
- `System.EventArgs`

#### See Also

- `ICSharpCode.SharpZipLib.Core Namespace`
ICSharpCode SharpZipLib Class Library
ICSharpCode.SharpZipLib.Encryption Hierarchy

System.Object
System.IDisposable

See Also
ICSharpCode.SharpZipLib.Encryption Namespace
ICSharpCode SharpZipLib Class Library
ICSharpCode.SharpZipLib.GZip Hierarchy

System.Object
ICSharpCode.SharpZipLib.GZip.GZipConstants
System.Exception ---- System.Runtime.Serialization.ISerializable
System.ApplicationException
ICSharpCode.SharpZipLib.SharpZipBaseException
ICSharpCode.SharpZipLib.GZip.GZipException
System.MarshalByRefObject
System.IO.Stream ---- System.IDisposable
ICSharpCode.SharpZipLib.GZip.GZipOutputStream
ICSharpCode.SharpZipLib.GZip.GZipInputStream

See Also
ICSharpCode.SharpZipLib.GZip Namespace
ICSharpCode SharpZipLib Class Library
ICSharpCode.SharpZipLib.Tar Hierarchy

System.Object
ICSharpCode.SharpZipLib.Tar.TarArchive ---- System.IDisposable
ICSharpCode.SharpZipLib.Tar.TarHeader ---- System.ICloneable
ICSharpCode.SharpZipLib.Tar.TarInputStream.EntryFactoryAdapter -

System.Delegate ---- System.ICloneable,
System.Runtime.Serialization.ISerializable
System.MulticastDelegate
System.Exception ---- System.Runtime.Serialization.ISerializable
System.ApplicationException
ICSharpCode.SharpZipLib.SharpZipBaseException
ICSharpCode.SharpZipLib.Tar.TarException
ICSharpCode.SharpZipLib.Tar.InvalidHeaderException
System.MarshalByRefObject
System.IO.Stream ---- System.IDisposable

See Also
ICSharpCode.SharpZipLib.Tar Namespace

System.Object
ICSharpCode.SharpZipLib.Zip.IArchiveStorage
ICSharpCode.SharpZipLib.Zip.IArchiveStorage
ICSharpCode.SharpZipLib.Zip.IStaticDataSource
System.Collections.IEnumerable, System.IDisposable
ICSharpCode.SharpZipLib.Core.INameTransform
System.Delegate ---- System.ICloneable,
System.Runtime.Serialization.ISerializable

System.MulticastDelegate
System.EventArgs
System.Exception ---- System.Runtime.Serialization.ISerializable
System.ApplicationException
ICSharpCode.SharpZipLib.SharpZipBaseException
System.MarshalByRefObject
System.IO.Stream ---- System.IDisposable
System.ValueType
System.Enum ---- System.IComparable, System.IConvertible, System.IFormattable

See Also
ICSharpCode SharpZipLib Class Library

System.Object
System.ValueType
System.Enum ---- System.IComparable, System.IConvertible, System.IFormattable

See Also
ICSharpCode SharpZipLib Class Library

System.Object
System.MarshalByRefObject
System.IO.Stream ---- System.IDisposable

See Also