**SharpZip Compression Library**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## I SharpCode.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>
SharpZip Compression Library
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.


**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.

**Requirements**

**Namespace:** ICSHarpCode.SharpZipLib

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

**See Also**

SharpZipBaseException Members | ICSHarpCode.SharpZipLib Namespace
SharpZip Compression Library
### SharpZipBaseException Members

#### SharpZipBaseException overview

**Public Instance Constructors**

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

**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/Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>GetObjectData</strong> (inherited from <code>Exception</code>)</td>
<td>When overridden in a derived class, sets the <code>SerializationInfo</code> with information about the exception.</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>ToString</strong> (inherited from <code>Exception</code>)</td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HResult</strong> (inherited from <code>Exception</code>)</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 <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

- [SharpZipBaseException Class](#) | [ICSharpCode.SharpZipLib Namespace](#)
SharpZip Compression Library
SharpZipBaseException Constructor

Initializes a new instance of the SharpZipLibraryException class.

Overload List

Initializes a new instance of the SharpZipLibraryException class.

public SharpZipBaseException();

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

public SharpZipBaseException(string);

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

public SharpZipBaseException(string, Exception);

See Also

SharpZipBaseException Class | ICSharpCode.SharpZipLib Namespace
SharpZip Compression Library
SharpZipBaseException Constructor ()

Initializes a new instance of the SharpZipLibraryException class.

```java
public SharpZipBaseException();
```

See Also

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

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

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

See Also

SharpZipBaseException Class | ICSharpCode.SharpZipLib Namespace | SharpZipBaseException Constructor Overload List
SharpZip Compression Library
SharpZipBaseException Constructor (String, Exception)

Initializes a new instance of the SharpZipLibraryException 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  |  ISharpCode.SharpZipLib Namespace  |  SharpZipBaseException Constructor Overload List
## ISharpCode.SharpZipLib.BZip2 Namespace

### Namespace hierarchy

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BZip2</strong></td>
<td>Does all the compress and decompress pre-operation stuff. Sets up the streams and file header characters. Uses multiply overloaded methods to call for the compress/decompress.</td>
</tr>
<tr>
<td><strong>BZip2Constants</strong></td>
<td>Defines internal values for both compression and decompression</td>
</tr>
<tr>
<td><strong>BZip2Exception</strong></td>
<td>BZip2Exception represents exceptions specific to Bzip2 algorithm</td>
</tr>
<tr>
<td><strong>BZip2InputStream</strong></td>
<td>An input stream that decompresses files in the BZip2 format</td>
</tr>
<tr>
<td><strong>BZip2OutputStream</strong></td>
<td>An output stream that compresses into the BZip2 format including file header chars into another stream.</td>
</tr>
</tbody>
</table>
SharpZip Compression Library
BZip2 Class

Does all the compress and decompress pre-operation stuff. Sets up the streams and file header characters. Uses multiply overloaded methods to call for the compress/decompress.

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
# BZip2 Members

## BZip2 overview

### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Compress</code></td>
<td>Compress <code>instream</code> sending result to <code>outputstream</code></td>
</tr>
<tr>
<td><code>Decompress</code></td>
<td>Decompress <code>instream</code> writing decompressed data to <code>outstream</code></td>
</tr>
</tbody>
</table>

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BZip2 Constructor</code></td>
<td>Initializes a new instance of the BZip2 class.</td>
</tr>
</tbody>
</table>

### 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>ToString</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](#)
SharpZip Compression Library
BZip2 Constructor

Initializes a new instance of the BZip2 class.

public BZip2();

See Also

BZip2 Class | ISharpCode.SharpZipLib.BZip2 Namespace
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 <code>instream</code> sending result to <code>outputstream</code></td>
</tr>
<tr>
<td>Decompress</td>
<td>Decompress <code>instream</code> writing decompressed data to <code>outstream</code></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

BZip2 Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
### BZip2.Compress Method

Compress *instream* sending result to *outputstream*

```csharp
public static void Compress(
    Stream instream,
    Stream outstream,
    int blockSize
);
```

See Also

- [BZip2 Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
SharpZip Compression Library
BZip2.Decompress Method

Decompress *instream* writing decompressed data to *outstream*

```csharp
public static void Decompress(
    Stream instream,
    Stream outstream
);
```

See Also

BZip2 Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression 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

```csharp
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: ISharpCode.SharpZipLib.BZip2


See Also

BZip2Constants Members | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2Constants Members

**BZip2Constants overview**

**Public Static Fields**

<table>
<thead>
<tr>
<th>Variable</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>

**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><strong>Get Type</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
- ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2Constants Fields

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><strong>baseBlockSize</strong></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><strong>G_SIZE</strong></td>
<td>Backend constant</td>
</tr>
<tr>
<td><strong>MAX_ALPHA_SIZE</strong></td>
<td>Backend constant</td>
</tr>
<tr>
<td><strong>MAX_CODE_LEN</strong></td>
<td>Backend constant</td>
</tr>
<tr>
<td><strong>MAX_SELECTORS</strong></td>
<td>Backend constant</td>
</tr>
<tr>
<td><strong>N_GROUPS</strong></td>
<td>Backend constant</td>
</tr>
<tr>
<td><strong>N_ITERS</strong></td>
<td>Backend constant</td>
</tr>
<tr>
<td><strong>NUM_OVERSHOOT_BYTES</strong></td>
<td>Backend constant</td>
</tr>
<tr>
<td><strong>rNums</strong></td>
<td>Random numbers used to randomise repetitive blocks</td>
</tr>
<tr>
<td><strong>RUNA</strong></td>
<td>Backend constant</td>
</tr>
<tr>
<td><strong>RUNB</strong></td>
<td>Backend constant</td>
</tr>
</tbody>
</table>

**See Also**

[BZip2Constants Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
SharpZip Compression 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 static readonly int baseBlockSize;
```

See Also

BZip2Constants Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2Constants.G_SIZE Field

Backend constant

```csharp
public static readonly int G_SIZE;
```

See Also

- [BZip2Constants Class](#)
- [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
BZip2Constants.MAX_ALPHA_SIZE Field

Backend constant

```csharp
public static readonly int MAX_ALPHA_SIZE;
```

See Also

BZip2Constants Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2Constants.MAX_CODE_LEN Field

Backend constant

```
public static readonly int MAX_CODE_LEN;
```

See Also

[ BZip2Constants Class ]  |  [ ISharpCode.SharpZipLib.BZip2 Namespace ]
BZip2Constants.MAX_SELECTORS Field

Backend constant

```csharp
public static readonly int MAX_SELECTORS;
```

See Also

BZip2Constants Class | ICSharpCode.SharpZipLib.BZip2 Namespace
BZip2Constants.N_GROUPS Field

Backend constant

```csharp
public static readonly int N_GROUPS;
```

See Also

BZip2Constants Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2Constants.N_ITERS Field

Backend constant

```csharp
public static readonly int N_ITERS;
```

See Also

[BZip2Constants Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
SharpZip Compression Library
BZip2Constants.NUM_OVERSHOOT_BYTES Field

Backend constant

```csharp
public static readonly int NUM_OVERSHOOT_BYTES;
```

See Also

BZip2Constants Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2Constants.rNums Field

Random numbers used to randomise repetitive blocks

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

See Also

[BZip2Constants Class] | [ICSharpCode.SharpZipLib.BZip2 Namespace]
BZip2Constants.RUNA Field

Backend constant

```csharp
public static readonly int RUNA;
```

See Also

- [BZip2Constants Class](#)
- [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
SharpZip Compression Library
BZip2Constants.RUNB Field

Backend constant

```csharp
public static readonly int RUNB;
```

See Also

[ICSharpCode.SharpZipLib.BZip2 Namespace](#)
SharpZip Compression 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
SharpZip Compression Library
# BZip2Exception Members

## BZip2Exception overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BZip2Exception</code></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><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</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 Properties

<table>
<thead>
<tr>
<th>Property</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

BZip2Exception Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2Exception Constructor

Initialise a new instance of BZip2Exception.

Overload List

Initialise a new instance of BZip2Exception.

public BZip2Exception();

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

public BZip2Exception(string);

See Also

BZip2Exception Class | ICSharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2Exception Constructor ()

Initialise a new instance of BZip2Exception.

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

See Also

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

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

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

Parameters

- **message**
  - The message describing the error.

See Also

- [BZip2Exception Class](#)
- [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
- [BZip2Exception Constructor Overload List](#)
SharpZip Compression 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.

System.Object   System.MarshalByRefObject
                System.IO.Stream
                ICSharpCode.SharpZipLib.BZip2.BZip2InputStream

```
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


See Also

BZip2InputStream Members | ICSharpCode.SharpZipLib.BZip2 Namespace
# BZip2InputStream Members

## BZip2InputStream overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BZip2InputStream 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>CanRead</td>
<td>Gets a value indicating if the 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. This property always returns false</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 streams position. Setting the position is not supported and will throw a NotSupportedException</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>(inherited from Stream) Begins an asynchronous read operation.</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>(inherited from Stream) Begins an asynchronous write operation.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the stream, releasing any associated resources.</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>(inherited from) Creates an object that contains all the relevant information required to</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>MarshalByRefObject</td>
<td>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>Flushes the stream.</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>✷ 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 sequence of bytes and advances the read position by one byte.</td>
</tr>
<tr>
<td>✷ ReadByte</td>
<td>Read a byte from stream advancing position</td>
</tr>
<tr>
<td>✷ Seek</td>
<td>Set the streams position. This operation is not supported and will throw a NotSupportedException</td>
</tr>
<tr>
<td>✷ SetLength</td>
<td>Sets the length of this stream to the given value. 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><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 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><code>WriteByte</code></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>
<tr>
<td><code>CreateWaitHandle</code> (inherited from Stream)</td>
<td>Allocates a <code>WaitHandle</code> object.</td>
</tr>
<tr>
<td><code>Finalize</code> (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><code>MemberwiseClone</code> (inherited from Object)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

## See Also

BZip2InputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
Construct instance for reading from stream

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

Parameters

*stream*  
Data source

See Also

[BZip2InputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
SharpZip Compression Library
BZip2InputStream Properties

The properties of the `BZip2InputStream` class are listed below. For a complete list of `BZip2InputStream` class members, see the `BZip2InputStream Members` topic.

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Gets a value indicating if the 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. This property always returns false</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 streams position. Setting the position is not supported and will throw a NotSupportException</td>
</tr>
</tbody>
</table>

See Also

`BZip2InputStream Class` | `ICSharpCode.SharpZipLib.BZip2 Namespace`
SharpZip Compression 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
SharpZip Compression 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
SharpZip Compression 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
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
SharpZip Compression Library
BZip2InputStream.Position Property

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

```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
SharpZip Compression Library
**BZip2InputStream Methods**

The methods of the `BZip2InputStream` class are listed below. For a complete list of `BZip2InputStream` class members, see the [BZip2InputStream 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 <code>Stream</code>) Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong></td>
<td>(inherited from <code>Stream</code>) 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>(inherited from <code>MarshalByRefObject</code>) 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 <code>Stream</code>) Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong></td>
<td>(inherited from <code>Stream</code>) Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>(inherited from <code>Object</code>) Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the stream.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>(inherited from <code>Object</code>) 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>(inherited from <code>MarshalByRefObject</code>) 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 <code>Object</code>) Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td>Read</td>
<td>Read a sequence of bytes and advances the read position by one byte.</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Read a byte from stream advancing position</td>
</tr>
<tr>
<td>Seek</td>
<td>Set the streams position. This operation is not supported and will throw a NotSupportedException.</td>
</tr>
<tr>
<td>SetLength</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>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td>Write</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>WriteByte</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>CreateWaitHandle (inherited from Stream)</td>
<td>Allocates a WaitHandle object.</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>
</tbody>
</table>
MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object.

See Also

BZip2InputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2InputStream.Close Method

Closes the stream, releasing any associated resources.

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

See Also

- [BZip2InputStream Class](#)
- [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
BZip2InputStream.Flush Method

Flushes the stream.

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

See Also

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

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

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

Parameters

- **b**
  
  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](https://icsharpcode.net/SharpZipLib.BZip2/Namespace) | [ICSharpCode.SharpZipLib.BZip2 Namespace](https://icsharpcode.net/SharpZipLib.BZip2/Namespace)
SharpZip Compression 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 | ICSharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression 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
SharpZip Compression Library
BZip2InputStream.SetLength Method

Sets the length of this stream to the given value. This operation is not supported and will throw a NotSupportedException or ArgumentException.

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

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
SharpZip Compression Library
BZip2InputStream.Write Method

Wrote 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[] array, 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
SharpZip Compression 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 val
);
```

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
SharpZip Compression Library
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.

**System.Object**  **System.MarshalByRefObject**
**System.IO.Stream**
**ICSharpCode.SharpZipLib.BZip2.BZip2OutputStream**

```csharp
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**: IICSharpCode.SharpZipLib.BZip2


**See Also**

BZip2OutputStream Members | IICSharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
# BZip2OutputStream Members

## BZip2OutputStream overview

### Public Instance Constructors

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

### Public Instance Properties

<table>
<thead>
<tr>
<th><strong>CanRead</strong></th>
<th>Gets a value indicating whether the current stream supports reading</th>
</tr>
</thead>
<tbody>
<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>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 current position of this stream.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th><strong>BeginRead</strong> (inherited from Stream)</th>
<th>Begins an asynchronous read operation.</th>
</tr>
</thead>
<tbody>
<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</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>EndRead</code> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><code>EndWrite</code> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><code>Equals</code> (inherited from Object)</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>Flush output buffers</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>GetLifetimeService</code> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><code>GetType</code> (inherited from Object)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>InitializeLifetimeService</code> (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><code>Read</code></td>
<td>Read a block of bytes</td>
</tr>
<tr>
<td><code>ReadByte</code></td>
<td>Read a byte from the stream advancing the position.</td>
</tr>
<tr>
<td><code>Seek</code></td>
<td>Sets the current position of this stream to the given value.</td>
</tr>
<tr>
<td><code>SetLength</code></td>
<td>Sets the length of this stream to the given value.</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>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</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></td>
<td>Allocates a <a href="#">WaitHandle</a> object.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Free any resources and other cleanup before garbage collection reclaims memory</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <a href="#">Object</a>.</td>
</tr>
</tbody>
</table>

See Also

[BZip2OutputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
SharpZip Compression Library
**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)
[BZip2OutputStream Class](ICSharpCode.SharpZipLib.BZip2)
SharpZip Compression Library
BZip2OutputStream Constructor (Stream)

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 | ISharpCode.SharpZipLib.BZip2 Namespace | BZip2OutputStream Constructor Overload List
SharpZip Compression 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](#)
SharpZip Compression Library
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><strong>CanRead</strong></td>
<td>Gets a value indicating whether the current 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>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 current position of this stream.</td>
</tr>
</tbody>
</table>

**See Also**

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

Gets a value indicating whether the current stream supports reading

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

See Also

[BZip2OutputStream Class](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
SharpZip Compression Library
BZip2OutputStream.CanSeek Property

Gets a value indicating whether the current stream supports seeking

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

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2OutputStream.CanWrite Property

Gets a value indicating whether the current stream supports writing.

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

See Also

[BZip2OutputStream Class] | [ICSharpCode.SharpZipLib.BZip2 Namespace]
SharpZip Compression Library
BZip2OutputStream.Length Property

Gets the length in bytes of the stream

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

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression 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
SharpZip Compression 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><strong>BeginRead</strong> (inherited from <strong>Stream</strong>)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from <strong>Stream</strong>)</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 <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>Flush</strong></td>
<td>Flush output buffers</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>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>Finalize</strong></td>
<td>Free any resources and other cleanup before garbage collection reclaims memory</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**
SharpZip Compression 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
SharpZip Compression Library
Free any resources and other cleanup before garbage collection reclaims memory

```csharp
protected override void Finalize();
```

See Also

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

Flush output buffers

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

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2OutputStream.Read Method

Read a block of bytes

```csharp
public override int Read(byte[] b, int off, int len);
```

See Also

BZip2OutputStream Class | ICSharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2OutputStream.ReadByte Method

Read a byte from the stream advancing the position.

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

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
**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](#) | [ICSharpCode.SharpZipLib.BZip2 Namespace](#)
SharpZip Compression Library
BZip2OutputStream.SetLength Method

Sets the length of this stream to the given value.

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

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2OutputStream.Write Method

Write a block of bytes to the stream

```csharp
public override void Write(
    byte[] buf,
    int off,
    int len
);
```

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
BZip2OutputStream.WriteByte Method

Write a byte to the stream.

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

See Also

BZip2OutputStream Class | ISharpCode.SharpZipLib.BZip2 Namespace
SharpZip Compression Library
**ICSharpCode.SharpZipLib.Checksums Namespace**

**Namespace hierarchy**

**Classes**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adler32</strong></td>
<td>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 of the Fletcher algorithm, used in the ITU-T X.224 / ISO 8073 standard. The algorithm is composed of two sums accumulated per byte: s1 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. 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 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 checksum unchanged. (The Fletcher checksum uses 255, which is not prime and also makes the Fletcher check insensitive to single byte changes.) The sum s1 is initialized to 1 instead of zero to make the length part of s2, so that the length does not have to be checked separately. (Any sequence of zeroes has a Fletcher checksum of zero.)</td>
</tr>
<tr>
<td><strong>Crc32</strong></td>
<td>Generate a table for a byte-wise 32-bit CRC calculation. Polynomials over 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 b as a polynomial also with the lowest power in the most significant bit (so b = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1) then the CRC of p and b is the polynomial that is the result of performing b * x^11 + x^10 + x^8 + x^7 + x^6 + x^5 + x^2 + x + 1 and reducing it modulo p. This means that the CRC of b with respect to p is the result of subtracting p from b * x^11 + x^10 + x^8 + x^7 + x^6 + x^5 + x^2 + x + 1 and then reducing the result modulo p. This is also known as the CRC32 algorithm and is used in many popular data formats, including ZIP, many ZIP-like compressed files, gzip, and others.</td>
</tr>
</tbody>
</table>
polynomial $x^7 + x^3 + x + 1)$, then the CRC is $(q \cdot x^{32}) \mod p$, 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 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 by one). We start with the highest power (least significant bit) of $q$ and repeat for all eight bits. This calculation 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.

<table>
<thead>
<tr>
<th>StrangeCRC</th>
<th>Bzip2 checksum algorithm</th>
</tr>
</thead>
</table>

**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 <strong>getValue</strong>. The complete checksum object can also be reset so it can be used again with new data.</td>
</tr>
</tbody>
</table>
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. System.Object ICSharpCode.SharpZipLib.Checksums.Adler32

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

**Namespace:** [ICSharpCode.SharpZipLib.Checksums](https://icsharpcode.net/SharpZipLib/)

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

See Also

[Adler32 Members](https://icsharpcode.net/SharpZipLib/) | [ICSharpCode.SharpZipLib.Checksums Namespace](https://icsharpcode.net/SharpZipLib/)


SharpZip Compression Library
# Adler32 Members

## Adler32 overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adler32 Constructor</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>Value</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>Equals</td>
<td>Determines whether the specified &lt;code&gt;Object&lt;/code&gt; is equal to the current &lt;code&gt;Object&lt;/code&gt;.</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 &lt;code&gt;Type&lt;/code&gt; of the current instance.</td>
</tr>
<tr>
<td>Reset</td>
<td>Resets the Adler32 checksum to the initial value.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a &lt;code&gt;String&lt;/code&gt; that represents the current &lt;code&gt;Object&lt;/code&gt;.</td>
</tr>
<tr>
<td>Update</td>
<td>Overloaded. Updates the checksum with the bytes taken from the array.</td>
</tr>
</tbody>
</table>

### See Also

SharpZip Compression 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

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

See Also

Adler32 Class | ICSHarpCode.SharpZipLib.Checksums Namespace
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
SharpZip Compression Library
Adler32 Methods

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></td>
<td>Determines whether the specified <a href="#">Object</a> is equal to the current <a href="#">Object</a>.</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 <a href="#">Type</a> 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></td>
<td>Returns a <a href="#">String</a> that represents the current <a href="#">Object</a>.</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

SharpZip Compression Library
Adler32.Reset Method

Resets the Adler32 checksum to the initial value.

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

Implements

IChecksum.Reset

See Also

Adler32 Class | ISharpCode.SharpZipLib.Checksums Namespace
**Adler32.Update Method**

Updates the checksum with an array of bytes.

**Overload List**

Updates the checksum with an array of bytes.

```java
public void Update(byte[]);
```

Updates the checksum with the bytes taken from the array.

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

Updates the checksum with the byte b.

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

**See Also**

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 Class](#)  
- [ICSharpCode.SharpZipLib.Checksums Namespace](#)  
- [Adler32.Update Overload List](#)
SharpZip Compression Library
Adler32.Update Method (Byte[], Int32, Int32)

Updates the checksum with the bytes taken from the array.

```csharp
public void Update(
    byte[] buf,
    int off,
    int len
);
```

Parameters

- `buf` an array of bytes
- `off` the start of the data used for this update
- `len` the number of bytes to use for this update

Implements

- `IChecksum.Update`

See Also

SharpZip Compression Library
Adler32.Update Method (Int32)

Updates the checksum with the byte \( b \).

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

Parameters

- \( bval \)
  
The data value to add. The high byte of the int is ignored.

Implements

- IChecksum.Update

See Also

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 \(x^{32} \mod p = x^{26} + \ldots + 1\)), 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: ISharpCode.SharpZipLib (in
ICSharpCode.SharpZipLib.dll)

See Also

Crc32 Members | ICSharpCode.SharpZipLib.Checksums Namespace
SharpZip Compression Library
## Crc32 Members

### Crc32 overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Crc32 Constructor</strong></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><strong>Value</strong></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><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 CRC32 data checksum as if no update was ever called.</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. Adds the byte array to the data checksum.</td>
</tr>
</tbody>
</table>

### See Also

[Crc32 Class](#) | [ICSharpCode.SharpZipLib.Checksums Namespace](#)
SharpZip Compression Library
Crc32 Constructor

Initializes a new instance of the Crc32 class.

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

See Also

SharpZip Compression 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><strong>Value</strong></td>
<td>Returns the CRC32 data checksum computed so far.</td>
</tr>
</tbody>
</table>

**See Also**

[Crc32 Class](#) | [ICSharpCode.SharpZipLib.Checksums Namespace](#)
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
SharpZip Compression Library
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 <strong>Object</strong> is equal to the current <strong>Object</strong>.</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>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 <strong>String</strong> that represents the current <strong>Object</strong>.</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] | [ICSharpCode.SharpZipLib.Checksums Namespace]
SharpZip Compression Library
Crc32.Reset Method

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

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

Implements

IChecksum.Reset

See Also

Crc32 Class | ISharpCode.SharpZipLib.Checksums Namespace
SharpZip Compression Library
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
SharpZip Compression 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

SharpZip Compression Library
Crc32.Update Method (Byte[], Int32, Int32)

 Adds the byte array to the data checksum.

```csharp
public void Update(
    byte[] buf,
    int off,
    int len
);
```

**Parameters**

- `buf`  
  the buffer which contains the data

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

- `len`  
  the length of the data

**Implements**

- `IChecksum.Update`

**See Also**

- `Crc32 Class`
- `Crc32.Update Overload List`
SharpZip Compression Library
## Crc32.Update Method (Int32)

Updates the checksum with the int bval.

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

### Parameters

- **bval**
  - The byte is taken as the lower 8 bits of bval

### Implements

- `IChecksum.Update`

### See Also

- [Crc32 Class](#)
- [ICSharpCode.SharpZipLib.Checksums Namespace](#)
- [Crc32.Update Overload List](#)
SharpZip Compression 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.

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 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 (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. The Adler-32 checksum 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, then 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:

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

Polynomials over \( GF(2) \) are represented in binary, one bit per coefficient, with the lowest powers in the most significant bit. Then 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 also with the lowest power in the most significant bit (so the polynomial \( x^7 + x^3 + x + 1 \)), then the CRC is \((q \cdot x^{32}) \mod p \) means 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 \( x^{32} \mod p = p + x^{32} = x^{26} + \ldots + 1 \)), 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, 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**


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

**See Also**

SharpZip Compression Library
IChecksum Members

IChecksum overview

Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</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>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 | ICSHarpCode.SharpZipLib.Checksums

Namespace
SharpZip Compression 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

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

See Also

IChecksum Interface | ISharpCode.SharpZipLib.Checksums Namespace
SharpZip Compression Library
IChecksum.Value Property

Returns the data checksum computed so far.

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

See Also

[IChecksum Interface](#) | [ICSharpCode.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><img src="image" alt="Reset" /></td>
<td>Resets the data checksum as if no update was ever called.</td>
</tr>
<tr>
<td><img src="image" alt="Update" /></td>
<td>Overloaded. Adds the byte array to the data checksum.</td>
</tr>
</tbody>
</table>

See Also

IChecksum Interface | ISharpCode.SharpZipLib.Checksums Namespace
SharpZip Compression Library
IChecksum.Reset Method

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

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

See Also

IChecksum Interface | ISharpCode.SharpZipLib.Checksums Namespace
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 | ICSharpCode.SharpZipLib.Checksums Namespace
SharpZip Compression 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

- IChecksum Interface
- ISharpCode.SharpZipLib.Checksums Namespace
- IChecksum.Update Overload List
SharpZip Compression Library
IChecksum.Update Method (Byte[], Int32, Int32)

Adds the byte array to the data checksum.

```csharp
void Update(
    byte[] buf,
    int off,
    int len
);
```

Parameters

- `buf`  
  the buffer which contains the data

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

- `len`  
  the length of the data

See Also

- [IChecksum Interface](#)  
- [ICSharpCode.SharpZipLib.Checksums Namespace](#)  
- [IChecksum.Update Overload List](#)
SharpZip Compression Library
IChecksum.Update Method (Int32)

Adds one byte to the data checksum.

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

Parameters

*bval*

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

See Also

SharpZip Compression 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
SharpZip Compression Library
StrangeCRC Members

**StrangeCRC overview**

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StrangeCRC Constructor</strong></td>
<td>Initialise a default instance of StrangeCRC</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>Get the current Crc 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 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>Reset the state of Crc.</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. 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> (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 <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]
SharpZip Compression Library
StrangeCRC Constructor

 Initialise a default instance of StrangeCRC

\[
\text{public StrangeCRC();}
\]

See Also

StrangeCRC Class | ISharpCode.SharpZipLib.Checksums Namespace
SharpZip Compression 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 | ISharpCode.SharpZipLib.Checksums Namespace
SharpZip Compression Library
StrangeCRC.Value Property

Get the current Crc value.

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

Implements

IChecksum.Value

See Also

StrangeCRC Class | ISharpCode.SharpZipLib.Checksums Namespace
SharpZip Compression 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> (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>Reset the state of Crc.</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. 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> (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

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

Reset the state of Crc.

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

Implements

IChecksum.Reset

See Also

StrangeCRC Class | ISharpCode.SharpZipLib.Checksums Namespace
SharpZip Compression Library
StrangeCRC.Update Method

Update Crc based on a block of data

Overload List

Update Crc based on a block of data

\[
\text{public void Update(byte[])};
\]

Update Crc based on a portion of a block of data

\[
\text{public void Update(byte[],int,int)};
\]

Update the Crc value.

\[
\text{public void Update(int)};
\]

See Also

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

Update Crc based on a block of data

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

Implements

IChecksum.Update

See Also

StrangeCRC Class | ISharpCode.SharpZipLib.Checksums Namespace | StrangeCRC.Update Overload List
SharpZip Compression Library
StrangeCRC.Update Method (Byte[], Int32, Int32)

Update Crc based on a portion of a block of data

```csharp
public void Update(
    byte[] buf,
    int off,
    int len
);
```

Parameters

- `buf`
  - block of data
- `off`
  - index of first byte to use
- `len`
  - number of bytes to use

Implements

- `IChecksum.Update`

See Also

SharpZip Compression Library
StrangeCRC.Update Method (Int32)

Update the Crc value.

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

**Parameters**

*inCh*

data update is based on

**Implements**

IChecksum.Update

**See Also**

StrangeCRC Class | ISharpCode.SharpZipLib.Checksums Namespace | StrangeCRC.Update Overload List
SharpZip Compression Library
### Namespace hierarchy

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileSystemScanner</td>
<td>FileSystemScanner provides facilities scanning of files and directories.</td>
</tr>
<tr>
<td>NameAndSizeFilter</td>
<td>NameAnsSizeFilter filters based on name and file size.</td>
</tr>
<tr>
<td>NameFilter</td>
<td>NameFilter is a string matching class which allows for both positive and</td>
</tr>
<tr>
<td></td>
<td>negative matching. A filter is a sequence of independent Regex regular</td>
</tr>
<tr>
<td></td>
<td>expressions separated by semi-colons ';'. Each expression can be prefixed</td>
</tr>
<tr>
<td></td>
<td>by a plus '+' sign or a minus '-' sign to denote the expression is intended</td>
</tr>
<tr>
<td></td>
<td>to include or exclude names. If neither a plus or minus sign is found</td>
</tr>
<tr>
<td></td>
<td>include is the default A given name is tested for inclusion before checking</td>
</tr>
<tr>
<td></td>
<td>exclusions. Only names matching an include spec and not matching an</td>
</tr>
<tr>
<td></td>
<td>exclude spec are deemed to match the filter. An empty filter matches any</td>
</tr>
<tr>
<td></td>
<td>name.</td>
</tr>
<tr>
<td>PathFilter</td>
<td>PathFilter filters directories and files by full path name.</td>
</tr>
</tbody>
</table>
**Interfaces**

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

**Delegates**

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

Event arguments for directories.

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

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

```csharp
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

SharpZip Compression 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. (inherited from ScanEventArgs)</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. (inherited from ScanEventArgs)</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. (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>GetType</td>
<td>Gets the Type of the current instance. (inherited from Object)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents the current Object. (inherited from 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**

SharpZip Compression 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

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

See Also

- `DirectoryEventArgs Class` | `ICSharpCode.SharpZipLib.Core Namespace`
SharpZip Compression Library
DirectoryEventArgs.HasMatchingFiles Property

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

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

See Also

SharpZip Compression Library
DirectoryFailureDelegate Delegate

Delegate invoked when a directory failure is detected.

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

Requirements

**Namespace:** ICSherpCode.SharpZipLib.Core

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

See Also

ICSherpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
Delegate invoked when a file failure is detected.

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

Requirements

**Namespace:** ICSHarpCode.SharpZipLib.Core

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

See Also

ICSSharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
FileSystemScanner Class

FileSystemScanner provides facilities scanning of files and directories.

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

System.Object

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: ISharpCode.SharpZipLib.Core


See Also

FileSystemScanner Members | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
## FileSystemScanner Members

### FileSystemScanner overview

### Public Instance Constructors

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

### Public Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectoryFailure</td>
<td>Delegate to invoke when a directory failure is detected.</td>
</tr>
<tr>
<td>FileFailure</td>
<td>Delegate to invoke when a file failure is detected.</td>
</tr>
<tr>
<td>ProcessDirectory</td>
<td>Delegate to invoke when a directory is processed.</td>
</tr>
<tr>
<td>ProcessFile</td>
<td>Delegate to invoke when a file is processed.</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>OnDirectoryFailure</td>
<td>Raise the DirectoryFailure event.</td>
</tr>
<tr>
<td>OnFileFailure</td>
<td>Raise the FileFailure event.</td>
</tr>
<tr>
<td>Event/Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OnProcessFile</td>
<td>Raise the ProcessFile event.</td>
</tr>
<tr>
<td>Scan</td>
<td>Scan a directory.</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

| **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

- [FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression Library
File\System\Scanner Constructor

Initialise a new instance of File\System\Scanner

**Overload List**

Initialise a new instance of File\System\Scanner

`public FileSystemScanner(IScanFilter);`

Initialise a new instance of File\System\Scanner

`public FileSystemScanner(IScanFilter,IScanFilter);`

Initialise a new instance of File\System\Scanner

`public FileSystemScanner(string);`

Initialise a new instance of File\System\Scanner

`public FileSystemScanner(string,string);`

**See Also**

File\System\Scanner Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
Initialise a new instance of `FileSystemScanner`

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

**Parameters**

`filter`

The file filter to apply when scanning.

**See Also**

SharpZip Compression Library
FileSystemScanner Constructor (String, String)

Initialise a new instance of FileSystemScanner

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

Parameters

`fileFilter`
- The file NameFilter filter to apply.

`directoryFilter`
- The directory NameFilter filter to apply.

See Also

SharpZip Compression Library
FileSystemScanner Constructor (IScanFilter)

Initialise a new instance of FileSystemScanner

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

Parameters

fileFilter
The file NameFilter filter to apply.

See Also

SharpZip Compression Library
FileSystemScanner Constructor (IScanFilter, IScanFilter)

Initialise a new instance of FileSystemScanner

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

**Parameters**

- `fileFilter`
  The file IScanFilter filter to apply.

- `directoryFilter`
  The directory IScanFilter filter to apply.

**See Also**

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

**See Also**

[FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression 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
SharpZip Compression Library
**FileSystemScanner.FileFailure Field**

Delegate to invoke when a file failure is detected.

```
public FileFailureDelegate FileFailure;
```

See Also

[FileSystemScanner Class] | [ICSharpCode.SharpZipLib.Core Namespace]
SharpZip Compression Library
FileSystemScanner.ProcessDirectory Field

Delegate to invoke when a directory is processed.

```csharp
```

See Also

FileSystemScanner Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
Delegate to invoke when a file is processed.

```csharp
public ProcessFileDelegate ProcessFile;
```

See Also

[FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression 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><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 DirectoryFailure event.</td>
</tr>
<tr>
<td><strong>OnFileFailure</strong></td>
<td>Raise the FileFailure event.</td>
</tr>
<tr>
<td><strong>OnProcessFile</strong></td>
<td>Raise the ProcessFile event.</td>
</tr>
<tr>
<td><strong>Scan</strong></td>
<td>Scan a directory.</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)</td>
<td>Creates a shallow copy of the</td>
</tr>
</tbody>
</table>
See Also

FileSystemScanner Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression 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](#)
SharpZip Compression Library
FileSystemScanner.OnFileFailure Method

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](#)
SharpZip Compression Library
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`
SharpZip Compression Library
Raise the ProcessFile event.

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

**Parameters**

- `file`  
  The file name.

**See Also**

- [FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression 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 do a single directory.

**See Also**

- [FileSystemScanner Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression 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.

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 name as per the Zip file convention.</td>
</tr>
</tbody>
</table>

Requirements


See Also

INameTransform Members | ICSharpCode.SharpZipLib.Core Namespace
SharpZip Compression 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 equivalent.</td>
</tr>
<tr>
<td>TransformFile</td>
<td>Given a file name determine the transformed equivalent.</td>
</tr>
</tbody>
</table>

See Also

INameTransform Interface | ICSharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
INameTransform Methods

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>TransformDirectory</td>
<td>Given a directory name determine the transformed equivalent.</td>
</tr>
<tr>
<td>TransformFile</td>
<td>Given a file name determine the transformed equivalent.</td>
</tr>
</tbody>
</table>

See Also

INameTransform Interface | ICSharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
**INameTransform.TransformDirectory Method**

Given a directory name determine the transformed equivalent.

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

**Parameters**

`name`

The name to transform.

**Return Value**

The transformed directory name

**See Also**

[INameTransform Interface](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression Library
Given a file name determine the transformed equivalent.

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

Parameters

- `name`
  - The name to transform.

Return Value

- The transformed name.

See Also

- `INameTransform` Interface | `ICSharpCode.SharpZipLib.Core Namespace`
SharpZip Compression Library
Scanning filters support these operations.

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>NameAndSizeFilter</td>
<td>NameAnsSizeFilter filters based on name and file size.</td>
</tr>
<tr>
<td>PathFilter</td>
<td>PathFilter filters directories and files by full path name.</td>
</tr>
</tbody>
</table>

Requirements

Namespace: ICSHarpCode.SharpZipLib.Core


See Also

IScanFilter Members | ICSHarpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
IScanFilter Members

IScanFilter overview

Public Instance Methods

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

See Also

IScanFilter Interface | ICSharpCode.SharpZipLib.Core Namespace
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 is 'matches' the filter.</td>
</tr>
</tbody>
</table>

See Also

[IScanFilter Interface](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression Library
IScanFilter.IsMatch Method

Test a name to see if '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` | `ICSharpCode.SharpZipLib.Core Namespace`
**SharpZip Compression Library**
NameAndSizeFilter filters based on name and file size.

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

ICSharpCode.SharpZipLib.Core.NameAndSizeFilter

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.

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</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>The maximum size for a file that will match this filter.</td>
</tr>
<tr>
<td>MinSize</td>
<td>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</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>IsMatch</td>
<td>Test a filename to see if it matches the filter.</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 other cleanup operations before the Object is reclaimed by</td>
</tr>
</tbody>
</table>


garbage collection.

| ![MemberwiseClone](inheritedinfromObject) | Creates a shallow copy of the current Object. |

See Also

NameAndSizeFilter Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
NameAndSizeFilter Constructor

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.SharpZipLib.Core Namespace
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>MaxSize</td>
<td>The maximum size for a file that will match this filter.</td>
</tr>
<tr>
<td>MinSize</td>
<td>The minimum size for a file that will match this filter.</td>
</tr>
</tbody>
</table>

See Also

NameAndSizeFilter Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
NameAndSizeFilter.MaxSize Property

The maximum size for a file that will match this filter.

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

See Also

NameAndSizeFilter Class | ICSharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
NameAndSizeFilter.MinSize Property

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](#)
**SharpZip Compression 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></td>
<td>(inherited from <strong>Object</strong>) Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>(inherited from <strong>Object</strong>) 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 <strong>Object</strong>) 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></td>
<td>(inherited from <strong>Object</strong>) 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>(inherited from <strong>Object</strong>) 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>(inherited from <strong>Object</strong>) Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

NameAndSizeFilter Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
NameAndSizeFilter.IsMatch Method

Test a filename to see if it matches the filter.

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

Parameters

fileName

The filename to test.

Return Value

True if the filter matches, false otherwise.

Implements

IScanFilter.IsMatch

See Also

NameAndSizeFilter Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
NameFilter Class

NameFilter is a string matching class which allows for both positive and negative matching. A filter is a sequence of independent Regex 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.


```csharp
public class NameFilter
```

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

NameFilter Members  |  ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression 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>Constructor</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](#)
SharpZip Compression Library
Construct an instance based on the filter expression passed

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

Parameters

- `filter`
  The filter expression.

See Also

- NameFilter Class  |  ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression 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>$ 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 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>$ Finalize (inherited from Object)</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**

- [NameFilter Class](#)
- [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression Library
NameFilter.IsExcluded Method

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

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

Parameters

testValue
The value to test.

Return Value
True if the value is excluded, false otherwise.

See Also

NameFilter Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
NameFilter.IsIncluded Method

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

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

Parameters

testValue  
The value to test.

Return Value

True if the value is included, false otherwise.

See Also

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

Test a value to see if it matches the filter.

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

Parameters

- `testValue`  
  The value to test.

Return Value

True if the value matches, false otherwise.

See Also

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

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

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

Parameters

- `e` The expression to test.

Return Value

True if expression is a valid `Regex` false otherwise.

See Also

- [NameFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression 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
SharpZip Compression Library
NameFilter.ToString Method

Convert this filter to its string equivalent.

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

Return Value

The string equivalent for this filter.

See Also

NameFilter Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
PathFilter filters directories and files by full path name.

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

PathFilter Class

```
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
SharpZip Compression Library
## PathFilter Members

### PathFilter overview

### Public Instance Constructors

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

PathFilter Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
PathFilter Constructor

Initialise a new instance of **PathFilter**.

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

**Parameters**

* filter
  
  The **NameFilter** filter expression to apply.

**See Also**

[PathFilter Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
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></td>
<td>(inherited from Object) Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</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 <strong>Type</strong> 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></td>
<td>(inherited from Object) 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>(inherited from Object) 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>(inherited from Object) Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### See Also

- PathFilter Class | ICShapCode.SharpZipLib.Core Namespace
SharpZip Compression 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` | `ICSharpCode.SharpZipLib.Core Namespace`
Delegate invoked when a directory is processed.

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

Requirements

**Namespace:** [ICSharpCode.SharpZipLib.Core](https://example.com)

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

See Also

[ICSharpCode.SharpZipLib.Core Namespace](https://example.com)
Delegate invoked when a file is processed.

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

**Requirements**

- **Namespace:** [ICSharpCode.SharpZipLib.Core](#)
- **Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

**See Also**

[ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression Library
ScanEventArgs Class

Event arguments for scanning.

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


```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


See Also

ScanEventArgs Members | ICSharpCode.SharpZipLib.Core Namespace
SharpZip Compression 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 <a href="#">ScanEventArgs</a></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 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>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>
</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 <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><strong>MemberwiseClone</strong>(inherited)</td>
<td>Creates a shallow copy of the</td>
</tr>
</tbody>
</table>
from Object) | current Object.

See Also

ScanEventArgs Class | ICSharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
ScanEventArgs Constructor

Initialise a new instance of ScanEventArgs

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

Parameters

`name`

See Also

ScanEventArgs Class | ICSharpCode.SharpZipLib.Core Namespace
SharpZip Compression 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>ContinueRunning</td>
<td>Get set a value indicating if scanning should continue or not.</td>
</tr>
<tr>
<td>Name</td>
<td>The name for this event.</td>
</tr>
</tbody>
</table>

See Also

ScanEventArgs Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression 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](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression Library
ScanEventArgs.Name Property

The name for this event.

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

See Also

[ScanEventArgs Class](index.html) | [ICSharpCode.SharpZipLib.Core Namespace](index.html)
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: ISharpCode.SharpZipLib.Core


See Also

ScanFailureEventArgs Members | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
## ScanFailureEventArgs Members

### ScanFailureEventArgs overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ScanFailureEventArgs Constructor</strong></td>
<td>Initialise a new instance of <strong>ScanFailureEventArgs</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 whether scanning should continue.</td>
</tr>
<tr>
<td><strong>Exception</strong></td>
<td>The applicable exception.</td>
</tr>
<tr>
<td><strong>Name</strong></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><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 from <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

[ScanFailureEventArgs Class](#) | [ICSharpCode.SharpZipLib.Core Namespace](#)
SharpZip Compression Library
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 | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
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](#)
SharpZip Compression Library
Get / set a value indicating whether scanning should continue.

public bool ContinueRunning {get; set;}

See Also

ScanFailureEventArgs Class | ICSharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
ScanFailureEventArgs.Exception Property

The applicable exception.

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

See Also

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

The applicable name.

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

See Also

ScanFailureEventArgs Class | ISharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
# ISharpCode.SharpZipLib.Encryption Namespace

## Namespace hierarchy

## 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>
SharpZip Compression 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](#).

```
```

### 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:** ISharpCode.SharpZipLib.Encryption

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

#### See Also

SharpZip Compression 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>Gets or sets the block size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td><strong>FeedbackSize</strong></td>
<td>Gets or sets the feedback size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td><strong>IV</strong></td>
<td>Gets or sets the initialization vector (IV) for the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>Gets or sets the secret key for the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>KeySize</strong></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>Gets the block sizes that are supported by the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>LegalKeySizes</strong></td>
<td>Gets the key sizes that are supported by the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>Mode</strong></td>
<td>Gets or sets the mode for operation of the symmetric algorithm.</td>
</tr>
<tr>
<td><strong>Padding</strong></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</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> (inherited from <strong>SymmetricAlgorithm</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> (inherited from <strong>SymmetricAlgorithm</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> (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> (inherited from <strong>SymmetricAlgorithm</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> (inherited from <strong>SymmetricAlgorithm</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> (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>
<tr>
<td><strong>ValidKeySize</strong> (inherited from <strong>SymmetricAlgorithm</strong>)</td>
<td>Determines whether the <strong>algorithm</strong>.</td>
</tr>
</tbody>
</table>
SymmetricAlgorithm) specified key size is valid for the current algorithm.

### Protected Instance Constructors

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

### Protected Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BlockSizeValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the block size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td>FeedbackSizeValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the feedback size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td>IVValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the initialization vector (IV) for the symmetric algorithm.</td>
</tr>
<tr>
<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>KeyValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the secret key for the symmetric algorithm.</td>
</tr>
<tr>
<td>LegalBlockSizesValue (inherited from SymmetricAlgorithm)</td>
<td>Specifies the block sizes that are supported by the symmetric algorithm.</td>
</tr>
<tr>
<td>LegalKeySizesValue (inherited from SymmetricAlgorithm)</td>
<td>Specifies the key sizes that are supported by the symmetric algorithm.</td>
</tr>
<tr>
<td>ModeValue (inherited from SymmetricAlgorithm)</td>
<td>Represents the cipher mode used in the symmetric algorithm.</td>
</tr>
<tr>
<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>Dispose (inherited from SymmetricAlgorithm)</td>
<td>Releases the unmanaged resources used by the SymmetricAlgorithm and optionally releases the managed resources.</td>
</tr>
<tr>
<td>Finalize (inherited from SymmetricAlgorithm)</td>
<td>Releases the unmanaged resources used by the SymmetricAlgorithm.</td>
</tr>
<tr>
<td>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression Library
PktzipClassic Constructor

Initializes a new instance of the PktzipClassic class.

```java
protected PktzipClassic();
```

See Also

SharpZip Compression Library
PkzipClassic Methods

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>GenerateKeys</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>Clear</td>
<td>Releases all resources used by the <a href="#">SymmetricAlgorithm</a>.</td>
</tr>
<tr>
<td>CreateDecryptor</td>
<td>Overloaded. When overridden in a derived class, creates a symmetric decryptor object with the specified Key and initialization vector (IV).</td>
</tr>
<tr>
<td>CreateEncryptor</td>
<td>Overloaded. When overridden in a derived class, creates a symmetric encryptor object with the specified Key and initialization vector (IV).</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GenerateIV</td>
<td>When overridden in a derived class, generates a random initialization vector (IV) to be used for the algorithm.</td>
</tr>
<tr>
<td>GenerateKey</td>
<td>When overridden in a derived class, generates a random Key to be used for the algorithm.</td>
</tr>
<tr>
<td>GetHashCode</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 (inherited from)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td><strong>ValidKeySize</strong> (inherited from <strong>SymmetricAlgorithm</strong>)</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 (inherited from)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dispose</strong> (inherited from <strong>SymmetricAlgorithm</strong>)</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 <strong>SymmetricAlgorithm</strong>)</td>
<td>Releases the unmanaged resources used by the <strong>SymmetricAlgorithm</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**

- **PkzipClassic Class** | **ICSSharpCode.SharpZipLib.Encryption Namespace**
SharpZip Compression Library
Generates new encryption keys based on given seed

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

See Also

SharpZip Compression 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
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


See Also

PkzipClassicManaged Members | ISharpCode.SharpZipLib.Encryption Namespace
## PkzipClassicManaged Members

### PkzipClassicManaged overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PkzipClassicManagedConstructor</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>BlockSize</td>
<td>Get / set the applicable block size.</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.</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>

### Public Instance Methods
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clear</strong> (inherited from SymmetricAlgorithm)</td>
<td>Releases all resources used by the SymmetricAlgorithm.</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 Key and initialization vector (IV).</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 Key and initialization vector (IV).</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>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 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>
<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](#) | [ICSharpCode.SharpZipLib.Encryption Namespace](#)
SharpZip Compression Library
PkszClassicManaged Constructor

Initializes a new instance of the PkszClassicManaged class.

```csharp
public PkszClassicManaged();
```

See Also

PkszClassicManaged Class | ISharpCode.SharpZipLib.Encryption Namespace
SharpZip Compression Library
PkpzipClassicManaged Properties

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

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.</td>
</tr>
<tr>
<td><strong>FeedbackSize</strong></td>
<td>Gets or sets the feedback size of the cryptographic operation in bits.</td>
</tr>
<tr>
<td>(inherited from SymmetricAlgorithm)</td>
<td></td>
</tr>
<tr>
<td><strong>IV</strong></td>
<td>Gets or sets the initialization vector (IV) for the symmetric algorithm.</td>
</tr>
<tr>
<td>(inherited from SymmetricAlgorithm)</td>
<td></td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>Get / set the key value applicable.</td>
</tr>
<tr>
<td><strong>KeySize</strong></td>
<td>Gets or sets the size of the secret key used by the symmetric algorithm in bits.</td>
</tr>
<tr>
<td>(inherited from SymmetricAlgorithm)</td>
<td></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></td>
<td>Gets or sets the mode for operation of the symmetric algorithm.</td>
</tr>
<tr>
<td>(inherited from SymmetricAlgorithm)</td>
<td></td>
</tr>
<tr>
<td><strong>Padding</strong></td>
<td>Gets or sets the padding mode used in the symmetric algorithm.</td>
</tr>
<tr>
<td>(inherited from SymmetricAlgorithm)</td>
<td></td>
</tr>
</tbody>
</table>

See Also

- **PkpzipClassicManaged Class** | **ICSharpCode.SharpZipLib.Encryption Namespace**
SharpZip Compression Library
PkzipClassicManaged.BlockSize Property

Get / set the applicable block size.

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

Remarks

The only valid block size is 8.

See Also

- PkzipClassicManaged Class
SharpZip Compression Library
PkzipClassicManaged.Key Property

Get / set the key value applicable.

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

See Also

- PkzipClassicManaged Class | ICSharpCode.SharpZipLib.Encryption Namespace
SharpZip Compression Library
Get an array of legal block sizes.


See Also

PkzipClassicManaged Class | ISharpCode.SharpZipLib.Encryption Namespace
SharpZip Compression Library
Get an array of legal `key sizes`.

```csharp
```

See Also

[PkzipClassicManaged Class] | [ICSharpCode.SharpZipLib.Encryption Namespace]
SharpZip Compression 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`.

`public virtual ICryptoTransform CreateDecryptor();`
Create a decryptor.

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

**See Also**

[PkzipClassicManaged Class](Link) | [ICSharpCode.SharpZipLib.Encryption Namespace](Link)
SharpZip Compression Library
**PkzipClassicManaged.CreateDecryptor Method (Byte[], Byte[])**

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](#)
- [ICSharpCode.SharpZipLib.Encryption Namespace](#)
- [PkzipClassicManaged.CreateDecryptor Overload List](#)
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](#)
SharpZip Compression Library
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
SharpZip Compression Library
Generate a new random key.

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

See Also

[PKZipClassicManaged Class](#) | [ICSharpCode.SharpZipLib.Encryption Namespace](#)
SharpZip Compression Library
## ISharpCode.SharpZipLib.GZip Namespace

### 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 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>$ FCLOCK</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><strong>Equals</strong></td>
<td>Determines whether the specified <a href="#">Object</a> is equal to the current <a href="#">Object</a>.</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 <a href="#">Type</a> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong></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><strong>Finalize</strong></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><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**

[GZipConstants Class] | [ICSharpCode.SharpZipLib.GZip Namespace]
SharpZip Compression Library
GZipConstants Fields

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 | ISharpCode.SharpZipLib.GZip Namespace
GZipConstants.FCOMMENT Field

flag bit mask indicating comment is present

```
public const int FCOMMENT = 16;
```

See Also

GZipConstants Class | ISharpCode.SharpZipLib.GZip Namespace
GZipConstants.FEXTRA Field

Flag bit mask for extra

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

See Also

[GZipConstants Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
SharpZip Compression Library
GZipConstants.FHCRC Field

Flag bitmask for Crc

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

See Also

GZipConstants Class | ISharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
GZipConstants.FNAME Field

flag bitmask for name

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

See Also

[GZipConstants Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
SharpZip Compression Library
GZipConstants.FTEXT Field

Flag bit mask for text

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

See Also

GZipConstants Class | ISharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
GZipConstants.GZIP_MAGIC Field

Magic number found at start of GZIP header

```
public static readonly int GZIP_MAGIC;
```

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  
ICSharpCode.SharpZipLib.SharpZipBaseException  
ICSharpCode.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
SharpZip Compression Library
### GZipException Members

#### GZipException overview

**Public 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>

**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</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>Property</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 Properties

<table>
<thead>
<tr>
<th>Property</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**

GZipException Class  |  ISharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
GZipException Constructor

Initialise a new instance of GZipException

Overload List

Initialise a new instance of GZipException

public GZipException();

Initialise a new instance of GZipException with its message string.

public GZipException(string);

See Also

GZipException Class | ISharpCode.SharpZipLib.GZip Namespace
GZipException Constructor ()

Initialise a new instance of GZipException

```java
public GZipException();
```

See Also

- GZipException Class
- ISharpCode.SharpZipLib.GZip Namespace
- GZipException Constructor Overload List
SharpZip Compression Library
GZipException Constructor (String)

Initialise a new instance of GZipException with its message string.

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

Parameters

`message`  
A `String` string that describes the error.

See Also

- GZipException Class  |  ISharpCode.SharpZipLib.GZip Namespace  
- GZipException Constructor Overload List
SharpZip Compression 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
                 ISharpCode.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

using System;
using System.IO;

using ISharpCode.SharpZipLib.GZip;

class MainClass
{
    public static void Main(string[] args)
    {
        Stream s = new GZipInputStream(File.OpenRead(FileStream fs = File.Create(Path.GetFileNameWithoutExtension(args[0]));
        int size = 2048;
        byte[] writeData = new byte[2048];
while (true) {
    size = s.Read(writeData, 0, size);
    if (size > 0) {
        fs.Write(writeData, 0, size);
    } else {
        break;
    }
}
s.Close();

Requirements

Namespace: ISharpCode.SharpZipLib.GZip


See Also

GZipInputStream Members | ISharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
# GZipInputStream Members

## GZipInputStream overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GZipInputStream</td>
<td>Overloaded. Initializes a new instance of the GZipInputStream 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>Returns 0 once the end of the stream (EOF) has been reached. Otherwise returns 1.</td>
</tr>
<tr>
<td><strong>CanRead</strong></td>
<td>Gets a value indicating whether the current stream supports reading.</td>
</tr>
<tr>
<td><strong>CanSeek</strong></td>
<td>Gets a value of false indicating seeking is not supported for this stream.</td>
</tr>
<tr>
<td><strong>CanWrite</strong></td>
<td>Gets a value of false indicating that this stream is not writeable.</td>
</tr>
<tr>
<td><strong>IsStreamOwner</strong></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>A value representing the length of the stream in bytes.</td>
</tr>
<tr>
<td><strong>Position</strong></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>Inherited From</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong></td>
<td>(inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong></td>
<td>(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>(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></td>
<td>(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></td>
<td>(inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><strong>EndWrite</strong></td>
<td>(inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>(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>(inherited from InflaterInputStream)</td>
<td>Flushes the baseInputStream</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>(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></td>
<td>(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></td>
<td>(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>
<td></td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Reads uncompressed data into an array of bytes</td>
<td></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>
<td></td>
</tr>
<tr>
<td><strong>Seek</strong> (inherited fromInflaterInputStream)</td>
<td>Sets the position within the current stream Always throws a NotSupportedException</td>
<td></td>
</tr>
<tr>
<td><strong>SetLength</strong> (inherited fromInflaterInputStream)</td>
<td>Set the length of the current stream Always throws a NotSupportedException</td>
<td></td>
</tr>
<tr>
<td><strong>Skip</strong> (inherited fromInflaterInputStream)</td>
<td>Skip specified number of bytes of uncompressed data</td>
<td></td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited fromObject)</td>
<td>Returns a String that represents the current Object.</td>
<td></td>
</tr>
<tr>
<td><strong>Write</strong> (inherited fromInflaterInputStream)</td>
<td>Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException</td>
<td></td>
</tr>
<tr>
<td><strong>WriteByte</strong> (inherited fromInflaterInputStream)</td>
<td>Writes one byte to the current stream and advances the current position Always throws a NotSupportedException</td>
<td></td>
</tr>
</tbody>
</table>

**Protected Instance Fields**

<p>| <strong>baseInputStream</strong> (inherited from InflaterInputStream) | Base stream the inflater reads from. |
| <strong>crc</strong> | CRC-32 value for |</p>
<table>
<thead>
<tr>
<th>Property</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](#)
SharZip Compression Library
GZipInputStream Constructor

Creates a GZipInputStream with the default buffer size

Overload List

Creates a GZipInputStream with the default buffer size

```csharp
public GZipInputStream(Stream);
```

Creates a GZIPInputStream with the specified buffer size

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

See Also

[GZipInputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
SharpZip Compression 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 (baseInputSteam GZIP format)

See Also

GZipInputStream Class | ICSharpCode.SharpZipLib.GZip Namespace | GZipInputStream Constructor Overload List
SharpZip Compression 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

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>baseInputStream (inherited from InflaterInputStream)</td>
<td>Base stream the inflater reads from.</td>
</tr>
<tr>
<td>crc</td>
<td>CRC-32 value for uncompressed data</td>
</tr>
<tr>
<td>csize (inherited from InflaterInputStream)</td>
<td>The compressed size</td>
</tr>
<tr>
<td>eos</td>
<td>Indicates end of stream</td>
</tr>
<tr>
<td>inf (inherited from InflaterInputStream)</td>
<td>Decompressor for this stream</td>
</tr>
<tr>
<td>inputBuffer (inherited from InflaterInputStream)</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

```
protected Crc32 crc;
```

See Also

[GZipInputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
SharpZip Compression Library
**GZipInputStream.eos Field**

Indicates end of stream

```csharp
protected bool eos;
```

See Also

[GZipInputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
### GZipInputStream Methods

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><img src="#" alt="BeginRead" /> (inherited from Stream)</td>
<td>Begins an asynchronous read operation.</td>
</tr>
<tr>
<td><img src="#" alt="BeginWrite" /> (inherited from InflaterInputStream)</td>
<td>Entry point to begin an asynchronous write. Always throws a NotSupportedException.</td>
</tr>
<tr>
<td><img src="#" alt="Close" /> (inherited from InflaterInputStream)</td>
<td>Closes the input stream. When IsStreamOwner is true the underlying stream is also closed.</td>
</tr>
<tr>
<td><img src="#" alt="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><img src="#" alt="EndRead" /> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><img src="#" alt="EndWrite" /> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</td>
</tr>
<tr>
<td><img src="#" alt="Equals" /> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><img src="#" alt="Flush" /> (inherited from InflaterInputStream)</td>
<td>Flushes the baseInputStream</td>
</tr>
<tr>
<td><img src="#" alt="GetHashCode" /> (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>
- **GetLifetimeService** (inherited from MarshalByRefObject): Retrieves the current lifetime service object that controls the lifetime policy for this instance.

- **GetType** (inherited from Object): Gets the **Type** of the current instance.

- **InitializeLifetimeService** (inherited from MarshalByRefObject): Obtains a lifetime service object to control the lifetime policy for this instance.

- **Read**: Reads uncompressed data into an array of bytes.

- **ReadByte** (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.

- **Seek** (inherited from InflaterInputStream): Sets the position within the current stream Always throws a NotSupportedException.

- **SetLength** (inherited from InflaterInputStream): Set the length of the current stream Always throws a NotSupportedException.

- **Skip** (inherited from InflaterInputStream): Skip specified number of bytes of uncompressed data.

- **ToString** (inherited from Object): Returns a **String** that represents the current **Object**.

- **Write** (inherited from InflaterInputStream): Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException.

- **WriteByte** (inherited from InflaterInputStream): Writes one byte to the current stream and advances the
current position Always throws a NotSupportedException

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>✉️ Fill (inherited from InflaterInputStream)</td>
<td>Fills the buffer with more data to decompress.</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>✉️ MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>✉️ StopDecrypting (inherited from InflaterInputStream)</td>
<td>Clear any cryptographic state.</td>
</tr>
</tbody>
</table>

See Also

GZipInputStream Class | ICSharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
GZipInputStream.Read Method

Reads uncompressed data into an array of bytes

```csharp
public override int Read(byte[] buf, int offset, int len);
```

Parameters

- `buf`
  - The buffer to read uncompressed data into

- `offset`
  - The offset indicating where the data should be placed

- `len`
  - The number of uncompressed bytes to be read

See Also

- [GZipInputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
SharpZip Compression Library
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
   ISharpCode.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;

class MainClass
{
   public static void Main(string[] args)
   {
      Stream s = new GZipOutputStream(File.Create(FileStream fs = File.OpenRead(args[0]));
      byte[] writeData = new byte[fs.Length];
      fs.Read(writeData, 0, (int)fs.Length);
s.Write(writeData, 0, writeData.Length);
s.Close();
}

Requirements

Namespace: ICSharpCode.SharpZipLib.GZip

See Also

GZipOutputStream Members | ICSharpCode.SharpZipLib.GZip Namespace
## GZipOutputStream Members

### GZipOutputStream overview

#### Public Instance Constructors

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

#### 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>(inherited from</td>
<td>DeflaterOutputStream)</td>
</tr>
<tr>
<td>CanRead</td>
<td>Gets value indicating stream can be read from.</td>
</tr>
<tr>
<td>(inherited from</td>
<td>DeflaterOutputStream)</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>(inherited from</td>
<td>DeflaterOutputStream)</td>
</tr>
<tr>
<td>CanWrite</td>
<td>Get value indicating if this stream supports writing.</td>
</tr>
<tr>
<td>(inherited from</td>
<td>DeflaterOutputStream)</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>(inherited from</td>
<td>DeflaterOutputStream)</td>
</tr>
<tr>
<td>Length</td>
<td>Get current length of stream.</td>
</tr>
<tr>
<td>(inherited from</td>
<td>DeflaterOutputStream)</td>
</tr>
<tr>
<td>Password</td>
<td>Get/set the password used for encryption. When null no encryption is performed.</td>
</tr>
<tr>
<td>(inherited from</td>
<td>DeflaterOutputStream)</td>
</tr>
<tr>
<td>Position</td>
<td>The current position within the stream. Always throws a</td>
</tr>
<tr>
<td>(inherited from</td>
<td>DeflaterOutputStream)</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 <strong>Object</strong> is equal to the current <strong>Object</strong>.</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.</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>GetLevel</strong></td>
<td>Get the current compression level.</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>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 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 String that represents the current Object.</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>Write given buffer to output updating crc</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 Fields**
| **baseOutputStream** (inherited from DeflaterOutputStream) | Base stream the deflater depends on. |
| **buf** (inherited from DeflaterOutputStream) | This buffer is used temporarily to retrieve the bytes from the deflater and write them to the underlying output stream. |
| **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**

[GZipOutputStream Class] | [ICSharpCode.SharpZipLib.GZip Namespace]
SharpZip Compression Library
GZipOutputStream Constructor

Creates a GzipOutputStream with the default buffer size

Overload List

Creates a GzipOutputStream with the default buffer size

\[
\text{public GZipOutputStream(Stream);} \]

Creates a GZipOutputStream with the specified buffer size

\[
\text{public GZipOutputStream(Stream, int);} \]

See Also

GZipOutputStream Class | ICSharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
GZipOutputStream Constructor (Stream)

Creates a GzipOutputStream with the default buffer size

```csharp
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
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

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><code>baseOutputStream</code> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>Base stream the deflater depends on.</td>
</tr>
<tr>
<td><code>buf</code> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>This buffer is used temporarily to retrieve the bytes from the deflater and write them to the underlying output stream.</td>
</tr>
<tr>
<td><code>crc</code></td>
<td>CRC-32 value for uncompressed data</td>
</tr>
<tr>
<td><code>def</code> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>The deflater which is used to deflate the stream.</td>
</tr>
</tbody>
</table>

See Also

- [GZipOutputStream Class](#) | [ICSharpCode.SharpZipLib.GZip Namespace](#)
SharpZip Compression Library
GZipOutputStream.crc Field

CRC-32 value for uncompressed data

protected Crc32 crc;

See Also

GZipOutputStream Class | ICSharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
GZipOutputStream Methods

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></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>Writes remaining compressed output data to the output stream and closes it.</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>Finish</strong></td>
<td>Finish compression and write any footer information required to stream.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the stream by calling</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>deflaterOutputStream.flush()</code></td>
<td>flush() on the deflater and then on the underlying stream. This ensures that all bytes are flushed.</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>GetLevel</code></td>
<td>Get the current compression level.</td>
</tr>
<tr>
<td><code>GetLifetimeService</code> (inherited from <code>MarshalByRefObject</code>)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</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>InitializeLifetimeService</code> (inherited from <code>MarshalByRefObject</code>)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><code>Read</code> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>Read a block of bytes from stream</td>
</tr>
<tr>
<td><code>ReadByte</code> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>Read a byte from stream advancing position by one</td>
</tr>
<tr>
<td><code>Seek</code> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>Sets the current position of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><code>SetLength</code> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>Sets the length of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><code>SetLevel</code></td>
<td>Sets the active compression level (1-9). The new level will be activated immediately.</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>
<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> (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>
<tr>
<td><strong>Deflate</strong> (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><strong>EncryptBlock</strong> (inherited from DeflaterOutputStream)</td>
<td>Encrypt a block of data</td>
</tr>
<tr>
<td><strong>EncryptByte</strong> (inherited from DeflaterOutputStream)</td>
<td>Encrypt a single byte</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>InitializePassword</strong> (inherited from DeflaterOutputStream)</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> (inherited from DeflaterOutputStream)</td>
<td>Update encryption keys</td>
</tr>
</tbody>
</table>

### See Also

- GZipOutputStream Class  | ISharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
GZipOutputStream.Close Method

Writes remaining compressed output data to the output stream and closes it.

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

See Also

GZipOutputStream Class  |  ISharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
Finish compression and write any footer information required to stream

public override void Finish();

See Also
GZipOutputStream Class | ICSharpCode.SharpZipLib.GZip Namespace
### GZipOutputStream.GetLevel Method

Get the current compression level.

```csharp
public int GetLevel();
```

**Return Value**

The current compression level.

**See Also**

SharpZip Compression Library
GZipOutputStream.SetLevel Method

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>ArgumentOutOfRangeException</td>
<td>Level specified is not supported.</td>
</tr>
</tbody>
</table>

See Also

GZipOutputStream Class | ISharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
GZipOutputStream.Write Method

Write given buffer to output updating crc

```csharp
public override void Write(
    byte[] buf,
    int off,
    int len
);
```

Parameters

- **buf**
  - Buffer to write

- **off**
  - Offset of first byte in buf to write

- **len**
  - Number of bytes to write

See Also

- GZipOutputStream Class | ISharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
### ISharpCode.SharpZipLib.Tar Namespace

#### Namespace hierarchy

#### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidHeaderException</td>
<td>This exception is used to indicate that there is a problem with a TAR archive header.</td>
</tr>
<tr>
<td>TarArchive</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 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</td>
</tr>
</tbody>
</table>
| **TarBuffer** | 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 I/O Streams. |
| **TarEntry** | 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.

<table>
<thead>
<tr>
<th>TarHeader</th>
<th>TarException are used for exceptions specific to tar classes and code.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TarException</strong></td>
<td><strong>TarHeader</strong> This class encapsulates the Tar Entry Header used in Tar Archives. The class also holds a number of tar constants, used mostly in headers.</td>
</tr>
<tr>
<td><strong>TarInputStream</strong></td>
<td>The TarInputStream reads a UNIX tar archive as an InputStream. Methods are provided to position at each</td>
</tr>
</tbody>
</table>
successive entry in the archive, and the read each entry as a normal input stream using read().

<table>
<thead>
<tr>
<th>TarInputStream.EntryFactoryAdapter</th>
<th>Standard entry factory class creating instances of the class TarEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>TarOutputStream</td>
<td>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().</td>
</tr>
</tbody>
</table>

### Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>

SharpZip Compression 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.

System.Object   System.Exception   System.ApplicationException
                ISharpCode.SharpZipLib.SharpZipBaseException
                ISharpCode.SharpZipLib.Tar.TarException
                ISharpCode.SharpZipLib.Tar.InvalidHeaderException

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: ISharpCode.SharpZipLib.Tar


See Also

InvalidHeaderException Members | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
### InvalidHeaderException Members

**InvalidHeaderException overview**

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidHeaderException</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>HelpLink (inherited from Exception)</td>
<td>Gets or sets a link to the help file associated with this exception.</td>
</tr>
<tr>
<td>InnerException (inherited from Exception)</td>
<td>Gets the Exception instance that caused the current exception.</td>
</tr>
<tr>
<td>Message (inherited from Exception)</td>
<td>Gets a message that describes the current exception.</td>
</tr>
<tr>
<td>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>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>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>Equals (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>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 Properties

<table>
<thead>
<tr>
<th>Property</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

InvalidHeaderException Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
InvalidHeaderException Constructor

Initialise a new instance of the InvalidHeaderException class.

Overload List

Initialise a new instance of the InvalidHeaderException class.

  public InvalidHeaderException();

Initialises a new instance of the InvalidHeaderException class with a specified message.

  public InvalidHeaderException(string);

See Also

InvalidHeaderException Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
InvalidHeaderException Constructor ()

Initialise a new instance of the InvalidHeaderException class.

```java
public InvalidHeaderException();
```

See Also

- [InvalidHeaderException Class](#)
- [ICSharpCode.SharpZipLib.Tar Namespace](#)
- [InvalidHeaderException Constructor Overload List](#)
InvalidHeaderException Constructor (String)

Initialises a new instance of the InvalidHeaderException class with a specified message.

```java
public InvalidHeaderException(
    string msg
);
```

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
SharpZip Compression Library
TarArchive Class

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.getCurrentRecordNum() and TarBuffer.getCurrentBlockNum() methods, this would be rather trivial.

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


public class TarArchive

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

TarArchive Members | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
## TarArchive Members

### TarArchive overview

#### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateInputTarArchive</strong></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><strong>CreateOutputTarArchive</strong></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><strong>ApplyUserInfoOverrides</strong></td>
<td>Get or set a value indicating if overrides defined by SetUserInfo should be applied.</td>
</tr>
<tr>
<td><strong>GroupId</strong></td>
<td>Get the archive group id. See ApplyUserInfoOverrides 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 ApplyUserInfoOverrides 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>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>RecordSize</strong></td>
<td>Get the archive's record size. Because of its history, tar supports the concept of buffered IO consisting of RECORDS of BLOCKS. This allowed tar to match the IO characteristics of the physical device being used. Of course, in the C# world, this makes no sense, WITH ONE EXCEPTION - archives are expected to be properly &quot;blocked&quot;. Thus, all of the horrible TarBuffer support boils down to simply getting the &quot;boundaries&quot; correct.</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 ApplyUserInfoOverrides 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 ApplyUserInfoOverrides 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>CloseArchive</strong></td>
<td>Close the archive. This simply calls the underlying tar stream's close() method.</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</td>
</tr>
</tbody>
</table>
archive.

<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>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<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>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>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</td>
</tr>
</tbody>
</table>
available on other operating systems. TarArchive allows the programmer to specify values to be used in their place.

<table>
<thead>
<tr>
<th><strong>ToString</strong> (inherited from Object)</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 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.</td>
</tr>
</tbody>
</table>

### Public Instance Events

| **ProgressMessageEvent** | Client hook allowing detailed information to be reported during processing |

### Protected Instance Constructors

| **TarArchive Constructor** | Constructor for a TarArchive. |

### 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>
<tr>
<td><strong>OnProgressMessageEvent</strong></td>
<td>Raises the ProgressMessage event</td>
</tr>
</tbody>
</table>
See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
TarArchive Constructor

Constructor for a TarArchive.

```java
protected TarArchive();
```

See Also

[TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
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>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>
<tr>
<td><strong>RecordSize</strong></td>
<td>Get the archive's record size. Because of its history, tar supports the concept of buffered IO consisting of RECORDS of BLOCKS. This allowed tar to match the IO characteristics of the physical device being used. Of course, in the C# world, this makes no sense, WITH ONE EXCEPTION - archives are expected to be properly &quot;blocked&quot;. Thus, all of the horrible TarBuffer support boils</td>
</tr>
</tbody>
</table>
down to simply getting the "boundaries" correct.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RootPath</td>
<td>RootPath is removed from entry names if it is found at the beginning of the name.</td>
</tr>
<tr>
<td>UserId</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>UserName</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>

See Also

[TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
**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] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
TarArchive.GroupId Property

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](#)
SharpZip Compression 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
class TarArchive
{
    public string GroupName {get;}
}
```

See Also

[TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression 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]
SharpZip Compression Library
Get the archive's record size. Because of its history, tar supports the concept of buffered IO consisting of RECORDS of BLOCKS. This allowed tar to match the IO characteristics of the physical device being used. Of course, in the C# world, this makes no sense, WITH ONE EXCEPTION - archives are expected to be properly "blocked". Thus, all of the horrible TarBuffer support boils down to simply getting the "boundaries" correct.

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

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarArchive.RootPath Property

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 | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
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](#)
SharpZip Compression Library
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](#)
SharpZip Compression Library
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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateInputTarArchive</strong></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><strong>CreateOutputTarArchive</strong></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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CloseArchive</strong></td>
<td>Close the archive. This simply calls the underlying tar stream's close() method.</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...</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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>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>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. TarArchive allows the programmer to specify values to be used in their place.</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

| **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**. |
| **OnProgressMessageEvent** | Raises the ProgressMessage event |

### See Also

- [TarArchive Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
**TarArchive.CloseArchive Method**

Close the archive. This simply calls the underlying tar stream's close() method.

```csharp
public void CloseArchive();
```

See Also

[TarArchive Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
**TarArchive.CreateInputTarArchive Method**

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.

```
public static TarArchive CreateInputTarArchive(Stream);
```
Create TarArchive for reading setting block factor

```
public static TarArchive CreateInputTarArchive(Stream,int);
```

**See Also**

[TarArchive Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression 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.

```java
public static TarArchive CreateInputTarArchive(Stream inputStream);
```

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace | TarArchive.CreateInputTarArchive Overload List
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](#)
SharpZip Compression 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]
SharpZip Compression Library
TarArchive.CreateOutputTarArchive Method (Stream)

Create a TarArchive for writing to, using the default blocking factor

```java
public static TarArchive CreateOutputTarArchive(Stream outputStream);
```

Parameters

`outputStream`
Stream to write to

See Also

SharpZip Compression Library
Create a TarArchive for writing to

```
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](#)
SharpZip Compression Library
TarArchive.ExtractContents Method

Perform the "extract" command and extract the contents of the archive.

```csharp
public void ExtractContents(
    string destDir
);
```

Parameters

`destDir`

The destination directory into which to extract.

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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!

```csharp
public void ListContents();
```

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
# 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](#)
SharpZip Compression Library
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 void SetAsciiTranslation(
    bool asciiTranslate
);
```

Parameters

`asciiTranslate`
If true, translate ascii text files.

See Also

TarArchive Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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]
**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.

```java
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](#)
SharpZip Compression 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
SharpZip Compression 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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗ ProgressMessageEvent</td>
<td>Client hook allowing detailed information to be reported during processing</td>
</tr>
</tbody>
</table>

**See Also**

[TarArchive Class] [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
TarArchive.ProgressMessageEvent Event

Client hook allowing detailed information to be reported during processing

```csharp
public event ProgressMessageHandler ProgressMessageEvent;
```

See Also

TarArchive Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarBuffer Class**

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](#).

```csharp
```

### 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:** [ICSharpCode.SharpZipLib.Tar](#)
- **Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

**See Also**

[TarBuffer Members](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
### TarBuffer Members

**TarBuffer overview**

### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✶ BlockSize</td>
<td>The size of a block in a tar archive in bytes.</td>
</tr>
<tr>
<td>✶ DefaultBlockFactor</td>
<td>The number of blocks in a default record.</td>
</tr>
<tr>
<td>✶ 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>✶ CreateInputTarBuffer</td>
<td>Overloaded. Create TarBuffer for reading with default BlockFactor</td>
</tr>
<tr>
<td>✶ 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>✶ BlockFactor</td>
<td>Get the Blocking factor for the buffer</td>
</tr>
<tr>
<td>✶ 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>✶ Close</td>
<td>Close the TarBuffer. If this is an output buffer, also flush the current block before closing.</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>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetBlockFactor</td>
<td>Get the TAR Buffer's block factor</td>
</tr>
<tr>
<td>GetCurrentBlockNum</td>
<td>Get the current block number, within the current record, zero based.</td>
</tr>
<tr>
<td>GetCurrentRecordNum</td>
<td>Get the current record number.</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>GetRecordSize</td>
<td>Get the TAR Buffer's record size.</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>IsEOFBlock</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>ReadBlock</td>
<td>Read a block from the input stream.</td>
</tr>
<tr>
<td>SkipBlock</td>
<td>Skip over a block on the input stream.</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
<tr>
<td>WriteBlock</td>
<td>Overloaded. Write a block of data to the archive.</td>
</tr>
</tbody>
</table>

**Protected Instance Constructors**
<table>
<thead>
<tr>
<th><strong>TarBuffer Constructor</strong></th>
<th>Construct a default TarBuffer</th>
</tr>
</thead>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th><strong>Finalize</strong> (inherited from <strong>Object</strong>)</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 <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
TarBuffer Constructor

Construct a default TarBuffer

```java
protected TarBuffer();
```

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
# 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>blockSize</td>
<td>The size of a block in a tar archive in bytes.</td>
</tr>
<tr>
<td>defaultBlockFactor</td>
<td>The number of blocks in a default record.</td>
</tr>
<tr>
<td>defaultRecordSize</td>
<td>The size in bytes of a default record.</td>
</tr>
</tbody>
</table>

## See Also

[TarBuffer Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
**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]
SharpZip Compression Library
**TarBuffer.DefaultBlockFactor Field**

The number of blocks in a default record.

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

Remarks

The default value is 20 block per record.

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
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
SharpZip Compression 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>RecordSize</td>
<td>Get the record size for this buffer</td>
</tr>
</tbody>
</table>

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
**TarBuffer.BlockFactor Property**

Get the Blocking factor for the buffer

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

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
TarBuffer.RecordSize Property

Get the record size for this buffer

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

See Also

TarBuffer Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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><code>CreateInputTarBuffer</code></td>
<td>Overloaded. Create TarBuffer for reading with default BlockFactor</td>
</tr>
<tr>
<td><code>CreateOutputTarBuffer</code></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><code>Close</code></td>
<td>Close the TarBuffer. If this is an output buffer, also flush the current block before closing.</td>
</tr>
<tr>
<td><code>Equals</code> (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><code>GetBlockFactor</code></td>
<td>Get the TAR Buffer's block factor</td>
</tr>
<tr>
<td><code>GetCurrentBlockNum</code></td>
<td>Get the current block number, within the current record, zero based.</td>
</tr>
<tr>
<td><code>GetCurrentRecordNum</code></td>
<td>Get the current record number.</td>
</tr>
<tr>
<td><code>GetHashCode</code> (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><code>GetRecordSize</code></td>
<td>Get the TAR Buffer's record size.</td>
</tr>
<tr>
<td><code>GetType</code> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</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><strong>ToString</strong> (inherited from Object)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>WriteBlock</strong></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><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**

[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.

```c#
public void Close();
```

See Also

[TarBuffer Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
**TarBuffer.CreateInputTarBuffer Method**

Create TarBuffer for reading with default BlockFactor

**Overload List**

Create TarBuffer for reading with default BlockFactor

```csharp
public static TarBuffer CreateInputTarBuffer(Stream);
```

Construct TarBuffer for reading inputStream setting BlockFactor

```csharp
public static TarBuffer CreateInputTarBuffer(Stream,int);
```

**See Also**

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
TarBuffer.CreateInputTarBuffer Method (Stream)

Create TarBuffer for reading with default BlockFactor

```csharp
public static TarBuffer CreateInputTarBuffer(Stream inputStream);
```

Parameters

- `inputStream`  
  Stream to buffer

Return Value

TarBuffer

See Also

- TarBuffer Class
- ISharpCode.SharpZipLib.Tar Namespace
- TarBuffer.CreateInputTarBuffer Overload List
SharpZip Compression Library
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**

Construct TarBuffer for writing with default BlockFactor

**Overload List**

Construct TarBuffer for writing with default BlockFactor

```csharp
public static TarBuffer CreateOutputTarBuffer(Stream);
```

Construct TarBuffer for writing Tar output to streams.

```csharp
public static TarBuffer CreateOutputTarBuffer(Stream,int);
```

**See Also**

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
TarBuffer.CreateOutputTarBuffer Method (Stream)

Construct TarBuffer for writing with default BlockFactor

```csharp
public static TarBuffer CreateOutputTarBuffer(Stream outputStream);
```

Parameters

- `outputStream` output stream for buffer

Return Value

- TarBuffer

See Also

SharpZip Compression 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**

SharpZip Compression Library
TarBuffer.GetBlockFactor Method

Get the TAR Buffer's block factor

```csharp
public int GetBlockFactor();
```

See Also

TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarBuffer.GetCurrentBlockNum Method**

Get the current block number, within the current record, zero based.

```java
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
SharpZip Compression Library
Get the current record number.

```csharp
public int GetCurrentRecordNum();
```

Return Value

The current zero based record number.

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
**TarBuffer.GetRecordSize Method**

Get the TAR Buffer's record size.

```csharp
public int GetRecordSize();
```

See Also

[TarBuffer Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression 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](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarBuffer.ReadBlock Method

Read a block from the input stream.

```
public byte[] ReadBlock();
```

Return Value

The block of data read.

See Also

[TarBuffer Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
TarBuffer.SkipBlock Method

Skip over a block on the input stream.

```csharp
public void SkipBlock();
```

See Also

[TarBuffer Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarBuffer.WriteBlock Method

Write a block of data to the archive.

Overload List

Write a block of data to the archive.

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.

public void WriteBlock(byte[], int);

See Also

TarBuffer Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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
SharpZip Compression Library
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[] buf,
    int offset
);
```

Parameters

- **buf**
  The buffer containing the record data to write.

- **offset**
  The offset of the record data within buf.

See Also

TarBuffer Class | ISharpCode.SharpZipLib.Tar Namespace | TarBuffer.WriteBlock Overload List
SharpZip Compression Library
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.

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

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
ICSharpCode.SharpZipLib.dll)

See Also

TarEntry Members | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarEntry Members

TarEntry overview

Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateEntryFromFile</td>
<td>Construct an entry for a file. File is set to file, and the header is</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>the entry's header &quot;by hand&quot;.</td>
</tr>
</tbody>
</table>

Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</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>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><strong>UserName</strong></td>
<td>Get/set this entry's user name.</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
</tr>
</tbody>
</table>

### Public Instance 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>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>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>IsDescendent</strong></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><strong>NameTarHeader</strong></td>
<td>Fill in a TarHeader given only the entry's name.</td>
</tr>
<tr>
<td><strong>SetIds</strong></td>
<td>Convenience method to set this entry's group and user ids.</td>
</tr>
<tr>
<td><strong>SetNames</strong></td>
<td>Convenience method to set this entry's group and user names.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from)</td>
<td>Returns a <strong>String</strong> that represents</td>
</tr>
<tr>
<td><strong>Object)</strong></td>
<td>the current <strong>Object</strong>.</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]
**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]
SharpZip Compression Library
**TarEntry Constructor (Byte[])**

Construct an entry from an archive's header bytes. File is set to null.

```java
public TarEntry(
    byte[] headerBuf
);
```

**Parameters**

*headerBuf*

The header bytes from a tar archive entry.

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#) | [TarEntry Constructor Overload List](#)
SharpZip Compression 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
SharpZip Compression 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>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>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>

See Also

TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarEntry.File Property

Get this entry's file.

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

See Also

TarEntry Class  |  ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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]
SharpZip Compression Library
TarEntry.GroupName Property

Get/set this entry's group name.

```csharp
public string GroupName {get; set;}
```

See Also

TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
**TarEntry.IsDirectory Property**

Return true if this entry represents a directory, false otherwise

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

**See Also**

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarEntry.ModTime Property

Get/Set the modification time for this entry

public System.DateTime ModTime {get; set;}

See Also

TarEntry Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarEntry.Name Property**

Get/Set this entry's name.

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

See Also

[TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
**TarEntry.Size Property**

Get/set this entry's recorded file size.

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

See Also

- [TarEntry Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarEntry.TarHeader Property

Get this entry's header.

```csharp
public TarHeader TarHeader { get; }
```

See Also

TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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](#)
SharpZip Compression Library
TarEntry.UserName Property

Get/set this entry’s user name.

```csharp
public string UserName {get; set;}
```

See Also

TarEntry Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarEntry Methods**

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><code>CreateEntryFromFile</code></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><code>CreateTarEntry</code></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>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AdjustEntryName</code></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><code>Clone</code></td>
<td>Clone this tar entry.</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>Determine if the two entries are equal. Equality is determined by the header names being equal.</td>
</tr>
<tr>
<td><code>GetDirectoryEntries</code></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><code>GetFileTarHeader</code></td>
<td>Fill in a TarHeader with information from a File.</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Must be overridden when you override Equals.</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IsDescendent</strong></td>
<td>Determine if the given entry is a descendant of this entry.</td>
</tr>
<tr>
<td></td>
<td>Descendancy is determined by the name of the descendant starting with this entry's name.</td>
</tr>
<tr>
<td><strong>NameTarHeader</strong></td>
<td>Fill in a TarHeader given only the entry's name.</td>
</tr>
<tr>
<td><strong>SetIds</strong></td>
<td>Convenience method to set this entry's group and user ids.</td>
</tr>
<tr>
<td><strong>SetNames</strong></td>
<td>Convenience method to set this entry's group and user names.</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>WriteEntryHeader</strong></td>
<td>Write an entry's header information to a header buffer.</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

[TarEntry Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
TarEntry.AdjustEntryName Method

Convenience method that will modify an entry's name directly in place in an entry header buffer byte array.

```csharp
public void AdjustEntryName(
    byte[] outbuf,
    string newName
);
```

Parameters

- `outbuf`
  - The buffer containing the entry header to modify.

- `newName`
  - The new name to place into the header buffer.

See Also

- TarEntry Class
- ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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
- ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarEntry.CreateEntryFromFile Method

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
SharpZip Compression Library
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 | ISharpCode.SharpZipLib.Tar Namespace
Determine if the two entries are equal. Equality is determined by the header names being equal.

```csharp
public override bool Equals(
    object it
);
```

**Return Value**

True if the entries are equal.

**See Also**

TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
Get entries for all files present in this entries directory. If this entry doesn't 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
SharpZip Compression Library
TarEntry.GetFileTarHeader Method

Fill in a TarHeader with information from a File.

```csharp
public void GetFileTarHeader(
    TarHeader hdr,
    string file
);
```

Parameters

- **hdr**
  The TarHeader to fill in.

- **file**
  The file from which to get the header information.

See Also

- TarEntry Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarEntry.GetHashCode Method

Must be overridden when you override Equals.

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

See Also

[ TarEntry Class ](#)  |  [ ISharpCode.SharpZipLib.Tar Namespace ](#)
SharpZip Compression 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 desc);
```

**Parameters**

`desc`  
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](#)
SharpZip Compression Library
TarEntry.NameTarHeader Method

Fill in a TarHeader given only the entry's name.

```csharp
public void NameTarHeader(
    TarHeader hdr,
    string name
);
```

Parameters

hdr
The TarHeader to fill in.

name
The tar entry name.

See Also

TarEntry Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
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
SharpZip Compression 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
SharpZip Compression Library
TarEntry.WriteEntryHeader Method

Write an entry's header information to a header buffer.

```csharp
public void WriteEntryHeader(
    byte[] outbuf
);
```

Parameters

`outbuf`  
The tar entry header buffer to fill in.

See Also

TarEntry Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
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.

System.Object System.Exception System.ApplicationException
   ISharpCode.SharpZipLib.SharpZipBaseException
   ISharpCode.SharpZipLib.Tar.TarException
   ISharpCode.SharpZipLib.Tar.InvalidHeaderException

```csharp
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:** ISharpCode.SharpZipLib.Tar

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

See Also

TarException Members | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
## TarException Members

### TarException overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TarException</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>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</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>.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>GetObjectData</td>
<td>When overridden in a derived class, sets the <a href="#">SerializationInfo</a> with information about the exception.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <a href="#">Type</a> of the current instance.</td>
</tr>
<tr>
<td>ToString</td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

### Protected Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HResult</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</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>MemberwiseClone</td>
<td>Creates a shallow copy of the current <a href="#">Object</a>.</td>
</tr>
</tbody>
</table>

See Also

TarException Class | ISharpCode.SharpZipLib.Tar Namespace
TarException Constructor

Initialises a new instance of the TarException class.

Overload List

Initialises a new instance of the TarException class.

public TarException();

Initialises a new instance of the TarException class with a specified message.

public TarException(string);

See Also

TarException Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarException Constructor ()**

Initialises a new instance of the `TarException` class.

```java
public TarException();
```

**See Also**

[ TarException Class ](#) | [ ICSharpCode.SharpZipLib.Tar Namespace ](#) | [ TarException Constructor Overload List ](#)
SharpZip Compression Library
TarException Constructor (String)

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] | [ICSharpCode.SharpZipLib.Tar Namespace] | TarException Constructor Overload List
SharpZip Compression Library
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**  **ICSharpCode.SharpZipLib.Tar.TarHeader**

```
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**: IICSharpCode.SharpZipLib.Tar

**See Also**

- TarHeader Members  |  IICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
## TarHeader Members

### TarHeader overview

### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CHKSUMLEN</strong></td>
<td>The length of the checksum field in a header buffer.</td>
</tr>
<tr>
<td><strong>CHKSUMOFS</strong></td>
<td>Offset of checksum in a header buffer.</td>
</tr>
<tr>
<td><strong>DEVLEN</strong></td>
<td>The length of the devices field in a header buffer.</td>
</tr>
<tr>
<td><strong>GIDLEN</strong></td>
<td>The length of the group id field in a header buffer.</td>
</tr>
<tr>
<td><strong>GNAMELEN</strong></td>
<td>The length of the group name field in a header buffer.</td>
</tr>
<tr>
<td><strong>GNU_TMAGIC</strong></td>
<td>The magic tag representing an old GNU tar archive where version is included in magic and overwrites it</td>
</tr>
<tr>
<td><strong>LF_ACL</strong></td>
<td>Solaris access control list file type</td>
</tr>
<tr>
<td><strong>LF_BLK</strong></td>
<td>Block device file type.</td>
</tr>
<tr>
<td><strong>LF_CHR</strong></td>
<td>Character device file type.</td>
</tr>
<tr>
<td><strong>LF_CONTIG</strong></td>
<td>Contiguous file type.</td>
</tr>
<tr>
<td><strong>LF_DIR</strong></td>
<td>Directory file type.</td>
</tr>
<tr>
<td><strong>LF_EXTATTR</strong></td>
<td>Solaris Extended Attribute File</td>
</tr>
<tr>
<td><strong>LF_FIFO</strong></td>
<td>FIFO (pipe) file type.</td>
</tr>
<tr>
<td><strong>LF_GHDR</strong></td>
<td>Posix.1 2001 global extended header</td>
</tr>
<tr>
<td><strong>LF_GNU_DUMPDIR</strong></td>
<td>GNU dir dump file type This is a dir entry that contains the names of files that were in the</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>dir</code></td>
<td>dir at the time the dump was made</td>
</tr>
<tr>
<td><code>LF_GNU_LONGLINK</code></td>
<td>Identifies the next file on the tape as having a long link name</td>
</tr>
<tr>
<td><code>LF_GNU_LONGNAME</code></td>
<td>Identifies the next file on the tape as having a long name</td>
</tr>
<tr>
<td><code>LF_GNU_MULTIVOL</code></td>
<td>Continuation of a file that began on another volume</td>
</tr>
<tr>
<td><code>LF_GNU_NAMES</code></td>
<td>For storing filenames that don't fit in the main header (old GNU)</td>
</tr>
<tr>
<td><code>LF_GNU_SPARSE</code></td>
<td>GNU Sparse file</td>
</tr>
<tr>
<td><code>LF_GNU_VOLHDR</code></td>
<td>GNU Tape/volume header ignore on extraction</td>
</tr>
<tr>
<td><code>LF_LINK</code></td>
<td>Link file type.</td>
</tr>
<tr>
<td><code>LF_META</code></td>
<td>Inode (metadata only) no file content</td>
</tr>
<tr>
<td><code>LF_NORMAL</code></td>
<td>Normal file type.</td>
</tr>
<tr>
<td><code>LF_OLDNORM</code></td>
<td>The &quot;old way&quot; of indicating a normal file.</td>
</tr>
<tr>
<td><code>LF_SYMLINK</code></td>
<td>Symbolic link file type.</td>
</tr>
<tr>
<td><code>LF_XHDR</code></td>
<td>Posix.1 2001 extended header</td>
</tr>
<tr>
<td><code>MAGICLEN</code></td>
<td>The length of the magic field in a header buffer.</td>
</tr>
<tr>
<td><code>MODELEN</code></td>
<td>The length of the mode field in a header buffer.</td>
</tr>
<tr>
<td><code>MODTIMELEN</code></td>
<td>The length of the modification time field in a header buffer.</td>
</tr>
<tr>
<td><code>NAMELEN</code></td>
<td>The length of the name field in a header buffer.</td>
</tr>
<tr>
<td><code>SIZELEN</code></td>
<td>The length of the size field in a header buffer.</td>
</tr>
<tr>
<td><strong>$TMAGIC</strong></td>
<td>The magic tag representing a POSIX tar archive. (includes trailing NULL)</td>
</tr>
<tr>
<td><strong>$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

| **$GetAsciiBytes**       | Add a string to a buffer as a collection of ascii bytes. |
| **$GetLongOctalBytes**   | Put an octal representation of a value into a buffer. |
| **$GetNameBytes**        | Overloaded. Add name to the buffer as a collection of bytes. |
| **$GetOctalBytes**       | Put an octal representation of a value into a buffer. |
| **$ParseName**           | Parse a name from a header buffer. |
| **$ParseOctal**          | Parse an octal string from a header buffer. |
| **$ResetValueDefaults**  | Reset value defaults to initial values. |
| **$SetValueDefaults**    | Set defaults for values used when constructing a TarHeader instance. |

### Public Instance Constructors

| **$TarHeader Constructor** | Initialise a default TarHeader instance |
**Public Instance Properties**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Checksum</code></td>
<td>Get the entry's checksum. This is only valid/updated after writing or reading an entry.</td>
</tr>
<tr>
<td><code>DevMajor</code></td>
<td>Get/set the entry's major device number.</td>
</tr>
<tr>
<td><code>DevMinor</code></td>
<td>Get/set the entry's minor device number.</td>
</tr>
<tr>
<td><code>GroupId</code></td>
<td>Get/set the entry's group id.</td>
</tr>
<tr>
<td><code>GroupName</code></td>
<td>Get/set the entry's group name.</td>
</tr>
<tr>
<td><code>IsChecksumValid</code></td>
<td>Get value of true if the header checksum is valid, false otherwise.</td>
</tr>
<tr>
<td><code>LinkName</code></td>
<td>The entry's link name.</td>
</tr>
<tr>
<td><code>Magic</code></td>
<td>Get/set the entry's magic tag.</td>
</tr>
<tr>
<td><code>Mode</code></td>
<td>Get/set the entry's Unix style permission mode.</td>
</tr>
<tr>
<td><code>ModTime</code></td>
<td>Get/set the entry's modification time.</td>
</tr>
<tr>
<td><code>Name</code></td>
<td>Get/set the name for this tar entry.</td>
</tr>
<tr>
<td><code>Size</code></td>
<td>Get/set the entry's size.</td>
</tr>
<tr>
<td><code>TypeFlag</code></td>
<td>Get/set the entry's type flag.</td>
</tr>
<tr>
<td><code>UserId</code></td>
<td>The entry's user id.</td>
</tr>
<tr>
<td><code>UserName</code></td>
<td>The entry's user name.</td>
</tr>
<tr>
<td><code>Version</code></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><code>Clone</code></td>
<td>Clone a TAR header.</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>Determines if this instance is</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>Obsolete. Get the name of this entry.</td>
</tr>
<tr>
<td><strong>GetTypeInfo</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ParseBuffer</strong></td>
<td>Parse TarHeader information from a header buffer.</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>WriteHeader</strong></td>
<td>'Write' header information to buffer provided, updating the check sum.</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**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
**TarHeader Constructor**

Initialise a default TarHeader instance

```
public TarHeader();
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression 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>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 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</td>
</tr>
<tr>
<td>Name</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</td>
</tr>
<tr>
<td></td>
<td>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></td>
<td>header buffer.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>✶ TMAGIC</td>
<td>The magic tag representing a POSIX tar archive. (includes trailing NULL)</td>
</tr>
<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 | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.CHKSUMLEN Field

The length of the checksum field in a header buffer.

```
public static readonly int CHKSUMLEN;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression 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](#)
SharpZip Compression Library
TarHeader.DEVLEN Field

The length of the devices field in a header buffer.

```csharp
public static readonly int DEVLEN;
```

See Also

- [TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
The length of the group id field in a header buffer.

```csharp
public static readonly int GIDLEN;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.GNAMELEN Field

The length of the group name field in a header buffer.

```csharp
public static readonly int GNAMELEN;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
The magic tag representing an old GNU tar archive where version is included in magic and overwrites it

```csharp
public static readonly string GNU_TMAGIC;
```

See Also

TarHeader Class | ICSHarpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarHeader.LF_ACL Field**

Solaris access control list file type

```csharp
public const byte LF_ACL = 65;
```

**See Also**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarHeader.LF_BLK Field

Block device file type.

```
public const byte LF_BLK = 52;
```

See Also

[ TarHeader Class ] [ ISharpCode.SharpZipLib.Tar Namespace ]
SharpZip Compression Library
**TarHeader.LF_CHR Field**

Character device file type.

```csharp
public const byte LF_CHR = 51;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
**TarHeader.LF_CONTIG Field**

Contiguous file type.

```
public const byte LF_CONTIG = 55;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarHeader.LF_DIR Field

Directory file type.

```csharp
public const byte LF_DIR = 53;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
Solaris Extended Attribute File

```csharp
public const byte LF_EXTATTR = 69;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarHeader.LF_FIFO Field

FIFO (pipe) file type.

```csharp
public const byte LF_FIFO = 54;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.LF_GHDR Field

Posix.1 2001 global extended header

```csharp
public const byte LF_GHDR = 103;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression 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

```
public const byte LF_GNU_DUMPDIR = 68;
```

See Also

- TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
**TarHeader.LF_GNU_LONGLINK Field**

Identifies the next file on the tape as having a long link name

```public const byte LF_GNU_LONGLINK = 75;```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression 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](#)
SharpZip Compression 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]
SharpZip Compression Library
For storing filenames that don't fit in the main header (old GNU)

```java
public const byte LF_GNU_NAMES = 78;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
GNU Sparse file

```java
public const byte LF_GNU_SPARSE = 83;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
**TarHeader.LF_GNU_VOLHDR Field**

GNU Tape/volume header ignore on extraction

```
public const byte LF_GNU_VOLHDR = 86;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarHeader.LF_LINK Field

Link file type.

```csharp
public const byte LF_LINK = 49;
```

See Also

[TarHeader Class](http) | [ICSharpCode.SharpZipLib.Tar Namespace](http)
SharpZip Compression Library
TarHeader.LF_META Field

Inode (metadata only) no file content

```csharp
public const byte LF_META = 73;
```

See Also

TarHeader Class | ICSsharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.LF.NORMAL Field

Normal file type.

```java
public const byte LF_NORMAL = 48;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
**TarHeader.LF_OLDNORM Field**

The "old way" of indicating a normal file.

```csharp
public const byte LF_OLDNORM = 0;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
TarHeader.LF_SYMLINK Field

Symbolic link file type.

```
public const byte LF_SYMLINK = 50;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.LF_XHDR Field

Posix.1 2001 extended header

```csharp
public static readonly byte LF_XHDR;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.MAGICLEN Field

The length of the magic field in a header buffer.

public static readonly int MAGICLEN;

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.MODELEN Field

The length of the mode field in a header buffer.

```csharp
public static readonly int MODELEN;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
The length of the modification time field in a header buffer.

public static readonly int MODTIMELEN;

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.NAMELEN Field

The length of the name field in a header buffer.

public static readonly int NAMELEN;

See Also

TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
The length of the size field in a header buffer.

```
public static readonly int SIZELEN;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
The magic tag representing a POSIX tar archive. (includes trailing NULL)

```
public static readonly string TMAGIC;
```

See Also

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
TarHeaderUIDLEN Field

The length of the user id field in a header buffer.

```
public static readonly int UIDLEN;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.UNAMELEN Field

The length of the user name field in a header buffer.

```csharp
public static readonly int UNAMELEN;
```

See Also

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarHeader.VERSIONLEN Field

The length of the version field in a header buffer.

```csharp
public static readonly int VERSIONLEN;
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
## 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>Version</td>
<td>The entry's version.</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------</td>
</tr>
</tbody>
</table>

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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](#)
SharpZip Compression Library
TarHeader.DevMinor Property

Get/set the entry's minor device number.

```csharp
public int DevMinor {get; set;}
```

See Also

TarHeader Class | ICSHarpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**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]
SharpZip Compression 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](#)
SharpZip Compression 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](#)
SharpZip Compression 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><code>ArgumentNullException</code></td>
<td>Thrown when attempting to set LinkName to null.</td>
</tr>
</tbody>
</table>

**See Also**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression 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>ArgumentNullException</td>
<td>Thrown when attempting to set Magic to null.</td>
</tr>
</tbody>
</table>

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
TarHeader.Mode Property

Get/set the entry's Unix style permission mode.

```csharp
public int Mode {get; set;}
```

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
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
SharpZip Compression Library
## 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><code>ArgumentNullException</code></td>
<td>Thrown when attempting to set the property to null.</td>
</tr>
</tbody>
</table>

### See Also

- [TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
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](#)
SharpZip Compression 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]
SharpZip Compression Library
TarHeader.UserID Property

The entry's user id.

```
public int UserID {get; set;}
```

Remarks

This is only directly relevant to unix systems. The default is zero.

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarHeader.UserName Property**

The entry's user name.

```csharp
public string UserName {get; set;}
```

**Remarks**

See [ResetValueDefaults](#) for detail on how this value is derived.

**See Also**

[TarHeader Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
**TarHeader.Version Property**

The entry's version.

```
public string Version {get; set;}
```

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentNullException</td>
<td>Thrown when attempting to set Version to null.</td>
</tr>
</tbody>
</table>

**See Also**

[TarHeader Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression 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 name 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>
<tr>
<td><code>ResetValueDefaults</code></td>
<td>Reset value defaults to initial values.</td>
</tr>
<tr>
<td><code>SetValueDefaults</code></td>
<td>Set defaults for values used when constructing a TarHeader instance.</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>Obsolete. Get the name of this entry.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>ParseBuffer</strong></td>
<td>Parse TarHeader information from a header buffer.</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>WriteHeader</strong></td>
<td>'Write' header information to buffer provided, updating the check sum.</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**

TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarHeader.Clone Method**

Clone a TAR header.

```csharp
public object Clone();
```

**Implements**

ICloneable.Clone

**See Also**

TarHeader Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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](#)
SharpZip Compression Library
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
- ICSharpCode.SharpZipLib.Tar Namespace
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]
SharpZip Compression Library
Put an octal representation of a value into a buffer

```csharp
public static int GetLongOctalBytes(
    long val,
    byte[] buf,
    int offset,
    int length
);
```

**Parameters**

- `val`  
  Value to be convert to octal

- `buf`  
  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](#)
SharpZip Compression Library
NOTE: This method is now obsolete.

Get the name of this entry.

```csharp
public string GetName();
```

Return Value

The entry's name.

Remarks

This is obsolete use the Name property instead.

See Also

[ TarHeader Class ](index.html) | [ ICSharpCode.SharpZipLib.Tar Namespace ](index.html)
SharpZip Compression Library
TarHeader.GetNameBytes Method

Add an entry name to the buffer

Overload List

Add an entry name to the buffer

    public static int GetNameBytes(string,byte[],int,int);

Add *name* to the buffer as a collection of bytes

    public static int GetNameBytes(string,int,byte[],int,int);

Add an entry name to the buffer

    public static int GetNameBytes(StringBuilder,byte[],int,int);

Add *name* to the buffer as a collection of bytes

    public static int GetNameBytes(StringBuilder,int,byte[],int,int);

See Also

    TarHeader Class  |  ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarHeader.GetNameBytes Method (String, Byte[], Int32, Int32)

Add an entry name to the buffer

```csharp
public static int GetNameBytes(
    string name,
    byte[] buf,
    int offset,
    int length
);
```

Parameters

- **name**
  - The name to add

- **buf**
  - 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
SharpZip Compression Library
Add name to the buffer as a collection of bytes

```csharp
public static int GetNameBytes(
    string name,
    int nameOffset,
    byte[] buf,
    int bufferOffset,
    int length
);
```

**Parameters**

*name*

The name to add

*nameOffset*

The offset of the first character

*buf*

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
SharpZip Compression Library
TarHeader.GetNameBytes Method (StringBuilder, Byte[], Int32, Int32)

Add an entry name to the buffer

```csharp
public static int GetNameBytes(
    StringBuilder name,
    byte[] buf,
    int offset,
    int length
);
```

Parameters

- **name**
  The name to add

- **buf**
  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
SharpZip Compression Library
Add name to the buffer as a collection of bytes

```csharp
public static int GetNameBytes(
    StringBuilder name,
    int nameOffset,
    byte[] buf,
    int bufferOffset,
    int length
);
```

**Parameters**

- **name**
  - The name to add

- **nameOffset**
  - The offset of the first character

- **buf**
  - 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](#)
SharpZip Compression Library
Put an octal representation of a value into a buffer

```csharp
public static int GetOctalBytes(
    long val,
    byte[] buf,
    int offset,
    int length
);
```

Parameters

- `val`  
  the value to be converted to octal

- `buf`  
  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
SharpZip Compression 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](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression 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
SharpZip Compression 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 ]
SharpZip Compression Library
TarHeader.ResetValueDefaults Method

Reset value defaults to initial values.

```csharp
public static void ResetValueDefaults();
```

Remarks

The default values are user id=0, group id=0, groupname="None", user name=null. When the default user name is null the value from Environment.UserName is used. Or "PocketPC" for the Compact framework. When the default group name is null the value "None" is used.

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
TarHeader.SetValueDefaults Method

Set defaults for values used when constructing a TarHeader instance.

```csharp
public static void SetValueDefaults(
    int userId,
    string userName,
    int groupId,
    string groupName
);
```

Parameters

- **userId**
  Value to apply as a default for userId.

- **userName**
  Value to apply as a default for userName.

- **groupId**
  Value to apply as a default for groupId.

- **groupName**
  Value to apply as a default for groupName.

See Also

- [TarHeader Class](#)
- [ICSharpCode.SharpZipLib.Tar Namespace](#)
TarHeader.WriteHeader Method

'Write' header information to buffer provided, updating the checksum.

```java
public void WriteHeader(
    byte[] outbuf
);
```

Parameters

`outbuf`
output buffer for header information

See Also

TarHeader Class | ICSharpCode.SharpZipLib.Tar Namespace
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.

**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**

TarInputStream Members | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
## TarInputStream Members

### TarInputStream overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TarInputStream</strong></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><strong>Available</strong></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><strong>CanRead</strong></td>
<td>Gets a value indicating whether the current stream supports reading</td>
</tr>
<tr>
<td><strong>CanSeek</strong></td>
<td>Gets a value indicating whether the current stream supports seeking. This property always returns false.</td>
</tr>
<tr>
<td><strong>CanWrite</strong></td>
<td>Gets a value indicating if the stream supports writing. This property always returns false.</td>
</tr>
<tr>
<td><strong>IsMarkSupported</strong></td>
<td>Since we do not support marking just yet, we return false.</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>The length in bytes of the stream</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>Gets or sets the position within the stream. Setting the Position is not supported and throws a NotSupportedException.</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRead</strong></td>
<td>(inherited from) Begins an asynchronous read</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Stream)</strong></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>Closes this stream. Calls the TarBuffer’s close() method. The underlying</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>stream.</td>
</tr>
<tr>
<td><strong>CreateObjRef</strong> (inherited from MarshalByRefObject)</td>
<td>Creates an object that contains all the relevant information required to</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>data in the current</td>
</tr>
</tbody>
</table>
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><strong>GetRecordSize</strong></td>
<td>Get the record size being used by this stream’s TarBuffer.</td>
</tr>
<tr>
<td><strong>GetTypeInfo</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>Mark</strong></td>
<td>Since we do not support marking just yet, we do nothing.</td>
</tr>
<tr>
<td><strong>Read</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, int).</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Since we do not support marking just yet, we do nothing.</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>SetEntryFactory</strong></td>
<td>Set the entry factory for this</td>
</tr>
</tbody>
</table>
### `SetLength`
Sets the length of the stream. This operation is not supported and will throw a `NotSupportedException`.

### `Skip`
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.

### `ToString` (inherited from `Object`)
Returns a `String` that represents the current `Object`.

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

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

---

### 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>currEntry</code></td>
<td>Current entry being read</td>
</tr>
<tr>
<td><code>eFactory</code></td>
<td>Factory used to create <code>TarEntry</code> 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>Property</td>
<td>Description</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>readBuf</td>
<td>Buffer used with calls to Read()</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>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
SharpZip Compression Library
TarInputStream Constructor

Construct a TarInputStream with default block factor

Overload List

Construct a TarInputStream with default block factor

public TarInputStream(Stream);

Construct a TarInputStream with user specified block factor

public TarInputStream(Stream,int);

See Also

TarInputStream Class  |  ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
Construct a `TarInputStream` with default block factor

```java
public TarInputStream(
    Stream inputStream
);
```

**Parameters**

`inputStream`
stream to source data from

**See Also**

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace] | [TarInputStream Constructor Overload List]
SharpZip Compression Library
Construct a TarInputStream with user specified block factor

```java
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](#)
SharpZip Compression Library
### 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>buffer</td>
<td>Working buffer</td>
</tr>
<tr>
<td>currEntry</td>
<td>Current entry being read</td>
</tr>
<tr>
<td>eFactory</td>
<td>Factory used to create TarEntry or descendant class instance</td>
</tr>
<tr>
<td>entryOffset</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>hasHitEOF</td>
<td>Flag set when last block has been read</td>
</tr>
<tr>
<td>readBuf</td>
<td>Buffer used with calls to Read()</td>
</tr>
</tbody>
</table>

**See Also**

- [TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarInputStream.buffer Field

Working buffer

```csharp
protected TarBuffer buffer;
```

See Also

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
Current entry being read

```java
protected TarEntry currEntry;
```

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarInputStream.eFactory Field**

Factory used to create TarEntry or descendant class instance

```csharp
protected IEntryFactory eFactory;
```

**See Also**

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
**TarInputStream.entryOffset Field**

Number of bytes read for this entry so far

```csharp
protected long entryOffset;
```

See Also

[TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
TarInputStream.entrySize Field

Size of this entry as recorded in header

```java
protected long entrySize;
```

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarInputStream.hasHitEOF Field

Flag set when last block has been read

```csharp
protected bool hasHitEOF;
```

See Also

[TarInputStream Class]  |  [ICSharpCode.SharpZipLib.Tar Namespace]
TarInputStream.readBuf Field

Buffer used with calls to

Read()

protected byte[] readBuf;

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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><code>Available</code></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><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 indicating whether the current stream supports seeking. This property always returns false.</td>
</tr>
<tr>
<td><code>CanWrite</code></td>
<td>Gets a value indicating if the stream supports writing. This property always returns false.</td>
</tr>
<tr>
<td><code>IsMarkSupported</code></td>
<td>Since we do not support marking just yet, we return false.</td>
</tr>
<tr>
<td><code>Length</code></td>
<td>The length in bytes of the stream.</td>
</tr>
<tr>
<td><code>Position</code></td>
<td>Gets or sets the position within the stream. Setting the Position is not supported and throws a NotSupportedException.</td>
</tr>
</tbody>
</table>

**See Also**

TarInputStream Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
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.

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

See Also

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
Gets a value indicating whether the current stream supports reading

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

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
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 | ISharpCode.SharpZipLib.Tar Namespace
TarInputStream.CanWrite Property

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

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

See Also

TarInputStream Class | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
Since we do not support marking just yet, we return false.

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

See Also

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
**TarInputStream.Length Property**

The length in bytes of the stream

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

See Also

[TarInputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
**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><code>NotSupportedException</code></td>
<td>Any attempt to set position</td>
</tr>
</tbody>
</table>

### See Also

[TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
**TarInputStream Methods**

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 <strong>MarshalByRefObject</strong>)</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>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>Mark</strong></td>
<td>Since we do not support marking just yet, we do nothing.</td>
</tr>
<tr>
<td><strong>Read</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, int).</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Since we do not support marking just yet, we do nothing.</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>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 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>
**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](#)
SharpZip Compression 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](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
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
### TarInputStream.Flush Method

Flushes the baseInputStream

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

**See Also**

[TarInputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
TarInputStream.GetNextEntry Method

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 | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarInputStream.GetRecordSize Method

Get the record size being used by this stream's TarBuffer.

```csharp
public int GetRecordSize();
```

Return Value

TarBuffer record size.

See Also

TarInputStream Class | ISharpCode.SharpZipLib.Tar Namespace
TarInputStream.Mark Method

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 | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarInputStream.Read Method

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

```plaintext
public override int Read(
    byte[] outputBuffer,
    int offset,
    int count
);
```

Parameters

outputBuffer
   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
| SharpZip Compression Library |
TarInputStream.ReadByte Method

Reads a byte from the current tar archive entry. This method simply calls read(byte[], int, int).

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

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
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]
SharpZip Compression 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
SharpZip Compression 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
SharpZip Compression Library
Sets the length of the stream. This operation is not supported and will throw a NotSupportedException.

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

### 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](#)
SharpZip Compression 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 numToSkip
);
```

Parameters

numToSkip
The number of bytes to skip.

See Also

TarInputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
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[] array,
    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
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 val);
```

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](#)
SharpZip Compression Library
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

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
ICSharpCode.SharpZipLib.dll)

See Also

TarInputStream.EntryFactoryAdapter Members |
ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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 headerBuf</td>
</tr>
<tr>
<td>CreateEntryFromFile</td>
<td>Create a tar entry with details obtained from fileName</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>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

**TarInputStream.EntryFactoryAdapter Constructor**

Initializes a new instance of the `TarInputStream.EntryFactoryAdapter` class.

```java
public TarInputStream.EntryFactoryAdapter();
```

**See Also**

SharpZip Compression 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
SharpZip Compression Library
TarInputStream.EntryFactoryAdapter.CreateEntry Method

Create and entry based on details in headerBuf

Overload List

Create and entry based on details in headerBuf

public TarEntry CreateEntry(byte[]);

Create a TarEntry based on named

public TarEntry CreateEntry(string);

See Also

SharpZip Compression Library
Create and entry based on details in `headerBuf`

```csharp
public TarEntry CreateEntry(byte[] headerBuf);
```

Implements

`IEntryFactory.CreateEntry`

See Also

SharpZip Compression 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

- `TarInputStream.EntryFactoryAdapter` Class
- `ICSharpCode.SharpZipLib.Tar` Namespace
SharpZip Compression 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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TarInputStream.EntryFactoryAdapter</td>
<td>Standard entry factory class creating instances of the class TarEntry</td>
</tr>
</tbody>
</table>

Requirements


See Also

TarInputStream.IEntryFactory Members | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarInputStream.IEntryFactory Members

TarInputStream.IEntryFactory overview

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

TarInputStream.IEntryFactory Interface | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
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><img src="#" alt="CreateEntry" /></td>
<td>Overloaded. Create a tar entry based on the header information passed</td>
</tr>
<tr>
<td><img src="#" alt="CreateEntryFromFile" /></td>
<td>Create an instance based on an actual file</td>
</tr>
</tbody>
</table>

**See Also**

- [TarInputStream.IEntryFactory Interface](#)
- [ICSharpCode.SharpZipLib.Tar Namespace](#)
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 | ISharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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**

- `TarInputStream.IEntryFactory Interface`  
- `ICSharpCode.SharpZipLib.Tar Namespace`  
- `TarInputStream.IEntryFactory.CreateEntry Overload List`
SharpZip Compression Library
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
- ISharpCode.SharpZipLib.Tar Namespace
- TarInputStream.IEntryFactory.CreateEntry Overload List
**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  |  *ICSharpCode.SharpZipLib.Tar Namespace*
TarOutputStream Class

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


See Also

TarOutputStream Members | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
# TarOutputStream Members

## TarOutputStream overview

### Public Instance Constructors

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

### Public Instance Properties

| CanRead | true if the stream supports reading; otherwise, false. |
| CanSeek | true if the stream supports seeking; otherwise, false. |
| CanWrite | true if stream supports writing; otherwise, false. |
| Length | length of stream in bytes |
| Position | gets or sets the position within the current stream. |

### Public Instance Methods

| BeginRead (inherited from Stream) | Begins an asynchronous read operation. |
| BeginWrite (inherited from Stream) | Begins an asynchronous write operation. |
| Close | Ends the TAR archive and closes the underlying OutputStream. This means that finish() is called followed by calling the TarBuffer's close(). |
| CloseEntry | 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>Ends the TAR archive without closing the underlying OutputStream. The result is that the EOF record 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 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>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetRecordSize</strong></td>
<td>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 <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>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 completely written to the output stream.</td>
</tr>
<tr>
<td><strong>Read</strong></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><strong>.ReadByte</strong></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><strong>Seek</strong></td>
<td>set the position within the current stream</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>set the length of the current stream</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>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>---</td>
<td>---</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**

| 'assembleBuf' | 'Assembly' buffer used to assemble data before writing |
| 'assembleLen' | current 'Assembly' buffer length |
| 'blockBuf' | single block working buffer |
| 'buffer' | TarBuffer used to provide correct blocking factor |
| 'currBytes' | bytes written for this entry so far |
| 'currSize' | Size for the current entry |
| 'debug' | flag indicating debugging code should be activated or not |
| 'outputStream' | the destination stream for the archive contents |

**Protected Instance Methods**

| 'CreateWaitHandle' (inherited) | 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]
**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]
SharpZip Compression Library
Construct TarOutputStream using default block factor

```csharp
public TarOutputStream(
    Stream outputStream
);
```

**Parameters**

`outputStream`

stream to write to

**See Also**

SharpZip Compression Library
TarOutputStream Constructor (Stream, Int32)

Construct TarOutputStream with user specified block factor

```java
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
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>assemBuf</td>
<td>'Assembly' buffer used to assemble data before writing</td>
</tr>
<tr>
<td>assemLen</td>
<td>current 'Assembly' buffer length</td>
</tr>
<tr>
<td>blockBuf</td>
<td>single block working buffer</td>
</tr>
<tr>
<td>buffer</td>
<td>TarBuffer used to provide correct blocking factor</td>
</tr>
<tr>
<td>currBytes</td>
<td>bytes written for this entry so far</td>
</tr>
<tr>
<td>currSize</td>
<td>Size for the current entry</td>
</tr>
<tr>
<td>debug</td>
<td>flag indicating debugging code should be activated or not</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
TarOutputStream.assemBuf Field

'Assembly' buffer used to assemble data before writing

```csharp
protected byte[] assemBuf;
```

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarOutputStream.assemLen Field**

Current 'Assembly' buffer length

```csharp
protected int assemLen;
```

See Also

[ TarOutputStream Class ](#) | [ ICSharpCode.SharpZipLib.Tar Namespace ](#)
SharpZip Compression Library
**TarOutputStream.blockBuf Field**

single block working buffer

```java
protected byte[] blockBuf;
```

See Also

[TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarBuffer used to provide correct blocking factor

protected TarBuffer buffer;

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library


**TarOutputStream.currBytes Field**

bytes written for this entry so far

```
protected long currBytes;
```

See Also

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
TarOutputStream.currSize Field

Size for the current entry

```java
protected long currSize;
```

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
**TarOutputStream.debug Field**

flag indicating debugging code should be activated or not

```csharp
protected bool debug;
```

See Also

[TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
TarOutputStream.outputStream Field

the destination stream for the archive contents

protected Stream outputStream;

See Also

TarOutputStream Class  |  ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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>
</tbody>
</table>

**See Also**

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
TarOutputStream.CanRead Property

true if the stream supports reading; otherwise, false.

public override bool CanRead {get;}

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarOutputStream.CanSeek Property

true if the stream supports seeking; otherwise, false.

public override bool CanSeek {get;}
SharpZip Compression 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](#)
TarOutputStream.Length Property

length of stream in bytes

public override long Length {get;}

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
**TarOutputStream.Position Property**

gets or sets the position within the current stream.

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

See Also

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression 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. This means that finish() is called followed by calling the TarBuffer's close().</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>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>✯ <strong>EndWrite</strong> <em>(inherited from Stream)</em></td>
<td>Ends an asynchronous write operation.</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>Finish</strong></td>
<td>Ends the TAR archive without closing the underlying OutputStream. The result is that the EOF record 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> <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>GetLifetimeService</strong> <em>(inherited from MarshalByRefObject)</em></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>Get the record size being used by this stream's TarBuffer.</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>InitializeLifetimeService</strong> <em>(inherited from MarshalByRefObject)</em></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</td>
</tr>
</tbody>
</table>
Contents are written, closeEntry() **MUST** be called to ensure that all buffered data is completely written to the output stream.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Read</strong></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><strong>ReadByte</strong></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><strong>Seek</strong></td>
<td>set the position within the current stream</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>set the length of the current stream</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>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</td>
</tr>
</tbody>
</table>
simply calls `Write(byte[], int, int)`.

### Protected Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ <strong>CreateWaitHandle</strong></td>
<td>Allocates a <code>WaitHandle</code> object.</td>
</tr>
<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**

[TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
Ends the TAR archive and closes the underlying OutputStream. This means that finish() is called followed by calling the TarBuffer's close().

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

See Also

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression 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
SharpZip Compression Library
**TarOutputStream.Finish Method**

Ends the TAR archive without closing the underlying OutputStream. The result is that the EOF record of nulls is written.

```java
public void Finish();
```

**See Also**

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarOutputStream.Flush Method

All buffered data is written to destination

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

See Also

[TarOutputStream Class] | [ICSharpCode.SharpZipLib.Tar Namespace]
SharpZip Compression Library
TarOutputStream.GetRecordSize Method

Get the record size being used by this stream's TarBuffer.

```csharp
public int GetRecordSize();
```

Return Value

The TarBuffer record size.

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression 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.

```java
public void PutNextEntry(TarEntry entry);
```

Parameters

- `entry`
  - The TarEntry to be written to the archive.

See Also

- TarOutputStream Class
- ICSharpCode.SharpZipLib.Tar Namespace
TarOutputStream.Read Method

read bytes from the current stream and advance the position within the stream by the number of bytes read.

public override int Read(
    byte[] b,
    int off,
    int len
);

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
SharpZip Compression 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 | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarOutputStream.Seek Method

set the position within the current stream

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

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
SharpZip Compression Library
TarOutputStream.SetLength Method

set the length of the current stream

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

See Also

TarOutputStream Class | ICSharpCode.SharpZipLib.Tar Namespace
**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[] wBuf,
    int wOffset,
    int numToWrite
);
```

**Parameters**

- `wBuf`
  - The buffer to write to the archive.

- `wOffset`
  - The offset in the buffer from which to get bytes.

- `numToWrite`
  - The number of bytes to write.

**See Also**

- [TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression 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 b);
```

Parameters

- `b` The byte to be written.

See Also

- [TarOutputStream Class](#) | [ICSharpCode.SharpZipLib.Tar Namespace](#)
SharpZip Compression Library
# ISharpCode.SharpZipLib.Zip Namespace

## Namespace hierarchy

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FastZip</strong></td>
<td>FastZip provides facilities for creating and extracting zip files. Only relative paths are supported.</td>
</tr>
<tr>
<td><strong>FastZipEvents</strong></td>
<td>FastZipEvents supports all events applicable to FastZip operations.</td>
</tr>
<tr>
<td><strong>KeysRequiredEventArgs</strong></td>
<td>Arguments used with KeysRequiredEvent</td>
</tr>
<tr>
<td><strong>ZipConstants</strong></td>
<td>This class contains constants used for Zip format files</td>
</tr>
<tr>
<td><strong>ZipEntry</strong></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</td>
</tr>
<tr>
<td></td>
<td>this class as information about the members in an archive. ZipOutputStream uses an instance of this class when creating an entry in a Zip</td>
</tr>
<tr>
<td></td>
<td>file.</td>
</tr>
<tr>
<td></td>
<td>Author of the original java version : Jochen Hoenicke</td>
</tr>
<tr>
<td><strong>ZipException</strong></td>
<td>Represents errors specific to Zip file handling</td>
</tr>
<tr>
<td><strong>ZipFile</strong></td>
<td>This class represents a Zip archive. You can ask for the contained entries, or get an</td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ZipInputStream</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>ZipNameTransform</strong></td>
<td>ZipNameTransform transforms name as per the Zip file convention.</td>
</tr>
<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>
</tbody>
</table>
Delegates

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FastZip.ConfirmOverwriteDelegate</code></td>
<td>Delegate called when confirming overwriting of files.</td>
</tr>
</tbody>
</table>

Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CompressionMethod</code></td>
<td>The kind of compression used for an entry in an archive</td>
</tr>
<tr>
<td><code>FastZip.Overwrite</code></td>
<td>Defines the desired handling when overwriting files.</td>
</tr>
</tbody>
</table>
SharpZip Compression 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</td>
</tr>
<tr>
<td>BZip2</td>
<td>Not supported by #Zip</td>
</tr>
<tr>
<td>WinZipAES</td>
<td>WinZip special for AES encryption, Not supported by #Zip</td>
</tr>
</tbody>
</table>

Requirements

**Namespace:** ICSnarpCode.SharpZipLib.Zip

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

See Also

SharpZip Compression Library
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

SharpZip Compression Library
### FastZip Members

#### FastZip overview

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FastZip</code></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><code>CreateEmptyDirectories</code></td>
<td>Get/set a value indicating whether empty directories should be created.</td>
</tr>
<tr>
<td><code>NameTransform</code></td>
<td>Get or set the active when creating Zip files.</td>
</tr>
</tbody>
</table>

**Public Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateZip</code></td>
<td>Overloaded. Create a zip file.</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>ExtractZip</code></td>
<td>Overloaded. Extract the contents of a zip file.</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</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td>free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>------------</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>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
FastZip Constructor

Initialize a default instance of FastZip.

Overload List

Initialize a default instance of FastZip.

public FastZip();

Initialise a new instance of FastZip

public FastZip(FastZipEvents);

See Also

SharpZip Compression Library
FastZip Constructor ()

Initialize a default instance of FastZip.

```csharp
public FastZip();
```

See Also

- FastZip Class
- FastZip Constructor Overload List
SharpZip Compression Library
FastZip Constructor (FastZipEvents)

Initialise a new instance of FastZip

```java
public FastZip(
    FastZipEvents events
);
```

Parameters

*events*

See Also

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>NameTransform</td>
<td>Get or set the active when creating Zip files.</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
Get/set a value indicating whether empty directories should be created.

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

See Also

- FastZip Class
SharpZip Compression Library
Get or set the active when creating Zip files.

```csharp
public ZipNameTransform NameTransform {get; }
```

See Also

- [FastZip Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression 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>CreateZip</td>
<td>Overloaded. Create a zip file.</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>ExtractZip</td>
<td>Overloaded. Extract the contents of a zip file.</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>
</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

SharpZip Compression Library
FastZip.CreateZip Method

Create a zip file/archive.

**Overload List**

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**

SharpZip Compression Library
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**

SharpZip Compression 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**

SharpZip Compression 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

SharpZip Compression Library
FastZip.ExtractZip Method (String, String, Overwrite, ConfirmOverwriteDelegate, String, String)

Exatract the contents of a zip file.

```csharp
public void ExtractZip(
    string zipFileName,
    string targetDirectory,
    Overwrite overwrite,
    ConfirmOverwriteDelegate confirmDelegate,
    string fileFilter,
    string directoryFilter
);
```

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.

See Also

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
- FastZip.ExtractZip Overload List
SharpZip Compression Library
**FastZip.ConfirmOverwriteDelegate Delegate**

Delegate called when confirming overwriting of files.

```csharp
public delegate bool FastZip.ConfirmOverwriteDelegate(string fileName);
```

requirements

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

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

see also

SharpZip Compression Library
# FastZip.Overwrite Enumeration

Defines the desired handling when overwriting files.

```csharp
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

**Namespace:** [ICSharpCode.SharpZipLib.Zip](https://icsharpcode.net/code/nick/SharpZipLib/

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

## See Also


SharpZip Compression Library
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


See Also

SharpZip Compression Library
# FastZipEvents Members

## FastZipEvents overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastZipEvents Constructor</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>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>

### 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>OnDirectoryFailure</td>
<td>Raise the directory failure event.</td>
</tr>
<tr>
<td>OnFileFailure</td>
<td>Raises the file failure event.</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

SharpZip Compression Library
FastZipEvents Constructor

Initializes a new instance of the FastZipEvents class.

```csharp
public FastZipEvents();
```

See Also

SharpZip Compression 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

SharpZip Compression Library
Delegate to invoke when processing directory failures.

```csharp
public DirectoryFailureDelegate DirectoryFailure;
```

See Also

SharpZip Compression Library
FastZipEvents.FileFailure Field

Delegate to invoke when processing file failures.

```csharp
public FileFailureDelegate FileFailure;
```

See Also

SharpZip Compression Library
Delegate to invoke when processing directories.

```csharp
```

See Also

- [FastZipEvents Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression Library
FastZipEvents.ProcessFile Field

Delegate to invoke when processing files.

```csharp
public ProcessFileDelegate ProcessFile;
```

See Also

SharpZip Compression Library
FastZipEvents Methods

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><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>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>OnDirectoryFailure</strong></td>
<td>Raise the directory failure event.</td>
</tr>
<tr>
<td><strong>OnFileFailure</strong></td>
<td>Raises the file failure event.</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**
FastZipEvents.OnDirectoryFailure Method

Raise the directory failure event.

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

Parameters

- **directory**
  - The directory.

- **e**
  - The exception for this event.

See Also

- FastZipEvents Class
SharpZip Compression Library
FastZipEvents.OnFileFailure Method

Raises the file failure event.

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

Parameters

- **file**
  The file for this event.

- **e**
  The exception for this event.

See Also

SharpZip Compression Library

Raises the ProcessDirectoryEvent.

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

Parameters

- **directory**
  The directory for this event.

- **hasMatchingFiles**
  Flag indicating if directory has matching files as determined by the current filter.

See Also

SharpZip Compression Library
FastZipEvents.OnProcessFile Method

Raises the ProcessFileEvent.

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

Parameters

- `file`  
  The file for this event.

See Also

**KeysRequiredEventArgs Class**

Arguments used with KeysRequiredEvent

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


```csharp
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**

SharpZip Compression Library
## KeysRequiredEventArgs Members

### KeysRequiredEventArgs overview

### Public Instance Constructors

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

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>FileName</code></td>
<td>Get the name of the file for which keys are required.</td>
</tr>
<tr>
<td><code>Key</code></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><code>Equals</code> (inherited from Object)</td>
<td>Determines whether the specified Object is equal to the current Object.</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 Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><code>ToString</code> (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><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)</td>
<td>Creates a shallow copy of the</td>
</tr>
</tbody>
</table>
See Also

SharpZip Compression Library
KeysRequiredEventArgs Constructor

Initialise a new instance of `KeysRequiredEventArgs`.

**Overload List**

Initialise a new instance of `KeysRequiredEventArgs`

```csharp
public KeysRequiredEventArgs(string);
```

Initialise a new instance of `KeysRequiredEventArgs`

```csharp
public KeysRequiredEventArgs(string[], byte[]);
```

**See Also**

SharpZip Compression Library
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**

- `KeysRequiredEventArgs Class`
- `KeysRequiredEventArgs Constructor Overload List`
SharpZip Compression 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

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>FileName</td>
<td>Get the name of the file for which keys are required.</td>
</tr>
<tr>
<td>Key</td>
<td>Get/set the key value</td>
</tr>
</tbody>
</table>

### See Also

SharpZip Compression Library
Get the name of the file for which keys are required.

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

See Also

SharpZip Compression Library
KeysRequiredEventArgs.Key Property

Get/set the key value

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

See Also

- [KeysRequiredEventArgs Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression Library
**ZipConstants Class**

This class contains constants used for Zip format files.

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

```csharp
```

```csharp
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**

- **Namespace:** [ICSharpCode.SharpZipLib.Zip](#)
- **Assembly:** ICSharpCode.SharpZipLib (in ICSharpCode.SharpZipLib.dll)

**See Also**

# ZipConstants Members

## ZipConstants overview

### Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>S CENATT</code></td>
<td>Offset of internal file attributes in central file header</td>
</tr>
<tr>
<td><code>S CENATX</code></td>
<td>Offset of external file attributes in central file header</td>
</tr>
<tr>
<td><code>S CENCOM</code></td>
<td>Offset of file comment length in central file header</td>
</tr>
<tr>
<td><code>S CENCRC</code></td>
<td>Offset of crc-32 in central file header</td>
</tr>
<tr>
<td><code>S CENDIGITALSIG</code></td>
<td>Central header digital signature</td>
</tr>
<tr>
<td><code>S CENDSK</code></td>
<td>Offset of disk start number in central file header</td>
</tr>
<tr>
<td><code>S CENEXT</code></td>
<td>Offset of extra field length in central file header</td>
</tr>
<tr>
<td><code>S CENFLG</code></td>
<td>Offset of general purpose bit flag in central file header</td>
</tr>
<tr>
<td><code>S CENHDR</code></td>
<td>Size of central header entry</td>
</tr>
<tr>
<td><code>S CENHOW</code></td>
<td>Offset of compression method in central file header</td>
</tr>
<tr>
<td><code>S CENLEN</code></td>
<td>Offset of uncompressed size in central file header</td>
</tr>
<tr>
<td><code>S CENNAM</code></td>
<td>Offset of file name length in central file header</td>
</tr>
<tr>
<td><code>S CENOFF</code></td>
<td>Offset of relative offset of</td>
</tr>
<tr>
<td>Local Header in Central File Header</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---</td>
</tr>
<tr>
<td>s CENSIG</td>
<td>Signature for central header</td>
</tr>
<tr>
<td>s CENSIG64</td>
<td>Signature for Zip64 central file header</td>
</tr>
<tr>
<td>s CENSIZ</td>
<td>Offset of compressed size in central file header</td>
</tr>
<tr>
<td>s CENTIM</td>
<td>Offset of time/date in central file header</td>
</tr>
<tr>
<td>s CENVEM</td>
<td>Offset of version made by in central file header</td>
</tr>
<tr>
<td>s CENVER</td>
<td>Offset of version needed to extract in central file header</td>
</tr>
<tr>
<td>s CRYPTO_HEADER_SIZE</td>
<td>Size of cryptographic header stored before entry data</td>
</tr>
<tr>
<td>s ENDCOM</td>
<td>Offset of ZIP file comment length</td>
</tr>
<tr>
<td>s ENDDCD</td>
<td>Offset of number of disk with start of central directory</td>
</tr>
<tr>
<td>s ENDHDR</td>
<td>Size of end of central record (excluding variable fields)</td>
</tr>
<tr>
<td>s ENDRND</td>
<td>Offset of number of this disk</td>
</tr>
<tr>
<td>s ENDOFF</td>
<td>Offset of offset of start of central directory with respect to starting disk number</td>
</tr>
<tr>
<td>s ENDSIG</td>
<td>End of central directory</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENDSIZ</td>
<td>Offset of size of central directory</td>
</tr>
<tr>
<td>ENDSUB</td>
<td>Offset of number of entries in the central directory of this disk</td>
</tr>
<tr>
<td>ENDTOT</td>
<td>Offset of total number of entries in the central directory</td>
</tr>
<tr>
<td>EXTCRC</td>
<td>Offset of crc-32 in data descriptor</td>
</tr>
<tr>
<td>EXTHDR</td>
<td>Size of data descriptor</td>
</tr>
<tr>
<td>EXTLEN</td>
<td>Offset of uncompressed length in data descriptor</td>
</tr>
<tr>
<td>EXTSIG</td>
<td>Signature for data descriptor</td>
</tr>
<tr>
<td>EXTSIZ</td>
<td>Offset of compressed size in data descriptor</td>
</tr>
<tr>
<td>LOCCRC</td>
<td>Offset of crc-32 in local entry header</td>
</tr>
<tr>
<td>LOCEXT</td>
<td>Offset of extra field length in local entry header</td>
</tr>
<tr>
<td>LOCFLG</td>
<td>Offset of general purpose flags in local entry header</td>
</tr>
<tr>
<td>LOCHDR</td>
<td>Size of local entry header (excluding variable length fields at end)</td>
</tr>
<tr>
<td>LOCHOW</td>
<td>Offset of compression method in local entry header</td>
</tr>
<tr>
<td>LOCLEN</td>
<td>Offset of uncompressed size in local entry header</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>LOCNAM</strong></td>
<td>Offset of file name length in local entry header</td>
</tr>
<tr>
<td><strong>LOCSIG</strong></td>
<td>Signature for local entry header</td>
</tr>
<tr>
<td><strong>LOCSIZ</strong></td>
<td>Offset of compressed size in local entry header</td>
</tr>
<tr>
<td><strong>LOCTIM</strong></td>
<td>Offset of last mod file time + date in local entry header</td>
</tr>
<tr>
<td><strong>LOCVER</strong></td>
<td>Offset of version to extract in local entry header</td>
</tr>
<tr>
<td><strong>SPANNINGSIG</strong></td>
<td>Signature for spanning entry</td>
</tr>
<tr>
<td><strong>SPANTEMPSIG</strong></td>
<td>Signature for temporary spanning entry</td>
</tr>
<tr>
<td><strong>VERSION_MADE_BY</strong></td>
<td>The version made by field for entries in the central header when created by this library</td>
</tr>
<tr>
<td><strong>VERSION_STRONG_ENCRYPTION</strong></td>
<td>The minimum version required to support strong encryption</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>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</td>
</tr>
</tbody>
</table>
### Public Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ConvertToArray</code></td>
<td>Convert a string to a byte array</td>
</tr>
<tr>
<td><code>ConvertToString</code></td>
<td>Overloaded. Convert a portion of a byte array to a string.</td>
</tr>
</tbody>
</table>

### Public Instance Constructors

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

SharpZip Compression Library
ZipConstants Constructor

Initializes a new instance of the ZipConstants class.

```java
public ZipConstants();
```

See Also

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression 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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S CENATT</strong></td>
<td>Offset of internal file attributes in central file header</td>
</tr>
<tr>
<td><strong>S CENATX</strong></td>
<td>Offset of external file attributes in central file header</td>
</tr>
<tr>
<td><strong>S CENCOM</strong></td>
<td>Offset of file comment length in central file header</td>
</tr>
<tr>
<td><strong>S CENCRC</strong></td>
<td>Offset of crc-32 in central file header</td>
</tr>
<tr>
<td><strong>S CENDIGITALSIG</strong></td>
<td>Central header digital signature</td>
</tr>
<tr>
<td><strong>S CENDSK</strong></td>
<td>Offset of disk start number in central file header</td>
</tr>
<tr>
<td><strong>S CENEXT</strong></td>
<td>Offset of extra field length in central file header</td>
</tr>
<tr>
<td><strong>S CENFLG</strong></td>
<td>Offset of general purpose bit flag in central file header</td>
</tr>
<tr>
<td><strong>S CENHDR</strong></td>
<td>Size of central header entry</td>
</tr>
<tr>
<td><strong>S CENHOW</strong></td>
<td>Offset of compression method in central file header</td>
</tr>
<tr>
<td><strong>S CENLEN</strong></td>
<td>Offset of uncompressed size in central file header</td>
</tr>
<tr>
<td><strong>S CENNAM</strong></td>
<td>Offset of file name length in central file header</td>
</tr>
<tr>
<td><strong>S CENOFF</strong></td>
<td>Offset of relative offset of local header in central file header</td>
</tr>
<tr>
<td><strong>S CENSIG</strong></td>
<td>Signature for central header</td>
</tr>
<tr>
<td><strong>S CENSIG64</strong></td>
<td>Signature for Zip64 central file header</td>
</tr>
<tr>
<td><strong>S CENSIZ</strong></td>
<td>Offset of compressed size in central file header</td>
</tr>
<tr>
<td><strong>S CENTIM</strong></td>
<td>Offset of time/date in central file header</td>
</tr>
<tr>
<td><strong>S CENVEM</strong></td>
<td>Offset of version made by in central file header</td>
</tr>
<tr>
<td><strong>S CENVER</strong></td>
<td>Offset of version needed to extract in central file header</td>
</tr>
<tr>
<td><strong>S CRYPTO_HEADER_SIZE</strong></td>
<td>Size of cryptographic header stored before entry data</td>
</tr>
<tr>
<td><strong>S ENDCOM</strong></td>
<td>Offset of ZIP file comment length</td>
</tr>
<tr>
<td><strong>S ENDDCD</strong></td>
<td>Offset of number of disk with start of central directory</td>
</tr>
<tr>
<td><strong>S ENDHDR</strong></td>
<td>Size of end of central record (excluding variable fields)</td>
</tr>
<tr>
<td><strong>S ENDRNRD</strong></td>
<td>Offset of number of this disk</td>
</tr>
<tr>
<td><strong>S ENDOFF</strong></td>
<td>Offset of offset of start of central directory with respect to starting disk number</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;ENDSIG</td>
<td>End of central directory record signature</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;ENDSIZ</td>
<td>Offset of size of central directory</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;ENDSUB</td>
<td>Offset of number of entries in the central directory of this disk</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;ENDTOT</td>
<td>Offset of total number of entries in the central directory</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;EXTCRC</td>
<td>Offset of crc-32 in data descriptor</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;EXTHDR</td>
<td>Size of data descriptor</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;EXTLEN</td>
<td>Offset of uncompressed length in data descriptor</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;EXTSIG</td>
<td>Signature for data descriptor</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;EXTSIZ</td>
<td>Offset of compressed size in data descriptor</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;LOCCRC</td>
<td>Offset of crc-32 in local entry header</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;LOCEXT</td>
<td>Offset of extra field length in local entry header</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;LOCFLG</td>
<td>Offset of general purpose flags in local entry header</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;LOCHDR</td>
<td>Size of local entry header (excluding variable length fields at end)</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;LOCHOW</td>
<td>Offset of compression method in local entry header</td>
</tr>
<tr>
<td><strong>S</strong>&lt;br/&gt;LOCLEN</td>
<td>Offset of uncompressed</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOCNAM</td>
<td>Offset of file name length in local entry header</td>
</tr>
<tr>
<td>LOCSIG</td>
<td>Signature for local entry header</td>
</tr>
<tr>
<td>LOCSIZ</td>
<td>Offset of compressed size in local entry header</td>
</tr>
<tr>
<td>LOCTIM</td>
<td>Offset of last mod file time + date in local entry header</td>
</tr>
<tr>
<td>LOCVER</td>
<td>Offset of version to extract in local entry header</td>
</tr>
<tr>
<td>SPANNINGSIG</td>
<td>Signature for spanning entry</td>
</tr>
<tr>
<td>SPANTEMPSIG</td>
<td>Signature for temporary spanning entry</td>
</tr>
<tr>
<td>VERSION_MADE_BY</td>
<td>The version made by field for entries in the central header when created by this library</td>
</tr>
<tr>
<td>VERSION_STRONG_ENCRYPTION</td>
<td>The minimum version required to support strong encryption</td>
</tr>
</tbody>
</table>

See Also

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipConstants.CENATT Field

Offset of internal file attributes in central file header

```
public const int CENATT = 36;
```

See Also

SharpZip Compression Library
ZipConstants.CENATX Field

Offset of external file attributes in central file header

```csharp
public const int CENATX = 38;
```

See Also

SharpZip Compression Library
ZipConstants.CENCOM Field

Offset of file comment length in central file header

```
public const int CENCOM = 32;
```

See Also

SharpZip Compression Library
ZipConstants.CENCRC Field

Offset of crc-32 in central file header

public const int CENCRC = 16;

See Also

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipConstants.CENDIGITALSIG Field

Central header digital signature

```csharp
public const int CENDIGITALSIG = 84233040;
```

See Also

SharpZip Compression Library
# ZipConstants.CENDSK Field

Offset of disk start number in central file header

```csharp
public const int CENDSK = 34;
```

See Also

SharpZip Compression Library
ZipConstants.CENEXT Field

Offset of extra field length in central file header

```csharp
public const int CENEXT = 30;
```

See Also

SharpZip Compression Library
ZipConstants.CENFLG Field

Offset of general purpose bit flag in central file header

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

See Also

SharpZip Compression Library
ZipConstants.CENHDR Field

Size of central header entry

```csharp
public const int CENHDR = 46;
```

See Also

SharpZip Compression Library
ZipConstants.CENHOW Field

Offset of compression method in central file header

```csharp
public const int CENHOW = 10;
```

See Also

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipConstants.CENLEN Field

Offset of uncompressed size in central file header

```
public const int CENLEN = 24;
```

See Also

SharpZip Compression Library
ZipConstants.CENNAM Field

Offset of file name length in central file header

```csharp
public const int CENNAM = 28;
```

See Also

SharpZip Compression Library
ZipConstants.CENOFF Field

Offset of relative offset of local header in central file header

```java
public const int CENOFF = 42;
```

See Also

[ZipConstants Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
SharpZip Compression Library
ZipConstants.CENSIG Field

Signature for central header

```
public const int CENSIG = 33639248;
```

See Also

SharpZip Compression Library
ZipConstants.CENSIG64 Field

Signature for Zip64 central file header

```csharp
public const int CENSIG64 = 101075792;
```

See Also

SharpZip Compression Library
ZipConstants.CENSIZ Field

Offset of compressed size in central file header

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

See Also

SharpZip Compression Library
ZipConstants.CENTIM Field

Offset of time/date in central file header

```java
public const int CENTIM = 12;
```

See Also

SharpZip Compression Library
ZipConstants.CENVEM Field

Offset of version made by in central file header

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

See Also

SharpZip Compression Library
ZipConstants.CENVER Field

Offset of version needed to extract in central file header

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

See Also

SharpZip Compression Library
ZipConstants.CRYPTO_HEADER_SIZE Field

Size of cryptographic header stored before entry data

```csharp
public const int CRYPTO_HEADER_SIZE = 12;
```

See Also

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipConstants.ENDCOM Field

Offset of ZIP file comment length

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

See Also

SharpZip Compression Library
ZipConstants.ENDDCD Field

Offset of number of disk with start of central directory

```
public const int ENDDCD = 6;
```

See Also

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipConstants.ENDHDR Field

Size of end of central record (excluding variable fields)

```csharp
public const int ENDHDR = 22;
```

See Also

SharpZip Compression Library
ZipConstants.ENDNRD Field

Offset of number of this disk

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

See Also

ZipConstants.ENDOFF Field

Offset of offset of start of central directory with respect to starting disk number

```
public const int ENDOFF = 16;
```

See Also

- [ZipConstants Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression Library
ZipConstants.ENDSIG Field

End of central directory record signature

```
public const int ENDSIG = 101010256;
```

See Also

SharpZip Compression Library
ZipConstants.ENDSIZ Field

Offset of size of central directory

```java
public const int ENDSIZ = 12;
```

See Also

SharpZip Compression Library
ZipConstants.ENDSUB Field

Offset of number of entries in the central directory of this disk

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

See Also

SharpZip Compression Library
ZipConstants.ENDTOT Field

Offset of total number of entries in the central directory

```csharp
public const int ENDTOT = 10;
```

See Also

SharpZip Compression Library
ZipConstants.EXTCRC Field

Offset of crc-32 in data descriptor

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

See Also

SharpZip Compression Library
ZipConstants.EXTHDR Field

Size of data descriptor

```csharp
public const int EXTHDR = 16;
```

See Also

ZipConstants.EXTLEN Field

Offset of uncompressed length in data descriptor

```csharp
public const int EXTLEN = 12;
```

See Also

SharpZip Compression Library
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

SharpZip Compression Library
ZipConstants.EXTSIZ Field

Offset of compressed size in data descriptor

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

See Also

- ZipConstants Class
SharpZip Compression Library
ZipConstants.LOCCRC Field

Offset of crc-32 in local entry header

```
public const int LOCCRC = 14;
```

See Also

SharpZip Compression Library
ZipConstants.LOCEXT Field

Offset of extra field length in local entry header

```
public const int LOCEXT = 28;
```

See Also

SharpZip Compression Library
ZipConstants.LOCFLG Field

Offset of general purpose flags in local entry header

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

See Also

SharpZip Compression Library
ZipConstants.LOCHDR Field

Size of local entry header (excluding variable length fields at end)

```java
public const int LOCHDR = 30;
```

See Also

- [ZipConstants Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression Library
Offset of compression method in local entry header

```
public const int LOCHOW = 8;
```

See Also

SharpZip Compression Library
**ZipConstants.LOCLEN Field**

Offset of uncompressed size in local entry header

```csharp
public const int LOCLEN = 22;
```

See Also

SharpZip Compression Library
ZipConstants.LOCNAM Field

Offset of file name length in local entry header

```
public const int LOCNAM = 26;
```

See Also

SharpZip Compression Library
ZipConstants.LOCSIG Field

Signature for local entry header

```csharp
public const int LOCSIG = 67324752;
```

See Also

SharpZip Compression Library
ZipConstants.LOCSIZ Field

Offset of compressed size in local entry header

```csharp
public const int LOCSIZ = 18;
```

See Also

SharpZip Compression Library
ZipConstants.LOCTIM Field

Offset of last mod file time + date in local entry header

\[
\text{public const int LOCTIM = 10;}
\]

See Also

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipConstants.LOCVER Field

Offset of version to extract in local entry header

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

See Also

SharpZip Compression Library
ZipConstants.SPANNINGSIG Field

Signature for spanning entry

```csharp
public const int SPANNINGSIG = 134695760;
```

See Also

SharpZip Compression Library
ZipConstants.SPANTEMPSIG Field

Signature for temporary spanning entry

```csharp
public const int SPANTEMPSIG = 808471376;
```

See Also

- [ZipConstants Class](#) | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipConstants.VERSION_MADE_BY Field

The version made by field for entries in the central header when created by this library

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

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

ZipConstants Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
The minimum version required to support strong encryption

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

See Also

SharpZip Compression 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><code>DefaultCodePage</code></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 compatibility.</td>
</tr>
</tbody>
</table>

### See Also

SharpZip Compression 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 neccessarily there are many variable factors, codepage 850 is often a good choice for European users, however be careful about compatability.

```public static int DefaultCodePage {get; set;}
```

See Also

SharpZip Compression Library
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>ToArray</strong></td>
<td>Convert a string to a byte array</td>
</tr>
<tr>
<td>✷ <strong>ToString</strong></td>
<td>Overloaded. Convert a portion of a byte array to a 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></td>
<td>(inherited from <strong>Object</strong>) Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td>✷ <strong>GetHashCode</strong></td>
<td>(inherited from <strong>Object</strong>) 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 <strong>Object</strong>) Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>✷ <strong>ToString</strong></td>
<td>(inherited from <strong>Object</strong>) Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
ZipConstants.ConvertToArray Method

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

SharpZip Compression Library
ZipConstants.ConvertToString Method

Convert byte array to string

Overload List

Convert 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

SharpZip Compression Library
ZipConstants.ConvertToString Method (Byte[])  

Convert 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  

SharpZip Compression Library
ZipConstants.ConvertToString Method (Byte[], Int32)

Convert a portion of a byte array to a string.

```csharp
public static string ConvertToString(
    byte[] data,
    int length
);
```

Parameters

- **data**
  Data to convert to string

- **length**
  Number of bytes to convert starting from index 0

Return Value

- `data[0]..data[length - 1]` converted to a string

See Also

SharpZip Compression 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

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
SharpZip Compression 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>Overloaded. 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 ('/').</td>
</tr>
</tbody>
</table>

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</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>Comment</strong></td>
<td>Gets/Sets the entry comment.</td>
</tr>
<tr>
<td><strong>CompressedSize</strong></td>
<td>Gets/Sets the size of the compressed data.</td>
</tr>
<tr>
<td><strong>CompressionMethod</strong></td>
<td>Gets/Sets the compression method. Only Deflated and Stored are supported.</td>
</tr>
<tr>
<td><strong>Crc</strong></td>
<td>Gets/Sets the crc of the uncompressed data.</td>
</tr>
<tr>
<td><strong>DateTime</strong></td>
<td>Gets/Sets the time of last modification of the entry.</td>
</tr>
<tr>
<td><strong>DosTime</strong></td>
<td>Get/Set DosTime</td>
</tr>
<tr>
<td><strong>ExternalFileAttributes</strong></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><strong>ExtraData</strong></td>
<td>Gets/Sets the extra data.</td>
</tr>
<tr>
<td><strong>Flags</strong></td>
<td>Get/Set general purpose bit flag for entry</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. A directory is determined by an entry name with a trailing slash '/'. The external file attributes can also mark a file as a directory. The trailing slash convention should always be followed 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>Name</strong></td>
<td>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 <a href="#">CleanName</a>.</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>RequiresZip64</td>
<td>Gets a value indicating if the entry requires Zip64 extensions to be stored</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</td>
</tr>
<tr>
<td></td>
<td>indicates the major version number, and the value mod 10 is the minor</td>
</tr>
<tr>
<td></td>
<td>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><code>Clone</code></td>
<td>Creates a copy of this zip entry.</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>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 Object)</td>
<td>Gets the Type of the current instance.</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><strong>Finalize</strong></td>
<td>(inherited from <strong>Object</strong>) 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>(inherited from <strong>Object</strong>) Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression Library
ZipEntry Constructor

Creates a zip entry with the given name.

Overload List

Creates a copy of the given zip entry.

public ZipEntry(ZipEntry);

Creates a zip entry with the given name.

public ZipEntry(string);

See Also

ZipEntry Class  |  ICSharpCode.SharpZipLib.Zip Namespace
SharpZip Compression 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 no device names and path elements separated by '/\' characters. This is not enforced see `CleanName` on how to ensure names are valid if this is desired.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ArgumentNullException</code></td>
<td>The name passed is null</td>
</tr>
</tbody>
</table>

**See Also**

- ZipEntry Class
- ZipEntry Constructor Overload List
**ZipEntry Constructor (ZipEntry)**

Creates a copy of the given zip entry.

```java
public ZipEntry(
    ZipEntry e
);
```

**Parameters**

- `e`
  - The entry to copy.

**See Also**

SharpZip Compression Library
## 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>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 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>HostSystem</td>
<td>Gets the compatibility information for the <a href="#">external file attribute</a> 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</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. A directory is determined by an entry name with a trailing slash '/'. The external file attributes can also mark a file as a directory. The trailing slash convention should always be followed 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>Name</strong></td>
<td>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 CleanName.</td>
</tr>
<tr>
<td><strong>Offset</strong></td>
<td>Get/set offset for use in central header</td>
</tr>
<tr>
<td><strong>RequiresZip64</strong></td>
<td>Gets a value indicating if the entry requires Zip64 extensions to be stored</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Gets/Sets the size of the uncompressed data.</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>Get minimum Zip feature version required to extract this entry</td>
</tr>
<tr>
<td><strong>VersionMadeBy</strong></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><strong>ZipFileIndex</strong></td>
<td>Get/Set index of this entry in Zip file</td>
</tr>
</tbody>
</table>

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipEntry.Comment Property

Gets/Sets the entry comment.

```
public string Comment {get; set;}
```

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>If comment is longer than 0xffff.</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
ZipEntry.CompressedSize Property

Gets/Sets the size of the compressed data.

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>Size is not in the range 0..0xffffffff</td>
</tr>
</tbody>
</table>

See Also

- [ZipEntry Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression 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.Zip Namespace]
**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**

SharpZip Compression Library
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](#)
SharpZip Compression Library
ZipEntry.DosTime Property

Get/Set DosTime

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

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
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

SharpZip Compression 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><code>ArgumentOutOfRangeException</code></td>
<td>Extra data is longer than 0xffff bytes.</td>
</tr>
</tbody>
</table>

**See Also**

- [ZipEntry Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
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 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 | ICSharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipEntry.HostSystem Property

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.

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

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.Zip Namespace
SharpZip Compression 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
ZipEntry.IsDirectory Property

Gets a value indicating of the if the entry is a directory. A directory is determined by an entry name with a trailing slash '/'. The external file attributes can also mark a file as a directory. The trailing slash convention should always be followed however.

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

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipEntry.IsFile Property

Get a value of true if the entry appears to be a file; false otherwise

    public bool IsFile {get;}

Remarks

This only takes account Windows attributes. Other operating systems are ignored. For linux and others the result may be incorrect.

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
SharpZip Compression 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 CleanName.

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

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipEntry.Offset Property

Get/set offset for use in central header

```csharp
public int Offset {get; set;}
```

See Also

SharpZip Compression Library
ZipEntry.RequiresZip64 Property

Gets a value indicating if the entry requires Zip64 extensions to be stored

public bool RequiresZip64 {get;}

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
ZipEntry.Size Property

Gets/Sets the size of the uncompressed data.

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

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>If the size is not in the range 0..0xffffffffL</td>
</tr>
</tbody>
</table>

See Also

ZipEntry Class  | ISharpCode.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)

See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
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
SharpZip Compression Library
**ZipEntry.ZipFileIndex Property**

Get/Set index of this entry in Zip file

```
public int ZipFileIndex {get; set;}
```

See Also

SharpZip Compression 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><code>CleanName</code></td>
<td>Overloaded. 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 (‘/’).</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>Creates a copy of this zip entry.</td>
</tr>
<tr>
<td><code>Equals</code> (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><code>GetHashCode</code> (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><code>GetType</code> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</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 <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><code>MemberwiseClone</code> (inherited)</td>
<td>Creates a shallow copy of the</td>
</tr>
</tbody>
</table>
See Also

ZipEntry Class | ISharpCode.SharpZipLib.Zip Namespace
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 Windows-XPs built in Zip file handling.

Overload List

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 Windows-XPs built in Zip file handling.

    public static string CleanName(string);

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 ('/').

    public static string CleanName(string,bool);

See Also

ZipEntry Class | ICSharpCode.SharpZipLib.Zip Namespace
ZipEntry.CleanName Method (String)

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 Windows-XP's built in Zip file handling.

```csharp
public static string CleanName(
    string name
);
```

Parameters

- `name`
  Name to clean

See Also

- ZipEntry Class
- ZipEntry.CleanName Overload List
SharpZip Compression Library
ZipEntry.CleanName Method (String, Boolean)

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 ('/').

```csharp
public static string CleanName(
    string name,
    bool relativePath
);
```

Parameters

- `name`
  Name to clean

- `relativePath`
  Make names relative if true or absolute if false

See Also

- [ZipEntry Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [ZipEntry.CleanName Overload List](#)
SharpZip Compression Library
ZipEntry.Clone Method

Creates a copy of this zip entry.

```
public object Clone();
```

Implements

`ICloneable.Clone`

See Also

`ZipEntry Class` | `ICSharpCode.SharpZipLib.Zip Namespace`
SharpZip Compression Library
ZipEntry.ToString Method

Gets the string representation of this ZipEntry.

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

See Also

SharpZip Compression Library
ZipException Class

Represents errors specific to Zip file handling

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


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


See Also

## ZipException Members

### ZipException overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</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</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>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>GetObjectData</strong> (inherited from <strong>Exception</strong>)</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 <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> (inherited from <strong>Exception</strong>)</td>
<td>Creates and returns a string representation of the current exception.</td>
</tr>
</tbody>
</table>

Protected Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HResult</strong> (inherited from <strong>Exception</strong>)</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>Property</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

[ZipException Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
SharpZip Compression Library
ZipException Constructor

Initializes a new instance of the ZipException class.

Overload List

Initializes a new instance of the ZipException class.

public ZipException();

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

public ZipException(string);

See Also

ZipException Class | ISharpCode.SharpZipLib.Zip Namespace
ZipException Constructor ()

Initializes a new instance of the ZipException class.

```csharp
public ZipException();
```

See Also

- [ZipException Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
- [ZipException Constructor Overload List](#)
**ZipException Constructor (String)**

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

```java
public ZipException(
    string msg
);
```

See Also

[ZipException Class] | [ICSharpCode.SharpZipLib.Zip Namespace] | [ZipException Constructor Overload List]
| SharpZip Compression 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. This class is thread safe: 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

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)
    {
        ZipFile zFile = new ZipFile(args[0]);
        Console.WriteLine("Listing of : " + zFile.Name);
        Console.WriteLine("");
        Console.WriteLine("Raw Size		Size						Date					Time					Name");
        Console.WriteLine("--------		--------		--------		------		---------");
    }
}
foreach (ZipEntry e in zFile) {
    DateTime d = e.DateTime;
    Console.WriteLine("{0,-10}{1,-10}{2}		{3}			{4}",
                      e.Size, e.CompressedSize, d.ToString("dd-MM-yy"),
                      d.ToString("t"),
                      e.Name);
}

Requirements


See Also

SharpZip Compression Library
## ZipFile Members

### ZipFile overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</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>EntryByIndex</td>
<td>Indexer property for ZipEntries</td>
</tr>
<tr>
<td>Name</td>
<td>Gets the name of this zip file.</td>
</tr>
<tr>
<td>Password</td>
<td>Password to be used for encrypting/decrypting files.</td>
</tr>
<tr>
<td>Size</td>
<td>Gets the number of entries in this zip file.</td>
</tr>
<tr>
<td>ZipFileComment</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>Close</td>
<td>Closes the ZipFile. If the stream is IsStreamOwner then this also closes the underlying input stream. Once closed, no further instance methods should be called.</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>FindEntry</td>
<td>Return the index of the entry</td>
</tr>
</tbody>
</table>
with a matching name

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY(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</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</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td>TestArchive</td>
<td>Test an archive for integrity/validity</td>
</tr>
<tr>
<td>ToString</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>ENTRY(Finalize)</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>ENTRY(MemberwiseClone)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>

### See Also

SharpZip Compression Library
ZipFile Constructor

Opens a Zip file with the given name for reading.

**Overload List**

Opens a Zip file reading the given FileStream

```csharp
public ZipFile(FileStream);
```

Opens a Zip file reading the given Stream

```csharp
public ZipFile(Stream);
```

Opens a Zip file with the given name for reading.

```csharp
public ZipFile(string);
```

**See Also**

ZipFile Constructor (String)

Opens a Zip file with the given name for reading.

```csharp
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 Class
- ZipFile Constructor Overload List
SharpZip Compression Library
ZipFile Constructor (FileStream)

Opens a Zip file reading the given FileStream

```csharp
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

SharpZip Compression Library
ZipFile Constructor (Stream)

Opens a Zip file reading the given Stream

```csharp
public ZipFile(
    Stream baseStream
);
```

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

- ZipFile Class
- ZipFile Constructor Overload List
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

[ZipFile Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
SharpZip Compression Library
ZipFile.KeysRequired Field

Event handler for handling encryption keys.

```csharp
public(KeysRequiredEventHandler) KeysRequired;
```

See Also

SharpZip Compression 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><strong>EntryByIndex</strong></td>
<td>Indexer property for ZipEntries</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Gets the name of this zip file.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Password to be used for encrypting/decrypting files.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Gets the number of entries in this zip file.</td>
</tr>
<tr>
<td><strong>ZipFileComment</strong></td>
<td>Gets the comment for the zip file.</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
ZipFile.EntryByIndex Property

Indexer property for ZipEntries

```csharp
public ZipEntry this[int index] {get;}
```

See Also

SharpZip Compression Library
ZipFile.Name Property

Gets the name of this zip file.

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

See Also

- [ZipFile Class](#)
- [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression 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

SharpZip Compression Library
ZipFile.Size Property

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>InvalidOperationException</td>
<td>The Zip file has been closed.</td>
</tr>
</tbody>
</table>

See Also

[ZipFile Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
SharpZip Compression Library
Gets the comment for the zip file.

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

See Also

[ZipFile Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
SharpZip Compression 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 Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Close</strong></td>
<td>Closes the ZipFile. If the stream is <strong>IsStreamOwner</strong> then this also closes the underlying input stream. Once closed, no further instance methods should be called.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>(inherited from <strong>Object</strong>) Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>FindEntry</strong></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><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 <strong>Object</strong>) 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 <strong>Object</strong>) Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>TestArchive</td>
<td>Test an archive for integrity/validity</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

| **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

SharpZip Compression Library
ZipFile.Close Method

Closes the ZipFile. If the stream is `IsStreamOwner` 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><code>IOException</code></td>
<td>An i/o error occurs.</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression 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

SharpZip Compression 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

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

SharpZip Compression Library
**ZipFile.GetEnumerator Method**

Returns an enumerator for the Zip entries in this Zip file.

```csharp
public IEnumerator GetEnumerator();
```

**Implements**

[IEnumerable.GetEnumerator](#)

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">InvalidOperationException</a></td>
<td>The Zip file has been closed.</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression 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(int);

See Also

SharpZip Compression 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

ZipFile.GetInputStream Overload List
SharpZip Compression Library
ZipFile.GetInputStream Method (Int32)

Creates an input stream reading a zip entry

```csharp
public Stream GetInputStream(int 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>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**

*ZipFile Class* | *ICSharpCode.SharpZipLib.Zip Namespace* | *ZipFile.GetInputStream Overload List*
**ZipFile.TestArchive Method**

Test an archive for integrity/validity

```csharp
public bool TestArchive(
    bool testData
);
```

**Parameters**

`testData`  
Perform low level data Crc check

**Return Value**

true iff the test passes, false otherwise

**See Also**

SharpZip Compression Library
**ZipFile.KeysRequiredEventHandler Delegate**

Delegate for handling keys/password setting during compression/decompression.

```csharp
public delegate void ZipFile.KeysRequiredEventHandler(
    object sender,
    KeysRequiredEventArgs e
);
```

**Requirements**

**Namespace**: ICSharpCode.SharpZipLib.Zip


**See Also**

SharpZip Compression 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](#).

```csharp
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

```csharp
[C#]
using System;
using System.Text;
using System.IO;


class MainClass
{
    public static void Main(string[] args)
```
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.Length);
            if (size > 0) {
                Console.Write(new ASCIIEncoding().GetString(data, 0, size));
            } else {
                break;
            }
        }
    }
    s.Close();
}
SharpZip Compression Library
### ZipInputStream Members

#### ZipInputStream overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipInputStream Constructor</td>
<td>Creates a new Zip input stream, for reading a zip archive.</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>Returns 1 if there is an entry available Otherwise returns 0.</td>
</tr>
<tr>
<td>CanDecompressEntry</td>
<td>Gets a value indicating if the entry can be decompressed</td>
</tr>
<tr>
<td>CanRead (inherited from InflaterInputStream)</td>
<td>Gets a value indicating whether the current stream supports reading</td>
</tr>
<tr>
<td>CanSeek (inherited from InflaterInputStream)</td>
<td>Gets a value of false indicating seeking is not supported for this stream.</td>
</tr>
<tr>
<td>CanWrite (inherited from InflaterInputStream)</td>
<td>Gets a value of false indicating that this stream is not writeable.</td>
</tr>
<tr>
<td>IsStreamOwner (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>Length (inherited from InflaterInputStream)</td>
<td>A value representing the length of the stream in bytes.</td>
</tr>
<tr>
<td>Password</td>
<td>Optional password used for encryption when non-null</td>
</tr>
<tr>
<td>Position (inherited from InflaterInputStream)</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 Name</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>BodyRead</strong></td>
<td>Reads a block of bytes from the current zip entry.</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>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetNextEntry</strong></td>
<td>Advances to the next entry in the archive</td>
</tr>
<tr>
<td><strong>GetType</strong> (inherited from Object)</td>
<td>Gets the <code>Type</code> 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 <code>String</code> that represents the current <code>Object</code>.</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 Fields**
<table>
<thead>
<tr>
<th>Property</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><em>Input buffer</em> 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></td>
<td>Allocates a <em>WaitHandle</em> 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></td>
<td>Allows an <em>Object</em> to attempt to free resources and perform other cleanup operations before the <em>Object</em> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <em>Object</em>.</td>
</tr>
<tr>
<td><code>StopDecrypting</code></td>
<td>Clear any cryptographic state.</td>
</tr>
</tbody>
</table>

**See Also**

*ZipInputStream Class* | *ICSharpCode.SharpZipLib.Zip Namespace*
SharpZip Compression Library
ZipInputStream Constructor

Creates a new Zip input stream, for reading a zip archive.

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

See Also

SharpZip Compression 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 the entry 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> (inherited from <code>InflaterInputStream</code>)</td>
<td>A value representing the length of the stream in bytes.</td>
</tr>
<tr>
<td>✔️ <strong>Password</strong></td>
<td>Optional password used for encryption when non-null</td>
</tr>
<tr>
<td>✔️ <strong>Position</strong></td>
<td>The current position within the stream. Throws a NotSupportedException when attempting to set the position</td>
</tr>
</tbody>
</table>

### See Also
SharpZip Compression Library
ZipInputStream.Available Property

Returns 1 if there is an entry available Otherwise returns 0.

```
public override int Available {get;}
```

See Also

SharpZip Compression Library
ZipInputStream.CanDecompressEntry Property

Gets a value indicating if the entry 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

SharpZip Compression Library
Optional password used for encryption when non-null

```csharp
public string Password {get; set;}
```

See Also

### ZipInputStream Methods

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>BeginRead</td>
<td>(inherited from Stream) Begins an asynchronous read operation.</td>
</tr>
<tr>
<td>BeginWrite</td>
<td>(inherited from InflaterInputStream) Entry point to begin an asynchronous write. Always throws a NotSupportedException.</td>
</tr>
<tr>
<td>BodyRead</td>
<td>Reads a block of bytes from the current zip entry.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the zip input stream</td>
</tr>
<tr>
<td>CloseEntry</td>
<td>Closes the current zip entry and moves to the next one.</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>Flush</td>
<td>(inherited from InflaterInputStream) Flashes the baseInputStream</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>(inherited from) Serves as a hash function for a</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td>GetNextEntry</td>
<td>Advances to the next entry in the archive.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td>InitializeLifetimeService</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</td>
<td>Sets the position within the current stream Always throws a NotSupportedException.</td>
</tr>
<tr>
<td>SetLength</td>
<td>Set the length of the current stream Always throws a NotSupportedException.</td>
</tr>
<tr>
<td>Skip</td>
<td>Skip specified number of bytes of uncompressed data.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</td>
</tr>
<tr>
<td>Write</td>
<td>Writes a sequence of bytes to stream and advances the current position This method always throws a NotSupportedException.</td>
</tr>
</tbody>
</table>
| WriteByte              | Writes one byte to the current

particular type, suitable for use in hashing algorithms and data structures like a hash table.
InflatedInputStream) stream and advances the current position Always throws a NotSupportedException

<table>
<thead>
<tr>
<th>Protected Instance Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CreateWaitHandle</strong> (inherited from Stream)</td>
</tr>
<tr>
<td><strong>Fill</strong> (inherited from InflaterInputStream)</td>
</tr>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
</tr>
<tr>
<td><strong>StopDecrypting</strong> (inherited from InflaterInputStream)</td>
</tr>
</tbody>
</table>

See Also

[ZipInputStream Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
SharpZip Compression Library
ZipInputStream.BodyRead Method

Reads a block of bytes from the current zip entry.

```csharp
public int BodyRead(
    byte[] b,
    int off,
    int len
);
```

Return Value

The number of bytes read (this may be less than the length requested, even before the end of stream), or 0 on end of stream.

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZipException</td>
<td>The deflated stream is corrupted.</td>
</tr>
<tr>
<td>InvalidOperationException</td>
<td>The stream is not open.</td>
</tr>
<tr>
<td></td>
<td>An i/o error occurred.</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
Closes the zip input stream

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

See Also

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>InvalidOperationException</td>
<td>The stream is closed</td>
</tr>
<tr>
<td>ZipException</td>
<td>The Zip stream ends early</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression 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

SharpZip Compression Library
ZipInputStream.Read Method

Read a block of bytes from the stream.

```csharp
public override int Read(
    byte[] destination,
    int index,
    int count
);
```

Parameters

- **destination**
  - The destination for the bytes.
- **index**
  - 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

SharpZip Compression 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

ZipNameTransform transforms name as per the Zip file 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.

Requirements


See Also

ZipNameTransform Members | ICSSharpCode.SharpZipLib.Zip Namespace
SharpZip Compression Library
# ZipNameTransform Members

## ZipNameTransform overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="ZipNameTransform" /> <strong>ZipNameTransform</strong></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><img src="Image" alt="TrimPrefix" /> <strong>TrimPrefix</strong></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><img src="Image" alt="Equals" /> <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><img src="Image" alt="GetHashCode" /> <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><img src="Image" alt="GetType" /> <strong>GetType</strong> (inherited from <strong>Object</strong>)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><img src="Image" alt="ToString" /> <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><img src="Image" alt="TransformDirectory" /> <strong>TransformDirectory</strong></td>
<td>Transform a directory name according to the Zip file naming conventions.</td>
</tr>
<tr>
<td><img src="Image" alt="TransformFile" /> <strong>TransformFile</strong></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><img src="Image" alt="Finalize" /> <strong>Finalize</strong> (inherited from <strong>Object</strong>)</td>
<td>Allows an <strong>Object</strong> to attempt to</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td>free resources and perform other cleanup operations before the <strong>Object</strong> is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>---</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>
</tr>
</tbody>
</table>

**See Also**

- [ZipNameTransform Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression Library
ZipNameTransform Constructor

Initialize a new instance of **ZipNameTransform**

**Overload List**

Initialize a new instance of **ZipNameTransform**

```csharp
public ZipNameTransform();
```

Initialize a new instance of **ZipNameTransform**

```csharp
public ZipNameTransform(bool);
```

Initialize a new instance of **ZipNameTransform**

```csharp
public ZipNameTransform(bool,string);
```

See Also

[ZipNameTransform Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression Library
ZipNameTransform Constructor ()

Initialize a new instance of ZipNameTransform

```csharp
public ZipNameTransform();
```

Remarks

Relative paths default to true with this constructor.

See Also

ZipNameTransform Class | ICSharpCode.SharpZipLib.Zip Namespace | ZipNameTransform Constructor Overload List
SharpZip Compression Library
Initialize a new instance of `ZipNameTransform`:

```csharp
public ZipNameTransform(
    bool useRelativePaths
);
```

**Parameters**

`useRelativePaths`
- If true relative paths are created, if false absolute paths are created.

**See Also**

- `ZipNameTransform Class` | `ICSharpCode.SharpZipLib.Zip Namespace` | `ZipNameTransform Constructor Overload List`
SharpZip Compression Library
ZipNameTransform Constructor (Boolean, String)

Initialize a new instance of `ZipNameTransform`

```csharp
public ZipNameTransform(
    bool useRelativePaths,
    string trimPrefix
);
```

Parameters

- `useRelativePaths`
  - If true relative paths are created, if false absolute paths are created.

- `trimPrefix`
  - The string to trim from front of paths if found.

See Also

- `ZipNameTransform Class` | `ICSharpCode.SharpZipLib.Zip Namespace` | `ZipNameTransform Constructor Overload List`
SharpZip Compression Library
ZipNameTransform Properties

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

<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>

See Also

[ZipNameTransform Class](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression 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](#) | [ICSharpCode.SharpZipLib.Zip Namespace](#)
SharpZip Compression Library
The methods of the **ZipNameTransform** class are listed below. For a complete list of **ZipNameTransform** class members, see the [ZipNameTransform Members](#) topic.

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong> <em>(inherited from <strong>Object</strong>)</em></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> <em>(inherited from <strong>Object</strong>)</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 <strong>Object</strong>)</em></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>ToString</strong> <em>(inherited from <strong>Object</strong>)</em></td>
<td>Returns a <strong>String</strong> that represents the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>TransformDirectory</strong></td>
<td>Transform a directory name according to the Zip file naming conventions.</td>
</tr>
<tr>
<td><strong>TransformFile</strong></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><strong>Finalize</strong> <em>(inherited from <strong>Object</strong>)</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 <strong>Object</strong>)</em></td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

### See Also
SharpZip Compression Library
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 | ISharpCode.SharpZipLib.Zip Namespace
SharpZip Compression 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` | `ICSharpCode.SharpZipLib.Zip Namespace`
SharpZip Compression Library
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 contain 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.

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;

class MainClass
{
    public static void Main(string[] args)
    {
```
string[] filenames = Directory.GetFiles(args[0]);

ZipOutputStream s = new ZipOutputStream(File.Create(args[1]));
s.SetLevel(5); // 0 - store only to 9 - mean

foreach (string file in filenames) {
    FileStream fs = File.OpenRead(file);

    byte[] buffer = new byte[fs.Length];
    fs.Read(buffer, 0, buffer.Length);

    ZipEntry entry = new ZipEntry(file);
    s.PutNextEntry(entry);
    s.Write(buffer, 0, buffer.Length);
}

s.Finish();
s.Close();

Requirements


See Also

SharpZip Compression 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 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. When null no encryption is performed</td>
</tr>
<tr>
<td>Position (inherited from DeflaterOutputStream)</td>
<td>The current position within the stream. Always throws a</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 <strong>DeflaterOutputStream</strong>)</td>
<td>Asynchronous reads are not supported. A <code>NotSupportedException</code> is always thrown.</td>
</tr>
<tr>
<td><strong>BeginWrite</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Asynchronous writes are not supported. A <code>NotSupportedException</code> is always thrown.</td>
</tr>
<tr>
<td><strong>Close</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Calls <code>finish()</code> and closes the underlying stream when <code>IsStreamOwner</code> 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 <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 <code>Object</code> is equal to the current <code>Object</code>.</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><strong>Flush</strong> (inherited from <strong>Stream</strong>)</td>
<td>Flushes the stream by calling</td>
</tr>
<tr>
<td>Method (inherited from)</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 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 <code>Type</code> 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><strong>Seek</strong> (inherited from)</td>
<td>Sets the current position of this</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>DeflaterOutputStream</code></td>
<td>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 <code>DeflaterOutputStream</code>)</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 <code>Object</code>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</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 <code>DeflaterOutputStream</code>)</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> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>Base stream the deflater depends on.</td>
</tr>
<tr>
<td><strong>buf</strong> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>This buffer is used temporarily to retrieve the bytes from the deflater and write them to the underlying output stream.</td>
</tr>
<tr>
<td><strong>def</strong> (inherited from <code>DeflaterOutputStream</code>)</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><strong>CreateWaitHandle</strong> (inherited from <code>Stream</code>)</td>
<td>Allocates a <code>WaitHandle</code> object.</td>
</tr>
<tr>
<td><strong>Deflate</strong> (inherited from <code>DeflaterOutputStream</code>)</td>
<td>Deflates everything in the input buffers. This will call <code>def deflate()</code></td>
</tr>
</tbody>
</table>
until all bytes from the input buffers are processed.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EncryptBlock (inherited from DeflaterOutputStream)</td>
<td>Encrypt a block of data</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

ZipOutputStream Constructor

Creates a new Zip output stream, writing a zip archive.

```csharp
public ZipOutputStream(
    Stream baseOutputStream
);
```

Parameters

`baseOutputStream`

The output stream to which the archive contents are written.

See Also

SharpZip Compression Library
ZipOutputStream Properties

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>✗ 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>✗ 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</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>Get current length of stream</td>
</tr>
<tr>
<td>✗ Password</td>
<td>Get/set the password used for encryption. When null no encryption is performed</td>
</tr>
<tr>
<td>✗ Position</td>
<td>The current position within the stream. Always throws a NotSupportedException</td>
</tr>
</tbody>
</table>
See Also

SharpZip Compression 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

SharpZip Compression 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><code>BeginRead</code> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous reads are not supported, a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><code>BeginWrite</code> (inherited from DeflaterOutputStream)</td>
<td>Asynchronous writes aren't supported, a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><code>Close</code> (inherited from DeflaterOutputStream)</td>
<td>Calls finish() and closes the underlying stream when IsStreamOwner is true.</td>
</tr>
<tr>
<td><code>CloseEntry</code></td>
<td>Closes the current entry, updating header and footer information as required</td>
</tr>
<tr>
<td><code>CreateObjRef</code> (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><code>EndRead</code> (inherited from Stream)</td>
<td>Waits for the pending asynchronous read to complete.</td>
</tr>
<tr>
<td><code>EndWrite</code> (inherited from Stream)</td>
<td>Ends an asynchronous write operation.</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>Finish</code></td>
<td>Finishes the stream. This will</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Flush</code> (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><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>GetLevel</code></td>
<td>Get the current deflate compression level</td>
</tr>
<tr>
<td><code>GetLifetimeService</code> (inherited from MarshalByRefObject)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><code>GetType</code> (inherited from Object)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><code>InitializeLifetimeService</code> (inherited from MarshalByRefObject)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><code>PutNextEntry</code></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><code>Read</code> (inherited from DeflaterOutputStream)</td>
<td>Read a block of bytes from stream</td>
</tr>
</tbody>
</table>
### Class Members

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>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 <strong>String</strong> that represents the current <strong>Object</strong>.</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 <strong>WaitHandle</strong> object.</td>
</tr>
<tr>
<td><strong>Deflate</strong> (inherited from DeflaterOutputStream)</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> (inherited from DeflaterOutputStream)</td>
<td>Encrypt a block of data</td>
</tr>
<tr>
<td><strong>EncryptByte</strong> (inherited from DeflaterOutputStream)</td>
<td>Encrypt a single byte</td>
</tr>
<tr>
<td>Method</td>
<td>Description</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>InitializePassword</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Initializes encryption keys based on given password.</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>
<tr>
<td>🔄 <strong>UpdateKeys</strong> (inherited from <strong>DeflaterOutputStream</strong>)</td>
<td>Update encryption keys</td>
</tr>
</tbody>
</table>

See Also

**ZipOutputStream Class** | **ICSharpCode.SharpZipLib.Zip Namespace**
SharpZip Compression 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**

SharpZip Compression Library
ZipOutputStream.Finish Method

Finishes the stream. This will write the central directory at the end of the zip file and flush the stream.

```
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 Class
SharpZip Compression Library
ZipOutputStream.GetLevel Method

Get the current deflate compression level

```csharp
public int GetLevel();
```

Return Value

The current compression level

See Also

SharpZip Compression 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

```java
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>IOException</td>
<td>if an I/O error occurred.</td>
</tr>
<tr>
<td>InvalidOperationException</td>
<td>if stream was finished</td>
</tr>
<tr>
<td>ZipException</td>
<td>Too many entries in the Zip file</td>
</tr>
<tr>
<td></td>
<td>Entry name is too long</td>
</tr>
<tr>
<td></td>
<td>Finish has already been called</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
Set the zip file comment.

```csharp
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

SharpZip Compression 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

[ZipOutputStream Class] | [ICSharpCode.SharpZipLib.Zip Namespace]
SharpZip Compression Library
ZipOutputStream.Write Method

Writes the given buffer to the current entry.

```
public override void Write(
    byte[] b,
    int off,
    int len
);
```

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

## 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>DeflaterHuffman.Tree</strong></td>
<td>Not documented</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</td>
</tr>
</tbody>
</table>
use constructor

```java
public Inflater(boolean 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:

- `needsInput()` returns true because the input buffer is empty. You have to provide more input with `setInput()`.
- `needsDictionary()` returns true, you have to provide a preset dictionary with `setDictionary()`.
- `finished()` 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>
SharpZip Compression 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


```
public class Deflater
```

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

SharpZip Compression Library
## Deflater Members

### Deflater overview

### Public Static Fields

<table>
<thead>
<tr>
<th>Enum</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>Method</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.

| **TotalIn** | Gets the number of input bytes processed so far. |
| **TotalOut** | Gets the number of output bytes so far. |

### 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> (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 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.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>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().</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>GetLevel</strong></td>
<td>Get current compression level</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>Resets the deflater. The deflater acts afterwards as if it was just created with the same compression level and strategy as it had before.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SetDictionary</strong></td>
<td>Overloaded. Sets the dictionary which should be used in the deflate process. This call is equivalent to setDictionary(dict, 0, dict.Length).</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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</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](image) (inherited from **Object**) | Creates a shallow copy of the current **Object**. |

**See Also**

- Deflater Class
SharpZip Compression Library
Deflater Constructor

Creates a new deflater with default compression level.

Overload List

Creates a new deflater with default compression level.

public Deflater();

Creates a new deflater with given compression level.

public Deflater(int);

Creates a new deflater with given compression level.

public Deflater(int,bool);

See Also

SharpZip Compression Library
Deflater Constructor ()

Creates a new deflater with default compression level.

public Deflater();

See Also

SharpZip Compression Library
Deflater Constructor (Int32)

Creates a new deflater with given compression level.

```java
public Deflater(
    int lvl
);
```

Parameters

- `lvl` 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 Class](#)
- [Deflater Constructor Overload List](#)
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

# SharpZip Compression Library
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>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>

See Also

Namespace
SharpZip Compression Library
The best and slowest compression level. This tries to find very long and distant string repetitions.

```java
public static int BEST_COMPRESSION;
```

See Also

- [Deflater Class](#)
SharpZip Compression Library
Deflater.BEST_SPEED Field

The worst but fastest compression level.

```java
public static int BEST_SPEED;
```

See Also

SharpZip Compression Library
Deflater.DEFAULT_COMPRESSION Field

The default compression level.

```java
public static int DEFAULT_COMPRESSION;
```

See Also

Deflater.DEFLATED Field

The compression method. This is the only method supported so far. There is no need to use this constant at all.

```csharp
public static int DEFLATED;
```

See Also

- Deflater Class
Deflater.NO_COMPRESSION Field

This level won't compress at all but output uncompressed blocks.

```java
public static int NO_COMPRESSION;
```

See Also

- Deflater Class
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><img src="#" alt="Adler" /></td>
<td>Gets the current adler checksum of the data that was processed so far.</td>
</tr>
<tr>
<td><img src="#" alt="IsFinished" /></td>
<td>Returns true if the stream was finished and no more output bytes are available.</td>
</tr>
<tr>
<td><img src="#" alt="IsNeedingInput" /></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><img src="#" alt="TotalIn" /></td>
<td>Gets the number of input bytes processed so far.</td>
</tr>
<tr>
<td><img src="#" alt="TotalOut" /></td>
<td>Gets the number of output bytes so far.</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
Deflater.Adler Property

Gets the current adler checksum of the data that was processed so far.

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

See Also

SharpZip Compression 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

SharpZip Compression 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

SharpZip Compression Library
Deflater.TotalIn Property

Gets the number of input bytes processed so far.

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

See Also

- Deflater Class
SharpZip Compression Library
Deflater.TotalOut Property

Gets the number of output bytes so far.

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

See Also

**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> <strong>(inherited from</strong> <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 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.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>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().</td>
</tr>
<tr>
<td><strong>GetHashCode</strong> <strong>(inherited from</strong> <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>GetLevel</strong></td>
<td>Get current compression level</td>
</tr>
<tr>
<td><strong>GetType</strong> <strong>(inherited from</strong> <strong>Object)</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td><strong>Reset</strong></td>
<td>Resets the deflater. The deflater acts afterwards as if it was just created 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. This call is equivalent to</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>setDictionary</code></td>
<td>Overloaded. Sets the data which should be compressed next. This should be only called when <code>needsInput</code> indicates that more input is needed. If you call <code>setInput</code> when <code>needsInput()</code> returns false, the previous input that is still pending will be thrown away. The given byte array should not be changed, before <code>needsInput()</code> returns true again. This call is equivalent to <code>setInput(input, 0, input.length)</code></td>
</tr>
<tr>
<td><code>SetInput</code></td>
<td>Sets the data which should be compressed next. This should be only called when <code>needsInput</code> indicates that more input is needed. If you call <code>setInput</code> when <code>needsInput()</code> returns false, the previous input that is still pending will be thrown away. The given byte array should not be changed, before <code>needsInput()</code> returns true again. This call is equivalent to <code>setInput(input, 0, input.length)</code></td>
</tr>
<tr>
<td><code>SetLevel</code></td>
<td>Sets the compression level. There is no guarantee of the exact position of the change, but if you call this when <code>needsInput</code> is true the change of compression level will occur somewhere near before the end of the so far given input.</td>
</tr>
<tr>
<td><code>SetStrategy</code></td>
<td>Sets the compression strategy. Strategy is one of <code>DEFAULT_STRATEGY</code>, <code>HUFFMAN_ONLY</code> and <code>FILTERED</code>. For the exact position where the strategy is changed, the same as for <code>setLevel()</code> applies.</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></td>
<td></td>
</tr>
<tr>
<td><strong>Protected Instance Methods</strong></td>
<td></td>
</tr>
<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>
</tbody>
</table>
MemberwiseClone (inherited from Object) | Creates a shallow copy of the current Object.

See Also

SharpZip Compression Library
Deflate 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**

[Deflater Class] | [ICSharpCode.SharpZipLib.Zip.Compression Namespace]
SharpZip Compression Library
Deflater.Deflate Method (Byte[])  

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 `needsInput()` or `finished()` returns true or length is zero.

See Also

SharpZip Compression Library
Deflater.Deflate Method (Byte[], Int32, Int32)

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>InvalidOperationException</td>
<td>If end() was previously called.</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>If offset and/or length don’t match the array length.</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
**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

SharpZip Compression Library
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

Deflater.GetLevel Method

Get current compression level

```csharp
public int GetLevel();
```

**Return Value**

Returns the current compression level

**See Also**

SharpZip Compression 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

SharpZip Compression Library
Deflater.SetDictionary Method

Sets the dictionary which should be used in the deflate process. This call is equivalent to

```
setDictionary(dict, 0, dict.Length)
```

Overload List

Sets the dictionary which should be used in the deflate process. This call is equivalent to

```
setDictionary(dict, 0, dict.Length)
```

```
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.

```
public void SetDictionary(byte[], int, int);
```

See Also

SharpZip Compression Library
**Deflater.SetDictionary Method (Byte[])**

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[] dict
);
```

**Parameters**

- `dict`  
  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**

SharpZip Compression 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[] dict, int offset, int length);
```

Parameters

* `dict` - The dictionary data
* `offset` - An offset into the dictionary.
* `length` - The length of the dictionary data to use

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

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 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)
```

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 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)
```

```
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.

```
public void SetInput(byte[],int,int);
```

See Also

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)
```

```
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

SharpZip Compression 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 off,
    int len
);
```

**Parameters**

- **input**: the buffer containing the input data.
- **off**: the start of the data.
- **len**: the length of the data.

**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 finished() or ended() or if previous input is still pending.</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression 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 lvl
);
```

Parameters

- **lvl**
  the new compression level.

See Also

- [Deflater Class](#)
SharpZip Compression 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.

```java
public void SetStrategy(
    DeflateStrategy strategy
);
```

Parameters

- `strategy`
  - The new compression strategy.

See Also

SharpZip Compression 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**

**Namespace:** ISharpCode.SharpZipLib.Zip.Compression

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

**See Also**

SharpZip Compression Library
# DeflaterConstants Members

**DeflaterConstants overview**

## Public Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$COMPR_FUNC</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$DEBUGGING</strong></td>
<td>Set to true to enable debugging</td>
</tr>
<tr>
<td><strong>$DEFAULT_MEM_LEVEL</strong></td>
<td>Sets internal buffer sizes for Huffman encoding</td>
</tr>
<tr>
<td><strong>$DEFLATE_FAST</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$DEFLATE_SLOW</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$DEFLATE_STORED</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$DYN_TREES</strong></td>
<td>Identifies dynamic tree in Zip file</td>
</tr>
<tr>
<td><strong>$GOOD_LENGTH</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$HASH_BITS</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$HASH_MASK</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$HASH_SHIFT</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$HASH_SIZE</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$MAX_BLOCK_SIZE</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$MAX_CHAIN</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>$MAX_DIST</strong></td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>Constant</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>$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>$DeflaterConstants$</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>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>
<tr>
<td>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression Library
DeflaterConstants Constructor

Initializes a new instance of the DeflaterConstants class.

```java
public DeflaterConstants();
```

See Also

SharpZip Compression 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>COMPR_FUNC</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>DEBUGGING</td>
<td>Set to true to enable debugging</td>
</tr>
<tr>
<td>DEFAULT_MEM_LEVEL</td>
<td>Sets internal buffer sizes for Huffman encoding</td>
</tr>
<tr>
<td>DEFLATE_FAST</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>DEFLATE_SLOW</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>DEFLATE_STORED</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>DYN_TREES</td>
<td>Identifies dynamic tree in Zip file</td>
</tr>
<tr>
<td>GOOD_LENGTH</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>HASH_BITS</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>HASH_MASK</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>HASH_SHIFT</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>HASH_SIZE</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>MAX_BLOCK_SIZE</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td>MAXCHAIN</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> MAX_DIST</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> MAX_LAZY</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> MAX_MATCH</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> MAX_WBITS</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> MIN_LOOKAHEAD</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> MIN_MATCH</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> NICE_LENGTH</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> PENDING_BUF_SIZE</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> PRESET_DICT</td>
<td>Header flag indicating a preset dictionary for deflation</td>
</tr>
<tr>
<td><strong>s</strong> STATIC_TREES</td>
<td>Identifies static tree in Zip file</td>
</tr>
<tr>
<td><strong>s</strong> STORED_BLOCK</td>
<td>Written to Zip file to identify a stored block</td>
</tr>
<tr>
<td><strong>s</strong> WMASK</td>
<td>Internal compression engine constant</td>
</tr>
<tr>
<td><strong>s</strong> WSIZE</td>
<td>Internal compression engine constant</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression Library
DeflaterConstants.COMPR_FUNC Field

Internal compression engine constant

```java
public static int[] COMPR_FUNC;
```

See Also

SharpZip Compression Library
**DeflaterConstants.DEBUGGING Field**

Set to true to enable debugging

```csharp
public const bool DEBUGGING = False;
```

See Also

SharpZip Compression 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
SharpZip Compression Library
DeflaterConstants.DEFLATE_FAST Field

Internal compression engine constant

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

See Also

- [DeflaterConstants Class](#)
SharpZip Compression Library
**DeflaterConstants.DEFLATE_SLOW Field**

Internal compression engine constant

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

See Also

SharpZip Compression Library
**DeflaterConstants.DEFLATE_STORED Field**

Internal compression engine constant

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

See Also

- [DeflaterConstants Class](#)
SharpZip Compression Library
DeflaterConstants.DYN_TREES Field

Identifies dynamic tree in Zip file

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

See Also

DeflaterConstants.GOOG_LENGTH Field

Internal compression engine constant

```java
public static int[] GOOD_LENGTH;
```

See Also

SharpZip Compression Library
DeflaterConstants.HASH_BITS Field

Internal compression engine constant

```csharp
public const int HASH_BITS = 15;
```

See Also

SharpZip Compression Library
**DeflaterConstants.HASH_MASK Field**

Internal compression engine constant

```
public const int HASH_MASK = 32767;
```

See Also

SharpZip Compression Library
DeflaterConstants.HASH_SHIFT Field

Internal compression engine constant

```csharp
public const int HASH_SHIFT = 5;
```

See Also

- DeflaterConstants Class
DeflaterConstants.HASH_SIZE Field

Internal compression engine constant

```csharp
public const int HASH_SIZE = 32768;
```

See Also

[DeflaterConstants Class](#)
SharpZip Compression Library
DeflaterConstants.MAX_BLOCK_SIZE Field

Internal compression engine constant

```java
public static int MAX_BLOCK_SIZE;
```

See Also

[DeflaterConstants Class](#)
DeflaterConstants.MAX_CHAIN Field

Internal compression engine constant

```java
public static int[] MAX_CHAIN;
```

See Also

- DeflaterConstants Class
SharpZip Compression Library
DeflaterConstants.MAX_DIST Field

Internal compression engine constant

public const int MAX_DIST = 32506;

See Also

DeflaterConstants Class
DeflaterConstants.MAX_LAZY Field

Internal compression engine constant

```java
public static int[] MAX_LAZY;
```

See Also

- DeflaterConstants Class
SharpZip Compression Library
DeflaterConstants.MAX_MATCH Field

Internal compression engine constant

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

See Also

- DeflaterConstants Class
SharpZip Compression Library
DeflaterConstants.MAX_WBITS Field

Internal compression engine constant

```csharp
public const int MAX_WBITS = 15;
```

See Also

- [DeflaterConstants Class](#)
SharpZip Compression Library
**DeflaterConstants.MIN_LOOKAHEAD Field**

Internal compression engine constant

```
public const int MIN_LOOKAHEAD = 262;
```

See Also

SharpZip Compression Library
**DeflaterConstants.MIN_MATCH Field**

Internal compression engine constant

```java
public const int MIN_MATCH = 3;
```

See Also

- [DeflaterConstants Class](#)
SharpZip Compression Library
DeflaterConstants.NICE_LENGTH Field

Internal compression engine constant

```java
public static int[] NICE_LENGTH;
```

See Also

- DeflaterConstants Class
SharpZip Compression Library
**DeflaterConstants.PENDING_BUF_SIZE Field**

Internal compression engine constant

```csharp
public const int PENDING_BUF_SIZE = 65536;
```

See Also

SharpZip Compression Library
DeflaterConstants.PRESET_DICT Field

Header flag indicating a preset dictionary for deflation

```
public const int PRESET_DICT = 32;
```

See Also

[DeflaterConstants Class](#) |
SharpZip Compression Library
DeflaterConstants.STATIC_TREES Field

Identifies static tree in Zip file

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

See Also

SharpZip Compression Library
DeflaterConstants.STORED_BLOCK Field

Written to Zip file to identify a stored block

```
public const int STORED_BLOCK = 0;
```

See Also

- [DeflaterConstants Class](#)
SharpZip Compression Library
### DeflaterConstants.WMASK Field

Internal compression engine constant

```csharp
public const int WMASK = 32767;
```

See Also

- [DeflaterConstants Class](#)
SharpZip Compression Library
DeflaterConstants.WSIZE Field

Internal compression engine constant

```public const int WSIZE = 32768;```

See Also

SharpZip Compression Library
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.

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

SharpZip Compression Library
## DeflaterEngine Members

### DeflaterEngine overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DeflaterEngine Constructor</strong></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><strong>Adler</strong></td>
<td>Get current value of Adler checksum</td>
</tr>
<tr>
<td><strong>Strategy</strong></td>
<td>Get/set the deflate strategy</td>
</tr>
<tr>
<td><strong>TotalIn</strong></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><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>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 NeedsInput() 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 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**

- [DeflaterEngine Class](#)
SharpZip Compression Library
**DeflaterEngine Constructor**

Construct instance with pending buffer

```csharp
public DeflaterEngine(
    DeflaterPending pending
);
```

**Parameters**

*pending*

Pending buffer to use

**See Also**

SharpZip Compression 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

SharpZip Compression Library
DeflaterEngine.Adler Property

Get current value of Adler checksum

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

See Also

SharpZip Compression Library
Get/set the **deflate strategy**

```csharp
public DeflateStrategy Strategy {get; set;}
```

**See Also**

**DeflaterEngine.TotalIn Property**

Total data processed

```
public int TotalIn {get;}
```

**See Also**

**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</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Deflate</code></td>
<td>Deflate drives actual compression of data</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>FillWindow</code></td>
<td>Fill the window</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>NeedsInput</code></td>
<td>Return true if input is needed via <code>SetInput</code></td>
</tr>
<tr>
<td><code>Reset</code></td>
<td>Reset internal state</td>
</tr>
<tr>
<td><code>ResetAdler</code></td>
<td>Reset Adler checksum</td>
</tr>
<tr>
<td><code>SetDictionary</code></td>
<td>Set compression dictionary</td>
</tr>
<tr>
<td><code>SetInput</code></td>
<td>Sets input data to be deflated. Should only be called when <code>NeedsInput()</code> returns true</td>
</tr>
<tr>
<td><code>SetLevel</code></td>
<td>Set the deflate level (0-9)</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>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>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
Deflate drives actual compression of data

```csharp
public bool Deflate(bool flush, bool finish);
```

See Also

DeflaterEngine.FillWindow Method

Fill the window

```csharp
public void FillWindow();
```

See Also

SharpZip Compression Library
DeflaterEngine.NeedsInput Method

Return true if input is needed via SetInput

```csharp
public bool NeedsInput();
```

See Also

DeflaterEngine.Reset Method

Reset internal state

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

See Also


SharpZip Compression Library
DeflaterEngine.ResetAdler Method

Reset Adler checksum

public void ResetAdler();

See Also

SharpZip Compression Library
Set compression dictionary

```csharp
public void SetDictionary(
    byte[] buffer,
    int offset,
    int length
);
```

See Also

SharpZip Compression Library
DeflaterEngine.SetInput Method

Sets input data to be deflated. Should only be called when

NeedsInput()

returns true

```csharp
public void SetInput(
    byte[] buf,
    int off,
    int len
);
```

Parameters

- **buf**
  The buffer containing input data.

- **off**
  The index of the first byte of data.

- **len**
  The number of bytes of data to use as input.

See Also

- [DeflaterEngine Class](#)
SharpZip Compression Library
DeflaterEngine.SetLevel Method

Set the deflate level (0-9)

```csharp
public void SetLevel(int lvl);
```

See Also

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


See Also

SharpZip Compression Library
**DeflaterHuffman Members**

**DeflaterHuffman overview**

**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 Constructors**

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflaterHuffman Constructor</td>
<td>Construct instance with pending buffer</td>
</tr>
</tbody>
</table>

**Public Instance Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pending</td>
<td>Pending buffer to use</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>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<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 <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><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 <a href="#">Object</a> 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

SharpZip Compression Library
DeflaterHuffman Constructor

Construct instance with pending buffer

```java
public DeflaterHuffman(
    DeflaterPending pending
);
```

Parameters

`pending`
Pending buffer to use

See Also

**DeflaterHuffman Fields**

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**

Pending buffer to use

```csharp
public DeflaterPending pending;
```

See Also

- [DeflaterHuffman Class](#)
SharpZip Compression 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>
</tbody>
</table>
**ToString** (inherited from **Object**)

Returns a **String** that represents the current **Object**.

### 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

DeflaterHuffman.BitReverse Method

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

SharpZip Compression Library
DeflaterHuffman.CompressBlock Method

Compress current buffer writing data to pending buffer

```
public void CompressBlock();
```

See Also

SharpZip Compression 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

DeflaterHuffman.FlushStoredBlock Method

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

SharpZip Compression Library
DeflaterHuffman.IsFull Method

Get value indicating if internal buffer is full

public bool IsFull();

Return Value

ture if buffer is full

See Also

SharpZip Compression Library
DeflaterHuffman.Reset Method

Reset internal state

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

See Also

SharpZip Compression Library
Write all trees to pending buffer

```csharp
public void SendAllTrees(
    int blTreeCodes
);
```

See Also

DeflaterHuffman.TallyDist Method

Add distance code and length to literal and distance trees

```csharp
public bool TallyDist(
    int dist,
    int len
);
```

Parameters

- `dist`  
  Distance code

- `len`  
  Length

Return Value

Value indicating if internal buffer is full

See Also

DeflaterHuffman.TallyLit Method

Add literal to buffer

```csharp
public bool TallyLit(int lit);
```

Parameters

- `lit`

Return Value

Value indicating internal buffer is full

See Also

SharpZip Compression Library
DeflaterHuffman.Tree Class

Not documented

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

System.Object


public class DeflaterHuffman.Tree

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

SharpZip Compression Library
## DeflaterHuffman.Tree Members

### DeflaterHuffman.Tree overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflaterHuffman.Tree Constructor</td>
<td>Not documented</td>
</tr>
</tbody>
</table>

#### Public Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>freqs</td>
<td>Not documented</td>
</tr>
<tr>
<td>length</td>
<td>Not documented</td>
</tr>
<tr>
<td>minNumCodes</td>
<td>Not documented</td>
</tr>
<tr>
<td>numCodes</td>
<td>Not documented</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BuildCodes</td>
<td>Build dynamic codes and lengths</td>
</tr>
<tr>
<td>BuildTree</td>
<td>Not documented</td>
</tr>
<tr>
<td>CalcBLFreq</td>
<td>Not documented</td>
</tr>
<tr>
<td>CheckEmpty</td>
<td>Check that at least one frequency is non-zero</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetEncodedLength</td>
<td>Get encoded length</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 internal state of the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SetStaticCodes</td>
<td>Set static codes and length</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>WriteSymbol</td>
<td>Not documented</td>
</tr>
<tr>
<td>WriteTree</td>
<td>Write tree values</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

- DeflaterHuffman.Tree Class
SharpZip Compression Library
DeflaterHuffman.Tree Constructor

Not documented

```csharp
public DeflaterHuffman.Tree(
    DeflaterHuffman dh,
    int elems,
    int minCodes,
    int maxLength
);
```

See Also

[DeflaterHuffman.Tree Class](#)
SharpZip Compression Library
DeflaterHuffman.Tree Fields

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

Public Instance Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>freqs</td>
<td>Not documented</td>
</tr>
<tr>
<td>length</td>
<td>Not documented</td>
</tr>
<tr>
<td>minNumCodes</td>
<td>Not documented</td>
</tr>
<tr>
<td>numCodes</td>
<td>Not documented</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
DeflaterHuffman.Tree.freqs Field

Not documented

```java
public short[] freqs;
```

See Also

- [DeflaterHuffman.Tree Class](#)
SharpZip Compression Library
DeflaterHuffman.Tree.length Field

Not documented

```java
public byte[] length;
```

See Also

SharpZip Compression Library
DeflaterHuffman.Tree.minNumCodes Field

Not documented

```java
public int minNumCodes;
```

See Also

- DeflaterHuffman.Tree Class
SharpZip Compression Library
DeflaterHuffman.Tree.numCodes Field

Not documented

```java
public int numCodes;
```

See Also

SharpZip Compression Library
The methods of the `DeflaterHuffman.Tree` class are listed below. For a complete list of `DeflaterHuffman.Tree` class members, see the `DeflaterHuffman.Tree Members` topic.

## Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BuildCodes</code></td>
<td>Build dynamic codes and lengths</td>
</tr>
<tr>
<td><code>BuildTree</code></td>
<td>Not documented</td>
</tr>
<tr>
<td><code>CalcBLFreq</code></td>
<td>Not documented</td>
</tr>
<tr>
<td><code>CheckEmpty</code></td>
<td>Check that at least one frequency is non-zero</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>GetEncodedLength</code></td>
<td>Get encoded length</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>Reset</code></td>
<td>Resets the internal state of the tree</td>
</tr>
<tr>
<td><code>SetStaticCodes</code></td>
<td>Set static codes and length</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>
<tr>
<td><code>WriteSymbol</code></td>
<td>Not documented</td>
</tr>
<tr>
<td><code>WriteTree</code></td>
<td>Write tree values</td>
</tr>
</tbody>
</table>

## Protected Instance Methods
<table>
<thead>
<tr>
<th><strong>Finalize</strong> (inherited from <strong>Object</strong>)</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 <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

- *DeflaterHuffman.Tree Class* | *ICSharpCode.SharpZipLib.Zip.Compression Namespace*
SharpZip Compression Library
DeflaterHuffman.Tree.BuildCodes Method

Build dynamic codes and lengths

```csharp
public void BuildCodes();
```

See Also

- DeflaterHuffman.Tree Class
DeflaterHuffman.Tree.BuildTree Method

Not documented

```csharp
public void BuildTree();
```

See Also

SharpZip Compression Library
DeflaterHuffman.Tree.CalcBLFreq Method

Not documented

```csharp
public void CalcBLFreq(
    Tree blTree
);
```

See Also

SharpZip Compression Library
Check that at least one frequency is non-zero

```csharp
public void CheckEmpty();
```

### Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SharpZipBaseException</td>
<td>No frequencies are non-zero</td>
</tr>
</tbody>
</table>

### See Also

- [DeflaterHuffman.Tree Class](#)
SharpZip Compression Library
**DeflaterHuffman.Tree.GetEncodedLength Method**

Get encoded length

```csharp
public int GetEncodedLength();
```

**Return Value**

Encoded length, the sum of frequencies * lengths

**See Also**

SharpZip Compression Library
DeflaterHuffman.Tree.Reset Method

Resets the internal state of the tree

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

See Also

SharpZip Compression Library
DeflaterHuffman.Tree.SetStaticCodes Method

Set static codes and length

```csharp
public void SetStaticCodes(
    short[] stCodes,
    byte[] stLength
);
```

Parameters

- **stCodes**
  - new codes

- **stLength**
  - length for new codes

See Also

- DeflaterHuffman.Tree Class
SharpZip Compression Library
DeflaterHuffman.Tree.WriteSymbol Method

Not documented

```csharp
public void WriteSymbol(
    int code
);
```

See Also

SharpZip Compression Library
DeflaterHuffman.Tree.WriteTree Method

Write tree values

```csharp
public void WriteTree(
    Tree blTree
);
```

Parameters

- `blTree`  
  Tree to write

See Also

SharpZip Compression 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

**Namespace:** ISharpCode.SharpZipLib.Zip.Compression

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

See Also

DeflaterPending Members |
SharpZip Compression Library
# DeflaterPending Members

## DeflaterPending overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflaterPending Constructor</td>
</tr>
<tr>
<td>Construct instance with default buffer size</td>
</tr>
</tbody>
</table>

### Public Instance Properties

<table>
<thead>
<tr>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitCount (inherited from PendingBuffer)</td>
</tr>
<tr>
<td>The number of bits written to the buffer</td>
</tr>
<tr>
<td>IsFlushed (inherited from PendingBuffer)</td>
</tr>
<tr>
<td>Indicates if buffer has been flushed</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlignToByte (inherited from PendingBuffer)</td>
</tr>
<tr>
<td>Align internal buffer on a byte boundary</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
</tr>
<tr>
<td>Determines whether the specified <a href="#">Object</a> is equal to the current <a href="#">Object</a>.</td>
</tr>
<tr>
<td>Flush (inherited from PendingBuffer)</td>
</tr>
<tr>
<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>GetHashCode (inherited from Object)</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>Gets the <a href="#">Type</a> of the current instance.</td>
</tr>
<tr>
<td>Reset (inherited from PendingBuffer)</td>
</tr>
<tr>
<td>Clear internal state/buffers</td>
</tr>
<tr>
<td>ToByteArray (inherited from PendingBuffer)</td>
</tr>
<tr>
<td>Convert internal buffer to byte array. Buffer is empty on</td>
</tr>
<tr>
<td>Protected Instance Fields</td>
</tr>
<tr>
<td>---------------------------</td>
</tr>
<tr>
<td><strong>buf</strong> (inherited from PendingBuffer)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protected Instance Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong> (inherited from Object)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
</tr>
</tbody>
</table>

**See Also**

DeflaterPending Constructor

Construct instance with default buffer size

public DeflaterPending();

See Also

**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](#)

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

### See Also

SharpZip Compression Library
Inflater Class

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

```
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:

- needsInput() returns true because the input buffer is empty. You have to provide more input with

```
setInput()
```

. NOTE: needsInput() also returns true when, the stream is finished.

- needsDictionary() returns true, you have to provide a preset dictionary with

```
setDictionary()
```

. 

- finished() 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

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

System.Object


```
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


See Also

SharpZip Compression Library
Inflater Members

Inflater overview

Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inflater</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>Adler</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>IsFinished</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>IsNeedingDictionary</td>
<td>Returns true, if a preset dictionary is needed to inflate the input.</td>
</tr>
<tr>
<td>IsNeedingInput</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>RemainingInput</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>
</tbody>
</table>
| **TotalOut** | Gets the total number of output bytes returned by `inflate()`.

## 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>Inflate</strong></td>
<td>Overloaded. Inflates the compressed stream to the output buffer. If this returns 0, you should check, whether <code>needsDictionary()</code>, <code>needsInput()</code> or <code>finished()</code> 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 <code>needsDictionary()</code> returns true and it should set the same dictionary, that was used for deflating. The <code>getAdler()</code> function returns the checksum</td>
</tr>
</tbody>
</table>
of the dictionary needed.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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 <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

SharpZip Compression 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);
SharpZip Compression Library
Inflater Constructor ()

Creates a new inflater or RFC1951 decompressor RFC1950/Zlib headers and footers will be expected in the input data

public Inflater();

See Also
SharpZip Compression Library
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

SharpZip Compression Library
The properties of the **Inflater** class are listed below. For a complete list of **Inflater** class members, see the **Inflater Members** topic.

## 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>. <strong>NOTE:</strong> 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

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

SharpZip Compression 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

- Inflater Class
SharpZip Compression 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
**SharpZip Compression 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

SharpZip Compression Library
Inflater.RemainingInput Property

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

SharpZip Compression Library
**Inflater.TotalIn Property**

Gets the total number of processed compressed input bytes.

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

**See Also**

SharpZip Compression Library
Inflater.TotalOut Property

Gets the total number of output bytes returned by inflate().

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

See Also

- Inflater Class
SharpZip Compression Library
**Inflater Methods**

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></td>
<td>Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</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>Inflate</strong></td>
<td>Overloaded. 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.</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><strong>SetInput</strong></td>
<td>Overloaded. Sets the input. This should only be called, if needsInput() returns true.</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------------</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><strong>Finalize</strong> (inherited from Object)</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<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.Inflate Method**

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.

**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.

```java
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.

```java
public int Inflate(byte[], int, int);
```

**See Also**

- [Inflater Class](#)
SharpZip Compression Library
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[] buf);
```

**Parameters**

- `buf`  
  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><code>ArgumentOutOfRangeException</code></td>
<td>if buf has length 0.</td>
</tr>
<tr>
<td><code>FormatException</code></td>
<td>if deflated stream is invalid.</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression Library
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[] buf,
    int offset,
    int len
);
```

**Parameters**

- **buf**
  the output buffer.

- **offset**
  the offset into buffer where the output should start.

- **len**
  the maximum length of the 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 len is &lt;= 0.</td>
</tr>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>if the offset and/or len 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
SharpZip Compression 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

SharpZip Compression 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

Namespace
SharpZip Compression Library
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**

SharpZip Compression 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 offset,
    int len
);
```

**Parameters**

- `buffer`  
  The dictionary.
- `offset`  
  The offset into buffer where the dictionary starts.
- `len`  
  The length of the dictionary.

**Exceptions**

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InvalidOperationException</code></td>
<td>No dictionary is needed.</td>
</tr>
<tr>
<td><code>SharpZipBaseException</code></td>
<td>The adler checksum for the buffer is invalid.</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression 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**

Inflater.SetInput Method (Byte[])  

Sets the input. This should only be called, if needsInput() returns true.

```csharp
public void SetInput(byte[] buf);
```

Parameters

- `buf`: the input.

See Also

SharpZip Compression 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 offset,
    int length
);
```

Parameters

- `buffer`  The source of input data
- `offset`  The offset into buffer where the input starts.
- `length`  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 off and/or len are wrong.</td>
</tr>
</tbody>
</table>

See Also

- Inflater Class
- Inflater.SetInput Overload List
SharpZip Compression Library
**InflaterHuffmanTree Class**

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**

- **Namespace:** ISharpCode.SharpZipLib.Zip.Compression
- **Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

**See Also**

- [InflaterHuffmanTree Members](#) | [ICSharpCode.SharpZipLib.Zip.Compression Namespace](#)
SharpZip Compression Library
## InflaterHuffmanTree Members

### InflaterHuffmanTree overview

#### Public Static Fields

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$defDistTree</td>
<td>Distance tree</td>
</tr>
<tr>
<td></td>
<td>$defLitLenTree</td>
<td>Literal length tree</td>
</tr>
</tbody>
</table>

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InflaterHuffmanTree Constructor</td>
<td>Constructs a Huffman tree from the array of code lengths.</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>$GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>$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 | ICSharpCode.SharpZipLib.Zip.Compression Namespace
SharpZip Compression 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.Zip.Compression Namespace
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**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>defDistTree</code></td>
<td>Distance tree</td>
</tr>
<tr>
<td><code>defLitLenTree</code></td>
<td>Literal length tree</td>
</tr>
</tbody>
</table>

**See Also**

[InflaterHuffmanTree Class](#) | [ICSharpCode.SharpZipLib.Zip.Compression Namespace](#)
SharpZip Compression Library
InflaterHuffmanTree.defDistTree Field

Distance tree

```java
public static InflaterHuffmanTree defDistTree
```

See Also

- InflaterHuffmanTree Class
SharpZip Compression Library
Literal length tree

public static InflaterHuffmanTree defLitLenTree

See Also

InflaterHuffmanTree Class |
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><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>GetSymbol</strong></td>
<td>Reads the next symbol from input. The symbol is encoded using the huffman tree.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>(inherited from Object) Gets the Type of the current instance.</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>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

InflaterHuffmanTree Class | ISharpCode.SharpZipLib.Zip.Compression Namespace
SharpZip Compression Library
InflatorHuffmanTree.GetSymbol Method

Reads the next symbol from input. The symbol is encoded using the huffman tree.

```csharp
public int GetSymbol(StreamManipulator input);
```

Parameters

- `input` input the input source.

Return Value

the next symbol, or -1 if not enough input is available.

See Also

InflatorHuffmanTree Class | ISharpCode.SharpZipLib.Zip.Compression Namespace
SharpZip Compression 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


See Also

SharpZip Compression Library
### PendingBuffer Overview

**Public Instance Constructors**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PendingBuffer</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>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 (inherited from Object)</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 (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>Clear internal state/buffers</td>
</tr>
<tr>
<td>ToByteArray</td>
<td>Convert internal buffer to byte array. Buffer is empty on</td>
</tr>
</tbody>
</table>
completion

<table>
<thead>
<tr>
<th>method</th>
<th>description</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td><code>WriteBits</code></td>
<td>Write bits to internal buffer</td>
</tr>
<tr>
<td><code>WriteBlock</code></td>
<td>Write a block of data to buffer</td>
</tr>
<tr>
<td><code>WriteByte</code></td>
<td>write a byte to buffer</td>
</tr>
<tr>
<td><code>WriteInt</code></td>
<td>write an integer LSB first</td>
</tr>
<tr>
<td><code>WriteShort</code></td>
<td>Write a short value to buffer LS first</td>
</tr>
<tr>
<td><code>WriteShortMSB</code></td>
<td>Write a short value to internal buffer most significant byte first</td>
</tr>
</tbody>
</table>

Protected Instance Fields

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>buf</code></td>
<td>Internal work buffer</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

SharpZip Compression 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

SharpZip Compression Library
PendingBuffer Constructor ()

construct instance using default buffer size of 4096

```
public PendingBuffer();
```

See Also

SharpZip Compression Library
PendingBuffer Constructor (Int32)

construct instance using specified buffer size

```csharp
public PendingBuffer(
    int bufsize
);
```

Parameters

`bufsize`

size to use for internal buffer

See Also

SharpZip Compression Library
PendingBuffer Fields

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

Protected Instance Fields

| buf | Internal work buffer |

See Also

SharpZip Compression Library
**PendingBuffer.buf Field**

Internal work buffer

```csharp
protected byte[] buf;
```

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**

SharpZip Compression Library
PendingBuffer.BitCount Property

The number of bits written to the buffer

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

See Also

SharpZip Compression Library
PendingBuffer.IsFlushed Property

Indicates if buffer has been flushed

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

See Also

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 too 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><strong>WriteShort</strong></td>
<td>Write a short value to buffer LSB first</td>
</tr>
<tr>
<td><strong>WriteShortMSB</strong></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></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

SharpZip Compression Library
PendingBuffer.AlignToByte Method

Align internal buffer on a byte boundary

```csharp
public void AlignToByte();
```

See Also

SharpZip Compression 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` length the maximum number of bytes to store;

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IndexOutOfBoundsException</td>
<td>IndexOutOfBoundsException if offset or length are invalid.</td>
</tr>
</tbody>
</table>

See Also

- PendingBuffer Class
SharpZip Compression Library
PendingBuffer.Reset Method

Clear internal state/buffers

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

See Also

SharpZip Compression Library
PendingBuffer.ToByteArray Method

Convert internal buffer to byte array. Buffer is empty on completion

```csharp
public byte[] ToByteArray();
```

Return Value

converted buffer contents contents

See Also

PendingBuffer.WriteBits Method

Write bits to internal buffer

```
public void WriteBits(
    int b,
    int count
);
```

Parameters

- **b**
  - source of bits
- **count**
  - number of bits to write

See Also

PendingBuffer.WriteBlock Method

Write a block of data to buffer

```csharp
public void WriteBlock(
    byte[] block,
    int offset,
    int len
);
```

Parameters

- **block**
  - data to write

- **offset**
  - offset of first byte to write

- **len**
  - number of bytes to write

See Also

- [PendingBuffer Class](#)
SharpZip Compression Library
PendingBuffer.WriteByte Method

write a byte to buffer

```csharp
public void WriteByte(
    int b
);
```

Parameters

- **b**
  - value to write

See Also

SharpZip Compression Library
PendingBuffer.WriteInt Method

write an integer LSB first

```csharp
public void WriteInt(
    int s
);
```

Parameters

s
value to write

See Also

SharpZip Compression Library
PendingBuffer.WriteShort Method

Write a short value to buffer LSB first

```csharp
public void WriteShort(
    int s
);
```

Parameters

- **s**: value to write

See Also

SharpZip Compression Library
PendingBuffer.WriteShortMSB Method

Write a short value to internal buffer most significant byte first

```csharp
public void WriteShortMSB(
    int s
);
```

Parameters

- `s` value to write

See Also

SharpZip Compression Library

Namespace hierarchy

Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflaterOutputStream</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>InflaterInputBuffer</td>
<td>An input buffer customised for use by InflaterInputStream</td>
</tr>
<tr>
<td>InflaterInputStream</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 GZipInputStream. Author of the original java version: John Leuner.</td>
</tr>
<tr>
<td>OutputWindow</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>StreamManipulator</td>
<td>This class allows us to retrieve a specified number of bits from the stream.</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
SharpZip Compression Library
DeflaterOutputStream Class

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
SharpZip Compression Library
# DeflaterOutputStream Members

## DeflaterOutputStream overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeflaterOutputStream</td>
<td>Overloaded. Initializes a new instance of the DeflaterOutputStream 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 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. When null no encryption is performed</td>
</tr>
<tr>
<td>Position</td>
<td>The current position within the stream. Always throws a</td>
</tr>
<tr>
<td></td>
<td>NotSupportedException NotSupportedException</td>
</tr>
</tbody>
</table>

### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginRead</td>
<td>Asynchronous reads are not supported a NotSupportedException is always thrown</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</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 finish() and closes the underlying stream when IsStreamOwner is true.</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 by calling finish() on the deflater.</td>
</tr>
<tr>
<td><strong>Flush</strong></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>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 object.</td>
</tr>
<tr>
<td><strong>Object</strong></td>
<td>instance.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> <em>(inherited from MarshalByRefObject)</em></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> <em>(inherited from Object)</em></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 Fields

| **baseOutputStream** | Base stream the deflater depends on. |
| **buf** | This buffer is used temporarily to retrieve the bytes from the deflater and write them to the underlying output stream. |
| **def** | The deflater which is used to deflate the stream. |

### Protected Instance Methods

<p>| <strong>CreateWaitHandle</strong> <em>(inherited)</em> | Allocates a <strong>WaitHandle</strong> object. |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deflate</td>
<td>Deflates everything in the input buffers. This will call <code>def.deflate()</code></td>
</tr>
<tr>
<td>EncryptBlock</td>
<td>Encrypt a block of data</td>
</tr>
<tr>
<td>EncryptByte</td>
<td>Encrypt a single byte</td>
</tr>
<tr>
<td>Finalize (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>InitializePassword</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 <strong>Object</strong>.</td>
</tr>
<tr>
<td>UpdateKeys</td>
<td>Update encryption keys</td>
</tr>
</tbody>
</table>

See Also
- DeflaterOutputStream Class
SharpZip Compression 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.

public DeflaterOutputStream(Stream);

Creates a new DeflaterOutputStream with the given Deflater and default buffer size.

public DeflaterOutputStream(Stream,Deflater);

Creates a new DeflaterOutputStream with the given Deflater and buffer size.

public DeflaterOutputStream(Stream,Deflater,int);

See Also

SharpZip Compression Library
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

[DeflaterOutputStream Class](#)
[DeflaterOutputStream Constructor Overload List](#)
SharpZip Compression Library
DeflaterOutputStream Constructor (Stream, Deflater)

Creates a new DeflaterOutputStream with the given Deflater and default buffer size.

```java
public DeflaterOutputStream(
    Stream baseOutputStream,
    Deflater defl
);
```

Parameters

- **baseOutputStream**
  - the output stream where deflated output should be written.

- **defl**
  - the underlying deflater.

See Also

- [DeflaterOutputStream Class](#)
- [DeflaterOutputStream Constructor Overload List](#)
SharpZip Compression Library
DeflaterOutputStream Constructor (Stream, Deflater, Int32)

Creates a new DeflaterOutputStream with the given Deflater and buffer size.

```csharp
public DeflaterOutputStream(
    Stream baseOutputStream,
    Deflater deflater,
    int bufsize
);
```

Parameters

- `baseOutputStream`
  The output stream where deflated output is written.

- `deflater`
  The underlying deflater to use

- `bufsize`
  The buffer size to use when deflating

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgumentOutOfRangeException</td>
<td>bufsize is less than or equal to zero.</td>
</tr>
<tr>
<td>ArgumentException</td>
<td>baseOutputStream does not support writing</td>
</tr>
<tr>
<td>ArgumentNullException</td>
<td>deflater instance is null</td>
</tr>
</tbody>
</table>

See Also

- DeflaterOutputStream Class
- DeflaterOutputStream Constructor Overload List
SharpZip Compression 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><code>baseOutputStream</code></td>
<td>Base stream the deflater depends on.</td>
</tr>
<tr>
<td><code>buf</code></td>
<td>This buffer is used temporarily to retrieve the bytes from the deflater and write them to the underlying output stream.</td>
</tr>
<tr>
<td><code>def</code></td>
<td>The deflater which is used to deflate the stream.</td>
</tr>
</tbody>
</table>

See Also

- `DeflaterOutputStream Class` |
SharpZip Compression Library
**DeflaterOutputStream.baseOutputStream Field**

Base stream the deflater depends on.

```csharp
protected Stream baseOutputStream;
```

See Also

SharpZip Compression Library
**DeflaterOutputStream,buf Field**

This buffer is used temporarily to retrieve the bytes from the deflater and write them to the underlying output stream.

```java
protected byte[] buf;
```

See Also

SharpZip Compression Library
The deflater which is used to deflate the stream.

```
protected Deflater def;
```

See Also

SharpZip Compression Library
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. When the flag is true Close will close the underlying stream also.</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 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. When null no encryption is performed</td>
</tr>
<tr>
<td>Position</td>
<td>The current position within the stream. Always throws a NotSupportedException</td>
</tr>
</tbody>
</table>

See Also

SharpZip Compression Library
DeflaterOutputStream.CanPatchEntries Property

Allows client to determine if an entry can be patched after its added

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

See Also

SharpZip Compression Library
DeflaterOutputStream.CanRead Property

Gets value indicating stream can be read from

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

See Also

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

SharpZip Compression Library
DeflaterOutputStream.CanWrite Property

Get value indicating if this stream supports writing

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

See Also

SharpZip Compression Library
DeflaterOutputStream.IsStreamOwner Property

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

public bool IsStreamOwner {get; set;}

See Also

SharpZip Compression Library
DeflaterOutputStream.Length Property

Get current length of stream

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

See Also

- DeflaterOutputStream Class
SharpZip Compression Library
DeflaterOutputStream.Password Property

Get/set the password used for encryption. When null no encryption is performed

```csharp
public string Password {get; set;}
```

See Also

SharpZip Compression Library
DeflaterOutputStream.Position Property

The current position within the stream. Always 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

- DeflaterOutputStream Class
SharpZip Compression 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 arent supported, a NotSupportedException is always thrown</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Calls finish() and closes the underlying stream when IsStreamOwner is true.</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 by calling finish() on the deflater.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes 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><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>GetLifetimeService</code> (inherited from <code>MarshalByRefObject</code>)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</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>GetLifetimeService</code> (inherited from <code>MarshalByRefObject</code>)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</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>GetLifetimeService</code> (inherited from <code>MarshalByRefObject</code>)</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td><code>Read</code></td>
<td>Read a block of bytes from stream</td>
</tr>
<tr>
<td><code>ReadByte</code></td>
<td>Read a byte from stream advancing position by one</td>
</tr>
<tr>
<td><code>Seek</code></td>
<td>Sets the current position of this stream to the given value. Not supported by this class!</td>
</tr>
<tr>
<td><code>SetLength</code></td>
<td>Sets the length of this stream to the given value. Not supported by this class!</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>
<tr>
<td><code>Write</code></td>
<td>Writes bytes from an array to the compressed stream.</td>
</tr>
<tr>
<td><code>WriteByte</code></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 <a href="#">WaitHandle</a> object.</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> (inherited from Object)</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><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 <a href="#">Object</a>.</td>
</tr>
<tr>
<td><strong>UpdateKeys</strong></td>
<td>Update encryption keys</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression 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`
- `offset`
- `count`
- `callback`
- `state`

Return 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

- DeflaterOutputStream Class
SharpZip Compression Library
Asynchronous writes are not supported, a NotSupportedException is always thrown.

```csharp
public override IAsyncResult BeginWrite(
    byte[] buffer,
    int offset,
    int count,
    AsyncCallback callback,
    object state
);```

### Parameters
- `buffer`
- `offset`
- `count`
- `callback`
- `state`

### Return 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
- DeflaterOutputStream Class
SharpZip Compression Library
Calls finish() and closes the underlying stream when IsStreamOwner is true.

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

See Also

- [DeflaterOutputStream Class](#)
SharpZip Compression 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

- [DeflaterOutputStream Class](#)
SharpZip Compression 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](#)
SharpZip Compression Library
# DeflaterOutputStream.EncryptByte Method

Encrypt a single byte

```csharp
protected byte EncryptByte();
```

## Return Value
The encrypted value

## See Also
- [DeflaterOutputStream Class](#)
SharpZip Compression 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>SharpZipBaseException</td>
<td>Not all input is deflated</td>
</tr>
</tbody>
</table>

**See Also**

- [DeflaterOutputStream Class](#)
SharpZip Compression 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

SharpZip Compression Library
DeflaterOutputStream.InitializePassword Method

Initializes encryption keys based on given password

```csharp
protected void InitializePassword(
    string password
);
```

See Also

DeflaterOutputStream Class |
SharpZip Compression Library
DeflaterOutputStream.Read Method

Read a block of bytes from stream

```csharp
public override int Read(byte[] b, int off, int len);
```

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](#)
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

- DeflaterOutputStream Class
SharpZip Compression 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](#)
DeflaterOutputStream.SetLength Method

Sets the length of this stream to the given value. Not supported by this class!

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

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

SharpZip Compression Library
**DeflaterOutputStream.UpdateKeys Method**

Update encryption keys

```csharp
protected void UpdateKeys(
    byte ch
);
```

**See Also**

SharpZip Compression Library
DeflaterOutputStream.Write Method

Writes bytes from an array to the compressed stream.

```csharp
public override void Write(
    byte[] buf,
    int off,
    int len
);
```

Parameters

- **buf**
  - The byte array

- **off**
  - The offset into the byte array where to start.

- **len**
  - The number of bytes to write.

See Also

- [DeflaterOutputStream Class](#)
SharpZip Compression Library
DeflaterOutputStream.WriteByte Method

Writes a single byte to the compressed output stream.

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

Parameters

`bval`

The byte value.

See Also

SharpZip Compression 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

SharpZip Compression Library
## InflaterInputBuffer Members

### InflaterInputBuffer overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InflaterInputBuffer</strong> Constructor</td>
<td>Initialise a new instance of <strong>InflaterInputBuffer</strong></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 <strong>ClearText</strong>.</td>
</tr>
<tr>
<td><strong>CryptoTransform</strong></td>
<td>Get/set the <strong>ICryptoTransform</strong> 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 <strong>RawData</strong>.</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>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></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 <strong>Object</strong>.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</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 <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**

InflaterInputBuffer Class
InflaterInputBuffer Constructor

Initialise a new instance of `InflaterInputBuffer`

```csharp
public InflaterInputBuffer(
    Stream stream
);
```

Parameters

`stream`
The stream to buffer.

See Also

- `InflaterInputBuffer Class`
SharpZip Compression 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

SharpZip Compression Library
**InflaterInputBuffer.Available Property**

Get/set the number of bytes available

```csharp
public int Available {get; set;}
```

See Also

- InflaterInputBuffer Class
SharpZip Compression Library
Get the contents of the clear text buffer.

```
public byte[] ClearText {get;}
```

See Also

SharpZip Compression Library
**InflatorInputBuffer.ClearTextLength Property**

Get the number of useable bytes in **ClearText**

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

**See Also**

- [InflatorInputBuffer Class](#)
SharpZip Compression 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

- InflaterInputBuffer Class
SharpZip Compression 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

- InflaterInputBuffer Class
SharpZip Compression Library
InflaterInputBuffer.RawLength Property

Get the length of bytes bytes in the RawData

```
public int RawLength {get;}
```

See Also

SharpZip Compression Library
### InflaterInputBuffer Methods

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></td>
<td>Determines whether the specified <code>Object</code> is equal to the current <code>Object</code>.</td>
</tr>
<tr>
<td><strong>Fill</strong></td>
<td>Fill the buffer from the underlying input 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>GetType</strong></td>
<td>Gets the <code>Type</code> 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 <code>SetInput</code> passing the current clear text buffer contents.</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>----------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
</tbody>
</table>

Protected Instance Methods

<table>
<thead>
<tr>
<th><strong>Finalize</strong> (inherited from <strong>Object</strong>)</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 <strong>Object</strong>)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>

See Also

- **InflaterInputBuffer Class**
SharpZip Compression Library
**InflaterInputBuffer.Fill Method**

Fill the buffer from the underlying input stream.

```csharp
public void Fill();
```

See Also

**InflaterInputBuffer.ReadClearTextBuffer Method**

Read clear text data from the input stream.

```
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 |
SharpZip Compression 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
SharpZip Compression Library
InflaterInputBuffer.ReadLeInt Method

Read an int in little endian byte order.

```csharp
public int ReadLeInt();
```

See Also

- InflaterInputBuffer Class
SharpZip Compression Library
InflaterInputBuffer.ReadLeLong Method

Read an int baseInputStream little endian byte order.

```csharp
public long ReadLeLong();
```

See Also

- InflaterInputBuffer Class
InflaterInputBuffer.ReadLeShort Method

Read an unsigned short in little endian byte order.

```csharp
public int ReadLeShort();
```

See Also

[InflaterInputBuffer Class](#)
SharpZip Compression Library
InflaterInputBuffer.ReadRawBuffer Method

Read a buffer directly from the input stream

**Overload List**

Read a buffer directly from the input stream

```java
public int ReadRawBuffer(byte[]);
```

Read a buffer directly from the input stream

```java
public int ReadRawBuffer(byte[],int,int);
```

See Also

- InflaterInputBuffer Class
SharpZip Compression 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](#)
SharpZip Compression Library
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
SharpZip Compression 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

SharpZip Compression 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


See Also

InflaterInputStream Members
SharpZip Compression Library
# InflaterInputStream Members

## InflaterInputStream overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="InflaterInputStream" /></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><img src="image" alt="Available" /></td>
<td>Returns 0 once the end of the stream (EOF) has been reached. Otherwise returns 1.</td>
</tr>
<tr>
<td><img src="image" alt="CanRead" /></td>
<td>Gets a value indicating whether the current stream supports reading</td>
</tr>
<tr>
<td><img src="image" alt="CanSeek" /></td>
<td>Gets a value of false indicating seeking is not supported for this stream.</td>
</tr>
<tr>
<td><img src="image" alt="CanWrite" /></td>
<td>Gets a value of false indicating that this stream is not writeable.</td>
</tr>
<tr>
<td><img src="image" alt="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><img src="image" alt="Length" /></td>
<td>A value representing the length of the stream in bytes.</td>
</tr>
<tr>
<td><img src="image" alt="Position" /></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></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<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>(inherited from MarshalByRefObject)</td>
<td>to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>✶ <strong>Read</strong></td>
<td>Decompresses data into the 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**

<p>| ✶ baseInputStream | Base stream the inflater reads from. |
| ✶ csize           | The compressed size |</p>
<table>
<thead>
<tr>
<th>inf</th>
<th>Decompressor for this stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputBuffer</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>CreateWaitHandle (inherited from Stream)</td>
<td>Allocates a <a href="#">WaitHandle</a> object.</td>
</tr>
<tr>
<td>Fill</td>
<td>Fills the buffer with more data to decompress.</td>
</tr>
<tr>
<td>Finalize (inherited from Object)</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>MemberwiseClone (inherited from Object)</td>
<td>Creates a shallow copy of the current <a href="#">Object</a>.</td>
</tr>
<tr>
<td>StopDecrypting</td>
<td>Clear any cryptographic state.</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression 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

SharpZip Compression Library
Create an InflaterInputStream with the default decompressor and a default buffer size of 4KB.

```
public InflaterInputStream(
    Stream baseInputStream
);
```

**Parameters**

*baseInputStream*

The InputStream to read bytes from

**See Also**

Create an InflaterInputStream with the specified decompressor and a default buffer size of 4KB.

```csharp
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**

- `InflaterInputStream Class`
- `InflaterInputStream Constructor Overload List`
SharpZip Compression Library
InflaterInputStream Constructor (Stream, Inflater, Int32)

Create an InflaterInputStream with the specified decompressor and the specified buffer size.

```csharp
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
SharpZip Compression Library
**InflaterInputStream Fields**

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>baseInputStream</td>
<td>Base stream the inflater reads from.</td>
</tr>
<tr>
<td>csize</td>
<td>The compressed size</td>
</tr>
<tr>
<td>inf</td>
<td>Decompressor for this stream</td>
</tr>
<tr>
<td>inputBuffer</td>
<td>Input buffer for this stream.</td>
</tr>
</tbody>
</table>

**See Also**

SharpZip Compression Library
InflaterInputStream.baseInputStream Field

Base stream the inflater reads from.

```java
protected Stream baseInputStream;
```

See Also

- InflaterInputStream Class
SharpZip Compression Library
InflaterInputStream.csize Field

The compressed size

```csharp
protected long csize;
```

See Also

SharpZip Compression Library
InflaterInputStream.inf Field

Decompressor for this stream

protected Inflater inf;

See Also

SharpZip Compression Library
**InflaterInputStream.inputBuffer Field**

`Input buffer` for this stream.

```java
protected InflaterInputBuffer inputBuffer;
```

See Also

- [InflaterInputStream Class](#)
SharpZip Compression 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

SharpZip Compression Library
InflaterInputStream.Available Property

Returns 0 once the end of the stream (EOF) has been reached. Otherwise returns 1.

public virtual int Available {get;}

See Also

SharpZip Compression Library
InflaterInputStream.CanRead Property

Gets a value indicating whether the current stream supports reading

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

See Also

- `InflaterInputStream Class`
SharpZip Compression Library
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
SharpZip Compression Library
InflaterInputStream.CanWrite Property

Gets a value of false indicating that this stream is not writeable.

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

See Also

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

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

Remarks
The default value is true.

See Also
- InflaterInputStream Class
**InflaterInputStream.Length Property**

A value representing the length of the stream in bytes.

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

See Also

SharpZip Compression Library
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](#)
SharpZip Compression 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> (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 <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Flushes the baseInputSteam</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>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong> (inherited from <em>MarshalByRefObject</em>)</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 <em>Object</em>)</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong> (inherited from <em>MarshalByRefObject</em>)</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Decompresses data into the byte array.</td>
</tr>
<tr>
<td><strong>ReadByte</strong> (inherited from <em>Stream</em>)</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 <code>NotSupportedException</code>.</td>
</tr>
<tr>
<td><strong>SetLength</strong></td>
<td>Set the length of the current stream. Always throws a <code>NotSupportedException</code>.</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 <em>Object</em>)</td>
<td>Returns a <code>String</code> that represents the current <code>Object</code>.</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 <code>NotSupportedException</code>.</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Writes one byte to the current stream and advances the current position. Always throws a <code>NotSupportedException</code>.</td>
</tr>
</tbody>
</table>
## Protected Instance Methods

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

### See Also

- [InflaterInputStream Class](#)
SharpZip Compression 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**
SharpZip Compression Library
Closes the input stream. When `IsStreamOwner` is true the underlying stream is also closed.

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

See Also

- [InflaterInputStream Class](#)
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 |
InflaterInputStream.Flush Method

Flushes the baseInputStream

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

See Also

SharpZip Compression Library
Decompresses data into the byte array

```csharp
public override int Read(
    byte[] b,
    int off,
    int len
);
```

### Parameters

*b*
- The array to read and decompress data into

*off*
- The offset indicating where the data should be placed

*len*
- 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`
SharpZip Compression Library
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

- InflaterInputStream Class
SharpZip Compression Library
**InflaterInputStream.SetLength Method**

Set the length of the current stream Always throws a NotSupportedException

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

## 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
SharpZip Compression Library
**InflaterInputStream.Skip Method**

Skip specified number of bytes of uncompressed data

```csharp
public long Skip(long n);
```

**Parameters**

- **n**
  - 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 zero or less</td>
</tr>
</tbody>
</table>

**See Also**

InflaterInputStream.StopDecrypting Method

Clear any cryptographic state.

```csharp
protected void StopDecrypting();
```

See Also

SharpZip Compression Library
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[] array,
    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

SharpZip Compression Library
**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 val);
```

<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
SharpZip Compression Library
**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](#).

```csharp
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**

- **Namespace:** ISharpCode.SharpZipLib.Zip.Compression.Streams
- **Assembly:** ISharpCode.SharpZipLib (in ISharpCode.SharpZipLib.dll)

**See Also**

- [OutputWindow Members](#)
## OutputWindow Members

### OutputWindow overview

### Public Instance Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutputWindow Constructor</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>CopyDict</td>
<td>Copy dictionary to window</td>
</tr>
<tr>
<td>CopyOutput</td>
<td>Copy contents of window to output</td>
</tr>
<tr>
<td>CopyStored</td>
<td>Copy from input manipulator to internal window</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td>GetAvailable</td>
<td>Get bytes available for output in window</td>
</tr>
<tr>
<td>GetFreeSpace</td>
<td>Get remaining unfilled space in 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>Repeat</td>
<td>Append a byte pattern already in the window itself</td>
</tr>
<tr>
<td>Reset</td>
<td>Reset by clearing window so GetAvailable returns 0</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>
Write

Write a byte to this output window

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

- [OutputWindow Class](#)
SharpZip Compression Library
OutputWindow Constructor

Initializes a new instance of the OutputWindow class.

```java
public OutputWindow();
```

See Also

- [OutputWindow Class](#)
SharpZip Compression Library
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> (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>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 <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>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 <code>GetAvailable</code> returns 0</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>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>✷ <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

- [OutputWindow Class](#)
SharpZip Compression Library
OutputWindow.CopyDict Method

Copy dictionary to window

```csharp
public void CopyDict(
    byte[] dict,
    int offset,
    int len
);
```

Parameters

- `dict` 
  source dictionary

- `offset` 
  offset of start in source dictionary

- `len` 
  length of dictionary

Exceptions

<table>
<thead>
<tr>
<th>Exception Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidOperationException</td>
<td>If window isn't empty</td>
</tr>
</tbody>
</table>

See Also

- [OutputWindow Class](#)
SharpZip Compression 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](#)
SharpZip Compression Library
OutputWindow.CopyStored Method

Copy from input manipulator to internal window

```csharp
public int CopyStored(
    StreamManipulator input,
    int len
);
```

Parameters

input
    source of data

len
    length of data to copy

Return Value

the number of bytes copied

See Also

SharpZip Compression Library
OutputWindow.GetAvailable Method

Get bytes available for output in window

```
public int GetAvailable();
```

Return Value

Number of bytes filled

See Also

- [OutputWindow Class](#)
Get remaining unfilled space in window

```csharp
public int GetFreeSpace();
```

Return Value
Number of bytes left in window

See Also
- OutputWindow Class
SharpZip Compression Library
**OutputWindow.Repeat Method**

Append a byte pattern already in the window itself

```csharp
public void Repeat(
    int len,
    int dist
);
```

**Parameters**

*len*
- length of pattern to copy

*dist*
- 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`
OutputWindow.Reset Method

Reset by clearing window so GetAvailable returns 0

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

See Also

SharpZip Compression Library
OutputWindow.Write Method

Write a byte to this output window

```csharp
public void Write(int aByte);
```

Parameters

`abyte` 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`
SharpZip Compression 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.

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

SharpZip Compression Library
### StreamManipulator Members

#### StreamManipulator overview

#### Public Instance Constructors

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StreamManipulator Constructor</td>
<td>Constructs a default StreamManipulator with all buffers empty</td>
</tr>
</tbody>
</table>

#### 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>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyBytes</td>
<td>Copies length 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.</td>
</tr>
<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 (inherited from Object)</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> (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 n bits but don't increase input pointer. n 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 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 <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**
SharpZip Compression Library
StreamManipulator Constructor

Constructs a default StreamManipulator with all buffers empty

```java
public StreamManipulator();
```

See Also

- [StreamManipulator Class](#)
SharpZip Compression 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><code>AvailableBits</code></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><code>AvailableBytes</code></td>
<td>Gets the number of bytes available.</td>
</tr>
<tr>
<td><code>IsNeedingInput</code></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.

```
public int AvailableBits {get;}
```

See Also

SharpZip Compression Library
StreamManipulator.AvailableBytes Property

Gets the number of bytes available.

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

See Also

- StreamManipulator Class
SharpZip Compression Library
StreamManipulator.IsNeedingInput Property

Returns true when SetInput can be called

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

See Also

SharpZip Compression 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 length 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.</td>
</tr>
<tr>
<td><strong>DropBits</strong></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><strong>Equals</strong></td>
<td>Determines whether the specified Object is equal to the current Object.</td>
</tr>
<tr>
<td><strong>GetBits</strong></td>
<td>Gets the next n bits and increases input pointer. This is equivalent to PeekBits followed by dropBits, except for correct error handling.</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>PeekBits</strong></td>
<td>Get the next n bits but don't increase input pointer. n must be less or equal 16 and if this</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</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 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 <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**

- [StreamManipulator Class](#)
StreamManipulator.CopyBytes Method

Copies length 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

SharpZip Compression 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 n
);
```

See Also

SharpZip Compression Library
StreamManipulator.GetBits Method

Gets 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 n);
```

Return Value

the value of the bits, or -1 if not enough bits available.

See Also

Get the next n bits but don't increase input pointer. n 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 n);
```

Return Value

the value of the bits, or -1 if not enough bits available. */

See Also

SharpZip Compression Library
StreamManipulator.Reset Method

resets state and empties internal buffers

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

See Also

- StreamManipulator Class
SharpZip Compression Library
StreamManipulator.SetInput Method

Add more input for consumption. Only call when IsNeedInput returns true

```csharp
public void SetInput(
    byte[] buf,
    int off,
    int len
);
```

Parameters

- `buf`
  - data to be input
- `off`
  - offset of first byte of input
- `len`
  - length of input

See Also

SharpZip Compression Library
StreamManipulator.SkipToByteBoundary Method

Skips to the next byte boundary.

```csharp
public void SkipToByteBoundary();
```

See Also

SharpZip Compression Library
ICSharpCode.SharpZipLib Hierarchy

System.Object
System.Exception ---- System.Runtime.Serialization.ISerializable
System.ApplicationException
ICSharpCode.SharpZipLib.SharpZipBaseException

See Also
ICSharpCode.SharpZipLib Namespace
SharpZip Compression Library
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.Core Hierarchy

System.Object
ICSharpCode.SharpZipLib.Core.INameTransform
ICSharpCode.SharpZipLib.Core.IScanFilter
ICSharpCode.SharpZipLib.Core.NameFilter
ICSharpCode.SharpZipLib.Core.PathFilter
ICSharpCode.SharpZipLib.Core.NameFilter
ICSharpCode.SharpZipLib.Core.NameAndSizeFilter
ICSharpCode.SharpZipLib.Core.ScanFailureEventArgs
System.Delegate ---- System.ICloneable,
System.Runtime.Serialization.ISerializable
System.MulticastDelegate
System.EventArgs
ICSharpCode.SharpZipLib.Core.ScanEventArgs

See Also

ICSharpCode.SharpZipLib.Core Namespace
SharpZip Compression Library
ICSharpCode.SharpZipLib.Encryption Hierarchy

System.Object
System.IDisposable

See Also

ICSharpCode.SharpZipLib.Encryption Namespace
# ISharpCode.SharpZipLib.GZip Hierarchy

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System.Object</td>
<td></td>
</tr>
<tr>
<td>ISharpCode.SharpZipLib.GZip.GZipConstants</td>
<td></td>
</tr>
<tr>
<td>System.Exception</td>
<td>System.Runtime.Serialization.ISerializable</td>
</tr>
<tr>
<td>System.ApplicationException</td>
<td></td>
</tr>
<tr>
<td>ISharpCode.SharpZipLib.SharpZipBaseException</td>
<td></td>
</tr>
<tr>
<td>ISharpCode.SharpZipLib.GZip.GZipException</td>
<td></td>
</tr>
<tr>
<td>System.MarshalByRefObject</td>
<td></td>
</tr>
<tr>
<td>System.IO.Stream</td>
<td>System.IDisposable</td>
</tr>
<tr>
<td>ISharpCode.SharpZipLib.GZip.GZipOutputStream</td>
<td></td>
</tr>
<tr>
<td>ISharpCode.SharpZipLib.GZip.GZipInputStream</td>
<td></td>
</tr>
</tbody>
</table>

## See Also

- ISharpCode.SharpZipLib.GZip Namespace
SharpZip Compression Library
ICSharpCode.SharpZipLib.Tar Hierarchy

System.Object
ICSharpCode.SharpZipLib.Tar.TarArchive
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
SharpZip Compression Library

System.Object
System.Collections.IEnumerable
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

SharpZip Compression Library

System.Object
System.ValueType
System.Enum ---- System.IComparable, System.IConvertible, System.IFormattable

See Also
SharpZip Compression Library

Hierarchy

System.Object
System.MarshalByRefObject
System.IO.Stream ---- System.IDisposable

See Also