DotNetZip - Zip file manipulation in .NET languages

DotNetZip is a small, easy-to-use class library for manipulating .zip files. It can enable .NET applications written in VB.NET, C#, or any .NET language, to easily create, read, and update zip files. Zip Compression is easy with DotNetZip. The DotNetZip project also includes a library for performing ZLIB, Deflate, or GZIP compression and decompression, a library for BZip2 compression and decompression, a GUI ZIP tool, and a few command line tools.

DotNetZip works on Windows-powered PCs with the full .NET Framework, and also runs on Windows Mobile devices that use the .NET Compact Framework. Create and read zip files in VB, C#, or any .NET language. The library can also be used from COM enironments, like PHP, Classivc ASP, or VBSCript. DotNetZip supports these scenarios:

- creating a zip archive, adding files or directories into the archive
- listing files in an archive, extracting files from an archive
- modifying an existing archive renaming entries, removing entries from an archive, or adding new entries to an archive
- creating zip files from stream content, saving to a stream, extracting to a stream, reading from a stream
- dynamically creating ZIP files from ASP.NET or Silverlight applications

If all you want is a better DeflateStream or GZipStream class to replace the one that is built-into the .NET BCL, that is here, too. DotNetZip's DeflateStream and GZipStream are available in a standalone assembly, based on a .NET port of Zlib. These streams

support compression levels and deliver much better performance that the built-in classes. There is also a ZlibStream to complete the set (RFC 1950, 1951, 1952).

This 100% managed code library can be used in any .NET application - Console, Winforms, WPF, ASP.NET, Sharepoint, Web services apps, Powershell scripts, and so on. It produces zip files that are fully interoperable with Windows Explorer, as well as Java applications, apps running on Linux.

It is designed to be simple and easy to use. DotNetZip is packaged as a single DLL, about 400k in size. It has no third-party dependencies. It is Medium Trust, so can be used on most hosters. Get zipping just by referencing the DLL. The library supports zip passwords, Unicode, ZIP64, stream input and output, AES encryption, multiple compression levels, self-extracting archives, and more.

The release includes the library, as well as some sample applications (with source) showing how to use the library.

DotNetZip is DonationWare

If you find DotNetZip useful, consider donating. I am now <u>accepting</u> <u>donations</u> on behalf of my favorite charity. and Yes, it is a real charity.

Example Usage

Example: creating a zip file

```
Copy
```

```
using (ZipFile zip = new ZipFile("MyZipFile.zip")

zip.AddFile("c:\\images\\personal\\7440-N49th.png'
 zip.AddFile("c:\\Desktop\\2008-Regional-Sales-Report zip.AddFile("ReadMe.txt");
 zip.Save();
}
```

Tons more examples on **Codeplex**, and later in this helpfile.

Frequently Asked Questions

How does this Zip Library work? DotNetZip is packaged as a single DLL, a single assembly. It is fully managed code, written in C#, and provides support for reading and writing Zip archive files and streams. The main type is ZipFile, featuring methods like Add(), Extract() and Save(). There are string and int indexers for the entries of the ZipFile. There are properties for things like password protection, unicode and codepage support, and ZIP64 behavior. And there are progress events for Reading, Saving, and Extracting.

What do I need, in order to be able to create and read zip files from within my application using this library?

To use the zip capability in your applications, you need to be using the .NET Framework 2.0 or later, and you need the DotNetZip assembly. You can use the Zip library from any application, whether a console application, a Windows-Forms application, a server-based application like an ASP.NET page, or something else. You can use C#, VB.NET, COBOL.NET, IronPython, IronRuby, F#, or any other .NET language.

What do I need to build this library from the source?

To use the zip capability in your applications, you need to be using the .NET Framework 2.0 or later, and you need the DotNetZip assembly. You can use the Zip library from any application, whether a console application, a Windows-Forms application, a server-based application like an ASP.NET page, a smart-device app, a Windows Service, or something else. You can use C#, VB.NET, COBOL.NET, IronPython, IronRuby, F#, or any other .NET language. You can use the full version of Visual Studio, or one of the Visual Studio Express tools, or just a text editor.

What if I just want to use the graphical tool?

No problem. Just download the Utils download from the latest releases tab, and you will get an easy-to-use fully-functional ZIP creation and extraction tool, a free functional-equialent replacement for WinZip. It supports all the DotNetZip capability, like SFX, AES, ZIP64, arbitrary Code Pages, Unicode, zip comments, file and entry selection, and more.

What do I need to _build_ this library from the source?f

You need the .NET Framework SDK v3.5, or later; or, Visual Studio 2008 or later. While the library is usable by .NET 2.0 and later, to build it you need .NET 3.5 or later. This is because the source code uses features that were introduced in the C# v3.0 compiler, vars and array initializers and so on. The .NET 3.5 SDK is available here. I don't know if the library will _build_ in Visual C# Express 2008; I never tried it. It should.

How big is the library?

As of version 1.9, the Ionic zip DLL is about 420k in size. The self-extracting capability comprises about 200k of that. There is a "Reduced" build of the library in the Developer's Kit, that eliminates the SFX capability and results in a DLL of about half the size. The Compact-Framework version of the library is about 130k. There is just one DLL. There is no other pre-requisite.

Why would you want to produce yet another Zip Library?

There are a number of options in the .NET world for manipulating zip files. Some are commercial libraries, some are open-source. You can even shoe-horn the System.IO.Packaging API, which is included in .NET 3.0, into a zip library. But there are tradeoffs for all of them. Some people are willing to give up commercial support for a lower price. Some people don't like the GPL. Some of those other packages are complex to use for zip files. Some of them don't have enough features (Eg, AES encryption, ZIP64). Some are too slow. What I

found is that most people want something simple and easy to use, that works well, has good features, compresses effectively and fast, and is the right price. DotNetZip is that option. It's fast, it's simple, it's free.

Does this library make self-extracting zip files?

Yes. It can make self-extracting zip files or standard zip files. The self-extracting archives can either be Windows (GUI) apps or command-line applications. The self-extractors require .NET 2.0 on the computer doing the extraction.

Does this library read self-extracting zip files?

Yes. As of v1.7, DotNetZip can read self-extracting zip files generated by WinZip. As of v1.8 (currently in preview), DotNetZip can read SFX archives generated by itself. DotNetZip does not (yet) read SFX archives built by some other tools. I've not tested it.

Are the zip files this library makes compatible with the java.util.zip classes in the Java class library?

Yes. This library makes standard zip files. If you use some advanced features not supported by the built-in Java library, like AES encryption or ZIP64, then the zip files won't be readable by Java. This is a shortcoming in Java, not in DotNetZip.

If I create a zipfile with this library, can I open it from within a Java/PHP/Python/C/Perl application? From within WinRar/WinZIP?

Yes. This library makes standard zip files, so anything that can read and write zip files, on any platform, can work with zipfiles generated from this library. In Java, you would use the java.util.zip library. In Python, you would use the zipfile module. Keep in mind that not all languages and environments support all ZIP features. For example, currently the Java library does not do ZIP64, and the zipfile module in Python does not support zipfile comments, while DotNetZip can create and read zip archives using those features. You need to be

careful with interop testing. If you don't know what these features are, you probably don't need them and you won't run into interop problems.

If I create a zipfile with a java/PHP/Python/C/Perl application, or with WinRar/WinZip/7zip, can I open it from within a .NET application, using this library?

Yes. This library reads compliant zip files.

Can I use Windows Explorer to open the zip files that this library creates, and vice versa?

Yes. This library reads and writes standard zip files. If you use extensions to the ZIP spec not supported by Windows, like ZIP64 or AES encryption, Windows Explorer will not be able to extract the files.

Does the library work with applications that depend on the .NET Compact Framework?

Yes. DotNetZip is built for the .NET Compact Framework (v2.0 or later) as well as for the regular .NET Framework (v2.0 or later). This means you can build a Smart Device application in Visual Studio 2008, that uses the capabilities of DotNetZip.

Does the library support zero-length zip entries, zipfile comments, zip entry comments, zipping up empty directories, recursive directory traversal, zipping up selected files by filename (with wildcards), and password-protecting entries?

Yes.

Does the library handle ZIP64?

Yes. This was added in December 2008 in v1.7. The original ZIP specification allowed up to 65535 entries in a zip archive, and archive and entry sizes up to 4.2g. The ZIP64 extensions raise those restrictions, at the expense of compatibility and interoperability.

DotNetZip can read or write "standard" zip files or ZIP64 zip files.

Does this library support any compression algorithm other than deflate?

No. Entries are either Stored or Deflated.

Does this library use System.IO.Compression.DeflateStream for the DEFLATE algorithm?

No. In earlier versions, it did. But the DeflateStream built-into the .NET Framework BCL exhibits anomalous compression behavior on previously compressed data, like mp3 or jpg files. DotNetZip includes a managed compression library, based on ZLIB. It's fast, fully DEFLATE compliant, and it compresses better than the built in version! And, the DeflateStream that is included in DotNetZip is available for any application to use, under the same terms as the rest of the DotNetZip library.

Why would a developer not just use the Packaging APIs included in the .NET Framework 3.0?

The Packaging APIs are optimized for producing .docx files and .xlsx files. This library is generally simpler and cleaner for creating or reading plain, generic zipfiles. Also this library takes advantage of ZIP features like ZIP64 or AES, in a much nicer way. And it provides nice features like progress events and file selectors.

Does the library support reading or writing encrypted or password protected zip files?

Yes, this library helps applications read and write zip files that use passwords to protect the entries in the zip archive. Either PKZIP ("weak") encryption or WinZip-compatible AES encryption.

Can Windows Explorer ("compressed folders") read and extract the zip files created by this library if the zip has password protection?

Yes. It just works.

Can I use the DotNetZip library to read .docx files, .xslx files?

Yes. They are just zip files. Keep in mind, the Packaging APIs included in .NET 3.0 are optimized for producing .docx files and .xlsx files.

Does the library support Unicode filenames and comments?

Yes, as of September 2008, the library can create and read zip files that have Unicode (UTF-8) filenames and comments. You can also specify arbitrary code pages when reading or writing zipfiles. But, Windows Explorer does not support UTF-8 encoded zip files. To read the zip you create, you will have to use a tool or library that supports unicode, such as WinRar, DotNetZip, etc.

Can the library be used to read a zip file that has entries with filenames containing Chinese characters, as produced by WinRAR?

Yes. To do this you would specify the "big5" code page (cp 950) when reading the zip. In addition to Chinese, it can handle any code page.

What about reading and writing zip files with other languages and code pages? Portugese? Hebrew? Arabic? Greek? Cyrillic? Norwegian? Finnish? etc?

Yes. Just specify the appropriate code page when reading or writing the zip archive.

Can Windows Explorer ("compressed folders") read the zip files created by this library if the zip has entries with Unicode-encoded filenames?

No. But that is a limitation of Windows XP and Windows Vista. It is not a limitation of this library.

Is there documentation for the library?

Yes, there is a CHM file in MSDN helpfile format, generated from the xml comments in the code. All the public interfaces are thoroughly documented. There are tons of code examples in the doc.

Does the library support zipping to a stream? Or unzipping from a stream?

Yes, you can zip up files and Save the zip archive to a stream. As well you can Read a zip archive from an open stream - I use this for embedded resources in apps: I call GetManifestResourceStream(), and then unzip that resource. Reading and writing streams complements the capability of being able to Save to a plain file or read from a plain file. The Save-to-a-stream capability allows you to write a zip archive out to, for example, the ASP.NET Response.Output stream, without creating an intermediate file. Very nice for ASP.NET applications.

Ok, the library can write a zip archive to a stream, and read a zip from a stream, But... can the library add an entry to a zipfile, grabbing content from a stream? can the library unzip a single entry into a stream? Can an application read an entry as a stream?

Yes. Yes. Yes. Unlike some other libraries, in most cases DotNetZip handles the streaming; your application does not need to implement a Read/Write data pump. The application needs only to open the streams. Using the stream support, you could, for example, open a zip archive, and then modify the files in the archive, and Save out to a Response.OutputStream in ASP.NET, without ever writing a file to the disk. All the zip file content can be maniulated in memory (using MemoryStream for example).

Does this library allow removal of entries from zip files, or updating of entries in zip files?
Yes.

Do I have to write programs to take advantage of this thing? or is

there some other way I can take advantage of the DotNetZip?

Typically people will embed this DLL into an application. But, the DotNetZip downloads includes a package of "example" applications: command-line utilities that utilize the library, that you can use out of the box to zip and unzip. I drop these in my own bin directory and it allows me to zip and unzip from the command line or from batch files. Also, you can take advantage of the DotNetZip library from within Powershell, if that is your bag. Creating or extracting zips within a script is pretty handy.

Can I grab the source for DotNetZip and embed it into my own project?

Yes - that's allowed by the license. But you may want to think about just redistributing the binary DLL - it is a much easier option.

Can I distribute the binary DLL with my own project?

Yes - that's allowed by the license.

What's the mainstream approach for using DotNetZip in a 3rd party app?

The mainstream approach is to distribute the binary DLL with your own app.

What if I want to use DotNetZip in my app, but I don't want to distribute it as a separate DLL?

You have options. The most mainstream and generally easiest way to embed DotNetZip into your application is to redistribute the DLL. But if you don't want to do that, there are alternatives. One option is to embed the source for DotNetZip into your own project. This may seem like the obvious approach, but it is probably not what you want, unless you really need to modify the source of DotNetZip. A better approach is to merge the DLL into your EXE, with ILMerge. This works with your app regardless of what language it uses, whether it is a Winforms app, a WPF app, a service, etc.

Getting Started with DotNetZip

This page describes how to get started using DotNetZip.

Which download to choose?

DotNetZip is free to use. You just need to download it and start using it. Different downloads of DotNetZip are available. Which one you choose depends on what you want to do.

If your goal is	Download this	which contains
To read the documentation for the library and tools	DotNetZipLib- *.chm	a compiled helpfile containing all the documentation.
To build .NET, COM, or PowerShell apps that use the Zip library.	DotNetZipLib- DevKit-vx.x.zip	the signed DLLs for Zip and Zlib, for both the desktop and Compact Framework; the XML Documentation file for intellisense; the compiled help file (CHM); an MSI file that installs help into Visual Studio; and the License.
To modify or view the source code for the Zip & Zlib Libraries, the setup (msi) project, or the examples.	DotNetzip-src- vx.x.x.x.zip	the VS2008 solution, containing a number of VS projects and the source code for the library and various examples that use the library.
To distribute .NET applications that use the Zip library, or if you want to run a .NET application that depends on this library.	DotNetZipLib- Runtime- vx.x.zip	the signed DLLs for Zip and Zlib, for both the desktop and Compact Framework; and the License. This is a strict subset of the DevKit download.
To distribute .NET applications that use the Zip library, or if you want to run a .NET application	Runtime-	an .MSI version of the Runtime release, containing signed DLLs for Zip and Zlib, for both the desktop and Compact Framework;

that depends on this library.		and the License.
To create, read, or modify ZIP files from within a Windows GUI interface, or from the command line, or from within batch files.	DotNetZipUtils- v1.8.zip	a set of exe tools (command-line and GUI) that rely on the DotNetZip library, to create and manage zip files.
	DotNetZipUtils- v1.8.msi	an .MSI version of the Utils download. This installs a set of exe tools (command-line and GUI) that rely on the DotNetZip library, to create and manage zip files. There's also an uninstaller.

Which DLL to use?

When you are building an application that manipulates ZIP files, you will want to use one of the binary releases of DotNetZip. These releases include multiple distinct DLLs or assemblies. Which one should you use?

The likely answer is: *Ionic.Zip.dll*

That's the mainstream library, the full library, and it includes all the capability of DotNetZip. The other DLLs are reduced in various ways, to suite particular requirements.

For example, you may want a smaller library, or you want to exclude the Self-Extracting capability, or you only want the ZLIB capability. In these cases, you may want to choose a different assembly.

Here's a summary of the options.

Usage scenario	DLL
Basic reading or writing of Zip files	Ionic.Zip.dll
raw block or stream compression using ZLIB, DEFLATE, or GZip	Ionic.Zlib.dll
compression using the BZip2 algorithm	Ionic.BZip2.dll
Some combination of ZLIB/DEFLATE/GZIP compression, BZip2 compression, and reading or writing Zip files	Ionic.Zip.dll
reading or writing Zip files on .NET Compact Framework	Ionic.Zip.CF.dll
ZLIB/DEFLATE/GZIP compression on .NET Compact Framework	Ionic.Zlib.CF.dll
compression using the BZip2 algorithm on .NET	Ionic.BZip2.CF.dll

Compact Framework	
Some combination of ZLIB/DEFLATE/GZIP compression, BZip2 compression, and reading or writing Zip files on .NET CF	Ionic.Zip.CF.dll
reading or writing Zip files, but never creating a self- extracting archive	Ionic.Zip.Reduced.dll

There are also Silverlight versions of the Zlib, BZip2, and Zip libraries.

Never reference both Ionic.Zlib.dll and Ionic.Zip.dll in the same application, or both Ionic.BZip2.dll and Ionic.Zip.dll. If your application does both Zlib and Zip things, or both BZip2 and Zip things, or all three, you need only add a reference to Ionic.Zip.dll. Ionic.Zip.dll includes all the capability in Ionic.Zlib.dll and Ionic.BZip2.dll. You always need to reference only a single Ionic DLL, regardless whether you use Zlib or BZip2 or Zip or some combination.

If you use COM, then you want to reference the main DLL, Ionic.Zip.dll. If you install the DotNetZip Runtime library via the .MSI installer, the correct DLL will automatically be installed and registered for use with COM. If you don't use the .MSI installer, you will have to perform these steps yourself. Consult the documentation on using DotNetZip with COM for more information.

DotNetZip works with the .NET Compact Framework

There is a special version of DotNetZip, specifically built for the .NET Compact Framework v2.0 or v3.5.

The programming model is exactly the same as for the regular DotNetZip library. There are a few features not supported in the CF version of DotNetZip:

- 1. **Self-extracting archives.** The ZipFile.SaveSelfExtractor() method is not available in the CF version of DotNetZip. There's no strong technical reason for this limnitation; I just didn't think it would be of mainstream interest, and in order to keep the size of the library down, I omitted that part from the CF version. It amounts to about a 120k savings.
- 2. **WinZip AES encryption.** The CF version does not support encryption using WinZipAes128 or WinZipAes256. This is because the crypto classes from the .NET Framework are not available on .NET CF. I would have to write my own SHA1HMAC() and so on, which I haven't got 'round to doing, especially because no one is clamoring for it.
- 3. **ParallelDeflateStream**. Most mobile devices do not use multicore processors, so a parallel deflater would offer no benefit. Even aside from that, parallel deflation shows its strength when cmopressing larger files, and when larger memory resources are available. Both of these are unlikely in .NET CF applications.

Using DotNetZip in a Smart Device project

To build a smart device application that uses DotNetZip, you will need to use Visual Studio 2008 or later.

- 1. Open Visual Studio 2008.
- 2. Create a smart device project. Specify that it will use the .NET Compact Framework v2.0 or v3.5.
- 3. Add a Reference to the CF Library for DotNetZip. The filename is Ionic.Zip.CF.dll
- 4. Build your application as normal.

About the Project Design

If you are a developer who wants to modify or extend the DotNetZip library itself, the following may be of interest.

The CF-Library project in the DotNetZip source distribution includes all of the source files as *Links* to the original files in the Library project. This means, the source files are not duplicated. Any changes need to be made only once. The CF-Library project is simply a "build project" in that regard. There is no additional source code specifically for the .NET Compact Framework.

NETCF is defined as a conditional compile symbol for the CF-Library project. This symbol is used in the source to exclude code that will not run on the .NET Compact Framework, for example, the calls to SetLastModifiedTime() method on the System.IO.File class.

DotNetZip - C# Examples

Here are a bunch of examples in C# that illustrate how to use the library.

There are a few complete, working example applications shipped in the source code distribution.

Create a zip file, and add items to it.

```
using (ZipFile zip = new ZipFile())

zip.AddFile("ReadMe.txt");
zip.AddFile("Resume.doc");
zip.AddFile("Portrait.png");
zip.Save("Package.zip");
}
```

Add items to a zip file, using Zip 2.0 encryption, and the same password for all items.

```
using (ZipFile zip = new ZipFile())

zip.Password= "123456!";

zip.AddFile("ReadMe.txt");

zip.AddFile("7440-N49th.png");

zip.AddFile("2005_Annual_Report.pdf");

zip.Save("Backup.zip");

}
```

Add files to a zip file, using Zip 2.0 encryption, and different passwords for different files.

```
using (ZipFile zip = new ZipFile())

zip.AddFile("ReadMe.txt"); // no password for thi
zip.Password= "123456!";
zip.AddFile("7440-N49th.png");
zip.Password= "!Secret1";
zip.AddFile("2005_Annual_Report.pdf");

zip.Save("Backup.zip");
}
```

Create a zip archive, and add files to it, using WinZip-compatible AES 256-bit encryption for one of the files.

Add items 4

```
using (ZipFile zip = new ZipFile())

zip.AddFile("ReadMe.txt"); // no password for this
zip.Password= "Cool.Hand.Luke!";
zip.Encryption= EncryptionAlgorithm.WinZipAes256;
zip.AddFile("Rawdata-2008-12-18.csv");
zip.Save("Backup-AES-Encrypted.zip");
}
```

Create a zip, and add a file, telling the library to not use compression when adding in the file. This makes sense with previously-compressed files such as those in .mp3 format. The Deflate algorithm can actually

increase the size of a data stream that has already been compressed. The DotNetZip library intelligently turns off compression for those files that get bigger with compression, but the ForceNoCompression property allows you to do it explicitly.

No Compression

Copy

```
using (ZipFile zip = new ZipFile())

zip.ForceNoCompression = true;

zip.AddFile(@"MyMusic\Handel\Messiah-01.mp3");

zip.Save(ZipFileToCreate);
}
```

Zip up an entire directory, recursively. Use unicode to encode entries in the archive, for those files in that directory tree that have characters outside the ANSI range. Finally, specify a comment on the zip archive when creating it. (Be careful using Unicode - it is not widely supported by other zip utilities)

Zip a directory

■ Copy

```
using (ZipFile zip = new ZipFile())

zip.UseUnicode= true; // utf-8
    zip.AddDirectory(@"MyDocuments\ProjectX");
    zip.Comment = "This zip was created at " + System.
    zip.Save(ZipFileToCreate);
}
```

Zip up an entire directory, recursively. Use the "big5" encoding for those files in that directory tree that have chinese characters.

Zip a directory 2

```
using (ZipFile zip = new ZipFile())

zip.Encoding = System.Text.Encoding.GetEncoding("k
   zip.AddDirectory(@"MyDocuments\ProjectX");
   zip.Save(ZipFileToCreate);
}
```

Zip up an file, using the ZIP64 extensions if necessary to support large files.

Zip64 □ Copy

```
using (ZipFile zip = new ZipFile())

zip.UseZip64WhenSaving = Zip64Option.AsNecessary;

zip.AddFile(@"RawData-HugeFile-13800.dat");

zip.Save(ZipFileToCreate);
}
```

Zip up a set of files, each protected by the same password. Also, explicitly override the last modified time for all of the files as they are stored in the zip archive.

Set Last Modified time

♣ Copy

```
<hr/>
1
2
 Zip up a set of files and directories, and re-ma
3
 different directory hierarchy in the zip file.
4
5
 6
   title="remap directories">
7
   using (ZipFile zip = new ZipFile())
8
   {
9
```

```
// files in the filesystem like MyDocuments\Pro
10
      zip.AddDirectory(@"MyDocuments\ProjectX", "back
11
      // files in the filesystem like MyMusic\Santana
12
      zip.AddDirectory("MyMusic", "tunes");
13
      // The Readme.txt file in the filesystem will b
14
      zip.AddDirectory("Readme.txt", "documents");
15
16
      zip.Comment = "This zip was created at " + Syst
17
      zip.Save(ZipFileToCreate);
18
    }
19
```

Add content obtained from a stream (MemoryStrean, FileStream, etc) into a zip archive. Also, add a comment to the entry that was added from the stream.

```
using (ZipFile zip = new ZipFile())

ZipEntry e= zip.AddEntry("Content-From-Stream.bin"
e.Comment = "The content for entry in the zip file
zip.AddFile("Readme.txt");
zip.Save(ZipToCreate);
}
```

Open an existing zip file, remove an entry from it, and save the archive. This was first supported in v1.5 of the library.

Remove an entry

♣ Copy

```
using (ZipFile zip = ZipFile.Read(ExistingZipFile))
{
   // use the indexer to remove the file from the zip
   zip["Readme.txt"] = null;
   zip.Comment = "This archive has been modified from
```

```
6  zip.Save();
7 }
```

Open an existing zip file, rename an entry, then save it. This was first supported in v1.7 of the library.

Rename Copy

```
int renameCount = 0;
  using (ZipFile zip2 = ZipFile.Read(ExistingZipFile)
  {
3
      foreach (ZipEntry e in zip2)
4
5
        if (e.FileName.EndsWith(".txt"))
6
        {
7
           var newname = "renamed_files\\" + e.FileNam
8
9
           e.FileName = newname;
10
           e.Comment = "renamed";
11
           renameCount++;
12
        }
13
14
      zip2.Comment = String.Format("This archive has b
15
      zip2.Save();
16
17 | }
```

Extract all files from a zip archive:

Extract 1 © Copy

```
using (ZipFile zip = ZipFile.Read(ExistingZipFile))

foreach (ZipEntry e in zip)

e.Extract(TargetDirectory);
```

```
6 | }
7 | }
```

The default behavior of extraction is to NOT overwrite existing files. In this example, the app Extracts all files, and overwrites existing files in the filesystem:

```
using (ZipFile zip = ZipFile.Read(ExistingZipFile))

foreach (ZipEntry e in zip)

e.Extract(TargetDirectory, true); // overwrite

}

}
```

Extract an entry from the zip archive into a previously-opened stream:

```
Extract 3 © Copy
```

```
using (ZipFile zip = ZipFile.Read(ExistingZipFile))

ZipEntry e = zip["MyReport.doc"];
e.Extract(OutputStream);
}
```

Extract an Entry that was encrypted with a password, into the specified base directory:

```
using (ZipFile zip = ZipFile.Read(ExistingZipFile))
{
```

```
ZipEntry e = zip["TaxInformation-2008.xls"];
e.ExtractWithPassword(BaseDirectory, Password);
}
```

List all the entries in a zip file:

List 1 © Copy

```
using (ZipFile zip = ZipFile.Read(ExistingZipFile))
  {
2
    foreach (ZipEntry e in zip)
3
4
       if (header)
5
       {
6
         System.Console.WriteLine("Zipfile: {0}", zip.
7
         if ((zip.Comment != null) && (zip.Comment !=
8
           System.Console.WriteLine("Comment: {0}", zi
9
         System.Console.WriteLine("\n{1, -22} \{2, 8\} \{3\}
10
                                    "Filename", "Modifie
11
         System.Console.WriteLine(new System.String('-
12
         header = false;
13
14
       System.Console.WriteLine("{1,-22} {2,8} {3,5:F0
15
                                  e.FileName,
16
                                  e.LastModified.ToStrin
17
                                  e.UncompressedSize,
18
                                  e.CompressionRatio,
19
                                  e.CompressedSize,
20
                                  (e.UsesEncryption) ?
21
22
23
24 }
```

Read in zip archive content from a stream, and extract the content for

one entry to another stream. In this example, the filename "NameOfEntryInArchive.doc", refers only to the name of the entry within the zip archive. A file by that name is not created in the filesystem. The I/O is done strictly with the given streams.

Stream 1 © Copy

```
using (ZipFile zip = ZipFile.Read(InputStream))

z {
    zip.Extract("NameOfEntryInArchive.doc", OutputStream)}
}
```

Create a zip dynamically within an ASP.NET postback method, then download that zipfile to the requesting browser through Response.OutputStream. This works in DotNetZip v1.5 and later.

ASP.NET 1 © Copy

```
public void btnGo_Click (Object sender, EventArgs e
  {
2
    Response.Clear();
3
    String ReadmeText= "This is a zip file dynamicall
4
    string filename = System.IO.Path.GetFileName(List
5
    Response.ContentType = "application/zip";
6
    Response.AddHeader("content-disposition", "filena
7
8
    using (ZipFile zip = new ZipFile())
     {
10
      zip.AddFile(ListOfFiles.SelectedItem.Text, "fil
11
      zip.AddStringAsFile(ReadmeText, "Readme.txt",
12
      zip.Save(Response.OutputStream);
13
    }
14
15
    Response.End();
16
17 }
```

Create a zip with a single entry, obtaining the content for that entry from a string. Attach a comment to that entry. Specify the name of the zip file at the time of save.

Content from String

Copy

```
string content = ".....whatever...";
using (ZipFile zip = new ZipFile())

{
    ZipEntry e = zip.AddEntry("README.txt", content);
    e.Comment = "This entry in the zip archive was crezip.Save("archive-2008july12.zip");
}
```

Open an existing zip archive and modify it: update one entry, remove another, and rename a third. Update the comment on the archive as well.

Update a Zip □ Copy

```
1 using (ZipFile zip = ZipFile.Read("ExistingArchive.
2
  {
    ZipEntry e = zip["README.txt"];
3
    e.RemoveEntry();
5
    // for this entry, update the archive with conte
6
     zip.UpdateItem("Portfolio.doc");
7
8
    // update the filename of an entry
9
    e = zip["Table1.jpg"];
10
    e.FileName = "Figure1.jpg";
11
12
     zip.Comment = "This zip archive was updated " + S
13
     zip.Save();
14
15 | }
```

DotNetZip - VB.NET Examples

Here are a bunch of examples in VB.NET showing how to use the library.

There are also a few complete, working example applications shipped in the source code distribution.

Add items to a zip file:

```
Create a Zip
```

Copy

```
1
  Try
      Using zip As ZipFile = New ZipFile
2
           zip.AddFile("c:\photos\personal\7440-N49th.
3
           zip.AddFile("c:\Desktop\2005_Annual_Report.
4
           zip.AddFile("ReadMe.txt")
5
           zip.Save("MyZipFile.zip")
6
      End Using
7
  Catch ex1 As Exception
8
      Console.Error.WriteLine("exception: {0}", ex1.T
10 End Try
```

Extract items from a zip file:

Extract entries from a Zip

Copy

```
Try
Using zip As ZipFile = ZipFile.Read(ZipFileToEx
Dim e As ZipEntry
For Each e In zip
e.Extract
Next
```

```
The console of t
```

Extract all entries, and set the StatusMessageTextWriter so that verbose messages are generated:

ExtractAll, and use the status TextWriter

Copy

```
Using zip As ZipFile = ZipFile.Read(FilePath)
zip.StatusMessageTextWriter= System.Console.Out
'Status Messages will be sent to the console durin
zip.ExtractAll()
End Using
```

Add a few files to a zip file, specifying different passwords for different items:

Create a Zip, entries get passwords

♣ Copy

```
1
  Try
    Using zip As New ZipFile
2
       'the first entry is not protected by a password
3
      zip.AddFile("c:\datafiles\ReadMe.txt", "")
4
      zip.Password = "123456!"
5
      zip.AddFile("c:\photos\personal\7440-N49th.png"
6
      zip.Password= "!Secret1";
7
      zip.AddFile("c:\Desktop\2005_Annual_Report.pdf"
8
      zip.Save("Secret.zip")
9
    End Using
10
  Catch ex1 As System. Exception
11
    System.Console.Error.WriteLine("exception: {0}",
12
13 End Try
```

Add a few files to a zip file, using WinZip-compatible AES encryption on the entries:

Create a Zip, using AES encryption for the entries

♣ Copy

```
Try
1
    Using zip As New ZipFile
2
      zip.Password = "The.Silvertones.Box.Set!"
3
      zip.Encryption = EncryptionAlgorithm.WinZipAes2
4
      zip.AddFile("c:\datafiles\RawData-2008-12-20.cs
5
      zip.AddFile("c:\photos\personal\7440-N49th.png"
6
      zip.AddFile("c:\Desktop\2005_Annual_Report.pdf"
7
      zip.Save("AES-Encrypted-Secret.zip")
8
    End Using
9
  Catch ex1 As System. Exception
10
    System.Console.Error.WriteLine("exception: {0}",
11
12 End Try
```

Extract entries using a password:

Extract entries using a password

Copy

```
Using zip As new ZipFile(FilePath)
1
       Dim e As ZipEntry
2
       For Each e In zip
3
           If (e.UsesEncryption)
4
             e.ExtractWithPassword("Secret!")
5
           Else
6
             e.Extract
7
           End If
8
       Next
9
10 End Using
```

This example creates a zip using ZIP64 extensions. ZIP64 allows you

to exceed 4gb in a zip, or 65535 entries in a zip.

Create a Zip that uses ZIP64 extensions

```
Copy
```

```
Try
1
      Using zip As ZipFile = New ZipFile
2
           zip.UseZip64WhenSaving = Zip64Option.AsNece
3
           zip.AddFile("c:\datafiles\RawData-2009-02-1
4
           zip.AddFile("ReadMe.txt")
5
           zip.Save(String.Format("backup-{0}.zip", Da
6
      End Using
7
  Catch ex1 As Exception
8
      Console.Error.WriteLine("exception: {0}", ex1.T
9
10 End Try
```

Create a zip file, add a file, and also add an entry from a string. When the zip is unzipped, the content from the string will be inserted into the file "Readme.txt".

Create 3

```
Dim Content As String = "This string will be the cor
Using zip1 As ZipFile = New ZipFile
    zip1.AddEntry("Readme.txt", "This is the readme
    zip1.AddFile("MyDocuments\Resume.doc", "files")
    zip1.Comment = ("This zip file was created at "
    zip1.Save("Content.zip")
Find Using
```

Create a zip file, and add an entry taking content from a stream, like a MemoryStream or a FileStream.

```
1 Dim Content As String = "This string will be the cor
```

```
Using zip1 As ZipFile = New ZipFile
zip1.AddEntry("Readme.txt", stream)
zip1.AddFile("MyDocuments\Resume.doc", "files")
zip1.Comment = ("This zip file was created at "zip1.Save("Content.zip")
End Using
```

Read in a zip file, remove a few entries, save the file:

Modify a Zip
□ Copy

```
Dim sw As New System. IO. StringWriter
  Using zip As ZipFile = ZipFile.Read("PackedDocument
       Dim Threshold As New DateTime(2007, 7, 4)
3
       ' We cannot remove the entry from the list, wit
4
       ' an enumeration of said list.
5
       ' So we add the doomed entry to a list to be re
6
       ' pass 1: mark the entries for removal
7
       Dim MarkedEntries As New System.Collections.Gen
8
       Dim e As ZipEntry
9
       For Each e In zip
10
           If (e.LastModified < Threshold) Then</pre>
11
               MarkedEntries.Add(e)
12
           End If
13
       Next
14
         pass 2: actually remove the entry.
15
       Dim zombie As ZipEntry
16
       For Each zombie In MarkedEntries
17
           zip.RemoveEntry(zombie)
18
       Next
19
       zip.Comment = "This archive has been updated."
20
       zip.Save
21
22 End Using
```

Add a bunch of items, whether files or directories:

```
Dim itempaths As String() = _
1
     New String() { "c:\temp\Readme.txt", _
2
                     "MyProposal.docx", _
3
                     "SupportingFiles", _
4
                     "images\Image1.jpg" }
5
  Try
6
       Using zip As New ZipFile(ZipToCreate, Console.0
7
           Dim i As Integer
8
           For i = 1 To itempaths. Length - 1
9
               ' will add Files or Dirs, recursing and
10
               zip.AddItem(itempaths(i), "flat")
11
           Next i
12
           zip.Save
13
       End Using
14
  Catch ex1 As Exception
15
       Console.Error.WriteLine("exception: {0}", ex1.T
16
17 End Try
```

Create a self-extracting archive:

Create a self-extractor

♣ Copy

```
Dim DirectoryPath As String = "c:\Documents\Project7
Using zip As New ZipFile()
zip.AddDirectory(DirectoryPath, System.IO.Path.@zip.Comment = "This will be embedded into a self zip.SaveSelfExtractor("archive.exe", SelfExtract End Using
```

Update some entries in a Zip file:

Update some entries in a Zip

```
Using zip1 As New ZipFile
1
       ' the UpdateFile method works even if the entry
2
       ' Really it should be called "AddOrUpdateFile"
3
      zip1.UpdateFile("MyDocuments\Readme.txt", "")
4
      zip1.UpdateFile("CustomerList.csv", "")
5
      zip1.Comment = "This zip archive has been creat
6
      zip1.Save("Content.zip")
7
  End Using
8
9
  Using zip2 As ZipFile = ZipFile.Read("Content.zip")
10
      zip2.UpdateFile("Updates\Readme.txt", "")
11
      zip2.Comment = "This zip archive has been updat
12
      zip2.Save
13
14 End Using
```

Produce a zip file that contains embedded zip files.

Zip containing a Zip

```
Public Sub Run()
      Using s1 As Stream = ZipIntoMemory("c:\temp\dir
2
           Using s2 As Stream = ZipIntoMemory("c:\temp
3
               Using zip1 as New ZipFile
4
                   zip1.AddEntry("test1.zip", s1)
5
                   zip1.AddEntry("test2.zip", s2)
6
                     save to a file. Could also save
7
                   zip1.Save("Tescher.zip")
8
               End Using
9
           End Using
10
      End Using
11
  End Sub
12
13
  Public Function ZipIntoMemory(ByVal path As String)
      Dim ms As New MemoryStream
15
      Using zip1 as New ZipFile
16
           zip1.AddDirectory(path, "Result")
17
```

```
zip1.Save(ms)

End Using

move the stream position to the beginning
ms.Seek(0,SeekOrigin.Begin)
Return ms

End Function
```

DotNetZip - ASP.NET Example in C#

ASP.NET Example in C#

Here's an Example ASP.NET page, using C#, that dynamically creates a zip file and saves it to Response.OutStream. From the browser, the user will be prompted with the familiar download dialog box, allowing Open, Save, or Cancel of the generated zip file.

ASP.NET example in C#

```
< @ Page
      Language="C#"
2
      Debug="true" %>
3
4
 <%@ Import Namespace="System.Text" %>
 <%@ Import Namespace="System.IO" %>
7 < @ Import Namespace="Ionic.Zip" %>
 <%@ Import Namespace="System.Collections.Generic"</pre>
8
  <script language="C#" runat="server">
10
11
12 // ZipExample.aspx
13 //
14 // This .aspx page demonstrates how to use the Dot
15 //
16 // To run it,
17 // 1. drop the Ionic.Zip.dll into the \bin direct
18 // 2. create a subdirectory called "fodder" in yo
      3. copy into that directory a variety of rando
19 //
      4. insure your web.config is properly set up (
20 //
21 //
22 //
23 // notes:
```

```
This requies the .NET Framework 3.5 - because
24 //
       new for ASP.NET in the .NET Framework v3.5.
25 //
26 //
       To use this control, you must add the new web
27 //
       IF you build your app in Visual Studio, this i
28 //
       here's an example web.config that works with t
29 //
30 //
         <configuration>
31 //
32 //
           <system.web>
             <trust level="Medium" />
33 //
             <compilation defaultLanguage="c#" />
34 //
35 //
             <pages>
36 //
               <controls>
                 <add tagPrefix="asp" namespace="Syst
37 //
38 //
               </controls>
39 //
             </pages>
           </system.web>
40 //
           <system.codedom>
41 //
42 //
             <compilers>
               <compiler language="c#;cs;csharp" exte</pre>
43 //
                 oroviderOption name="CompilerVersio"
44 //
                 oroviderOption name="WarnAsError" v
45 //
46 //
               </compiler>
             </compilers>
47 //
48 //
           </system.codedom>
49 //
         </configuration>
50 //
51 //
  //
52
53
54
  public String width = "100%";
55
56
  public void Page_Load (Object sender, EventArgs e)
57
  {
58
       try
59
       {
60
```

```
if ( !Page.IsPostBack ) {
61
               // populate the dropdownlist
62
               // must have a directory called "fodde
63
               String homeDir = Server.MapPath(".");
64
               String sMappedPath= Server.MapPath("fo
65
66
               var fqFilenames= new List<String>(Syst
67
               var filenames= fqFilenames.ConvertAll(
68
69
               ErrorMessage.InnerHtml = "";
70
71
               FileListView.DataSource = filenames;
72
               FileListView.DataBind();
73
           }
74
75
76
       catch (Exception)
77
78
           // Ignored
79
       }
80
  }
81
82
83
  public void btnGo_Click (Object sender, EventArgs
84
  {
85
       ErrorMessage.InnerHtml =""; // debugging onl
86
       var filesToInclude= new System.Collections.Gen
87
       String sMappedPath= Server.MapPath("fodder");
88
       var source= FileListView.DataKeys as DataKeyAr
89
90
       foreach (var item in FileListView.Items)
91
       {
92
           CheckBox chkbox= item.FindControl("include")
93
           Label lbl= item.FindControl("label") as La
94
95
           if (chkbox!=null && lbl != null)
96
           {
97
```

```
(chkbox.Checked)
98
                 {
99
                     ErrorMessage.InnerHtml += String.F
100
101
                     filesToInclude.Add(System.IO.Path.
102
                 }
103
            }
104
        }
105
106
           (filesToInclude.Count==0)
107
        {
108
            ErrorMessage.InnerHtml += "You did not sel
109
110
        }
111
        else
112
        {
113
            Response.Clear();
114
115
            System.Web.HttpContext c= System.Web.HttpC
116
            String ReadmeText= String.Format("README.T
117
                                                  "This is
118
                                                  "by an AS
119
                                                  "The serv
120
                                                  System.Da
121
                                                  System.En
122
                                                  c.Request
123
124
                                                  );
            string archiveName= String.Format("archive
125
            Response.ContentType = "application/zip";
126
            Response. AddHeader ("content-disposition",
127
128
            using (ZipFile zip = new ZipFile())
129
            {
130
                 foreach (var f in filesToInclude)
131
                 {
132
                     zip.AddFile(f, "files");
133
                 }
134
```

```
zip.AddEntry("Readme.txt", ReadmeText)
135
                 zip.Save(Response.OutputStream);
136
137
            Response.Close();
138
        }
139
140
   }
141
142
   </script>
143
144
145
146
   <html>
147
      <head>
148
        <link rel="stylesheet" href="style/basic.css">
149
      </head>
150
151
      <body>
152
153
        <form id="Form" runat="server">
154
155
          <h3> <span id="Title" runat="server" />Zip F
156
157
          This page uses the .NET Zip library
158
          (see <a href="http://www.codeplex/com/DotNet"
159
           to dynamically create a zip archive, and the
160
161
          <span class="SampleTitle"><b>Check the boxes
162
          <br/>
163
          <br/>
164
          <asp:Button id="btnGo" Text="Zip checked file"
165
166
          <br/>
167
          <br/>
168
          <span style="color:red" id="ErrorMessage" ru</pre>
169
          <br/>
170
171
```

```
<asp:ListView ID="FileListView" runat="serve</pre>
172
173
          <LayoutTemplate>
174
            175
              176
            177
          </LayoutTemplate>
178
179
          <ItemTemplate>
180
            181
              <asp:Checkbox ID="include" runat="
182
              <asp:Label id="label" runat="serve"
183
            184
          </ItemTemplate>
185
186
          <EmptyDataTemplate>
187
            <div>Nothing to see here...</div>
188
          </EmptyDataTemplate>
189
190
         </asp:ListView>
191
192
193
       </form>
194
195
     </body>
196
197
198 </html>
```

DotNetZip - ASP.NET VB Example

ASP.NET Example in VB

Here's an Example ASP.NET page, using VB, that dynamically creates a zip file and saves it to Response.OutStream. From the browser, the user will be prompted with the familiar download dialog box, allowing Open, Save, or Cancel of the generated zip file.

ASP.NET example in VB

```
< @ Page
       Language="VB"
2
       Debug="true"
3
  %>
4
5
  <%@ Import Namespace="System.Text" %>
7 <%@ Import Namespace="System.IO" %>
 <%@ Import Namespace="Ionic.Zip" %>
  <%@ Import Namespace="System.Collections.Generic"</pre>
10
  <script language="VB" runat="server">
11
12
   ' ZipExample.aspx
13
14
   ' This .aspx page demonstrates how to use the DotN
15
16
    To run it,
17
      1. drop the Ionic.Zip.dll into the \bin directo
18
      2. create a subdirectory called "fodder" in you
19
      3. copy into that directory a variety of random
20
      4. insure your web.config is properly set up (S
21
22
23
```

```
notes:
24
      This requies the .NET Framework 3.5 - because i
25
      new for ASP.NET in the .NET Framework v3.5.
26
27
      To use this control, you must add the new web c
28
      Here's an example web.config that works with th
29
30
        <configuration>
31
          <system.web>
32
            <trust level="Medium" />
33
            <compilation defaultLanguage="c#" />
34
            <pages>
35
              <controls>
36
                <add tagPrefix="asp" namespace="Syste</pre>
37
                      assembly="System.Web.Extensions,
38
              </controls>
39
            </pages>
40
          </system.web>
41
          <system.codedom>
42
            <compilers>
43
              <compiler language="c#;cs;csharp"</pre>
44
                     extension=".cs"
45
                     warningLevel="4"
46
                     type="Microsoft.CSharp.CSharpCode
47
                oroviderOption name="CompilerVersion
48
                oviderOption name="WarnAsError" va
49
              </compiler>
50
51
              <compiler language="vb;vbs;visualbasic;</pre>
52
                         extension=".vb"
53
                         warningLevel="4"
54
                         type="Microsoft.VisualBasic.V
55
                oroviderOption name="CompilerVersion
56
                option name="OptionInfer" va
57
                oviderOption name="WarnAsError" va
58
              </compiler>
59
60
```

```
</system.codedom>
61
        </configuration>
62
63
64
65
66
67
  Dim width as String = "100%"
68
69
  Public Sub Page_Load (ByVal sender As Object, ByVal
70
       Try
71
           If Not ( Page.IsPostBack ) Then
72
                 populate the dropdownlist
73
                 must have a directory called "fodder
74
               Dim homeDir as String = Server.MapPath
75
               Dim sMappedPath as String= Server.Map
76
77
               Dim fqFilenames As New List(Of String)
78
79
               Dim filenames as List(Of String) = _
80
                    fqFilenames.ConvertAll (Function(s
81
82
               ErrorMessage.InnerHtml = ""
83
84
               FileListView.DataSource = filenames
85
               FileListView.DataBind()
86
           End If
87
88
       Catch
89
             Ignored
90
       End Try
91
92
93 End Sub
94
95
96
97 Public Sub btnGo_Click (ByVal sender As System.Obj
```

```
98
       99
       Dim filesToInclude as New System.Collections.G
100
       Dim sMappedPath as String= Server.MapPath("fod
101
       Dim source As DataKeyArray= FileListView.DataK
102
103
       For Each item As ListViewDataItem in FileListV
104
105
           Dim chkbox As CheckBox= CType(item.FindCon
106
           Dim lbl As Label = CType(item.FindControl(
107
108
           If Not (chkbox Is Nothing
                                       0R
                                           lbl Is Noth
109
               If (chkbox.Checked) Then
110
                   ErrorMessage.InnerHtml = ErrorMess
111
                            String.Format("adding file
112
                   filesToInclude.Add(System.IO.Path.
113
               End If
114
           End If
115
       Next
116
117
       If (filesToInclude.Count=0) Then
118
           ErrorMessage.InnerHtml = ErrorMessage.Inne
119
       Else
120
           Response.Clear
121
           Response.BufferOutput= false
122
123
           Dim enc as Ionic.Zip.EncryptionAlgorithm =
124
           If (chkUseAes.Checked) Then
125
               enc = EncryptionAlgorithm.WinZipAes256
126
           End If
127
128
129
           Dim c As System.Web.HttpContext = System.W
           Dim ReadmeText As String= String.Format("R
130
                                              "This is
131
                                              "by an AS
132
                                              "The serv
133
                                              "The pass
134
```

```
"Encrypti
135
                                                 System.Da
136
                                                 System.En
137
138
                                                 c.Request
                                                 tbPasswor
139
                                                 enc.ToStr
140
            Dim archiveName as String= String.Format("
141
            Response.ContentType = "application/zip"
142
            Response. AddHeader ("content-disposition",
143
144
            Using zip as new ZipFile()
145
                   the Readme.txt file will not be pass
146
                 zip.AddEntry("Readme.txt", ReadmeText,
147
                 If Not String.IsNullOrEmpty(tbPassword
148
                     zip.Password = tbPassword.Text
149
                     zip.Encryption = enc
150
                 End If
151
152
                 ' filesToInclude is a string[] or List
153
                 zip.AddFiles(filesToInclude, "files")
154
155
                 zip.Save(Response.OutputStream)
156
            End Using
157
            Response.Close
158
159
        End If
160
161
   End Sub
162
163
164
   </script>
165
166
167
168
   <html>
169
      <head>
170
        <link rel="stylesheet" href="style/basic.css">
171
```

```
</head>
172
173
     <body>
174
175
       <form id="Form" runat="server">
176
177
         <h3> <span id="Title" runat="server" />Zip F
178
179
         This page uses the .NET Zip library (see
180
         href="http://DotNetZip.codeplex.com">http://
181
         to dynamically create a zip archive, and the
182
         browser through Response.OutputStream.
183
         in VB.NET.
184
185
         <span class="SampleTitle"><b>Check the boxes
186
         then click the button to zip them up.</b></s
187
         <br/>br/>
188
         <br/>
189
         Password: <asp:TextBox id="tbPassword" Passw
190
         <span style="color:Red">(Optional)</span>
191
         <br/>
192
         <br/>br/>
193
         Use AES?: <asp:CheckBox id="chkUseAes" AutoP
194
         <br/>
195
         <br/>
196
         <asp:Button id="btnGo" Text="Zip checked fil
197
198
         <br/>
199
         <br/>
200
         <span style="color:red" id="ErrorMessage" ru</pre>
201
         <br/>
202
203
         <asp:ListView ID="FileListView" runat="serve</pre>
204
205
           <LayoutTemplate>
206
              207
                208
```

```
209
            </LayoutTemplate>
210
211
            <ItemTemplate>
212
              213
                <asp:Checkbox ID="include" runat="
214
                <asp:Label id="label" runat="serve"
215
              216
            </ItemTemplate>
217
218
            <EmptyDataTemplate>
219
              <div>Nothing to see here...</div>
220
            </EmptyDataTemplate>
221
222
         </asp:ListView>
223
224
225
       </form>
226
227
     </body>
228
229
230 </html>
```

You can use DotNetZip from Powershell

The programming model is the same as for VB or C#, with a few syntactic changes for Powershell. This page will show some examples.

Create a zip file

This example just creates a simple zipfile

Create a Zip archive

```
Copy
```

```
# Load the assembly - replace this with your own pat
[System.Reflection.Assembly]::LoadFrom("c:\\dinoch\\

**SdirectoryToZip = "c:\\temp";

**Szipfile = new-object Ionic.Zip.ZipFile;

**Se= $zipfile.AddEntry("Readme.txt", "This is a zipfi

**Se= $zipfile.AddDirectory($directoryToZip, "home")

**Szipfile.Save("ZipFiles.ps1.out.zip");

**Szipfile.Dispose();
```

Create a zip file using AES encryption

This example creates a zipfile, using AES 256-bit encryption to encrypt the entries.

Create an encrypted zip

```
♣ Copy
```

```
1 # Load the assembly - replace this with your own pa
2 [System.Reflection.Assembly]::LoadFrom("c:\\dinoch\
```

```
$ $directoryToZip = "c:\\dinoch\\dev\\powershell"
$ $zipfile = new-object Ionic.Zip.ZipFile
$ $zipfile.Encryption = [Ionic.Zip.EncryptionAlgorith
$ $zipfile.Password = "Albatros$"

$ $e= $zipfile.AddEntry("Readme.txt", "This is a zipf
$ $e= $zipfile.AddSelectedFiles("name != *.zip", $dir
$ $zipfile.Save("ZipFiles.ps1.out.zip")
$ $zipfile.Dispose()
```

Create a number of zip files, each containing a single file

This example iterates through a set of files, and creates a zipfile containing a single file, for each one.

Create a set of Zipfiles

```
1 # Load the assembly - replace this with your own pa
  [System.Reflection.Assembly]::LoadFrom("c:\\dinoch\
3
  foreach ($file in gci $LogPath -filter $FileType -r
4
           where {\$_. LastWriteTime - lt [DateTime]:: Now.
5
  {
6
       $FileName = $File.FullName
7
       $ZipName = $File.FullName + ".zip"
8
       $zip = new-object Ionic.Zip.ZipFile
9
       $zip.AddFile($FileName, "");
10
       $zip.Save($ZipName)
11
       $zip.Dispose()
12
13 }
```

DotNetZip can be used from COM Environments

You can use DotNetZip from COM environments, via an IDispatch (late bound) interface. This means you can call into the DotNetZip library (the Ionic.Zip.dll assembly) from programming environments like PHP, Perl, Javascript, and VBScript (including old-school ASP pages), among others.

If you download and install the DotNetZip Runtime MSI package, it will set up DotNetZip for use with COM.

If for some reason you prefer to not use the MSI package, then you will need to perform some manual steps to enable DotNetZip for COM usage.

- 1. open a CMD.exe prompt.
- 2. Install into the GAC: gacutil -i Ionic.Zip.dll
- 3. Register for COM use: regasm Ionic.Zip.dll

Notes: You will need to be local administrator in order to perform these steps. You need to perform those steps just once, on each computer where COM applications will run that use DotNetZip. The gacutil and regasm tools are included in the .NET Framework SDK.

Using the COM interface to DotNetZip

Via the COM exposure, applications written in COM-capable environments like old-school ASP using VBScript or Javascript, VBScript or Javascript running in Windows Script Host, PHP, Perl, and others, can create instances of the ZipFile class, add entries, and save archives, as any .NET application could. Most of DotNetZip's advanced features are available to these COM environments, including ZIP64 support, self-extracting archives, password-protected zip files, AES Encryption, spanned Zip files, and Unicode support. Beyond creating zips, COM environments can also read zips, modify them, or extract files from zip files.

Some of the advanced features of DotNetZip are not available through COM, including eventing, and streaming.

IDispatch

The key classes in DotNetZip that are exposed to COM are: ZipFile, ZipEntry, and the various ZipExceptions. These are all exposed via IDispatch interfaces - late binding only. (For those familiar with .NET, DotNetZip is decorated with ClassInterfaceType.AutoDispatch.) There is no explicit typelib exposed by DotNetZip.

COM supports calling instance methods and properties on .NET classes; COM does not support calling static methods on .NET objects via interop. To allow the various static methods on the ZipFile class, such as ZipFile.CheckZip() and ZipFile.IsZipFile(), to be called from COM, DotNetZip includes a ComHelper class. Instead of calling ZipFile.IsZipFile(), which is inaccessible to COM clients because it is a static method, COM clients should call

Ionic.Zip.ComHelper.IsZipFile(). Use ComHelper as you would, any other class. The methods on the class are documented as with all the other classes in DotNetZip. If you are not programming DotNetZip from COM, you don't need the ComHelper class.

Overloaded Methods

COM does not directly support calling overloaded methods. In a .NET assembly exposed to COM via interop, only the simplest method in a method group is directly available to COM callers. For example, consider ZipFile.AddFile. There are two overloads. Only the overload that accepts a single string will be accessible via the name "AddFile" to COM clients.

In most cases the thing you need to accomplish is achievable anyway. In the AddFile() case, setting the FileName on the entry after calling AddFile() will do the trick.

Also, the overloaded methods *are available*, via "mangled" names: each successive overload gets a numeric suffix. Consider the ZipFile.ExtractSelectedEntries method group; there are 5 overloads. The first, simplest, is available via the name ExtractSelectedEntries. Then, additional overloads are available via ExtractSelectedEntries_2, ExtractSelectedEntries_3, and so on. The same is true for other overloaded methods on the ZipFile and ZipEntry classes.

For the ZipFile class:

Method Overload	IDispatch name
AddItem(string)	AddItem
AddItem(string, string)	AddItem_2
AddFile(string)	AddFile

AddFile(string, string)	AddFile_2
UpdateFile(string)	UpdateFile
UpdateFile(string, string)	UpdateFile_2
UpdateDirectory(string)	UpdateDirectory
UpdateDirectory(string, string)	UpdateDirectory_2
UpdateItem(string)	UpdateItem
UpdateItem(string, string)	UpdateItem_2
AddEntry(string,string)	AddEntry
AddEntry(string,string,Encoding)	AddEntry_2
AddEntry(string,Stream)	AddEntry_3
AddEntry(string,byte[])	AddEntry_4
UpdateEntry(string,string)	UpdateEntry
UpdateEntry(string,string,Encoding)	UpdateEntry_2
UpdateEntry(string,Stream)	UpdateEntry_3
UpdateEntry(string,byte[])	UpdateEntry_4
AddDirectory(string)	AddDirectory
AddDirectory(string,string)	AddDirectory_2
RemoveEntry(ZipEntry)	RemoveEntry
RemoveEntry(string)	RemoveEntry_2
ExtractAll(string)	ExtractAll
ExtractAll(string, bool)	ExtractAll_2
ExtractAll(string, ExtractExistingFileAction)	ExtractAll_3
Save()	Save
Save(string)	Save_2
Save(Stream)	Save_3
AddSelectedFiles(string)	AddSelectedFiles
AddSelectedFiles(string, bool)	AddSelectedFiles_2
AddSelectedFiles(string, string)	AddSelectedFiles_3
AddSelectedFiles(string, string, bool)	AddSelectedFiles_4
AddSelectedFiles(string, string, string)	AddSelectedFiles_5

AddSelectedFiles(string, string, string, bool)	AddSelectedFiles_6
RemoveSelectedEntries(string)	RemoveSelectedEntries
AddSelectedFiles(string, string)	RemoveSelectedEntries_2
ExtractSelectedEntries(string)	ExtractSelectedEntries
ExtractSelectedEntries(string, ExtractExistingFileAction)	ExtractSelectedEntries_2
ExtractSelectedEntries(string, string)	ExtractSelectedEntries_3
ExtractSelectedEntries(string, string, string)	ExtractSelectedEntries_4
ExtractSelectedEntries(string, string, string, ExtractExistingFileAction)	ExtractSelectedEntries_5
SaveSelfExtractor(string, SelfExtractorFlavor)	SaveSelfExtractor
SaveSelfExtractor(string, SelfExtractorFlavor, string)	SaveSelfExtractor_2
SaveSelfExtractor(string, SelfExtractorFlavor, string, string)	SaveSelfExtractor_3

For the ZipEntry class:

Method Overload	IDispatch name
Extract()	Extract
Extract(ExtractExistingFileAction)	Extract_2
Extract(Stream)	Extract_3
Extract(string)	Extract_4
Extract(string, ExtractExistingFileAction)	Extract_5
ExtractWithPassword(string)	ExtractWithPassword
ExtractWithPassword(string,string)	ExtractWithPassword_2
ExtractWith Password (Extract Existing File Action, string)	ExtractWithPassword_3
ExtractWithPassword(string, ExtractExistingFileAction, string)	ExtractWithPassword_4
ExtractWithPassword(Stream, string)	ExtractWithPassword_5

The bad news is that the mappings between names and actual methods will change over releases of DotNetZip. This means depending on these names is a bit fragile, but it will work.

Destroy the ZipFile object

After you call .Dispose() on the ZipFile object, you should destroy it completely. In VBSCript, this is done by setting the object reference to Nothing.

The usage model for the ZipFile object

After you initialize a ZipFile object, you can't reuse the same object instance for another zip file. If for example, you have a set of 6 zip files, and you want to read and extract them in a loop, then you'll need to instantiate a new ZipFile object to read each one. You should call Dispose() on each one when you're finished, and then set it to null/Nothing.

For enumerations

DotNetZip exposes various enumerations to specify behavior in the interface - for example, the Ionic.Zlib.CompressionLevel enum, and the Ionic.Zip.ExtractExistingFileAction enum. Enums are not available to IDispatch clients. COM applications that need to use enum values can simply use the integer equivalent for those enums. In most cases those integer values are specified in the documentation for the enum. For example, for Ionic.Zip.ExtractExistingFileAction, the value of Throw is 0, while OverwriteSilently is 1.

Some examples follow.

Using DotNetZip in PHP

This is a PHP script that dynamically creates a ZIP file on the server, then downloads it to the requesting client. The Zip archive will use 256-bit AES encryption.

Create a Zip

```
1 $fname = "zip-generated-from-php-" . date('Y-m-d-Hi
  $zipOutput = "c:\\temp\\" . $fname;
3 $zip = new COM("Ionic.Zip.ZipFile");
4 $zip->Name = $zipOutput;
  $dirToZip= "c:\\temp\\psh";
6 $zip->Encryption = 3;
7 $zip->Password = "AES-Encryption-Is-Secure";
  $zip->AddDirectory($dirToZip);
9 $zip->Save();
10 $zip->Dispose();
11
  if (file_exists($zipOutput))
12
13
    header('Cache-Control: no-cache, must-revalidate'
14
    header('Content-Type: application/x-zip');
15
    header('Content-Disposition: attachment; filename
16
    header('Content-Length: ' . filesize($zipOutput))
17
    readfile($zipOutput);
18
    unlink($zipOutput);
19
20 }
```

Using DotNetZip in Javascript

This example dynamically creates a zipfile, using AES 256-bit encryption.

Create a Zip

```
var filename = "C:\\temp\\ZipFile-created-from-java
2
3 try
4 {
      WScript.echo("Instantiating a ZipFile object...
5
       var zip = new ActiveXObject("Ionic.Zip.ZipFile"
6
7
      WScript.echo("setting the encryption...");
8
       // 3 = AES256, 2 = AES128, 1 = PKZIP, 0 = none
9
       zip.Encryption = 3;
10
11
       WScript.echo("setting the password...");
12
       zip.Password = "This is the Password.";
13
14
       WScript.echo("adding a selection of files...");
15
       zip.AddSelectedFiles("*.js");
16
       zip.AddSelectedFiles("*.vbs");
17
18
       WScript.echo("setting the save name...");
19
       zip.Name = filename;
20
21
       WScript.echo("Saving...");
22
       zip.Save();
23
24
       WScript.echo("Disposing...");
25
       zip.Dispose();
26
27
      WScript.echo("Done.");
28
```

This example lists the entries in a zipfile.

Listing entries in a Zip archive

```
var filename = "C:\\temp\\CompressedData.zip";
1
2
3 try
  {
4
       WScript.echo("Instantiating a ZipFile object...
5
       var zip = new ActiveXObject("Ionic.Zip.ZipFile"
6
7
       WScript.echo("Initialize (Read)...(" + filename
8
       zip.Initialize(filename);
9
10
       WScript.echo("listing entries...");
11
       var e = new Enumerator(zip);
12
       for (; !e.atEnd(); e.moveNext())
13
14
           var entry= e.item();
15
           WScript.Echo (" " + entry.FileName);
16
       }
17
18
       WScript.echo("Disposing...")
19
       zip.Dispose();
20
21
       WScript.echo("Done.");
22
23
  catch (e2)
24
  {
25
       WScript.echo(e2.number + ": " + e2.name);
26
```

```
WScript.echo(e2.message);
28 }
```

This example checks a ZipFile using the ComHelper class.

Checking a file to see if it is a Zip archive

```
function checkZip(filename)
2
  {
       var obj = new ActiveXObject("Ionic.Zip.ComHelpe
3
       return obj.IsZipFile(filename);
  }
5
  function checkZipWithExtract(filename)
8
       var obj = new ActiveXObject("Ionic.Zip.ComHelpe
       return obj.IsZipFileWithExtract(filename);
10
11
12
  function main()
13
14
       var result;
15
       var args = WScript.Arguments;
16
17
       if (args.Length == 1)
18
       {
19
           result = checkZip(args(0));
20
21
       else if (args.Length == 2 \&\& args(0) == "-x")
22
23
           result = checkZipWithExtract(args(1));
24
25
       else
26
       {
27
           WScript.Echo("TestCheckZip.js - check a zip
28
           WScript.Echo(" usage: TestCheckZip.js [-x]
29
           WScript.Quit(1);
30
```

Using DotNetZip in VBScript

This example creates a zipfile, using AES 256-bit encryption.

Create a Zip

```
1 dim filename
2 filename = "C:\temp\ZipFile-created-from-VBScript.z
4 WScript.echo("Instantiating a ZipFile object...")
5 dim zip
  set zip = CreateObject("Ionic.Zip.ZipFile")
7
  WScript.echo("Setting the encryption...")
8
  ' 3=AES256, 2=AES128, 1=PKZIP, 0=none
  zip.Encryption = 3
11
  WScript.echo("setting the password...")
  zip.Password = "This is the Password."
13
14
  WScript.echo("adding a selection of files...")
15
  zip.AddSelectedFiles "*.js"
  zip.AddSelectedFiles "*.vbs"
17
18
  WScript.echo("setting the save name...")
19
  zip.Name = filename
20
21
  WScript.echo("Saving...")
  zip.Save
23
24
  WScript.echo("Disposing...")
  zip.Dispose
26
27
  zip = Nothing
28
29
```

```
30 WScript.echo("Done.")
```

This example extracts all entries from a zipfile.

Extract all entries from a Zip

Copy

```
1 WScript.echo("Instantiating a ZipFile object...")
2 dim zip
  set zip = CreateObject("Ionic.Zip.ZipFile")
  WScript.echo("Initialize (Read)...")
5
  zip.Initialize("CompressedData.zip")
6
7
8 WScript.echo("setting the password for extraction...
  zip.Password = "This is the Password."
10
  WScript.echo("extracting all files...")
11
  zip.ExtractAll "DotNetZip-extract"
13
14 WScript.echo("Disposing...")
  zip.Dispose
15
16
17 WScript.echo("Done.")
```

This example lists the entries in a zipfile, and extracts some of them.

List and Maybe extract entries from a Zip

□ Copy

```
WScript.echo("")
Dim zip
WScript.echo("Instantiating a ZipFile object...")
Set zip = CreateObject("Ionic.Zip.ZipFile")
WScript.echo("Initialize (Read)...")
zip.Initialize filename

Set fso = CreateObject("Scripting.FileSystemObject"
```

```
If Not fso.FolderExists("unpack") Then
      fso.CreateFolder("unpack")
11
  End If
12
13
  ' Any call to ZipEntry.Extract() will put files int
14
  ' current working directory. So set it here:
15
  Set objShell = CreateObject("Wscript.Shell")
  objShell.CurrentDirectory = "unpack"
17
18
  WScript.echo("listing...")
19
  For Each entry in zip
20
     WScript.echo(" " & entry.FileName)
21
     ext = Right(entry.FileName, 4)
22
     If (ext = ".vbs") Then
23
           set password for extraction if necessary
24
         entry.Password = "This is the Password."
25
         entry.Extract
26
     End If
27
  Next
28
29
  WScript.echo("Disposing...")
  zip.Dispose
31
32
33 WScript.echo("Done.")
```

This example extracts selected entries from a zip archive, via the ExtractSelectedEntries overload. It uses the mangled name to access the method, as described above.

Extract selected entries from a Zip

```
Sub extractZip()
If Not (extractLocation = "") Then
Dim OverwriteSilently
OverwriteSilently = 1
Dim zip
```

```
Set zip = CreateObject("Ionic.Zip.ZipFile")
7
8
         zip.Initialize filename
9
10
         If password <> "" Then
11
           zip.Password = password
12
         End If
13
14
         zip.ExtractSelectedEntries_5 "name = *.xml",
15
16
         zip.Dispose
17
         zip = Nothing
18
       End If
19
20
21 End Sub
```

You can use DotNetZip from C++/CLI

A program written in C++/CLI can take advantage of any managed library. It's easy to use DotNetZip from a C++/CLI application. This page will show some examples.

Using C++/CLI, the key difference from VB and C#, is that there is no **using** statement in C++. C++ applications need to surround the use of the ZipFile class with a try..catch.. and call the ZipFile destructor, or call delete, in the finally clause.

Create a zip file

This example just creates a simple zipfile. It uses the destructor.

Create a Zip archive - destructor

```
using namespace System;
  using namespace Ionic::Zip;
3
  int main(array<System::String ^> ^args)
  {
5
       Console::WriteLine(L"Hello World");
6
7
       ZipFile ^ zip;
8
       try
9
       {
10
           zip = gcnew ZipFile();
11
           zip->AddEntry("Readme.txt", "This is the co
12
           zip-&gtAddFile("CreateZipFile.cpp");
13
           zip->Save("test.zip");
14
       }
15
```

```
finally
finally
{
    zip->~ZipFile();
}

Console::WriteLine(L"Press <ENTER> to quit.");
Console::ReadLine();
return 0;
}
```

This alternative uses the C++ deleee syntax:

Create a Zip archive - delete

```
using namespace System;
  using namespace Ionic::Zip;
3
  int main(array<System::String ^> ^args)
  {
5
       Console::WriteLine(L"Hello World");
6
7
       ZipFile ^ zip;
8
       try
9
       {
10
           zip = gcnew ZipFile();
11
           zip->AddEntry("Readme.txt", "This is the co
12
           zip-&gtAddFile("CreateZipFile.cpp");
13
           zip->Save("test.zip");
14
15
       finally
16
       {
17
           delete zip;
18
       }
19
20
       Console::WriteLine(L"Press <ENTER> to quit.");
21
       Console::ReadLine();
22
       return 0;
23
```

Building C++/CLI program that uses DotNetZip

Build a C++/CLI program that uses DotNetZip, just as you would build any C++/CLI program.

The easiest way is to use Visual Studio, and create a C++/CLI project. Right click on the project and select References.... Add a reference to the Ionic.Zip.dll assembly. Click OK, then build your application.

You can also build using command-line tools. To do this, you will need to compile using the c++ source using the cl.exe tool, specifying the /clr option, and specifying where to find the Ionic.Zip.dll assembly. A typical series of steps to build a simple C++/CLI program that uses DotNetZip from one source file, supposing the name of the source file is CreateZipFile.cpp, is:

```
\vc9\bin\cl.exe /Od /FD /EHa /MDd /Fo".\\" -I\vc9\include
/FU "c:\Windows\Microsoft.NET\Framework\v2.0.50727\Sys*
/FU Ionic.Zip.dll CreateZipFile.cpp
```

```
\vc9\bin\link.exe /OUT:CreateZipFile.exe /DEBUG /ASSEMBLYI
/MANIFEST /MANIFESTFILE:"CreateZipFile.exe.intermediate
/MANIFESTUAC:"level='asInvoker' uiAccess='false'"
/PDB:CreateZipFile.pdb /DYNAMICBASE /FIXED:No /NXCOMPA'
/LIBPATH:\vc9\lib /LIBPATH:\winsdk\lib CreateZipFile.ol
```

c:\netsdk2.0\Bin\mt.exe /outputresource:"CreateZipFile.exe
/manifest CreateZipFile.exe.intermediate.manifest

Create a zip file using AES encryption

This example creates a zipfile, using AES 128-bit encryption to encrypt the entries.

Create an encrypted zip

Copy

```
#include "stdafx.h"
1
2
3 using namespace System;
  using namespace Ionic::Zip;
5
  int main(array<System::String ^> ^args)
6
7
  {
       Console::WriteLine(L"Hello World");
8
9
       ZipFile ^ zip;
10
       try
11
       {
12
           zip = gcnew ZipFile();
13
           zip->Password = verySecret;
14
           zip->Encryption = EncryptionAlgorithm::WinZ
15
           zip->AddEntry("Readme.txt", "This is the co
16
           zip->AddFile("Data.csv");
17
           zip->Save("test.zip");
18
19
       finally
20
21
           zip->~ZipFile();
22
       }
23
24
       Console::WriteLine(L"Press <ENTER> to quit.");
25
       Console::ReadLine();
26
       return 0;
27
28 }
```

Use a SaveProgress event from C++

This example creates a zipfile, and uses a SaveProgress event.

Create an encrypted zip

♣ Copy

```
1 #include "stdafx.h"
2
3 using namespace System;
4 using namespace System::IO;
5 using namespace Ionic::Zip;
6
7
8 public ref class DnzHelloCppCli
9 {
10
  private:
11
       bool justHadByteUpdate;
12
13
14 public:
       DnzHelloCppCli()
15
16
           {
           }
17
18
19 public:
       void Run()
20
           {
21
               Console::WriteLine(L"Hello World");
22
               Console::WriteLine("Using DotNetZip ve
23
               array<String^>^ filesToAdd = System::I
24
25
               ZipFile ^ zip;
26
               try
27
                {
28
                    zip = gcnew ZipFile();
29
                    zip->Password = "Harbinger";
30
                    zip->Encryption = EncryptionAlgori
31
                    zip->SaveProgress += gcnew EventHa
32
                    zip->AddEntry("Readme.txt", "This
33
```

```
zip->AddFiles(filesToAdd, "files")
34
                    zip->Save("MyArchive.zip");
35
                }
36
                finally
37
                {
38
                    zip->~ZipFile();
39
                }
40
41
                Console::WriteLine(L"Press <ENTER> to
42
                Console::ReadLine();
43
                return;
44
           }
45
46
  public:
47
       void SaveProgress(Object^ sender, SaveProgress
48
           {
49
                switch (e->EventType)
50
                {
51
                    case ZipProgressEventType::Saving_
52
                    {
53
                         Console::WriteLine("Saving: {0
54
                         break;
55
56
                    case ZipProgressEventType::Saving_
57
58
                         if (this->justHadByteUpdate)
59
                         {
60
                             Console::WriteLine();
61
62
                         Console::WriteLine(" Writing:
63
                                              e->CurrentE
64
                                              (e->Entries
65
                                              e->EntriesT
66
                         this->justHadByteUpdate = fals
67
                         break;
68
                    }
69
                    case ZipProgressEventType::Saving_
70
```

```
{
71
                          if (e->CurrentEntry->InputStre
72
                          {
73
                              e->CurrentEntry->InputStre
74
                              e->CurrentEntry->InputStre
75
76
                          break;
77
                     }
78
                     case ZipProgressEventType::Saving_
79
                     {
80
                          this->justHadByteUpdate = fals
81
                          Console::WriteLine();
82
                          Console::WriteLine("Done: {0}"
83
                          break;
84
                     }
85
                     case ZipProgressEventType::Saving_
86
                     {
87
                          if (this->justHadByteUpdate)
88
                          {
89
                              Console::SetCursorPosition
90
91
                          Console::Write("
                                                  {0}/{1} (
92
                                           e->BytesTransfe
93
                                           e->TotalBytesTo
94
                                           (((double) e->B
95
                          this->justHadByteUpdate = true
96
                          break;
97
                     }
98
                 }
99
100
101
   };
102
103
104
   int main(array<System::String ^> ^args)
105
   {
106
        try
107
```

```
{
108
            DnzHelloCppCli^ me = gcnew DnzHelloCppCli(
109
            me->Run();
110
111
        catch (Exception^ ex1)
112
        {
113
            Console::Error->WriteLine(String::Concat("
114
115
        return 0;
116
117 }
```

Compression Command-Line Tools

The Tools/Utilities downloads for DotNetZip includes a pair of command-line tools that perform raw compression and decompression: GZip.exe and BZip2.exe. These tools can be used from the command line to compress or decompress .gz and .bz2 files, respectively, from within batch scripts, or CMD.exe windows.

GZip.exe

This tool can be used to compress a regular file into a .gz file, or to decompress a .gz files into its original, uncompressed form. It handles files in the .gz format created by other GZIP tools, such as the gzip tool that is included with Linux and other operating systems, and the .gz files it produces on Windows can be read by the gzip tool on other platforms.

Of interest, the GZip format handled by this tool is specified in IETF RFC 1952. While the GZIP format can handle multiple entries within a single archive, in general that feature is rarely used. Instead, most uses of GZip compress a single file to a single .gz file.

Usage

BZip2.exe

This tool can be used to create .bz2 files, that conform to the BZIP2 format. Bzip2 was created in 1996 as a compression method and format that can be applied to single files. Unlike ZIP, it is not a multientry archive format - you use bzip2 to compress a single file into a single compressed image. By convention, for a given filename, a bzip2 compressor creates a new file called filename.bz2.

Using the BZip2.exe command line tool included with DotNetZip, you can compress regular files into .bz2 files, and you can decompress .bz2 files into the corresponding original file.

This tool such is compatible with other bzip2 tools, such as the one that is included with Linux and other operating systems. The .bz2 files that this tool produces on Windows can be read by the bzip2 tool on other platforms.

Most Bzip2 utilities delete the original file after producing the compressed version, and delete the compressed version after decompressing. This utility follows that convention.

BZip2 is a compression format, similar to ZLIB, GZIP, or DEFLATE. BZIP2 tends to compress *better* than these other algorithms, but BZIP2 compressors tend to require more time to compress.

Usage

```
BZip2.exe <FileToProcess> [arguments]
  arguments:
```

- -v verbose output. -f force overwrite -keep don't delete the force overwrite of any existing files.
 don't delete the original file after compress or decompressing it.

Command Examples

Use this Command	to do this	
gzip.exe Larg	geFile.txt	
	compress the LargeFile.txt to produce LargeFile.txt.gz .	
gzip.exe Larg	geFile.txt.gz	
	decompress the LargeFile.txt.gz to produce LargeFile.txt , and delete LargeFile.txt.gz .	
gzip.exe Larg	geFile.txt.gz -keep	
	decompress the LargeFile.txt.gz to produce LargeFile.txt ; the original LargeFile.txt.gz will not be deleted.	
bzip2.exe Laı	geFile.txt	
	compress the LargeFile.txt to produce LargeFile.txt.bz2 , and delete the LargeFile.txt when finished.	
bzip2.exe Lai	geFile.txt -keep	
	compress the LargeFile.txt to produce LargeFile.txt.bz2 , and keep the LargeFile.txt when finished.	
bzip2.exe LargeFile.txt.bz2		
	decompress the LargeFile.txt.bz2 to produce LargeFile.txt , and delete the original compressed file when finished.	

DotNetZip Command-Line Tools

The Tools/Utilities downloads for DotNetZip include a pair of tools - Zipit.exe and Unzip.exe - that can be used from the command line to manipulate zip files from within batch scripts, or CMD.exe windows.

ZipIt.exe

This tool can be used to create ZIP files, or update existing ZIP files, from the command line or from a batch file.

Using the command line tools, you can do virtually everything you can do by using the library directly from a custom application. You can create self-extracting archives or archives that use ZIP64 extensions. You can create archives that use encryption or passwords. You can create split or spanned archives. And there are many other options.

Usage

```
ZipIt.exe <ZipFileToCreate> [arguments]
arguments:
  <directory> | <file> - a directory or file to add to the a
  <selector>
                       - a file selection expression.
                           *.txt
                            (name = *.txt) OR (name = *.xml)
                            (attrs = H) OR (name != *.xml)
                            (ctime < 2009/02/28-10:20:00)
                            (size > 1g) AND (mtime < 2009-06-1)
                            (ctime > 2009-04-29) AND (size < :
                         Filenames can include full paths. Ye
                         that includes spaces with single que
  -64
                       - use ZIP64 extensions, for large file
                       - use WinZip-compatible AES 256-bit e
  -aes
                         subsequently added to the archive.
  -cp <codepage>
                       - use the specified numeric codepage
                         or filenames that cannot be encoded
                         code page.
  -d <path>
                       - use the given directory path in the
```

succeeding items added to the archi -D <path> - find files in the given directory or -e<s,r,q,a> - when there is an error reading a fil the file, retry, quit, or ask the us - j -- do not traverse NTFS junctions - j+ - traverse NTFS junctions (default) -L <level> - compression level, 0..9 (Default is -p <password> apply the specified password for al. use "" to reset the password to nil - emit progress reports (good when cre -progress don't recurse directories (default) - r -- recurse directories. -r+ -s <entry> 'string' - insert an entry of the given name i archive, with the given string as i -sfx[w|c]- create a self-extracting archive, e. -split <maxsize> - produce a split zip, with the speci optionally use kb or mb as a suffix -split is not compatible with -sfx. This is not compatible with -sfx. - store Windows-format extended times -Tw+ - don't store Windows-format extended -Tw--Tu+ - store Unix-format extended times (de -Tu-- don't store Unix-format extended til - use uniform date/time, NOW, for all -UTnow -UTnewest - use uniform date/time, newest entry -UToldest - use uniform date/time, oldest entry - use uniform date/time, specified, for -UT <datetime> - use UTF-8 encoding for entries with -utf8 filenames that cannot be encoded wi code page. - use the given comment for the archi -zc <comment>

Command Examples

Use this Command	to do this	
zipit.exe NewZip.zip -64 *.doc		
	create a new zip file called NewZip.zip, adding all .doc files from	

	the current directory into the zip. Use ZIP64 extensions.		
zipit.exe Nev	zipit.exe NewZip.zip -r+ *.doc		
	create a new zip file called NewZip.zip, adding all .doc files, in the current directory and any child directories, into the zip.		
zipit.exe Bac	kup.zip -d c:\MyDocuments -r+ -j- *.*		
	create a new zip file called Backup.zip, adding all files within the c:\MyDocuments directory. Do not traverse NTFS Junction points (like My Videos).		
zipit.exe Pac	kage.exe -sfx -d c:\files -r+ *.*		
	create a new self-extracting archive file called Package.exe, adding all files within the c:\files directory hierarchy.		
zipit.exe Pac	kage.zip -split 1mb -d c:\files -r+ *.*		
	create a new "spanned" or multi-segment zip file called Package.zip, adding all files within the c:\files directory hierarchy. Each segment of the zipfile should be limited to 1mb in size, and the segments will be named Package.z01, Package.z02, etc.		
zipit.exe ImageLibrary.zip -split 1mb -d c:\images -r+ "name = *.jpg AND size > 100kb"			
	create a new "spanned" or multi-segment zip file called ImageLibrary.zip, adding all .jpg files within the c:\images directory hierarchy that have a size larger than 100kb. Each segment of the zipfile should be limited to 1mb in size, and the segments will be named Package.z01, Package.z02, etc.		
zipit.exe Por	tfolio.zip -p Secret! -aes -d c:\portfolio -r+ "name != *.tfs"		
	create a new aip file called Portfolio.zip, adding all files within the c:\portfolio directory hierarchy that do not have the .tfs extension. Encrypt all entries using AES 256-bit encryption.		
zipit.exe Portfolio.zip -p Secret! -aes -L 9 -d c:\portfolio -r+ "name != *.tfs"			
	create a new aip file called Portfolio.zip, adding all files within the c:\portfolio directory hierarchy that do not have the .tfs extension. Encrypt all entries using AES 256-bit encryption, and use level 9 (Best) compression.		

Syntax for the Selection Criteria

Using the a selector string, you can specify a set of criteria for the files to be added to the zip file. Specify the criteria in statements of 3 elements: a noun, an operator, and a value. Consider the string "name != *.doc" . The noun is "name". The operator is "!=", implying "Not Equal". The value is "*.doc". That criterion, in English, says "all files with a name that does not end in the .doc extension."

Supported nouns include **name** for the filename; **atime**, **mtime**, and **ctime** for last access time, last modfied time, and created time of the file, respectively; **attributes** for the file attributes; and **size** for the file length (uncompressed). The **attributes** and **name** nouns both support = and != as operators. The **size**, **atime**, **mtime**, and **ctime** nouns support = and !=, and >, >=, <, <= as well. The times are taken to be expressed in *local time*.

Specify values for the file attributes as a string with one or more of the characters H,R,S,A in any order, implying Hidden, ReadOnly, System, and Archive, respectively. To specify a time, use YYYY-MM-DD-HH:mm:ss or YYYY/MM/DD-HH:mm:ss as the format. If you omit the HH:mm:ss portion, it is assumed to be 00:00:00 (midnight). The value for a size criterion is expressed in integer quantities of bytes, kilobytes (use k or kb after the number), megabytes (m or mb), or gigabytes (g or gb). The value for a name is a pattern to match against the filename, potentially including wildcards. The pattern follows CMD.exe glob rules: * implies one or more of any character (not including dot), while ? implies one character (not including dot). If the name pattern contains any slashes, it is matched to the entire filename, including the path; otherwise, it is matched against only the filename without the path. This means a pattern of "**.*" matches all files one directory level deep, while a pattern of "**.*" matches all files in all

directories.

To specify a name pattern that includes spaces, use single quotes around the pattern. A pattern of "'* *.*" will match all files that have spaces in the filename. The full criteria string for that would be "name = '* *.*".

Examples of Selection Criteria

Use this Command	to do this	
name = *.jpg		
	any .jpg file	
mtime > 2009	9-07-14	
	any file with a <i>last modified time</i> after midnight on 14 July 2009. There is also ctime and atime for <i>created time</i> and <i>accessed time</i> .	
ctime > 2009	-07-14-07:53:00	
	any file with a <i>created time</i> after 7:53am (local time) on 14 July 2009.	
size > 320mb		
	any file with a size over 320mb. You can use kb or gb , too. Or omit the characters to specify a filesize in bytes. And you can use <, >, or = as operations.	
attr!= H		
	any file that does not have the Hidden attribute set. Other attributes include S=system, R=Readonly, A=Archive. Of course you can test that the attribute is ON as well, using = instead of !=.	
attr!= H and size > 320mb		
	include the files that satisfy both conditions. You can also use OR as a conjuction. Use parens to group complex expressions.	
name = *.jpg or name = *.gif		

include the files that satisfy one or the other condition.		
(name = *.jpg) or (name = *.gif)		
same as above.		
(mtime >= 2009-07-01) and (mtime < 2009-07-02)		
any file modified on July 1st. From midnight to midnight.		
(name = *.jpg) AND (mtime >= 2009/07/01) and (mtime < 2009/07/02)		
any .jpg file modified on July 1st.		
(name = *.jpg) and (size >= 100kb) and (mtime >= 2009/07/01) and (mtime < 2009/07/02)		
any .jpg file, 100kb or more in size, modified on July 1st.		

UnZip.exe

This tool can be used to view, extract, or test and verify ZIP files, from the command line or from a batch file.

Usage

```
UnZip.exe [options] <zipfile> [<entryname>...]
   unzips all files in the archive.
   options:
                       overwrite existing files if necessa
     -0
     -f
                       flatten directory structure when ex-
     -p <password>
                       specify password for extraction.
                       test the file for consistency.
     -t
     -q
                       operate quietly (no verbose message:
                       extract with the specified numeric
     -cp <codepage>
                       know the codepage, and it is neithe
                       codepage you specify here is different
                       the cmd.exe, then the verbose message
                       files will be extracted properly.
     -d <directory>
                       unpack to the specified directory.
                       unzip to the current directory.
                       unzip only the specified filename.
     <entryname>
unzip -l <zipfile>
   lists the entries in the zip archive.
unzip -i <zipfile>
   displays full information about all the entries in the
unzip -t <zipfile> [-p <password>] [-cp <codepage>]
   tests the zip archive.
unzip -?
   displays this message.
```

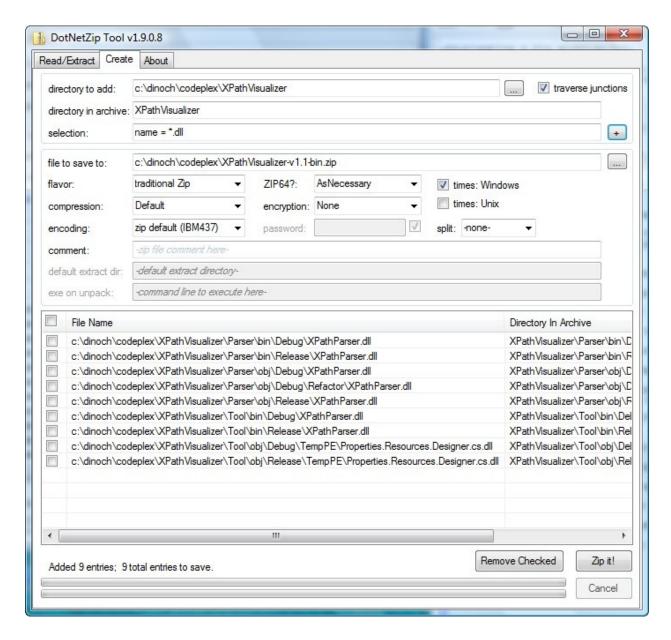
Command Examples

Use this Command	to do this	
UnZip.exe -l	Package.zip	
	list the entries in the zipfile, along with basic information about each entry.	
UnZip.exe Pa	nckage.zip	
	extract the entries in the zipfile, into the current directory.	
UnZip.exe Pa	nckage.zip -d ex1	
	extract the entries in the zipfile, into the directory 'ex1'. The directory will be created if it does not exist.	
UnZip.exe Pa	nckage.zip -f -d ex1	
	extract the entries in the zipfile, into the directory 'ex1'. The directory will be created if it does not exist. Any directory hierarchy within the zip file will be ignored or "flattened" upon extraction.	
UnZip.exe Package.zip -o -d ex1		
	extract the entries in the zipfile, into the directory 'ex1'. The directory will be created if it does not exist. During extraction, any existing files in the directory hierarchy will be overwritten if necessary.	

The DotNetZip Graphical Tool

The Tools/Utilities downloads for DotNetZip include a graphical tool for creating and extracting zip files.

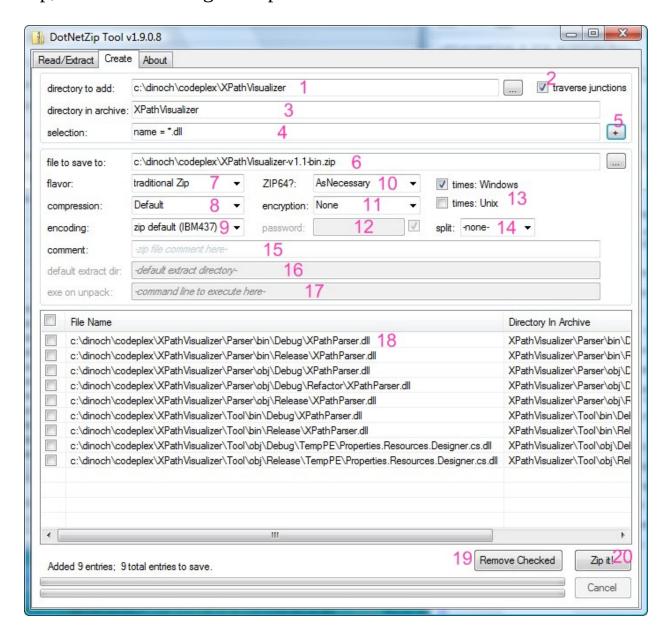
If you use the MSI installer, you will have the chance to associate .zip files to this tool. If you associate .zip files to this tool, double-clicking on a zip file from within Windows Explorer will open the zip file in the tool. If you don't associate .zip files to this tool, you can open the tool using the Start menu.



The tool uses a tabbed-pane interface. One tab is used for reading or extracting zip files, another is used for creating zip files.

Creating a Zip Archive

Create a zip file in two steps: first, select which files to add into the zip, then add creating the zip with those files.



An overview of the UI elements:

1. the directory to add. Specify where in the filesystem to find the

- files to be added into the zip file.
- 2. check this box to traverse Windows NTFS Junctions; leave it unchecked to not traverse junctions
- 3. the directory to be used within the zip archive for the files that get added.
- 4. The selection criteria for the files to be added to the zip.
- 5. Click this button to add the files that match the selection criteria, to the listbox below (label #18) To select additional files, you can then repeat steps 1-5 as many times as is necessary.

When you've selected the files you want to include in the archive,

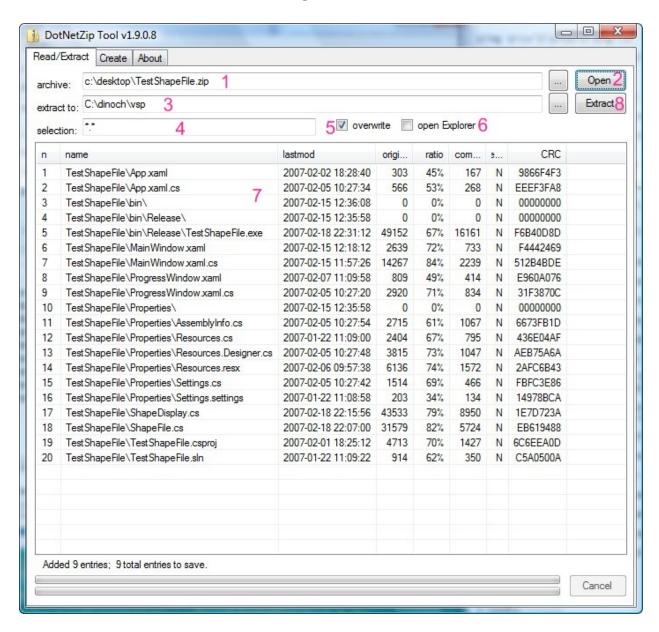
- 6. Specify the name of the zip file to create.
- 7. Specify whether to save a regular zip file, or a self-extracting archive.
- 8. Specify the compression level to use. Higher compression levels requires more compression time.
- 9. The text encoding to use for the filenames.
- 10. Whether to use ZIP64 extensions
- 11. whether to encrypt the file data, and if so, whether to use AES encryption.
- 12. the password to use. The adjacent checkbox allows you to show or obscure the password characters. This box is enabled only if encryption is in use.
- 13. Use these checkboxes to specify whether to include extended timestamps in the generated zip file, in Windows or Unix formats.
- 14. Whether to create a split archive, and if so, the size of the segment.
- 15. The comment attached to the zip file
- 16. The default extraction directory. This feature is enabled only for self-extracting archives.
- 17. The command to run after extraction. This feature is enabled only for self-extracting archives.
- 18. The list of files to be added to the zip archive. Each file has a

checkbox associated to it.

- 19. Click this button to remove any checked files from the list.
- 20. Finally, click this button to actually create the zip archive.

Extracting files from a Zip Archive

Extract files from a zip archive by opening the DotNetZip-WinformsTool.exe, and selecting the **Read/Extract** tab in the UI.



An overview of the UI elements:

1. the zip file to read or extract.

- 2. click here to open and read the specified zip file. The list of entries in the zip will be displayed in the listbox below (label #7).
- 3. The directory to extract entries into
- 4. The selection criteria for entries to extract.
- 5. Check this to overwrite existing files during extraction.
- 6. Check this box to open Windows Explorer after extraction.
- 7. The list of entries in the zip file.
- 8. Click this button to extract the specified entries in the zip file.

Syntax for the Selection Criteria

Using the a selector string, you can specify a set of criteria for the files to be added to the zip file. Specify the criteria in statements of 3 elements: a noun, an operator, and a value. Consider the string "name != *.doc" . The noun is "name". The operator is "!=", implying "Not Equal". The value is "*.doc". That criterion, in English, says "all files with a name that does not end in the .doc extension."

Supported nouns include **name** for the filename; **atime**, **mtime**, and **ctime** for last access time, last modfied time, and created time of the file, respectively; **attributes** for the file attributes; and **size** for the file length (uncompressed). The **attributes** and **name** nouns both support = and != as operators. The **size**, **atime**, **mtime**, and **ctime** nouns support = and !=, and >, >=, <, <= as well. The times are taken to be expressed in *local time*.

Specify values for the file attributes as a string with one or more of the characters H,R,S,A in any order, implying Hidden, ReadOnly, System, and Archive, respectively. To specify a time, use YYYY-MM-DD-HH:mm:ss as the format. If you omit the HH:mm:ss portion, it is assumed to be 00:00:00 (midnight). The value for a size criterion is expressed in integer quantities of bytes, kilobytes (use k or kb after the number), megabytes (m or mb), or gigabytes (g or gb). The value for a name is a pattern to match against the filename, potentially including wildcards. The pattern follows CMD.exe glob rules: * implies one or more of any character (not including dot), while ? implies one character (not including dot). If the name pattern contains any slashes, it is matched to the entire filename, including the path; otherwise, it is matched against only the filename without the path. This means a pattern of "**.*" matches all files one directory level deep, while a pattern of "*.*" matches all files in all directories.

To specify a name pattern that includes spaces, use single quotes around the pattern. A pattern of "'* *.*" will match all files that have spaces in the filename. The full criteria string for that would be "name = '* *.*".

Examples of Selection Criteria

Use this Command	to do this		
name = *.jpg	name = *.jpg		
	any .jpg file		
mtime > 07/0	1/2009		
	any file with a <i>last modified time</i> after midnight on 1 July 2009.		
	There is also ctime and atime for <i>created time</i> and <i>accessed time</i> .		
ctime > 07/01	1/2009-07:53:00		
	any file with a <i>created time</i> after 7:53am on 1 July 2009.		
size > 320mb			
	any file with a size over 320mb. You can use kb or gb , too. Or omit the characters for a size in bytes. And you can use <, >, or = as operations.		
attr!= H			
	any file that does not have the Hidden attribute set. Other attributes include S=system, R=Readonly, A=Archive. Of course you can test that the attribute is ON as well, using = instead of !=.		
attr != H and	size > 320mb		
	include the files that satisfy both conditions. You can also use OR as a conjuction. Use parens to group complex expressions.		
name = *.jpg or name = *.gif			
	include the files that satisfy one or the other condition.		
(name = *.jpg) or (name = *.gif)			

same as above.		
(mtime >= 07/01/2009) and (mtime < 07/02/2009)		
any file modified on July 1st. From midnight to midnight.		
(name = *.jpg) AND (mtime >= 07/01/2009) and (mtime < 07/02/2009)		
any .jpg file modified on July 1st.		
(name = *.jpg) and (size \geq 100kb) and (mtime \geq 07/01/2009) and (mtime \leq		
07/02/2009)		
any .jpg file, 100kb or more in size, modified on July 1st.		

Ionic Zip Library v1.9.1.6 Reference Reference C#

DotNetZip is a small, easy-to-use class library for manipulating .zip files It can enable .NET applications written in VB.NET, C#, any .NET language, to easily create, read, and update zip files. DotNetZip is donationware: http://cheeso.members.winisp.net/DotNetZipDonate.asp

Namespaces

Namespace	Description
lonic	The Ionic namespace includes supporting classes used in the DotNetZip library.
Ionic.BZip2	The Ionic.BZip2 namespace includes classes for doing BZip2 compression and decompression, according to the BZip2 specification provided by Julian Seward.
Ionic.Crc	The Ionic.Crc namespace includes a classes that calculate 32-bit Cyclic Redundancy Checks (CRC32) for data.
Ionic.Zip	Classes in the Ionic.Zip namespace allow applications to read and write zip files according to the format described by PKWARE DotNetZip is donationware. Please donate.
Ionic.Zlib	Ionic.Zlib namespace includes classes for

doing ZLIB, DEFLATE, and GZIP compression and decompression, according to RFC 1950, RFC 1951, and RFC 1952. It includes stream classes tha can directly replace the DeflateStream an GZipStream classes that are included in the Base Class Library of the .NET Framework.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

lonic Zip Library v1.9.1.6 lonic Namespace Reference ► lonic C#

The Ionic namespace includes supporting classes used in the DotNetZi library.

■ Declaration Syntax

C#	Visual Basic	Visual C++
namespace Ionic		
Namespace Ionic		
namespace Ionic		

Types

All Ty	pes	Classes		
Icon	Туре		Description	
4	FileSelector		selects files from the filesystem criteria. This could the DotNetZip the AddSelect class can also	ncapsulates logic that om a source - a zip file c - based on a set of lass is used internally by library, in particular for edFiles() methods. This be used independently ability in DotNetZip.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 File Selector Class

Reference ► Ionic ► FileSelector

C#

FileSelector encapsulates logic that selects files from a source - a zip fil or the filesystem - based on a set of criteria. This class is used internally by the DotNetZip library, in particular for the AddSelectedFiles() method This class can also be used independently of the zip capability in DotNetZip.

■ Declaration Syntax

C#	Visual Basic	Visual C++		
public class FileSelector				
Public Class FileSelector				
public ref class	FileSelector			

■ Members

All M	embers	Constructors	ſ	Methods	Properties	
Public Protected				☑ Instance ☑ Static		Declare Inherite
Icon	n Member			Description		
∄	FileSele	ctor(String)			hat allows the celection criteria.	aller to
∄	FileSelector(String, Boolean)			Constructor that allows the caller to specify file selection criteria.		
=0	Equals(Object)			Determines whether the specified Object is equal to the current Object .		

		(Inherited from Object.)
ğ	Finalize()	Allows an Object to attempt to free resources and perform other cleanur operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
≡©	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
≡	GetType()	Gets the Type of the current instance (Inherited from Object.)
Ģ Φ	MemberwiseClone()	Creates a shallow copy of the curren Object . (Inherited from Object .)
≅∳	SelectEntries(ZipFile)	Retrieve the ZipEntry items in the ZipFile that conform to the specified criteria.
=♦	SelectEntries(ZipFile, String)	Retrieve the ZipEntry items in the ZipFile that conform to the specified criteria.
≅∲	SelectFiles(String)	Returns the names of the files in the specified directory that fit the selection criteria specified in the FileSelector.
=0	SelectFiles(String,	Returns the names of the files in the

	Boolean)	specified directory that fit the selection criteria specified in the FileSelector, optionally recursing through subdirectories.
	SelectionCriteria	The string specifying which files to include when retrieving.
≅	ToString()	Returns a string representation of the FileSelector object. (Overrides Object.ToString().)
	TraverseReparsePoints	Indicates whether searches will traverse NTFS reparse points, like Junctions.

■ Remarks

The FileSelector class is used internally by the ZipFile class for selectin files for inclusion into the ZipFile, when the AddSelectedFiles(String, String) method, or one of its overloads, is called. It's also used for the ExtractSelectedEntries(String) methods. Typically, an application that creates or manipulates Zip archives will not directly interact with the FileSelector class.

Some applications may wish to use the FileSelector class directly, to select files from disk volumes based on a set of criteria, without creating or querying Zip archives. The file selection criteria include: a pattern to match the filename; the last modified, created, or last accessed time of the file; the size of the file; and the attributes of the file.

Consult the documentation for <u>SelectionCriteria</u> for more information on specifying the selection criteria.

■ Inheritance Hierarchy

Object

FileSelector

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 File Selector Constructor

Reference ► Ionic ► FileSelector ► FileSelector()

C#

■ Members

lcon	Member	Description
∃	FileSelector(String)	Constructor that allows the caller to specify file selection criteria.
=	FileSelector(String, Boolean)	Constructor that allows the caller to specify file selection criteria.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 File Selector Constructor (selection Criteria)

Reference ► Ionic ► FileSelector ► FileSelector(String)

Constructor that allows the caller to specify file selection criteria.

■ Declaration Syntax

Parameters

selectionCriteria (String)

The criteria for file selection.

Remarks

This constructor allows the caller to specify a set of criteria for selection of files.

See <u>SelectionCriteria</u> for a description of the syntax of the selectionCriteria string.

By default the FileSelector will traverse NTFS Reparse Points. To chanç this, use <u>FileSelector(String, bool)</u>.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 File Selector Constructor (selection Criteria, traverse Directory Reparse Points)

Reference ► Ionic ► FileSelector ► FileSelector(String, Boolean)

Constructor that allows the caller to specify file selection criteria.

Declaration Syntax

bool traverseDirectoryReparsePoints

Parameters

selectionCriteria (String)

The criteria for file selection.

traverseDirectoryReparsePoints (Boolean)

whether to traverse NTFS reparse points (junctions).

Remarks

This constructor allows the caller to specify a set of criteria for selection

of files.

See <u>SelectionCriteria</u> for a description of the syntax of the selectionCriteria string.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SelectEntries Method

Reference ► Ionic ► FileSelector ► SelectEntries()

C#

■ Members

Icon	Member	Description
€ 🚱	SelectEntries(ZipFile)	Retrieve the ZipEntry items in the ZipFile that conform to the specified criteria.
≓ ₩	SelectEntries(ZipFile, String)	Retrieve the ZipEntry items in the ZipFile that conform to the specified criteria.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SelectEntries Method (zip)

Reference ► Ionic ► FileSelector ► SelectEntries(ZipFile)

Retrieve the ZipEntry items in the ZipFile that conform to the specified criteria.

Declaration Syntax

Parameters

zip (ZipFile)

The ZipFile from which to retrieve entries.

■ Return Value

a collection of ZipEntry objects that conform to the criteria.

■ Remarks

This method applies the criteria set in the FileSelector instance (as described in the <u>SelectionCriteria</u>) to the specified ZipFile. Using this method, for example, you can retrieve all entries from the given ZipFile that have filenames ending in .txt.

Normally, applications would not call this method directly. This method is used by the ZipFile class.

Using the appropriate SelectionCriteria, you can retrieve entries based size, time, and attributes. See <u>SelectionCriteria</u> for a description of the syntax of the SelectionCriteria string.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SelectEntries Method (zip, directory PathInArchive)

Reference ► Ionic ► FileSelector ► SelectEntries(ZipFile, String)

Retrieve the ZipEntry items in the ZipFile that conform to the specified criteria.

Declaration Syntax

```
Public Function SelectEntries ( _ zip As ZipFile, _ directoryPathInArchive As String _ ) As ICollection(Of ZipEntry)
```

Parameters

zip (ZipFile)

The ZipFile from which to retrieve entries.

directoryPathInArchive (String)

the directory in the archive from which to select entries. If null, then all directories in the archive are used.

■ Return Value

a collection of ZipEntry objects that conform to the criteria.

■ Remarks

This method applies the criteria set in the FileSelector instance (as described in the <u>SelectionCriteria</u>) to the specified ZipFile. Using this method, for example, you can retrieve all entries from the given ZipFile that have filenames ending in .txt.

Normally, applications would not call this method directly. This method is used by the ZipFile class.

This overload allows the selection of ZipEntry instances from the ZipFile to be restricted to entries contained within a particular directory in the ZipFile.

Using the appropriate SelectionCriteria, you can retrieve entries based size, time, and attributes. See <u>SelectionCriteria</u> for a description of the syntax of the SelectionCriteria string.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

■ Members

Icon	Member	Description
€ 🚱	SelectFiles(String)	Returns the names of the files in the specified directory that fit the selectio criteria specified in the FileSelector.
∃	SelectFiles(String, Boolean)	Returns the names of the files in the specified directory that fit the selectio criteria specified in the FileSelector, optionally recursing through subdirectories.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SelectFiles Method (directory)

Reference ► Ionic ► FileSelector ► SelectFiles(String) C#

Returns the names of the files in the specified directory that fit the selection criteria specified in the FileSelector.

Declaration Syntax

Parameters

directory (String)

The name of the directory over which to apply the FileSelector criteria.

■ Return Value

A collection of strings containing fully-qualified pathnames of files that match the criteria specified in the FileSelector instance.

Remarks

This is equivalent to calling <u>SelectFiles(String, Boolean)</u> with recurseDirectories = false.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SelectFiles Method (directory, recurseDirectories)

Reference ► Ionic ► FileSelector ► SelectFiles(String, Boolean)

Returns the names of the files in the specified directory that fit the selection criteria specified in the FileSelector, optionally recursing through subdirectories.

Declaration Syntax

Parameters

directory (String)

The name of the directory over which to apply the FileSelector criteria.

recurseDirectories (Boolean)

Whether to recurse through subdirectories when applying the file selection criteria.

Return Value

A collection of strings containing fully-qualified pathnames of files that match the criteria specified in the FileSelector instance.

■ Remarks

This method applies the file selection criteria contained in the FileSelector to the files contained in the given directory, and returns the names of files that conform to the criteria.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

The string specifying which files to include when retrieving.

■ Declaration Syntax

■ Remarks

Specify the criteria in statements of 3 elements: a noun, an operator, an a value. Consider the string "name != *.doc" . The noun is "name". The operator is "!=", implying "Not Equal". The value is "*.doc". That criterior in English, says "all files with a name that does not end in the .doc extension."

Supported nouns include "name" (or "filename") for the filename; "atime "mtime", and "ctime" for last access time, last modfied time, and created time of the file, respectively; "attributes" (or "attrs") for the file attributes; "size" (or "length") for the file length (uncompressed); and "type" for the type of object, either a file or a directory. The "attributes", "type", and "name" nouns all support = and != as operators. The "size", "atime", "mtime", and "ctime" nouns support = and !=, and >, >=, <, <= as well. The times are taken to be expressed in local time.

Specify values for the file attributes as a string with one or more of the characters H,R,S,A,I,L in any order, implying file attributes of Hidden, ReadOnly, System, Archive, NotContextIndexed, and ReparsePoint (symbolic link) respectively.

To specify a time, use YYYY-MM-DD-HH:mm:ss or YYYY/MM/DD-HH:mm:ss as the format. If you omit the HH:mm:ss portion, it is assume to be 00:00:00 (midnight).

The value for a size criterion is expressed in integer quantities of bytes, kilobytes (use k or kb after the number), megabytes (m or mb), or gigabytes (g or gb).

The value for a name is a pattern to match against the filename, potentially including wildcards. The pattern follows CMD.exe glob rules: implies one or more of any character, while ? implies one character. If the name pattern contains any slashes, it is matched to the entire filename, including the path; otherwise, it is matched against only the filename without the path. This means a pattern of "**.*" matches all files one directory level deep, while a pattern of "*.*" matches all files in all directories.

To specify a name pattern that includes spaces, use single quotes arounthe pattern. A pattern of "'* *.*" will match all files that have spaces in the filename. The full criteria string for that would be "name = '* *.*".

The value for a type criterion is either F (implying a file) or D (implying a directory).

Some examples:

criteria	Files retrieved
name != *.xls	any file with an extension that is not .xls
name = *.mp3	any file with a .mp3 extension.
*.mp3	(same as above) any file with a .mp3 extension.
attributes = A	all files whose attributes include the Archive bit.
attributes != H	all files whose attributes do not include the Hidder

	bit.
mtime > 2009-01- 01	all files with a last modified time after January 1st, 2009.
ctime > 2009/01/01- 03:00:00	all files with a created time after 3am (local time), on January 1st, 2009.
size > 2gb	all files whose uncompressed size is greater than 2gb.
type = D	all directories in the filesystem.

You can combine criteria with the conjunctions AND, OR, and XOR. Using a string like "name = *.txt AND size >= 100k" for the selectionCriteria retrieves entries whose names end in .txt, and whose uncompressed size is greater than or equal to 100 kilobytes.

For more complex combinations of criteria, you can use parenthesis to group clauses in the boolean logic. Absent parenthesis, the precedence of the criterion atoms is determined by order of appearance. Unlike the C# language, the AND conjunction does not take precendence over the logical OR. This is important only in strings that contain 3 or more criterion atoms. In other words, "name = *.txt and size > 1000 or attributes = H" implies "((name = *.txt AND size > 1000) OR attributes = H)" while "attributes = H OR name = *.txt and size > 1000" evaluates to ((attributes = H OR name = *.txt) AND size > 1000)". When in doubt, us parenthesis.

Using time properties requires some extra care. If you want to retrieve a entries that were last updated on 2009 February 14, specify "mtime >= 2009-02-14 AND mtime < 2009-02-15". Read this to say: all files update after 12:00am on February 14th, until 12:00am on February 15th. You can use the same bracketing approach to specify any time period - a year, a month, a week, and so on.

The syntax allows one special case: if you provide a string with no spaces, it is treated as a pattern to match for the filename. Therefore a string like "*.xls" will be equivalent to specifying "name = *.xls". This "shorthand" notation does not work with compound criteria.

There is no logic in this class that insures that the inclusion criteria are internally consistent. For example, it's possible to specify criteria that says the file must have a size of less than 100 bytes, as well as a size that is greater than 1000 bytes. Obviously no file will ever satisfy such criteria, but this class does not check for or detect such inconsistencies.

Exceptions

Exception	Condition
Exception	Thrown in the setter if the value has an invalid syntax.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ToString Method

Reference ► Ionic ► FileSelector ► ToString()

C#

Returns a string representation of the FileSelector object.

■ Declaration Syntax

```
C# Visual Basic Visual C++
```

public override string ToString()

Public Overrides Function ToString As String

```
public:
```

virtual String^ ToString() override

Return Value

The string representation of the boolean logic statement of the file selection criteria for this instance.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 TraverseReparsePoints Property

Reference ► <u>lonic</u> ► <u>FileSelector</u> ► **TraverseReparsePoints**

Indicates whether searches will traverse NTFS reparse points, like Junctions.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Ionic.BZip2 Namespace

Reference ► Ionic.BZip2

C#

The Ionic.BZip2 namespace includes classes for doing BZip2 compression and decompression, according to the BZip2 specification provided by Julian Seward.

■ Declaration Syntax

C#	Visual Basic	Visual C++
namespace Ionic.	BZip2	
Namespace Ionic.	BZip2	
namespace Ionic.	BZip2	

Types

All Types		Classes	
Icon	on Type		Description
9 \$	BZip2InputStream		A read-only decorator stream that performs BZip2 decompression or Read.
9 \$	BZip2OutputStr	eam	A write-only decorator stream that compresses data as it is written using the BZip2 algorithm.
43	ParallelBZip2Ou	<u>itputStream</u>	A write-only decorator stream that compresses data as it is written using the BZip2 algorithm. This stream compresses by block using multiple threads.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BZip2InputStream Class

Reference ► Ionic.BZip2 ► BZip2InputStream

C#

A read-only decorator stream that performs BZip2 decompression on Read.

■ Declaration Syntax

C# Visual Basic Visual C++

public class BZip2InputStream : Stream

Public Class BZip2InputStream _ Inherits Stream

public ref class BZip2InputStream : public Stream

■ Members

All M	embers	Constructors	M	ethods	Properties	
			✓ Instance ✓ Static		Declare Inherite	
lcon	Member			Description		
Ξ₩	BZip2InputStream(Stream)		Create a BZip2InputStream, wrapping it around the given input Stream.			
Ξ₩	BZip2Inp Boolean)	outStream(Stream		given stream whether to l stream ope	Zip2InputStream m, and specifyin leave the wrapp n when the Stream is closed	g ed

≅	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous read operation.
		(Inherited from <u>Stream</u> .)
≅	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous write operation.
		(Inherited from <u>Stream</u> .)
	CanRead	Indicates whether the stream can b read.
		(Overrides <u>Stream</u> . <u>CanRead</u> .)
	CanSeek	Indicates whether the stream supports Seek operations.
		(Overrides <u>Stream.CanSeek</u> .)
	CanTimeout	Gets a value that determines whether the current stream can tim out.
		(Inherited from <u>Stream</u> .)
	<u>CanWrite</u>	Indicates whether the stream can b written.
		(Overrides <u>Stream.CanWrite</u> .)
≡	Close()	Close the stream.
		(Overrides <u>Stream.Close()</u> .)
≓ ₩	CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from
		(IIIIIeIIIeu IIUIII

	MarshalByRefObject.)
<u>CreateWaitHandle()</u>	Obsolete. Allocates a <u>WaitHandle</u> object.
	(Inherited from <u>Stream</u> .)
Dispose(Boolean)	Dispose the stream.
	(Overrides <u>Stream.Dispose(Boolean)</u> .)
Dispose()	Releases all resources used by the <u>Stream</u> .
	(Inherited from <u>Stream</u> .)
EndRead(IAsyncResult)	Waits for the pending asynchronous read to complete.
	(Inherited from <u>Stream</u> .)
EndWrite(IAsyncResult)	Ends an asynchronous write operation.
	(Inherited from <u>Stream</u> .)
Equals(Object)	Determines whether the specified Object is equal to the current Object .
	(Inherited from Object.)
Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.
	(Inherited from Object.)
Flush()	Flush the stream.
	(Overrides <u>Stream.Flush()</u> .)
	Dispose(Boolean) Dispose() EndRead(IAsyncResult) EndWrite(IAsyncResult) Equals(Object) Finalize()

₫₩	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
≓	GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
≡	GetType()	Gets the <u>Type</u> of the current instance. (Inherited from <u>Object</u> .)
≝₩	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
	Length	Reading this property always throw a NotImplementedException. (Overrides Stream.Length.)
ē	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object (Inherited from MarshalByRefObject.)
Ģ [©]	MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object.)
	Position	The position of the stream pointer.

		(Overrides <u>Stream.Position</u> .)
≡	Read(Byte[], Int32, Int32)	Read data from the stream.
		(Overrides <u>Stream.Read(Byte[], Int32, Int32)</u> .)
≡	ReadByte()	Read a single byte from the stream
		(Overrides <u>Stream.ReadByte()</u> .)
	ReadTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to read before timing out.
- 🗛		(Inherited from <u>Stream</u> .)
≟	Seek(Int64, SeekOrigin)	Calling this method always throws a NotImplementedException.
		(Overrides <u>Stream.Seek(Int64, SeekOrigin)</u> .)
≡	SetLength(Int64)	Calling this method always throws a NotImplementedException.
		(Overrides Stream .SetLength(Int64).)
≡	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> .
		(Inherited from Object.)
≡	Write(Byte[], Int32, Int32)	Calling this method always throws a NotImplementedException.
		(Overrides <u>Stream.Write(Byte[], Int32, Int32)</u> .)
≟	WriteByte(Byte)	Writes a byte to the current positior in the stream and advances the position within the stream by one

	byte. (Inherited from Stream.)
WriteTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to write before timing out.
	(Inherited from <u>Stream</u> .)

■ Inheritance Hierarchy

Object

<u>MarshalByRefObject</u>

Stream

BZip2InputStream

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BZip2InputStream Constructor

Reference ► Ionic.BZip2 ► BZip2InputStream ► BZip2InputStream()

C#

■ Members

Icon	Member	Description
€₩	BZip2InputStream(Stream)	Create a BZip2InputStream, wrapping it around the given input Stream.
≓ ₩	BZip2InputStream(Stream, Boolean)	Create a BZip2InputStream with the given stream, and specifying whether to leave the wrapped stream open when the BZip2InputStream is closed.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BZip2InputStream Constructor

(input)

Reference ► Ionic.BZip2 ► BZip2InputStream ► BZip2InputStream(Stream)

Create a BZip2InputStream, wrapping it around the given input Stream.

Declaration Syntax

Parameters

input (Stream)

The stream from which to read compressed data

■ Remarks

The input stream will be closed when the BZip2InputStream is closed.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 BZip2InputStream Constructor

(input, leaveOpen)

Reference ► Ionic.BZip2 ► BZip2InputStream ►

BZip2InputStream(Stream, Boolean)

Create a BZip2InputStream with the given stream, and specifying whether to leave the wrapped stream open when the BZip2InputStream is closed.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public BZip2InputStream(
    Stream input,
    bool leaveOpen
)
```

```
Public Sub New ( _ _ _ input As Stream, _ leaveOpen As Boolean _ )
```

Parameters

input (Stream)

The stream from which to read compressed data

leaveOpen (Boolean)

Whether to leave the input stream open, when the BZip2InputStrea closes.

Examples

This example reads a bzip2-compressed file, decompresses it, and write the decompressed data into a newly created file.

C# □ Cop

```
var fname = "logfile.log.bz2";
using (var fs = File.OpenRead(fname))
{
    using (var decompressor = new Ionic.BZip2.BZip2I
    {
        var outFname = fname + ".decompressed";
        using (var output = File.Create(outFname))
            byte[] buffer = new byte[2048];
            int n;
            while ((n = decompressor.Read(buffer, 0,
            {
                output.Write(buffer, ⊙, n);
            }
        }
    }
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

4

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.BZip2 ► BZip2InputStream ► CanRead

Indicates whether the stream can be read.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:
virtual property bool CanRead {
    bool get () override;
}
```

■ Remarks

The return value depends on whether the captive stream supports reading.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Can Seek Property

Reference ► Ionic.BZip2 ► BZip2InputStream ► CanSeek

Indicates whether the stream supports Seek operations.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:

virtual property bool CanSeek {

bool get () override;
```

■ Remarks

}

Always returns false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanWrite Property

Reference ► Ionic.BZip2 ► BZip2InputStream ► CanWrite

Indicates whether the stream can be written.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
    virtual property bool CanWrite {
        bool get () override;
}
```

■ Remarks

The return value depends on whether the captive stream supports writing.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Close Method

Reference ► Ionic.BZip2 ► BZip2InputStream ► Close()

Close the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Close()

Public Overrides Sub Close

public:
virtual void Close() override

C#

Send comments on this topic to $\underline{dpchiesa@hotmail.com}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Dispose Method (disposing)

Reference ► Ionic.BZip2 ► BZip2InputStream ► Dispose(Boolean)

C#

Dispose the stream.

■ Declaration Syntax

Parameters

disposing (Boolean)

indicates whether the Dispose method was invoked by user code.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Method

Reference ► Ionic.BZip2 ► BZip2InputStream ► Flush()

Flush the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

public:
virtual void Flush() override

C#

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Length Property

Reference ► Ionic.BZip2 ► BZip2InputStream ► Length

Reading this property always throws a NotImplemented Exception.

■ Declaration Syntax

```
Visual C++
C#
                  Visual Basic
public override long Length { get; }
Public Overrides ReadOnly Property Length As Long
        Get
public:
virtual property long long Length {
        long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Position Property

Reference ► Ionic.BZip2 ► BZip2InputStream ► Position

The position of the stream pointer.

Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

C#

■ Remarks

Setting this property always throws a <u>NotImplementedException</u>. Reading will return the total number of uncompressed bytes read in.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

Reference ► Ionic.BZip2 ► BZip2InputStream ► Read(Byte[], Int32, Int32)

Read data from the stream.

C# 🔻

Declaration Syntax

```
C# Visual Basic Visual C++

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

```
public:
virtual int Read(
          array<unsigned char>^ buffer,
          int offset,
          int count
) override
```

Parameters

buffer (Byte[])

The buffer into which the read data should be placed.

offset (Int32)

the offset within that data array to put the first byte read.

count (Int32)

the number of bytes to read.

Return Value

the number of bytes actually read

■ Remarks

To decompress a BZip2 data stream, create a BZip2InputStream providing a stream that reads compressed data. Then call Read() on the BZip2InputStream, and the data read will be decompressed as you read.

A BZip2InputStream can be used only for Read(), not for Write().

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ReadByte Method

Reference ► Ionic.BZip2 ► BZip2InputStream ► ReadByte()

C#

Read a single byte from the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override int ReadByte()

Public Overrides Function ReadByte As Integer

public:

virtual int ReadByte() override

Return Value

the byte read from the stream, or -1 if EOF

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.BZip2 ► BZip2InputStream ► Seek(Int64, SeekOrigin)

Calling this method always throws a NotImplementedException.

Declaration Syntax

```
C# Visual Basic Visual C++

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

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

Parameters

offset (Int64)

this is irrelevant, since it will always throw!

origin (SeekOrigin)

this is irrelevant, since it will always throw!

Return Value

irrelevant!

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.BZip2 ► BZip2InputStream ► SetLength(Int64)

Calling this method always throws a NotImplementedException.

■ Declaration Syntax

Parameters

value (Int64)

this is irrelevant, since it will always throw!

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

Reference ► Ionic.BZip2 ► BZip2InputStream ► Write(Byte[], Int32, Int32)

Calling this method always throws a NotImplementedException.

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

```
buffer (Byte[])
this parameter is never used

offset (Int32)
this parameter is never used

count (Int32)
this parameter is never used
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BZip2OutputStream Class

Reference ► Ionic.BZip2 ► BZip2OutputStream

C#

A write-only decorator stream that compresses data as it is written using the BZip2 algorithm.

■ Declaration Syntax

C# Visual Basic Visual C++

public class BZip2OutputStream : Stream

Public Class BZip2OutputStream _ Inherits <u>Stream</u>

public ref class BZip2OutputStream : public Stream

■ Members

All Members Constru		Constructors	Me	ethods	Properties	
✓ Public✓ Protected				Instance Static		Declare Inherite
lcon	Icon Member			Description		
≅ ©	BZip2OutputStream(Stream)		Constructs a new BZip2OutputStream, that sends its compressed output to the given output stream.			
≡ ∅	BZip2Ou Int32)	itputStream(Strea	m,		tputStream	with
≡						

	BZip2OutputStream(Stream, Boolean)	Constructs a new BZip2OutputStream.
₫ 🚱	BZip2OutputStream(Stream, Int32, Boolean)	Constructs a new BZip2OutputStream with specified blocksize, and explicitly specifies whether to leave the wrapped stream open.
Ξ Φ	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous read operation. (Inherited from <u>Stream</u> .)
∄	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous write operation. (Inherited from <u>Stream</u> .)
	BlockSize	The blocksize parameter specified at construction time.
	CanRead	Indicates whether the stream can be read. (Overrides <u>Stream.CanRead</u> .)
	CanSeek	Indicates whether the stream supports Seek operations. (Overrides <u>Stream.CanSeek.</u>)
	CanTimeout	Gets a value that determines whether the current stream can tim out.

		(Inherited from <u>Stream</u> .)
	<u>CanWrite</u>	Indicates whether the stream can be written.
		(Overrides <u>Stream.CanWrite</u> .)
≡	Close()	Close the stream.
		(Overrides <u>Stream.Close()</u> .)
≅©	CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from
		MarshalByRefObject.)
[™]	<u>CreateWaitHandle()</u>	Obsolete. Allocates a <u>WaitHandle</u> object.
		(Inherited from <u>Stream</u> .)
≟	Dispose()	Releases all resources used by the Stream.
		(Inherited from <u>Stream</u> .)
Ģ [¯] ••	Dispose(Boolean)	Releases the unmanaged resource used by the <u>Stream</u> and optionally releases the managed resources.
≡	F	(Inherited from <u>Stream</u> .)
	EndRead(IAsyncResult)	Waits for the pending asynchronou read to complete.
		(Inherited from <u>Stream</u> .)
≡	EndWrite(IAsyncResult)	Ends an asynchronous write operation.

		(Inherited from <u>Stream</u> .)
≡••	Equals(Object)	Determines whether the specified Object is equal to the current Object .
		(Inherited from Object.)
Ģ̄ ♥	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.
		(Inherited from Object.)
∃	Flush()	Flush the stream.
		(Overrides <u>Stream.Flush()</u> .)
≟	GetHashCode()	Serves as a hash function for a particular type.
		(Inherited from Object.)
₫ 🍪	GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
∃	GetType()	Gets the <u>Type</u> of the current instance. (Inherited from <u>Object</u> .)
≡•	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance.
		(Inherited from MarshalByRefObject.)

	Length	Reading this property always throw a NotImplementedException. (Overrides Stream.Length.)
<u></u>	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object (Inherited from MarshalByRefObject.)
· · · · · · · · · · · · · · · · · · ·	MemberwiseClone()	Creates a shallow copy of the current <u>Object</u> . (Inherited from <u>Object</u> .)
	Position	The position of the stream pointer. (Overrides <u>Stream.Position.</u>)
₫ ◊	Read(Byte[], Int32, Int32)	Calling this method always throws NotImplementedException. (Overrides Stream.Read(Byte[], Int32, Int32).)
₫∳	ReadByte()	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. (Inherited from Stream .)
	ReadTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to read before timing out. (Inherited from Stream .)
≟∳	Seek(Int64, SeekOrigin)	Calling this method always throws NotImplementedException.

		(Overrides <u>Stream.Seek(Int64, SeekOrigin)</u> .)
=	SetLength(Int64)	Calling this method always throws NotImplementedException.
		(Overrides Stream . SetLength(Int64).)
=	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> .
		(Inherited from Object.)
=	Write(Byte[], Int32, Int32)	Write data to the stream.
		(Overrides <u>Stream.Write(Byte[], Int32, Int32</u>).)
≡	WriteByte(Byte)	Writes a byte to the current position in the stream and advances the position within the stream by one byte.
		(Inherited from <u>Stream</u> .)
	WriteTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to write before timing out.
		(Inherited from <u>Stream</u> .)

■ Inheritance Hierarchy

Object

MarshalByRefObject

<u>Stream</u>

BZip2OutputStream

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BZip2OutputStream Constructor

Reference ► Ionic.BZip2 ► BZip2OutputStream ►

BZip2OutputStream()

■ Members



Icon	Member	Description
€₩	BZip2OutputStream(Stream)	Constructs a new BZip2OutputStream, that sends its compressed output to the given output stream.
ΞΦ	BZip2OutputStream(Stream, Int32)	Constructs a new BZip2OutputStream with specified blocksize.
≅	BZip2OutputStream(Stream, Boolean)	Constructs a new BZip2OutputStream.
∃	BZip2OutputStream(Stream, Int32, Boolean)	Constructs a new BZip2OutputStream with specified blocksize, and explicitly specifies whether to leave the wrapped stream open.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BZip2OutputStream Constructor

(output)

Reference ► Ionic.BZip2 ► BZip2OutputStream ►

BZip2OutputStream(Stream)

Ç#

Constructs a new BZip20utputStream, that sends its compressed output to the given output stream.

Declaration Syntax

Parameters

output (Stream)

The destination stream, to which compressed output will be sent.

Examples

This example reads a file, then compresses it with bzip2 file, and writes the compressed data into a newly created file.

```
C# Cop
```

```
var fname = "logfile.log";
using (var fs = File.OpenRead(fname))
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

Ionic Zip Library v1.9.1.6 BZip2OutputStream Constructor

(output, leaveOpen)

Reference ► Ionic.BZip2 ► BZip2OutputStream ► BZip2OutputStream(Stream, Boolean)

Constructs a new BZip2OutputStream.



Declaration Syntax

Parameters

output (Stream)

the destination stream.

leaveOpen (Boolean)

whether to leave the captive stream open upon closing this stream.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BZip2OutputStream Constructor

(output, blockSize)

Reference ► Ionic.BZip2 ► BZip2OutputStream ► BZip2OutputStream(Stream, Int32)

Constructs a new BZip20utputStream with specified blocksize.

Declaration Syntax

```
Public Sub New ( __ output As Stream, _ blockSize As Integer _ )
```

Parameters

output (Stream)

the destination stream.

blockSize (Int32)

The blockSize in units of 100000 bytes. The valid range is 1..9.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6BZip2OutputStream Constructor (output, blockSize, leaveOpen)

Reference ► Ionic.BZip2 ► BZip2OutputStream ► BZip2OutputStream(Stream, Int32, Boolean)

Constructs a new BZip20utputStream with specified blocksize, and explicitly specifies whether to leave the wrapped stream open.

Declaration Syntax

Parameters

```
output (Stream)
```

the destination stream.

blockSize (Int32)

The blockSize in units of 100000 bytes. The valid range is 1..9.

leaveOpen (Boolean)

whether to leave the captive stream open upon closing this stream.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BlockSize Property

Reference ► Ionic.BZip2 ► BZip2OutputStream ► BlockSize

The blocksize parameter specified at construction time.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public int BlockSize { get; }

Public ReadOnly Property BlockSize As Integer
    Get

public:
property int BlockSize {
    int get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.BZip2 ► BZip2OutputStream ► CanRead

Indicates whether the stream can be read.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:
virtual property bool CanRead {
    bool get () override;
}
```

■ Remarks

The return value is always false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Can Seek Property

Reference ► Ionic.BZip2 ► BZip2OutputStream ► CanSeek

Indicates whether the stream supports Seek operations.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:

virtual property bool CanSeek {

bool get () override;
```

■ Remarks

}

Always returns false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanWrite Property

Reference ► Ionic.BZip2 ► BZip2OutputStream ► CanWrite

Indicates whether the stream can be written.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
    virtual property bool CanWrite {
        bool get () override;
}
```

C#

■ Remarks

The return value should always be true, unless and until the object is disposed and closed.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Close Method

Reference ► Ionic.BZip2 ► BZip2OutputStream ► Close()

Close the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Close()

Public Overrides Sub Close

public:
virtual void Close() override

C#

■ Remarks

This may or may not close the underlying stream. Check the constructo that accept a bool value.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Method

Reference ► Ionic.BZip2 ► BZip2OutputStream ► Flush()

Flush the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

public:
virtual void Flush() override

C#

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Length Property

Reference ► Ionic.BZip2 ► BZip2OutputStream ► Length

C#

Reading this property always throws a NotImplemented Exception.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:

virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Position Property

Reference ► Ionic.BZip2 ► BZip2OutputStream ► Position

The position of the stream pointer.

■ Declaration Syntax

C#

■ Remarks

Setting this property always throws a <u>NotImplementedException</u>. Reading will return the total number of uncompressed bytes written through.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

Reference ► Ionic.BZip2 ► BZip2OutputStream ► Read(Byte[], Int32, Int32)

Calling this method always throws a NotImplementedException.

Declaration Syntax

```
C# Visual Basic Visual C++

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

Parameters

```
buffer (Byte[])
this parameter is never used

offset (Int32)
this parameter is never used

count (Int32)
this parameter is never used
```

Return Value

never returns anything; always throws

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.BZip2 ► BZip2OutputStream ► Seek(Int64, SeekOrigin)

Calling this method always throws a NotImplementedException.

Declaration Syntax

```
C# Visual Basic Visual C++

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

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

Parameters

offset (Int64)

this is irrelevant, since it will always throw!

origin (SeekOrigin)

this is irrelevant, since it will always throw!

Return Value

irrelevant!

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.BZip2 ► BZip2OutputStream ► SetLength(Int64)

Calling this method always throws a NotImplementedException.

■ Declaration Syntax

Parameters

value (<u>Int64</u>)

this is irrelevant, since it will always throw!

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

Reference ► Ionic.BZip2 ► BZip2OutputStream ► Write(Byte[], Int32, Int32)

Write data to the stream.

C# _

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

buffer (Byte[])

The buffer holding data to write to the stream.

offset (Int32)

the offset within that data array to find the first byte to write.

count (Int32)

the number of bytes to write.

■ Remarks

Use the BZip2OutputStream to compress data while writing: crea a BZip2OutputStream with a writable output stream. Then call Write() on that BZip2OutputStream, providing uncompressed data as input. The data sent to the output stream will be the compresse form of the input data.

A BZip2OutputStream can be used only for Write() not for Read().

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel BZip 2 Output Stream Class

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream

A write-only decorator stream that compresses data as it is written using the BZip2 algorithm. This stream compresses by block using multiple threads.

Declaration Syntax

C# Visual Basic Visual C++
public class ParallelBZip2OutputStream : <u>Stream</u>

Public Class ParallelBZip2OutputStream _ Inherits <u>Stream</u>

public ref class ParallelBZip2OutputStream : public

■ Members

All M	embers	Constructors	Methods	Prope	rties	
✓ Public✓ Protected		✓ Instanc✓ Static	e		Declare Inherite	
lcon	Member	-		Description	on	
€₩	ParallelBZip2OutputStream(Stream)		Constructs a new ParallelBZip2Outputhat sends its compressed of given output stream.			
≓	ParallelBZip2OutputStream(Stream, Int32)		Constructs Paralle with specif	elBZip2	•	

=∳	ParallelBZip2OutputStream(Stream, Boolean)	Constructs a new ParallelBZip2Outpu
₫ ₩	ParallelBZip2OutputStream(Stream, Int32, Boolean)	Constructs a new ParallelBZip2Outpt with specified blocksize, an specifies whether to leave t stream open.
Ξ ψ	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous re operation. (Inherited from <u>Stream</u> .)
€	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous will operation. (Inherited from <u>Stream</u> .)
	BlockSize	The blocksize parameter sp construction time.
	BytesWrittenOut	The total number of bytes withe stream.
	CanRead	Indicates whether the strea read. (Overrides <u>Stream.CanRea</u>
	CanSeek	Indicates whether the strea Seek operations.

	value that determine
	rent stream can time
(minerite	ed from <u>Stream</u> .)
CanWrite Indicate written.	es whether the strea
(Overrice	des <u>Stream</u> . <u>CanWrit</u>
Close the Close	he stream.
(Overrice	des <u>Stream</u> . <u>Close(</u>).)
relevan generat	s an object that contact information require te a proxy used to commote object.
(Inherite	ed from <u>MarshalByR</u>
CreateWaitHandle() Obsolet Allocate	te. es a <u>WaitHandle</u> obje
(Innerite	ed from <u>Stream</u> .)
Dispose() Release Stream	es all resources use
(Inherite	ed from <u>Stream</u> .)
used by	es the unmanaged resources the managed reso
(Inherite	ed from <u>Stream</u> .)
	or the pending async complete.
	od franc Otro ()
(Inherite	ed from <u>Stream</u> .)

(Inherited from Stream.) Determines whether the spobject is equal to the currer (Inherited from Object.) Finalize() Allows an Object to attempt resources and perform othe operations before the Object reclaimed by garbage coller (Inherited from Object.) Flush() Flush the stream. (Overrides Stream Flush().) GetHashCode() Serves as a hash function f particular type. (Inherited from Object.) Retrieves the current lifetim object that controls the lifetif for this instance. (Inherited from MarshalByR GetType() Gets the Type of the curren (Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway NotImplementedException.		EndWrite(IAsyncResult)	Ends an asynchronous writ
Determines whether the spr Object is equal to the currer (Inherited from Object.) Allows an Object to attempt resources and perform othe operations before the Object reclaimed by garbage coller (Inherited from Object.) Flush() Flush the stream. (Overrides Stream.Flush().) GetHashCode() Serves as a hash function f particular type. (Inherited from Object.) Retrieves the current lifetim object that controls the lifetifor this instance. (Inherited from MarshalByR GetType() Gets the Type of the curren (Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway			(Inherited from <u>Stream</u> .)
Finalize() Allows an Object to attempt resources and perform othe operations before the Object reclaimed by garbage collection (Inherited from Object.) Flush() Flush the stream. (Overrides Stream.Flush().) GetHashCode() Serves as a hash function f particular type. (Inherited from Object.) GetLifetimeService() Retrieves the current lifetim object that controls the lifetif for this instance. (Inherited from MarshalByR GetType() Gets the Type of the current (Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway	≡	Equals(Object)	·
resources and perform othe operations before the Object reclaimed by garbage collections before the Object.) Flush() Flush the stream. (Overrides Stream.Flush().) GetHashCode() Serves as a hash function f particular type. (Inherited from Object.) Retrieves the current lifetim object that controls the lifetifor this instance. (Inherited from MarshalByR GetType() Gets the Type of the current (Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway			(Inherited from Object.)
Flush() Flush the stream. (Overrides Stream.Flush().) GetHashCode() Serves as a hash function f particular type. (Inherited from Object.) Retrieves the current lifetim object that controls the lifeti for this instance. (Inherited from MarshalByR GetType() Gets the Type of the curren (Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway	₹	Finalize()	resources and perform othe operations before the Object reclaimed by garbage collections.
(Overrides Stream.Flush().) GetHashCode() Serves as a hash function f particular type. (Inherited from Object.) Retrieves the current lifetim object that controls the lifeti for this instance. (Inherited from MarshalByR GetType() Gets the Type of the curren (Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway	=♦	Flush()	, ,
GetLifetimeService() GetLifetimeService() GetLifetimeService() GetLifetimeService() Retrieves the current lifetim object that controls the lifeti for this instance. (Inherited from MarshalByR GetType() Gets the Type of the curren (Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway		 -	(Overrides <u>Stream.Flush()</u> .)
GetLifetimeService() Retrieves the current lifetim object that controls the lifeti for this instance. (Inherited from MarshalByR GetType() Gets the Type of the curren (Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway	≡	GetHashCode()	particular type.
object that controls the lifeti for this instance. (Inherited from MarshalByR GetType() Gets the Type of the curren (Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway	= (,
GetType() Gets the Type of the curren (Inherited from Object.) Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Reading this property alway		GetLifetimeService()	object that controls the lifeti
(Inherited from Object.) InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Length Reading this property alway			(Inherited from MarshalByR
InitializeLifetimeService() Obtains a lifetime service o control the lifetime policy fo instance. (Inherited from MarshalByR Length Reading this property alway	∉	GetType()	Gets the Type of the curren
Control the lifetime policy fo instance. (Inherited from MarshalByR Length Reading this property alway			(Inherited from Object.)
Ength Reading this property alway	€	InitializeLifetimeService()	control the lifetime policy fo instance.
		Longth	
		Lengu	

		(Overrides <u>Stream</u> . <u>Length</u> .)
	MaxWorkers	The maximum number of compression worker thread
Ģ	MemberwiseClone(Boolean)	Creates a shallow copy of the MarshalByRefObject object
		(Inherited from MarshalByR
	MemberwiseClone()	Creates a shallow copy of to Object.
		(Inherited from Object.)
	Position	The position of the stream p
		(Overrides <u>Stream</u> . <u>Position</u>
≅	Read(Byte[], Int32, Int32)	Calling this method always NotImplementedException. (Overrides Stream.Read(B)
		Int32).)
≡	ReadByte()	Reads a byte from the strea advances the position within by one byte, or returns -1 if of the stream.
		(Inherited from <u>Stream</u> .)
	ReadTimeout	Gets or sets a value, in mili- that determines how long the will attempt to read before t
		(Inherited from <u>Stream</u> .)
= ♦	Seek(Int64, SeekOrigin)	Calling this method always NotImplementedException.
		(Overrides Stream.Seek(Int

		SeekOrigin).)
≡	SetLength(Int64)	Calling this method always NotImplementedException.
		(Overrides Stream.SetLeng
=	ToString()	Returns a <u>String</u> that repres current <u>Object</u> .
		(Inherited from Object.)
∉	Write(Byte[], Int32, Int32)	Write data to the stream.
		(Overrides <u>Stream.Write(By Int32)</u> .)
≅©	WriteByte(Byte)	Writes a byte to the current the stream and advances the within the stream by one by
		(Inherited from <u>Stream</u> .)
	WriteTimeout	Gets or sets a value, in mili- that determines how long the will attempt to write before t
		(Inherited from <u>Stream</u> .)

■ Inheritance Hierarchy

Object

MarshalByRefObject

Stream

ParallelBZip2OutputStream

■ See Also

BZip2OutputStream

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa | 4 | _______**>** |

${\sf Ionic\ Zip\ Library\ v1.9.1.6} \\ {\color{red}\textbf{ParallelBZip2OutputStream}}$

Constructor

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

ParallelBZip2OutputStream()

■ Members

Icon	Member	Description
∃ ₩	ParallelBZip2OutputStream(Stream)	Constructs a new ParallelBZip2Outputhat sends its compressed of given output stream.
≓	ParallelBZip2OutputStream(Stream, Int32)	Constructs a new ParallelBZip2Outpu with specified blocksize.
≅	ParallelBZip2OutputStream(Stream, Boolean)	Constructs a new ParallelBZip2Outpu
≅	ParallelBZip2OutputStream(Stream, Int32, Boolean)	Constructs a new ParallelBZip2Outpu with specified blocksize, an specifies whether to leave t stream open.

C#

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

•

Ionic Zip Library v1.9.1.6 Parallel BZip 2 Output Stream

Constructor (output)

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

ParallelBZip2OutputStream(Stream)

Constructs a new ParallelBZip2OutputStream, that sends its compressed output to the given output stream.

Declaration Syntax

Parameters

output (Stream)

The destination stream, to which compressed output will be sent.

Examples

This example reads a file, then compresses it with bzip2 file, and writes the compressed data into a newly created file.

C# Cop

```
var fname = "logfile.log";
using (var fs = File.OpenRead(fname))
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Parallel BZip 2 Output Stream

Constructor (output, leaveOpen)

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

ParallelBZip2OutputStream(Stream, Boolean)

C# •

Constructs a new ParallelBZip2OutputStream.

Declaration Syntax

Parameters

output (Stream)

the destination stream.

leaveOpen (Boolean)

whether to leave the captive stream open upon closing this stream.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel BZip 2 Output Stream

Constructor (output, blockSize)

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

ParallelBZip2OutputStream(Stream, Int32)

C#... ... 🔽

Constructs a new ParallelBZip2OutputStream with specified blocksize.

Declaration Syntax

Parameters

```
output (Stream)
```

the destination stream.

blockSize (Int32)

The blockSize in units of 100000 bytes. The valid range is 1..9.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ParallelBZip2OutputStream

Constructor (output, blockSize, leaveOpen)

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

ParallelBZip2OutputStream(Stream, Int32, Boolean)

Constructs a new ParallelBZip2OutputStream with specified blocksize, and explicitly specifies whether to leave the wrapped stream open.

Declaration Syntax

Parameters

```
output (Stream)
the destination stream.
blockSize (Int32)
```

The blockSize in units of 100000 bytes. The valid range is 1..9.

leaveOpen (Boolean)

whether to leave the captive stream open upon closing this stream.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BlockSize Property

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ► BlockSize

The blocksize parameter specified at construction time.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public int BlockSize { get; }

Public ReadOnly Property BlockSize As Integer
    Get

public:
property int BlockSize {
    int get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BytesWrittenOut Property

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

BytesWrittenOut

The total number of bytes written out by the stream.

C# -

■ Declaration Syntax

```
C# Visual Basic Visual C++
public long BytesWrittenOut { get; }

Public ReadOnly Property BytesWrittenOut As Long Get

public:
property long long BytesWrittenOut {
    long long get ();
}
```

■ Remarks

This value is meaningful only after a call to Close().

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanRead Property

<u>Reference</u> ► <u>Ionic.BZip2</u> ► <u>ParallelBZip2OutputStream</u> ► **CanRead**

Indicates whether the stream can be read.

C# _

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:

virtual property bool CanRead {

bool get () override;
}
```

■ Remarks

The return value is always false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Can Seek Property

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ► CanSeek

Indicates whether the stream supports Seek operations.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:
virtual property bool CanSeek {
    bool get () override;
}
```

■ Remarks

Always returns false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanWrite Property

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ► CanWrite

Indicates whether the stream can be written.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
virtual property bool CanWrite {
    bool get () override;
}
```

C#

■ Remarks

The return value depends on whether the captive stream supports writing.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Close Method

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ► Close()

Close the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Close()

Public Overrides Sub Close

public:
virtual void Close() override

C#

■ Remarks

This may or may not close the underlying stream. Check the constructo that accept a bool value.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Method

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ► Flush()

Flush the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

C#

public: virtual void Flush() override

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Length Property

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ► Length

Reading this property always throws a NotImplemented Exception.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:
virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 MaxWorkers Property

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

MaxWorkers

The maximum number of concurrent compression worker threads to use

■ Declaration Syntax

■ Remarks

This property sets an upper limit on the number of concurrent worker threads to employ for compression. The implementation of this stream employs multiple threads from the .NET thread pool, via <a href="https://doi.org/10.2016/nc.

A higher number of workers enables a higher degree of parallelism, which tends to increase the speed of compression on multi-cpu computers. On the other hand, a higher number of buffer pairs also implies a larger memory consumption, more active worker threads, and higher cpu utilization for any compression. This property enables the application to limit its memory consumption and CPU utilization behavio depending on requirements.

By default, DotNetZip allocates 4 workers per CPU core, subject to the upper limit specified in this property. For example, suppose the application sets this property to 16. Then, on a machine with 2 cores, DotNetZip will use 8 workers; that number does not exceed the upper limit specified by this property, so the actual number of workers used wi be 4 * 2 = 8. On a machine with 4 cores, DotNetZip will use 16 workers; again, the limit does not apply. On a machine with 8 cores, DotNetZip w use 16 workers, because of the limit.

For each compression "worker thread" that occurs in parallel, there is up to 2mb of memory allocated, for buffering and processing. The actual number depends on the <u>BlockSize</u> property.

CPU utilization will also go up with additional workers, because a larger number of buffer pairs allows a larger number of background threads to compress in parallel. If you find that parallel compression is consuming too much memory or CPU, you can adjust this value downward.

The default value is 16. Different values may deliver better or worse results, depending on your priorities and the dynamic performance characteristics of your storage and compute resources.

The application can set this value at any time, but it is effective only before the first call to Write(), which is when the buffers are allocated.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Position Property

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ► Position

C#

The position of the stream pointer.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

■ Remarks

Setting this property always throws a <u>NotImplementedException</u>. Reading will return the total number of uncompressed bytes written through.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

Read(Byte[], Int32, Int32)

Calling this method always throws a NotImplementedException.

Declaration Syntax

```
C# Visual Basic Visual C++

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

Parameters

```
buffer (Byte[])
  this parameter is never used

offset (Int32)
  this parameter is never used
```

count (Int32)

this parameter is never used

Return Value

never returns anything; always throws

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ► Seek(Int64, SeekOrigin)

Calling this method always throws a NotImplementedException.

Declaration Syntax

```
C# Visual Basic Visual C++

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

```
Public Overrides Function Seek ( _ offset As Long, _ origin As SeekOrigin _ ) As Long
```

Parameters

offset (Int64)

this is irrelevant, since it will always throw!

origin (SeekOrigin)

this is irrelevant, since it will always throw!

Return Value

irrelevant!

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

SetLength(Int64)

Calling this method always throws a NotImplementedException.

■ Declaration Syntax

Parameters

value (Int64)

this is irrelevant, since it will always throw!

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

Reference ► Ionic.BZip2 ► ParallelBZip2OutputStream ►

Write(Byte[], Int32, Int32)

Write data to the stream.

C#

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

buffer (Byte[])

The buffer holding data to write to the stream.

offset (Int32)

the offset within that data array to find the first byte to write.

count (Int32)

the number of bytes to write.

■ Remarks

Use the ParallelBZip2OutputStream to compress data while writing: create a ParallelBZip2OutputStream with a writable output stream. Then call Write() on that

ParallelBZip2OutputStream, providing uncompressed data ϵ input. The data sent to the output stream will be the compressed form o the input data.

A ParallelBZip2OutputStream can be used only for Write() not for Read().

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Ionic. Crc Namespace

Reference ► Ionic.Crc

C#

The Ionic.Crc namespace includes a classes that calculate 32-bit Cyclic Redundancy Checks (CRC32) for data.

■ Declaration Syntax

C#	Visual Basic	Visual C++
namespace Ionic.	Crc	
Namespace Ionic.	Crc	
namespace Ionic.	Crc	

Types

All Ty	rpes	Classes		
Icon	Туре		Description	
9 3	CRC32		algorithm is p set the polyn	CRC-32. The CRC-32 parameterized - you can omial and enable or versal. This can be used p2, or ZIP.
9 3	CrcCalculatorSt	ream		nt calculates a CRC32 (and all bytes read, or on a

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

lonic Zip Library v1.9.1.6 CRC32 Class Reference ► lonic.Crc ► CRC32 C#

Computes a CRC-32. The CRC-32 algorithm is parameterized - you can set the polynomial and enable or disable bit reversal. This can be used for GZIP, BZip2, or ZIP.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public class CRC	32	
Public Class CRC	32	
public ref class	CRC32	

■ Members

All M	embers	Constructors	M	lethods	Properties	
Pu Pro Pro	blic otected			Instance Static		Declare Inherite
Icon	Member			Description	1	
∄	CRC32())		class using	istance of the C the default settir and a polynomi 0.	ngs: no
₫	CRC32(Boolean)			istance of the C fying whether to not.	
≡	CRC32 (Int32, Boolean)		Create an in	stance of the C	RC32

		class, specifying the polynomial and whether to reverse data bits or not.
≟∳	Combine(Int32, Int32)	Combines the given CRC32 value with the current running total.
₫ 🚱	ComputeCrc32(Int32, Byte)	Get the CRC32 for the given (word,byte) combo. This is a computation defined by PKzip for PKZIP 2.0 (weak) encryption.
	Crc32Result	Indicates the current CRC for all blocks slurped in.
∃	Equals(Object)	Determines whether the specified Object is equal to the current Object (Inherited from Object .)
Ģ	Finalize()	Allows an <u>Object</u> to attempt to free resources and perform other cleanu operations before the <u>Object</u> is reclaimed by garbage collection. (Inherited from <u>Object</u> .)
≡•	GetCrc32(Stream)	Returns the CRC32 for the specified stream.
€ \	GetCrc32AndCopy(Stream, Stream)	Returns the CRC32 for the specified stream, and writes the input into the output stream.

≡©	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
≟	GetType()	Gets the <u>Type</u> of the current instance. (Inherited from <u>Object</u> .)
·	MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object.)
≟	Reset()	Reset the CRC-32 class - clear the CRC "remainder register."
≅ 🍑	SlurpBlock(Byte[], Int32, Int32)	Update the value for the running CRC32 using the given block of bytes. This is useful when using the CRC32() class in a Stream.
≡••	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
	TotalBytesRead	Indicates the total number of bytes applied to the CRC.
∄	<u>UpdateCRC(Byte)</u>	Process one byte in the CRC.
=0	UpdateCRC(Byte, Int32)	Process a run of N identical bytes

into the CRC.

■ Remarks

This type is used internally by DotNetZip; it is generally not used directly by applications wishing to create, read, or manipulate zip archive files.

■ Inheritance Hierarchy

Object

CRC32

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CRC32 Constructor

Reference ► Ionic.Crc ► CRC32 ► CRC32()

C#

■ Members

Icon	Member	Description
∃	CRC32()	Create an instance of the CRC32 class using the default settings: no birreversal, and a polynomial of 0xEDB88320.
≓	CRC32(Boolean)	Create an instance of the CRC32 class, specifying whether to reverse data bits or not.
∃	CRC32(Int32, Boolean)	Create an instance of the CRC32 class, specifying the polynomial and whether to reverse data bits or not.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 CRC32 Constructor | Reference ► Ionic.Crc ► CRC32 ► CRC32() | C#

Create an instance of the CRC32 class using the default settings: no bit reversal, and a polynomial of 0xEDB88320.

■ Declaration Syntax

C#	Visual Basic	Visual C++
<pre>public CRC32()</pre>		
Public Sub New		
<pre>public: CRC32()</pre>		

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CRC32 Constructor (reverseBits)

Reference ► Ionic.Crc ► CRC32 ► CRC32(Boolean)

C#

Create an instance of the CRC32 class, specifying whether to reverse data bits or not.

Declaration Syntax

Parameters

reverseBits (Boolean)

specify true if the instance should reverse data bits.

■ Remarks

In the CRC-32 used by BZip2, the bits are reversed. Therefore if you want a CRC32 with compatibility with BZip2, you should pass true here. In the CRC-32 used by GZIP and PKZIP, the bits are not reversed; Therefore if you want a CRC32 with compatibility with those, you should pass false.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CRC32 Constructor (polynomial,

reverseBits)

Reference ► Ionic.Crc ► CRC32 ► CRC32(Int32, Boolean)

Create an instance of the CRC32 class, specifying the polynomial and whether to reverse data bits or not.

■ Declaration Syntax

Parameters

polynomial (<u>Int32</u>)

The polynomial to use for the CRC, expressed in the reversed (LSE format: the highest ordered bit in the polynomial value is the coefficient of the 0th power; the second-highest order bit is the coefficient of the 1 power, and so on. Expressed this way, the polynomial for the CRC-32C used in IEEE 802.3, is 0xEDB88320.

reverseBits (Boolean)

specify true if the instance should reverse data bits.

■ Remarks

In the CRC-32 used by BZip2, the bits are reversed. Therefore if you want a CRC32 with compatibility with BZip2, you should pass true here for the reverseBits parameter. In the CRC-32 used by GZIP and PKZIP, the bits are not reversed; Therefore if you want a CRC32 with compatibility with those, you should pass false for the reverseBits parameter.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Combine Method (crc, length)

Reference ► Ionic.Crc ► CRC32 ► Combine(Int32, Int32)

C#, , ,

Combines the given CRC32 value with the current running total.

Declaration Syntax

Parameters

crc (<u>Int32</u>)

the crc value to be combined with this one

length (Int32)

the length of data the CRC value was calculated on

Remarks

This is useful when using a divide-and-conquer approach to calculating CRC. Multiple threads can each calculate a CRC32 on a segment of the data, and then combine the individual CRC32 values at the end.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compute Crc32 Method (W, B)

Reference ► Ionic.Crc ► CRC32 ► ComputeCrc32(Int32, Byte)

C# •

Get the CRC32 for the given (word,byte) combo. This is a computation defined by PKzip for PKZIP 2.0 (weak) encryption.

Declaration Syntax

```
C# Visual Basic Visual C++

public int ComputeCrc32(
    int W,
    byte B
)
```

```
public:
int ComputeCrc32(
    int W,
    unsigned char B
)
```

Parameters

```
W (Int32)
```

The word to start with.

B (Byte)

The byte to combine it with.

Return Value

The CRC-ized result.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Crc32Result Property

Reference ► Ionic.Crc ► CRC32 ► Crc32Result

C#

Indicates the current CRC for all blocks slurped in.

Declaration Syntax

```
C# Visual Basic Visual C++
public int Crc32Result { get; }

Public ReadOnly Property Crc32Result As Integer
   Get

public:
property int Crc32Result {
   int get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **GetCrc32 Method (input)**

Reference ► Ionic.Crc ► CRC32 ► GetCrc32(Stream) C#

C#

Returns the CRC32 for the specified stream.

■ Declaration Syntax

Parameters

input (Stream)

The stream over which to calculate the CRC32

■ Return Value

the CRC32 calculation

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 GetCrc32AndCopy Method (input,

output)

Reference ► Ionic.Crc ► CRC32 ► GetCrc32AndCopy(Stream, Stream)

Returns the CRC32 for the specified stream, and writes the input into th output stream.

Declaration Syntax

```
C# Visual Basic Visual C++

public int GetCrc32AndCopy(
    Stream input,
    Stream output
)
```

```
Public Function GetCrc32AndCopy ( _______input As Stream, ______output As Stream ______)

As Integer
```

Parameters

```
input (Stream)
```

The stream over which to calculate the CRC32

output (Stream)

The stream into which to deflate the input

■ Return Value

the CRC32 calculation

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Reset the CRC-32 class - clear the CRC "remainder register."

■ Declaration Syntax

```
C# Visual Basic Visual C++

public void Reset()

Public Sub Reset

public:
void Reset()
```

■ Remarks

Use this when employing a single instance of this class to compute multiple, distinct CRCs on multiple, distinct data blocks.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **SlurpBlock Method (block, offset, count)**

Reference ► Ionic.Crc ► CRC32 ► SlurpBlock(Byte[], Int32, Int32)

Update the value for the running CRC32 using the given block of bytes. This is useful when using the CRC32() class in a Stream.

■ Declaration Syntax

```
Public Sub SlurpBlock ( __ block As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

```
block (Byte[])
block of bytes to slurp

offset (Int32)
starting point in the block

count (Int32)
```

how many bytes within the block to slurp

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Total Bytes Read Property

Reference ► Ionic.Crc ► CRC32 ► TotalBytesRead

C#

Indicates the total number of bytes applied to the CRC.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public long TotalBytesRead { get; }

Public ReadOnly Property TotalBytesRead As Long
    Get

public:
property long long TotalBytesRead {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Update CRC Method

Reference ► Ionic.Crc ► CRC32 ► UpdateCRC()

C#

■ Members

Icon	Member	Description
≡	<u>UpdateCRC(Byte)</u>	Process one byte in the CRC.
≡	<u>UpdateCRC(Byte, Int32)</u>	Process a run of N identical bytes into the CRC.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Update CRC Method (b)

Reference ► Ionic.Crc ► CRC32 ► UpdateCRC(Byte) C#

C# _

Process one byte in the CRC.

Declaration Syntax

Parameters

b (Byte)

the byte to include into the CRC .

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Update CRC Method (b, n)

Reference ► Ionic.Crc ► CRC32 ► UpdateCRC(Byte, Int32)

Process a run of N identical bytes into the CRC.

■ Declaration Syntax

Parameters

b (Byte)

the byte to include into the CRC.

unsigned char b,

int *n*

n (Int32)

the number of times that byte should be repeated.

■ Remarks

This method serves as an optimization for updating the CRC when a rul of identical bytes is found. Rather than passing in a buffer of length n, containing all identical bytes b, this method accepts the byte value and the length of the (virtual) buffer - the length of the run.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Crc Calculator Stream Class

Reference ► <u>lonic.Crc</u> ► CrcCalculatorStream

C# -

A Stream that calculates a CRC32 (a checksum) on all bytes read, or or all bytes written.

Declaration Syntax

```
Public Class CrcCalculatorStream _ Inherits Stream _ Implements IDisposable
```

■ Members

All M	embers	Constructors	Me	thods	Properties	
✓ Pu ✓ Pro	blic otected			nstance Static		Declare Inherite
Icon	Member	-		Description	on	
≡	CrcCalc	ulatorStream(Strea	am)	The defau	It constructor.	
=	CrcCalculatorStream(Stream, Boolean)		specify ho	ructor allows the w to handle the g stream at close		
≡	CrcCalc	ulatorStream(Strea	am,	A construc	ctor allowing the	

	<u>Int64</u>)	specification of the length of the stream to read.
a ⊘	CrcCalculatorStream(Stream, Int64, Boolean)	A constructor allowing the specification of the length of the stream to read, as well as whethe to keep the underlying stream open upon Close().
≡₩	CrcCalculatorStream(Stream, Int64, Boolean, CRC32)	A constructor allowing the specification of the length of the stream to read, as well as whethe to keep the underlying stream open upon Close(), and the CRC32 instance to use.
≡ ©	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous read operation. (Inherited from <u>Stream</u> .)
≡ ©	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous write operation. (Inherited from <u>Stream</u> .)
	CanRead	Indicates whether the stream supports reading. (Overrides Stream.CanRead .)
	CanSeek	Indicates whether the stream supports seeking.
		(Overrides <u>Stream.CanSeek.</u>)

	CanTimeout	Gets a value that determines whether the current stream can time out. (Inherited from <u>Stream</u> .)
	<u>CanWrite</u>	Indicates whether the stream supports writing. (Overrides <u>Stream.CanWrite.</u>)
=0	Close()	Closes the stream. (Overrides <u>Stream.Close()</u> .)
	Crc	Provides the current CRC for all blocks slurped in.
∃	<u>CreateObjRef(Type)</u>	Creates an object that contains all the relevant information required t generate a proxy used to communicate with a remote object (Inherited from MarshalByRefObject.)
Ģ [©]	CreateWaitHandle()	Obsolete. Allocates a <u>WaitHandle</u> object. (Inherited from <u>Stream</u> .)
=	Dispose()	Releases all resources used by th Stream . (Inherited from Stream .)
⊕	<u>Dispose(Boolean)</u>	Releases the unmanaged resources used by the <u>Stream</u> and optionally releases the managed resources.

		(Inherited from <u>Stream</u> .)
≅©	EndRead(IAsyncResult)	Waits for the pending asynchronous read to complete. (Inherited from <u>Stream</u> .)
=	EndWrite(IAsyncResult)	Ends an asynchronous write operation. (Inherited from <u>Stream</u> .)
≟	Equals(Object)	Determines whether the specified Object is equal to the current Object . (Inherited from Object .)
*	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
=	Flush()	Flush the stream. (Overrides <u>Stream.Flush()</u> .)
=0	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
≓	GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
≓	GetType()	Gets the <u>Type</u> of the current instance.

		(Inherited from Object.)
≡	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance.
		(Inherited from MarshalByRefObject.)
	LeaveOpen	Indicates whether the underlying stream will be left open when the CrcCalculatorStream is Closed.
	<u>Length</u>	Returns the length of the underlying stream.
		(Overrides <u>Stream</u> . <u>Length</u> .)
Ģ [¯] ©	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object.
		(Inherited from MarshalByRefObject.)
Ģ [¯] ♥	MemberwiseClone()	Creates a shallow copy of the current <u>Object</u> . (Inherited from <u>Object</u> .)
	Position	The getter for this property returns the total bytes read. If you use the setter, it will throw NotSupportedException. (Overrides Stream.Position.)
≡	Read(Byte[], Int32, Int32)	Read from the stream
	rodulpytog, mtoz, mtoz)	road nom the stroum

		(Overrides <u>Stream.Read(Byte[], Int32, Int32)</u> .)
Ξ Φ	ReadByte()	Reads a byte from the stream and advances the position within the stream by one byte, or returns -1 i at the end of the stream. (Inherited from Stream .)
	D 17	,
	ReadTimeout	Gets or sets a value, in miliseconds, that determines how long the stream will attempt to rea before timing out.
		(Inherited from <u>Stream</u> .)
a ©	Seek(Int64, SeekOrigin)	Seeking is not supported on this stream. This method always throws NotSupportedException (Overrides Stream.Seek(Int64, SeekOrigin).)
≅	SetLength(Int64)	This method always throws NotSupportedException
		(Overrides <u>Stream.SetLength(Int64)</u> .)
₫ ڼ	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> .
		(Inherited from Object.)
	<u>TotalBytesSlurped</u>	Gets the total number of bytes rur through the CRC32 calculator.
≡	Write(Byte[], Int32, Int32)	Write to the stream.
		(Overrides <u>Stream.Write(Byte[]</u> ,

		<u>Int32, Int32</u>).)
∄	WriteByte(Byte)	Writes a byte to the current position in the stream and advances the position within the stream by one byte. (Inherited from Stream.)
	WriteTimeout	Gets or sets a value, in miliseconds, that determines how long the stream will attempt to write before timing out. (Inherited from Stream .)

■ Remarks

This class can be used to verify the CRC of a ZipEntry when reading from a stream, or to calculate a CRC when writing to a stream. The stream should be used to either read, or write, but not both. If you intermix reads and writes, the results are not defined.

This class is intended primarily for use internally by the DotNetZip librar

■ Inheritance Hierarchy

Object

MarshalByRefObject

Stream

CrcCalculatorStream

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CrcCalculator Stream Constructor

CrcCalculatorStream(Stream) The default constructor.

<u>CrcCalculatorStream(Stream,</u> A constructor allowing the

<u>CrcCalculatorStream(Stream,</u> A constructor allowing the

CrcCalculatorStream(Stream, A constructor allowing the

<u>CrcCalculatorStream(Stream,</u> The constructor allows the caller t

Description

stream to read.

specify how to handle the

underlying stream at close.

specification of the length of the

specification of the length of the stream to read, as well as whethe

to keep the underlying stream

specification of the length of the stream to read, as well as whethe

to keep the underlying stream open upon Close(), and the CRC32 instance to use.

open upon Close().

Reference ► Ionic.Crc ► CrcCalculatorStream ►

CrcCalculatorStream()

Boolean)

<u>Int64</u>)

Int64, Boolean)

Int64, Boolean, CRC32)

Members

Icon Member

C#	

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CrcCalculator Stream Constructor

(stream)

Reference ► <u>lonic.Crc</u> ► <u>CrcCalculatorStream</u> ► CrcCalculatorStream(Stream)

The default constructor.



■ Declaration Syntax

Parameters

stream (Stream)

The underlying stream

■ Remarks

Instances returned from this constructor will leave the underlying stream open upon Close(). The stream uses the default CRC32 algorithm, whic implies a polynomial of 0xEDB88320.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CrcCalculator Stream Constructor (stream, leaveOpen)

Reference ► Ionic.Crc ► CrcCalculatorStream ►

CrcCalculatorStream(Stream, Boolean)

The constructor allows the caller to specify how to handle the underlying stream at close.

Declaration Syntax

```
C# Visual Basic Visual C++

public CrcCalculatorStream(
    Stream stream,
    bool leaveOpen
)
```

Parameters

stream (Stream)

The underlying stream

leaveOpen (Boolean)

true to leave the underlying stream open upon close of the CrcCalculatorStream; false otherwise.

■ Remarks

The stream uses the default CRC32 algorithm, which implies a polynomial of 0xEDB88320.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CrcCalculator Stream Constructor (stream, length)

Reference ► Ionic.Crc ► CrcCalculatorStream ►

CrcCalculatorStream(Stream, Int64)

A constructor allowing the specification of the length of the stream to read.

Declaration Syntax

```
C# Visual Basic Visual C++

public CrcCalculatorStream(
    Stream stream,
    long length
)
```

```
Public Sub New ( __ stream As Stream, _ length As Long _ )
```

Parameters

```
stream (Stream)
The underlying stream

length (Int64)
The length of the stream to slurp
```

■ Remarks

The stream uses the default CRC32 algorithm, which implies a polynomial of 0xEDB88320.

Instances returned from this constructor will leave the underlying stream open upon Close().

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 CrcCalculatorStream Constructor (stream, length, leaveOpen)

Reference ► Ionic.Crc ► CrcCalculatorStream ►

CrcCalculatorStream(Stream, Int64, Boolean)

A constructor allowing the specification of the length of the stream to read, as well as whether to keep the underlying stream open upon Close().

Declaration Syntax

```
C# Visual Basic Visual C++

public CrcCalculatorStream(
    Stream stream,
    long length,
    bool leaveOpen
)
```

```
Public Sub New ( __ stream As Stream, _ length As Long, _ leaveOpen As Boolean _ )
```

Parameters

```
stream (Stream)
The underlying stream

length (Int64)
```

The length of the stream to slurp

leaveOpen (Boolean)

true to leave the underlying stream open upon close of the CrcCalculatorStream; false otherwise.

■ Remarks

The stream uses the default CRC32 algorithm, which implies a polynomial of 0xEDB88320.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CrcCalculatorStream Constructor (stream, length, leaveOpen, crc32)

Reference ► Ionic.Crc ► CrcCalculatorStream ►

CrcCalculatorStream(Stream, Int64, Boolean, CRC32)

A constructor allowing the specification of the length of the stream to read, as well as whether to keep the underlying stream open upon Close(), and the CRC32 instance to use.

Declaration Syntax

```
C# Visual Basic Visual C++

public CrcCalculatorStream(
    Stream stream,
    long length,
    bool leaveOpen,
    CRC32 crc32
)
```

```
Public Sub New ( __ stream As Stream, _ length As Long, _ leaveOpen As Boolean, _ crc32 As CRC32 _ )
```

Parameters

stream (Stream)

The underlying stream

length (Int64)

The length of the stream to slurp

leaveOpen (Boolean)

true to leave the underlying stream open upon close of the CrcCalculatorStream; false otherwise.

crc32 (CRC32)

the CRC32 instance to use to calculate the CRC32

■ Remarks

The stream uses the specified CRC32 instance, which allows the application to specify how the CRC gets calculated.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.Crc ► CrcCalculatorStream ► CanRead

Indicates whether the stream supports reading.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:

virtual property bool CanRead {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Can Seek Property

Reference ► Ionic.Crc ► CrcCalculatorStream ► CanSeek

Indicates whether the stream supports seeking.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:
virtual property bool CanSeek {
    bool get () override;
}
```

■ Remarks

Always returns false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanWrite Property

Reference ► Ionic.Crc ► CrcCalculatorStream ► CanWrite

Indicates whether the stream supports writing.

C# -

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
virtual property bool CanWrite {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Close Method

Reference ► Ionic.Crc ► CrcCalculatorStream ► Close()

Closes the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Close()

Public Overrides Sub Close

public:
virtual void Close() override

C#

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Provides the current CRC for all blocks slurped in.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public int Crc { get; }

Public ReadOnly Property Crc As Integer
    Get

public:
property int Crc {
    int get ();
}
```

Remarks

The running total of the CRC is kept as data is written or read through the stream. read this property after all reads or writes to get an accurate CF for the entire stream.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Method

Reference ► Ionic.Crc ► CrcCalculatorStream ► Flush()

Flush the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

public:
virtual void Flush() override

C#

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 LeaveOpen Property

Reference ► Ionic.Crc ► CrcCalculatorStream ► LeaveOpen

Indicates whether the underlying stream will be left open when the CrcCalculatorStream is Closed.

Declaration Syntax

■ Remarks

Set this at any point before calling **Close()**.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Length Property

Reference ► Ionic.Crc ► CrcCalculatorStream ► Length

Returns the length of the underlying stream.

C# -

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:

virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Position Property

Reference ► Ionic.Crc ► CrcCalculatorStream ► Position

C#

The getter for this property returns the total bytes read. If you use the setter, it will throw NotSupportedException.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

Reference ► Ionic.Crc ► CrcCalculatorStream ► Read(Byte[], Int32, Int32)

Read from the stream

C#

Declaration Syntax

```
C# Visual Basic Visual C++

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

Parameters

```
buffer (Byte[])
  the buffer to read

offset (Int32)
  the offset at which to start

count (Int32)
  the number of bytes to read
```

Return Value

the number of bytes actually read

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.Crc ► CrcCalculatorStream ► Seek(Int64, SeekOrigin)

Seeking is not supported on this stream. This method always throws NotSupportedException

Declaration Syntax

```
C# Visual Basic Visual C++

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

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

Parameters

```
offset (Int64)
N/A
origin (SeekOrigin)
N/A
```

Return Value

N/A

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.Crc ► CrcCalculatorStream ► SetLength(Int64)

C#

This method always throws **NotSupportedException**

■ Declaration Syntax

Parameters

value (Int64) N/A

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Total Bytes Slurped Property

Reference ► Ionic.Crc ► CrcCalculatorStream ► TotalBytesSlurped

Gets the total number of bytes run through the CRC32 calculator.

Declaration Syntax

```
C# Visual Basic Visual C++
public long TotalBytesSlurped { get; }

Public ReadOnly Property TotalBytesSlurped As Long
    Get

public:
property long long TotalBytesSlurped {
    long long get ();
}
```

■ Remarks

This is either the total number of bytes read, or the total number of bytes written, depending on the direction of this stream.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

Reference ► <u>lonic.Crc</u> ► <u>CrcCalculatorStream</u> ► Write(Byte[], Int32, Int32)

Write to the stream.

C# 🔻

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

```
buffer (Byte[])
the buffer from which to write

offset (Int32)
the offset at which to start writing

count (Int32)
the number of bytes to write
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Ionic.Zip Namespace

Reference ► Ionic.Zip

C#

Classes in the Ionic.Zip namespace allow applications to read and write zip files according to the format described by PKWARE

DotNetZip is donationware. Please donate.

■ Declaration Syntax

C#	Visual Basic	Visual C++
namespace Ionic.	Zip	
Namespace Ionic.	Zip	
namespace Ionic.	Zip	

Types

All Ty	pes	Classes	Enumerations	Delegates
Icon	Туре		Description	
4 \$	AddProgi	<u>ressEventArgs</u>	Provides inform Add operation.	nation about the pro
9 \$	BadCrcEx	<u>kception</u>		n CRC check fails u _l ntry from a zip archi
9 \$	BadPassy	wordException		n ExtractWithPa: ked with an incorrec

4 \$	BadReadException	Indicates that a read was attempted stream, and bad or incomplete data received.
4 \$	BadStateException	Indicates that an operation was attera ZipFile which was not possible gistate of the instance. For example, Save() on a ZipFile which has no set, you can get this exception.
	CloseDelegate	Delegate in which the application c stream, just-in-time, for the named
9 \$	ComHelper	This class exposes a set of COM-a wrappers for static methods availal ZipFile class. You don't need this c you are using DotNetZip from a CC environment.
	CompressionMethod	The method of compression to use particular ZipEntry.
4 \$	<u>CountingStream</u>	A decorator stream. It wraps anoth and performs bookkeeping to keep stream Position.
	EncryptionAlgorithm	An enum that provides the various algorithms supported by this library

	ExtractExistingFileAction	An enum for the options when extra entry would overwrite an existing fil
4 3	ExtractProgressEventArgs	Provides information about the project extract operation.
	<u>OpenDelegate</u>	Delegate in which the application o stream, just-in-time, for the named
9 \$	ReadOptions	A class for collecting the various of can be used when Reading zip files extraction or update.
? (\$	ReadProgressEventArgs	Provides information about the pro- Read operation.
P \$	<u>SaveProgressEventArgs</u>	Provides information about the prosave operation.
.	SelfExtractorFlavor	An enum that provides the different extractor flavors
P \$	<u>SelfExtractorSaveOptions</u>	The options for generating a self-exarchive.
3	SetCompressionCallback	Delegate for the callback by which application tells the library the CompressionLevel to use for a file.

P \$	<u>SfxGenerationException</u>	Issued when errors occur saving a extracting archive.
	<u>WriteDelegate</u>	Delegate in which the application w ZipEntry content for the named
	Zip64Option	Options for using ZIP64 extensions saving zip archives.
9 \$	ZipEntry	Represents a single entry in a ZipF Typically, applications get a ZipEnt enumerating the entries within a Zi adding an entry to a ZipFile.
	ZipEntrySource	An enum that specifies the source ZipEntry.
	ZipEntryTimestamp	An enum that specifies the type of available on the ZipEntry.
g ^p	ZipErrorAction	An enum providing the options who occurs during opening or reading o directory that is being saved to a zi
₽ \$	<u>ZipErrorEventArgs</u>	Provides information about the an occurred while zipping.

43	ZipException	Base class for all exceptions define throw by the Zip library.
æş.	<u>ZipFile</u>	The ZipFile type represents a zip a
4	<u>ZipInputStream</u>	Provides a stream metaphor for reafiles.
3	ZipOption	An enum representing the values of way toggle switch for various option library. This might be used to specific to employ a particular text encoding ZIP64 extensions, or some other of
4 3	ZipOutputStream	Provides a stream metaphor for ge files.
4 \$	<u>ZipProgressEventArgs</u>	Provides information about the prograve, read, or extract operation. The class; you will probably use one of derived from this one.
	<u>ZipProgressEventType</u>	In an EventArgs type, indicates wh progress event is being reported.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa



Ionic Zip Library v1.9.1.6 Add Progress Event Args Class

Reference ► Ionic.Zip ► AddProgressEventArgs

C#

Provides information about the progress of a Add operation.

■ Declaration Syntax

C# Visual Basic Visual C++

public class AddProgressEventArgs : ZipProgressEvent

Public Class AddProgressEventArgs _ Inherits ZipProgressEventArgs

public ref class AddProgressEventArgs : public <a>ZipPr

■ Members

All M	embers	ers Methods		Properties	
✓ Public ✓ Protected				✓ Instance✓ Static	Declare Inherite
Icon Member De		Descrip	Description		
	<u>ArchiveName</u>	Returns the archive name ass to this event. (Inherited from ZipProgressEventArgs.)		sociated	
	so far t		The number of bytes read or written so far for this entry. (Inherited from ZipProgressEventArgs.)		written
	Cancel			rent handler, set this t e or extract operation s.	

	(Inherited from ZipProgressEventArgs .)
CurrentEntry	The name of the last entry saved or extracted.
	(Inherited from ZipProgressEventArgs.)
<u>EntriesTotal</u>	The total number of entries to be saved or extracted.
	(Inherited from ZipProgressEventArgs .)
Equals(Object)	Determines whether the specified Object is equal to the current Object .
	(Inherited from Object.)
<u>EventType</u>	The type of event being reported.
	(Inherited from ZipProgressEventArgs.)
Finalize()	Allows an <u>Object</u> to attempt to free resources and perform other cleanup operations before the <u>Object</u> is reclaimed by garbage collection.
	(Inherited from Object.)
GetHashCode()	Serves as a hash function for a particular type.
	(Inherited from Object.)
GetType()	Gets the Type of the current instance
	(Inherited from Object.)
MemberwiseClone()	Creates a shallow copy of the current Object.
	EntriesTotal Equals(Object) EventType Finalize() GetHashCode() GetType()

		(Inherited from Object.)
∃	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
	<u>TotalBytesToTransfer</u>	Total number of bytes that will be read or written for this entry. This number will be -1 if the value cannot be determined. (Inherited from ZipProgressEventArgs.)

■ Inheritance Hierarchy

Object

EventArgs

<u>ZipProgressEventArgs</u>

AddProgressEventArgs

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 BadCrcException Class

Reference ► Ionic.Zip ► BadCrcException

C# -

Issued when an CRC check fails upon extracting an entry from a zip archive.

■ Declaration Syntax

C# Visual Basic Visual C++

[SerializableAttribute]

public class BadCrcException : ZipException

<SerializableAttribute>

Public Class BadCrcException _ Inherits <u>ZipException</u>

[SerializableAttribute]

public ref class BadCrcException : public <a>ZipExcepti

■ Members

All Members	Constructors	Methods	Properties	
Public Protected		✓ Instance✓ Static		Declare Inherite
Joon Mombo	•	D	occription	

Icon	Member	Description
	BadCrcException()	Default ctor.
∃ ₩	BadCrcException(String)	Come on, you know how exceptions work. Why are yo looking at this documentation?

₹	BadCrcException(SerializationInfo, StreamingContext)	Come on, you know how exceptions work. Why are yo looking at this documentation?
	<u>Data</u>	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
€₩	Equals(Object)	Determines whether the specified <u>Object</u> is equal to the current <u>Object</u> . (Inherited from <u>Object</u> .)
₹	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaime by garbage collection. (Inherited from Object.)
≓∲	GetBaseException()	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
=0	GetHashCode()	Serves as a hash function fo a particular type. (Inherited from Object.)
∉	GetObjectData(SerializationInfo, StreamingContext)	When overridden in a derived class, sets the

		SerializationInfo with information about the exception. (Inherited from Exception.)
=	CotTune()	
·	GetType()	Gets the runtime type of the current instance.
		(Inherited from Exception.)
	HelpLink	Gets or sets a link to the help file associated with this exception.
		(Inherited from Exception.)
	HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.
		(Inherited from Exception.)
	InnerException	Gets the <u>Exception</u> instance that caused the current exception.
		(Inherited from Exception.)
Ģ	MemberwiseClone()	Creates a shallow copy of the current Object.
		(Inherited from Object.)
	Message	Gets a message that describes the current exception.
		(Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that

		causes the error.
		(Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stac at the time the current exception was thrown. (Inherited from Exception.)
	<u>TargetSite</u>	Gets the method that throws the current exception. (Inherited from Exception.)
∃	ToString()	Creates and returns a string representation of the current exception. (Inherited from Exception.)

■ Inheritance Hierarchy

Object

Exception

ZipException

BadCrcException

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadCrcException Constructor

Reference ► Ionic.Zip ► BadCrcException ► BadCrcException()

C#

■ Members

lcon	Member	Description
=	BadCrcException()	Default ctor.
€	BadCrcException(String)	Come on, you know how exceptions work. Why are yo looking at this documentation?
Ģ ₩	BadCrcException(SerializationInfo, StreamingContext)	Come on, you know how exceptions work. Why are yo looking at this documentation?

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadCrcException Constructor

Reference ► Ionic.Zip ► BadCrcException ► BadCrcException()

Default ctor.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public BadCrcException()

Public Sub New

public:
BadCrcException()
```

C#

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadCrcException Constructor (info,

context)

Reference ► Ionic.Zip ► BadCrcException ►

BadCrcException(SerializationInfo, StreamingContext)

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

```
Visual C++
C#
                   Visual Basic
protected BadCrcException(
        SerializationInfo info,
        StreamingContext context
```

```
Protected Sub New (
       info As SerializationInfo, _
        context As StreamingContext _
```

```
protected:
BadCrcException(
        SerializationInfo^ info,
        StreamingContext context
```

Parameters

info (SerializationInfo)

The serialization info for the exception.

context (StreamingContext)

The streaming context from which to deserialize.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadCrcException Constructor

(message)

Reference ► Ionic.Zip ► BadCrcException ►

BadCrcException(String)

C#

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

Parameters

message (String)

The message in the exception.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Reference ► Ionic.Zip ► BadPasswordException

C#

Issued when an ZipEntry.ExtractWithPassword() method is invoked with an incorrect password.

■ Declaration Syntax

C# Visual Basic Visual C++

[SerializableAttribute]

public class BadPasswordException : ZipException

<<u>SerializableAttribute</u>> _ Public Class BadPasswordException _ Inherits <u>ZipException</u>

[SerializableAttribute]

public ref class BadPasswordException : public ZipEx

■ Members

All M	embers	Constructors	Methods	Properties	
✓ Public✓ Protected			✓ Instance✓ Static		Declare Inherite
Icon Member				Description	
≡	BadPasswordException()		Default ctor.		
≡ ₩	BadPasswordException(String)		Come on, you how exception Why are you lot this document	s work. ooking a	

₫••	BadPasswordException(String, Exception)	Come on, you know how exceptions work. Why are you looking a this documentation?
Ģ	BadPasswordException(SerializationInfo, StreamingContext)	Come on, you know how exceptions work. Why are you looking a this documentation?
	<u>Data</u>	Gets a collection of key/value pairs that provide additional use defined information about the exception.
		(Inherited from Exception.)
□	Equals(Object)	Determines whether the specified Object is equal to the current Object. (Inherited from Object.
Ģ ♥	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
=	GetBaseException()	When overridden in a

		derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from
_		Exception.)
≡	GetHashCode()	Serves as a hash function for a particula type.
		(Inherited from Object.
± Q	GetObjectData(SerializationInfo, StreamingContext)	When overridden in a derived class, sets the <u>SerializationInfo</u> with information about the exception.
		(Inherited from Exception.)
≡	GetType()	Gets the runtime type of the current instance
		(Inherited from Exception.)
	HelpLink	Gets or sets a link to the help file associated with this exception.
		(Inherited from Exception .)
	HResult	Gets or sets HRESUL a coded numerical value that is assigned to a specific exception

		(Inherited from Exception.)
	InnerException	Gets the Exception instance that caused the current exception.
		(Inherited from Exception.)
Ģ [¯] ©	MemberwiseClone()	Creates a shallow cop of the current Object.
		(Inherited from Object.
	<u>Message</u>	Gets a message that describes the current exception.
		(Inherited from Exception .)
	Source	Gets or sets the name of the application or th object that causes the error.
		(Inherited from Exception.)
in the second	StackTrace	Gets a string representation of the frames on the call stac at the time the current exception was thrown.
		(Inherited from Exception.)
	<u>TargetSite</u>	Gets the method that throws the current exception.

		(Inherited from Exception .)
≅	ToString()	Creates and returns a string representation o the current exception.
		(Inherited from Exception .)

■ Inheritance Hierarchy

Object

Exception

ZipException

BadPasswordException

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Constructor

Reference ► Ionic.Zip ► BadPasswordException ►

BadPasswordException()

■ Members

Icon	Member	Description
∉	BadPasswordException()	Default ctor.
€	BadPasswordException(String)	Come on, you know how exceptions work. Why are you looking a this documentation?
≓	BadPasswordException(String, Exception)	Come on, you know how exceptions work. Why are you looking a this documentation?
ē	BadPasswordException(SerializationInfo, StreamingContext)	Come on, you know how exceptions work. Why are you looking a this documentation?

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Constructor

Reference ► Ionic.Zip ► BadPasswordException ►

BadPasswordException()

Default ctor.



■ Declaration Syntax

C# Visual Basic Visual C++

public BadPasswordException()

Public Sub New

public:
BadPasswordException()

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Constructor (info, context)

Reference ► Ionic.Zip ► BadPasswordException ►

BadPasswordException(SerializationInfo, StreamingContext)

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

```
Visual C++
C#
                   Visual Basic
protected BadPasswordException(
        SerializationInfo info,
        StreamingContext context
```

```
Protected Sub New (
       info As SerializationInfo, _
        context As StreamingContext _
```

```
protected:
BadPasswordException(
        SerializationInfo^ info,
        StreamingContext context
```

Parameters

info (SerializationInfo)

The serialization info for the exception.

context (StreamingContext)

The streaming context from which to deserialize.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Constructor (message)

Reference ► Ionic.Zip ► BadPasswordException ►

BadPasswordException(String)

C#

Come on, you know how exceptions work. Why are you looking at this documentation?

■ Declaration Syntax

Parameters

message (String)

The message in the exception.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Constructor (message, innerException)

Reference ► Ionic.Zip ► BadPasswordException ►

BadPasswordException(String, Exception)

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

```
C# Visual Basic Visual C++

public BadPasswordException(
    string message,
    Exception innerException
)
```

```
Public Sub New ( _ message As String, _ innerException As Exception _ )
```

Parameters

message (String)

The message in the exception.

innerException (Exception)

The innerException for this exception.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadReadException Class

Reference ► Ionic.Zip ► BadReadException

C# -

Indicates that a read was attempted on a stream, and bad or incomplete data was received.

■ Declaration Syntax

C# Visual Basic Visual C++

[SerializableAttribute]

public class BadReadException : ZipException

<<u>SerializableAttribute</u>>

Public Class BadReadException _ Inherits ZipException

[SerializableAttribute]

public ref class BadReadException : public <a>ZipExcept

■ Members

All Members	Constructors	Methods	Properties	
Public Protected Protected		✓ Instance ✓ Static		Declare Inherite

Icon	Member	Description
≟	BadReadException()	Default ctor.
∃ ₩	BadReadException(String)	Come on, you know how exceptions work. Why are you looking at this documentation?

∄	BadReadException(String, Exception)	Come on, you know how exceptions work. Why are you looking at this documentation?
Ģ ♥	BadReadException(SerializationInfo, StreamingContext)	Come on, you know how exceptions work. Why are you looking at this documentation?
	<u>Data</u>	Gets a collection of key/value pairs that provide additional user-defined information about the exception.
		(Inherited from Exception.)
- II	Equals(Object)	Determines whether the specified <u>Object</u> is equal to the current <u>Object</u> . (Inherited from <u>Object</u> .)
·	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.
		(Inherited from Object.)
≅∲	GetBaseException()	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions.

		(Inherited from Exception.)
≡	GetHashCode()	Serves as a hash function for a particular type.
		(Inherited from Object.)
€	GetObjectData(SerializationInfo, StreamingContext)	When overridden in a derived class, sets the SerializationInfo with information about the exception.
		(Inherited from Exception.)
≡	GetType()	Gets the runtime type of the current instance.
		(Inherited from Exception.)
	HelpLink	Gets or sets a link to the help file associated with thi exception.
		(Inherited from Exception.)
	HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.
		(Inherited from Exception.)
	InnerException	Gets the Exception instanc that caused the current exception.
•		(Inherited from Exception.)
Ģ [™]	MemberwiseClone()	Creates a shallow copy of the current Object.
		(Inherited from Object.)

	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	StackTrace	Gets a string representatio of the frames on the call stack at the time the currer exception was thrown. (Inherited from Exception.)
	<u>TargetSite</u>	Gets the method that throws the current exception. (Inherited from Exception.)
≡	ToString()	Creates and returns a strin representation of the current exception. (Inherited from Exception.)

■ Inheritance Hierarchy

Object

Exception

ZipException

BadReadException

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadReadException Constructor

Reference ► Ionic.Zip ► BadReadException ► BadReadException()

C#

■ Members

Icon	Member	Description
=	BadReadException()	Default ctor.
 € \	BadReadException(String)	Come on, you know how exceptions work. Why are you looking at this documentation?
∃	BadReadException(String, Exception)	Come on, you know how exceptions work. Why are you looking at this documentation?
ğ	BadReadException(SerializationInfo, StreamingContext)	Come on, you know how exceptions work. Why are you looking at this documentation?

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadReadException Constructor

Reference ► Ionic.Zip ► BadReadException ► BadReadException()

C#

Default ctor.

■ Declaration Syntax

C# Visual Basic Visual C++

public BadReadException()

Public Sub New

public:
BadReadException()

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadReadException Constructor

(info, context)

Reference ► Ionic.Zip ► BadReadException ►

BadReadException(SerializationInfo, StreamingContext)

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

```
C# Visual Basic Visual C++

protected BadReadException(
    SerializationInfo info,
    StreamingContext context
)
```

Parameters

info (SerializationInfo)

The serialization info for the exception.

context (StreamingContext)

The streaming context from which to deserialize.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

(message)

Reference ► Ionic.Zip ► BadReadException ►

BadReadException(String)

C#______

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

Parameters

message (String)

The message in the exception.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadReadException Constructor (message, innerException)

Reference ► Ionic.Zip ► BadReadException ►

BadReadException(String, Exception)

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

```
C# Visual Basic Visual C++

public BadReadException(
    string message,
    Exception innerException
)
```

```
Public Sub New ( _ message As String, _ innerException As Exception _ )
```

Parameters

message (String)

The message in the exception.

innerException (Exception)

The innerException for this exception.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadStateException Class

Reference ► Ionic.Zip ► BadStateException

C# -

Indicates that an operation was attempted on a ZipFile which was not possible given the state of the instance. For example, if you call Save (on a ZipFile which has no filename set, you can get this exception.

■ Declaration Syntax

C# Visual Basic Visual C++

[SerializableAttribute]

public class BadStateException : ZipException

<<u>SerializableAttribute</u>> _ Public Class BadStateException _ Inherits <u>ZipException</u>

[<u>SerializableAttribute</u>]

public ref class BadStateException : public <a>ZipExcep

■ Members

All M	embers	Constructors	Methods		Properties	
✓ Pu	blic otected		✓ Instance ✓ Static	e		Declare Inherite
Icon	Member	•		Des	scription	
= ₩	BadStateException()		Default ctor.			
∃	BadState	eException(String))	you	me on, you kno eptions work. \ I looking at this cumentation?	Why are

≡	BadStateException(String, Exception)	Come on, you know how exceptions work. Why are you looking at this documentation?
Ģ 🍑	BadStateException(SerializationInfo, StreamingContext)	Come on, you know how exceptions work. Why are you looking at this documentation?
	<u>Data</u>	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
≡ ◊	Equals(Object)	Determines whether the specified <u>Object</u> is equal to the current <u>Object</u> . (Inherited from <u>Object</u> .)
Ģ	Finalize()	Allows an Object to attemp to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
∃	GetBaseException()	When overridden in a derived class, returns the Exception that is the root

		cause of one or more
		subsequent exceptions.
		(Inherited from Exception.)
Ξ₩	GetHashCode()	Serves as a hash function for a particular type.
		(Inherited from Object.)
€	GetObjectData(SerializationInfo, StreamingContext)	When overridden in a derived class, sets the SerializationInfo with information about the exception.
		(Inherited from Exception.)
∉	GetType()	Gets the runtime type of the current instance.
		(Inherited from Exception.)
	HelpLink	Gets or sets a link to the help file associated with thi exception.
		(Inherited from Exception.)
	HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception.
		(Inherited from Exception.)
	InnerException	Gets the Exception instance that caused the current exception.
		(Inherited from Exception.)
Ģ [₩]	MemberwiseClone()	Creates a shallow copy of

		the current Object.
		(Inherited from Object.)
	Message	Gets a message that describes the current exception.
		(Inherited from Exception.)
	Source	Gets or sets the name of the application or the objec that causes the error.
		(Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the currer exception was thrown. (Inherited from Exception.)
	<u>TargetSite</u>	Gets the method that throw the current exception.
- 🗛		(Inherited from Exception.)
≓ ♥	ToString()	Creates and returns a strin representation of the currer exception.
		(Inherited from Exception.)

■ Inheritance Hierarchy

Object

Exception

ZipException

BadStateException

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

•

Reference ► Ionic.Zip ► BadStateException ► BadStateException()

C#

■ Members

Icon	Member	Description
=	BadStateException()	Default ctor.
₫\	BadStateException(String)	Come on, you know how exceptions work. Why are you looking at this documentation?
∉	BadStateException(String, Exception)	Come on, you know how exceptions work. Why are you looking at this documentation?
ğ	BadStateException(SerializationInfo, StreamingContext)	Come on, you know how exceptions work. Why are you looking at this documentation?

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Reference ► Ionic.Zip ► BadStateException ► BadStateException()

C#

Default ctor.

■ Declaration Syntax

C# Visual Basic Visual C++

public BadStateException()

Public Sub New

public:
BadStateException()

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

(info, context)

Reference ► Ionic.Zip ► BadStateException ►

BadStateException(SerializationInfo, StreamingContext)

Come on, you know how exceptions work. Why are you looking at this documentation?

■ Declaration Syntax

```
C# Visual Basic Visual C++

protected BadStateException(
    SerializationInfo info,
    StreamingContext context
)
```

```
Protected Sub New ( __ info As <u>SerializationInfo</u>, _ context As <u>StreamingContext</u> _ )
```

Parameters

info (SerializationInfo)

The serialization info for the exception.

context (StreamingContext)

The streaming context from which to deserialize.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

(message)

Reference ► Ionic.Zip ► BadStateException ►

BadStateException(String)

C#______

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

```
C# Visual Basic Visual C++

public BadStateException(
    string message
)

Public Sub New ( _
    message As String _
)
```

Parameters

message (String)

The message in the exception.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BadStateException Constructor (message, innerException)

Reference ► Ionic.Zip ► BadStateException ►

BadStateException(String, Exception)

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

```
C# Visual Basic Visual C++

public BadStateException(
    string message,
    Exception innerException
)
```

```
Public Sub New ( _ message As String, _ innerException As Exception _ )
```

Parameters

message (String)

The message in the exception.

innerException (Exception)

The innerException for this exception.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Close Delegate Delegate

Reference ► Ionic.Zip ► CloseDelegate

C#

Delegate in which the application closes the stream, just-in-time, for the named entry.

■ Declaration Syntax

Parameters

entryName (String)

The name of the ZipEntry that the application should close the stream for.

stream (Stream)

The stream to be closed.

■ Remarks

When you add an entry via <u>AddEntry(String, OpenDelegate,</u> <u>CloseDelegate)</u>, the application code provides the logic that opens and closes the stream for the given ZipEntry.

See Also

AddEntry(String, OpenDelegate, CloseDelegate)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ComHelper Class

Reference ► Ionic.Zip ► ComHelper

C#

This class exposes a set of COM-accessible wrappers for static method available on the ZipFile class. You don't need this class unless you are using DotNetZip from a COM environment.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public class Com	Helper	
Public Class Com	Helper	
public ref class	ComHelper	

■ Members

All M	Members Constructors			Methods	
✓ Public✓ Protected			✓ Instance✓ Static	Declare Inherite	
Icon	Member		Description		
≅	ComHelper()		Initializes a new instance of the ComHelper class		the
≅©	CheckZip(Stri	ng)	A wrap ZipFile	per for . <u>CheckZip(string)</u>	
≅©	CheckZipPass String)	sword(String,		I-friendly wrapper for I <u>CheckZipPassword</u>	

∃	Equals(Object)	Determines whether the specified Object is equal to the current Object (Inherited from Object .)
ē	Finalize()	Allows an Object to attempt to free resources and perform other cleanu operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
≡	FixZipDirectory(String)	A wrapper for ZipFile.FixZipDirectory(string)
≡•	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
≡	GetType()	Gets the <u>Type</u> of the current instance. (Inherited from <u>Object</u> .)
=•	GetZipLibraryVersion()	A wrapper for <u>ZipFile.LibraryVersion</u>
₫ ◊	IsZipFile(String)	A wrapper for ZipFile.IsZipFile(string
≡©	IsZipFileWithExtract(String)	A wrapper for <u>ZipFile.IsZipFile(stringbool)</u>
Ģ [®]	MemberwiseClone()	Creates a shallow copy of the currer Object.

		(Inherited from Object.)
≡	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> .
		(Inherited from Object.)

■ Inheritance Hierarchy

Object

ComHelper

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ComHelper Constructor

Reference ► Ionic.Zip ► ComHelper ► ComHelper()

C#

Initializes a new instance of the **ComHelper** class

■ Declaration Syntax

```
C# Visual Basic Visual C++

public ComHelper()

Public Sub New

public:
ComHelper()
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CheckZip Method (filename)

Reference ► Ionic.Zip ► ComHelper ► CheckZip(String)

A wrapper for <a>ZipFile.CheckZip(string)

C#

Declaration Syntax

```
C# Visual Basic Visual C++

public bool CheckZip(
    string filename
)

Public Function CheckZip (
    filename As String _
) As Boolean

public:
bool CheckZip(
    String^ filename
)
```

Parameters

filename (String)

The filename to of the zip file to check.

■ Return Value

true if the named zip file checks OK. Otherwise, false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CheckZipPassword Method (filename, password)

Reference ► Ionic.Zip ► ComHelper ► CheckZipPassword(String, String)

A COM-friendly wrapper for the static method C# C# CHECKZipPassword(String).

Declaration Syntax

Parameters

filename (String)

The filename to of the zip file to check.

password (String)

The password to check.

■ Return Value

true if the named zip file checks OK. Otherwise, false.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Fix Zip Directory Method (filename)

C#

Reference ► Ionic.Zip ► ComHelper ► FixZipDirectory(String)

A wrapper for ZipFile.FixZipDirectory(string)

■ Declaration Syntax

Parameters

filename (String)

The filename to of the zip file to fix.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 GetZipLibrary Version Method

Reference ► Ionic.Zip ► ComHelper ► GetZipLibraryVersion()

C#

A wrapper for **ZipFile.LibraryVersion**

■ Declaration Syntax

C# Visual Basic Visual C++

public string GetZipLibraryVersion()

Public Function GetZipLibraryVersion As String

public:

String^ GetZipLibraryVersion()

■ Return Value

the version number on the DotNetZip assembly, formatted as a string.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Is Zip File Method (filename)

Reference ► Ionic.Zip ► ComHelper ► IsZipFile(String)

A wrapper for <a>ZipFile.IsZipFile(string)

Declaration Syntax

```
C# Visual Basic Visual C++

public bool IsZipFile(
    string filename
)
```

C#

Parameters

filename (String)

The filename to of the zip file to check.

■ Return Value

true if the file contains a valid zip file.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Is Zip File With Extract Method

(filename)

Reference ► Ionic.Zip ► ComHelper ► IsZipFileWithExtract(String)

A wrapper for **ZipFile.IsZipFile(string, bool)**

C# -

■ Declaration Syntax

Parameters

filename (String)

The filename to of the zip file to check.

Return Value

true if the file contains a valid zip file.

Remarks

We cannot use "overloaded" Method names in COM interop. So, here, we use a unique name.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compression Method Enumeration

Reference ► Ionic.Zip ► CompressionMethod

C#

The method of compression to use for a particular ZipEntry.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enum Compr	ressionMethod	
Public Enumeration	on CompressionMeth	od
public enum class	s CompressionMetho	d

■ Members

Member	Description
None	No compression at all. For COM environments, the value is 0 (zero).
Deflate	DEFLATE compression, as described in IETF RFC 1951 . This is the "normal" compression used in zip files. For COM environments, the value is 8.
BZip2	BZip2 compression, a compression algorithm developed by Julian Seward. For COM environments, the value is 12.

Remarks

<u>PKWare's ZIP Specification</u> describes a number of distinct cmopression methods that can be used within a zip file. DotNetZip supports a subset of them.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Counting Stream Class

Reference ► Ionic.Zip ► CountingStream

C#

A decorator stream. It wraps another stream, and performs bookkeeping to keep track of the stream Position.

■ Declaration Syntax

C# Visual Basic Visual C++
public class CountingStream : Stream

Public Class CountingStream _ Inherits <u>Stream</u>

public ref class CountingStream : public Stream

■ Members

All M	embers	Constructors	M	ethods	Properties	
✓ Public✓ Protected			✓ Instance✓ Static		Declare Inherite	
Icon	Member	•		Descriptio	n	
≡	Counting	gStream(Stream)		The constru	uctor.	
≡	Adjust(Ir	<u>nt64)</u>		Adjust the I	oyte count on th	e strean
∃	_	ead(Byte[], Int32, syncCallback,		Begins an a operation.	tionAttribute. asynchronous re rom <u>Stream</u> .)	ead
≡						

	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous write operation.
		(Inherited from <u>Stream</u> .)
	BytesRead	the count of bytes that have been read from the stream.
	BytesWritten	The count of bytes written out to the stream.
	CanRead	Whether the stream can be read.
		(Overrides <u>Stream</u> . <u>CanRead</u> .)
	CanSeek	Whether it is possible to call Seek() on the stream.
		(Overrides <u>Stream.CanSeek.</u>)
	CanTimeout	Gets a value that determines whether the current stream can tim out.
		(Inherited from <u>Stream</u> .)
	CanWrite	Whether it is possible to call Write() on the stream.
		(Overrides <u>Stream</u> . <u>CanWrite</u> .)
ΕΦ	Close()	Closes the current stream and releases any resources (such as sockets and file handles) associate with the current stream.
		(Inherited from <u>Stream</u> .)
	ComputedPosition	Returns the sum of number of byte:

		written, plus the initial offset before writing.
E₩	CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)
Ģ [¯] ©	CreateWaitHandle()	Obsolete. Allocates a <u>WaitHandle</u> object. (Inherited from <u>Stream</u> .)
≡	<u>Dispose()</u>	Releases all resources used by the Stream . (Inherited from Stream .)
ē	Dispose(Boolean)	Releases the unmanaged resource used by the <u>Stream</u> and optionally releases the managed resources. (Inherited from <u>Stream</u> .)
≡	EndRead(IAsyncResult)	Waits for the pending asynchronous read to complete. (Inherited from <u>Stream</u> .)
≡	EndWrite(IAsyncResult)	Ends an asynchronous write operation. (Inherited from <u>Stream</u> .)
₫₩	Equals(Object)	Determines whether the specified Object is equal to the current Object .

		(Inherited from Object.)
<u>~</u>	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
≓	Flush()	Flushes the underlying stream. (Overrides Stream.Flush() .)
=₩	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
∃	GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
=0	GetType()	Gets the <u>Type</u> of the current instance. (Inherited from <u>Object</u> .)
∃ ©	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
	Length	The length of the underlying stream (Overrides Stream.Length.)
		,
₹ ₩	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object

		(Inherited from MarshalByRefObject.)
*	MemberwiseClone()	Creates a shallow copy of the current Object.
		(Inherited from Object.)
	Position	The Position of the stream.
		(Overrides <u>Stream.Position.</u>)
≟	Read(Byte[], Int32, Int32)	The read method.
		(Overrides <u>Stream.Read(Byte[], Int32, Int32</u>).)
≟∳	ReadByte()	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.
		(Inherited from <u>Stream</u> .)
	ReadTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to read before timing out.
		(Inherited from <u>Stream</u> .)
≟	Seek(Int64, SeekOrigin)	Seek in the stream.
		(Overrides <u>Stream.Seek(Int64, SeekOrigin)</u> .)
∉	SetLength(Int64)	Set the length of the underlying stream. Be careful with this!
		(Overrides Stream . SetLength(Int64).)

	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
	WrappedStream	Gets the wrapped stream.
≅©	Write(Byte[], Int32, Int32)	Write data into the stream. (Overrides <u>Stream.Write(Byte[], Int32, Int32</u>).)
≝₩	WriteByte(Byte)	Writes a byte to the current positior in the stream and advances the position within the stream by one byte. (Inherited from <u>Stream</u> .)
	WriteTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to write before timing out. (Inherited from Stream .)

■ Remarks

In some cases, it is not possible to get the Position of a stream, let's say on a write-only output stream like ASP.NET's Response.OutputStream, or on a different write-only stream provided as the destination for the zip by the application. In this case, programmers can use this counting stream to count the bytes read or written.

Consider the scenario of an application that saves a self-extracting archive (SFX), that uses a custom SFX stub.

Saving to a filesystem file, the application would open the filesystem file

(getting a FileStream), save the custom sfx stub into it, and then ca ZipFile.Save(), specifying the same FileStream.

ZipFile.Save() does the right thing for the zipentry offsets, by inquiring the Position of the FileStream before writing any data, and then adding that initial offset into any ZipEntry offsets in the zip directory Everything works fine.

Now suppose the application is an ASPNET application and it saves directly to Response.OutputStream. It's not possible for DotNetZip to inquire the Position, so the offsets for the SFX will be wrong.

The workaround is for the application to use this class to wrap <code>HttpResponse.OutputStream</code>, then write the SFX stub and the ZipFile into that wrapper stream. Because <code>ZipFile.Save()</code> can inquire the <code>Position</code>, it will then do the right thing with the offsets.

■ Inheritance Hierarchy

Object

<u>MarshalByRefObject</u>

Stream

CountingStream

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Counting Stream Constructor

(stream)

Reference ► Ionic.Zip ► CountingStream ►

CountingStream(Stream)

The constructor.



■ Declaration Syntax

```
C# Visual Basic Visual C++

public CountingStream(
    Stream stream
)

Public Sub New ( __
    stream As Stream __
```

Parameters

stream (Stream)

The underlying stream

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Adjust Method (delta)

Reference ► Ionic.Zip ► CountingStream ► Adjust(Int64)

Adjust the byte count on the stream.

C#

■ Declaration Syntax

Parameters

delta (Int64)

the number of bytes to subtract from the count.

■ Remarks

Subtract delta from the count of bytes written to the stream. This is necessary when seeking back, and writing additional data, as happens some cases when saving Zip files.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BytesRead Property

Reference ► Ionic.Zip ► CountingStream ► BytesRead C#

the count of bytes that have been read from the stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public long BytesRead { get; }

Public ReadOnly Property BytesRead As Long
    Get

public:
property long long BytesRead {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BytesWritten Property

Reference ► Ionic.Zip ► CountingStream ► BytesWritten

The count of bytes written out to the stream.

C# 🔽

■ Declaration Syntax

```
C# Visual Basic Visual C++

public long BytesWritten { get; }

Public ReadOnly Property BytesWritten As Long
    Get

public:
property long long BytesWritten {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.Zip ► CountingStream ► CanRead C#

Whether the stream can be read.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:

virtual property bool CanRead {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Whether it is possible to call Seek() on the stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:
virtual property bool CanSeek {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | CanWrite Property | Reference | Ionic.Zip | CountingStream | CanWrite | C#

Whether it is possible to call Write() on the stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:

virtual property bool CanWrite {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Computed Position Property

Reference ► <u>Ionic.Zip</u> ► <u>CountingStream</u> ► **ComputedPosition**

Returns the sum of number of bytes written, plus the initial offset before writing.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public long ComputedPosition { get; }

Public ReadOnly Property ComputedPosition As Long Get

public:
property long long ComputedPosition {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Flushes the underlying stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

public:
virtual void Flush() override
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Length Property

Reference ► Ionic.Zip ► CountingStream ► Length
C#
```

The length of the underlying stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:

virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Position Property Reference ► Ionic.Zip ► CountingStream ► Position C#

The Position of the stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

Reference ► Ionic.Zip ► CountingStream ► Read(Byte[], Int32, Int32)

The read method.

C#

Declaration Syntax

```
public:
virtual int Read(
          array<unsigned char>^ buffer,
          int offset,
          int count
) override
```

Parameters

buffer (Byte[])

The buffer to hold the data read from the stream.

offset (Int32)

the offset within the buffer to copy the first byte read.

count (Int32)

the number of bytes to read.

Return Value

the number of bytes read, after decryption and decompression.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.Zip ► CountingStream ► Seek(Int64, SeekOrigin)

C#

Seek in the stream.

Declaration Syntax

```
C# Visual Basic Visual C++

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

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

Parameters

offset (Int64)

the offset point to seek to

origin (SeekOrigin)

the reference point from which to seek

■ Return Value

The new position

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.Zip ► CountingStream ► SetLength(Int64)

Set the length of the underlying stream. Be careful with this!

■ Declaration Syntax

Parameters

value (Int64)

the length to set on the underlying stream.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 WrappedStream Property

Reference ► Ionic.Zip ► CountingStream ► WrappedStream

Gets the wrapped stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public Stream WrappedStream { get; }

Public ReadOnly Property WrappedStream As Stream
    Get

public:
property Stream^ WrappedStream {
    Stream^ get ();
}
```

C#

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

Reference ► Ionic.Zip ► CountingStream ► Write(Byte[], Int32, Int32)

Write data into the stream.

C# _

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

buffer (Byte[])

The buffer holding data to write to the stream.

offset (Int32)

the offset within that data array to find the first byte to write.

count (Int32)

the number of bytes to write.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Encryption Algorithm Enumeration

Reference ► Ionic.Zip ► EncryptionAlgorithm

C#

An enum that provides the various encryption algorithms supported by this library.

Declaration Syntax

C#	Visual Basic	Visual C++
public enum Encry	/ptionAlgorithm	
Public Enumeration	on EncryptionAlgor	ithm
public enum class	s EncryptionAlgori	thm

Members

Member	Description
None	No encryption at all.
PkzipWeak	Traditional or Classic pkzip encryption.
WinZipAes128	WinZip AES encryption (128 key bits).
WinZipAes256	WinZip AES encryption (256 key bits).
Unsupported	An encryption algorithm that is not supported by DotNetZip.

■ Remarks

PkzipWeak implies the use of Zip 2.0 encryption, which is known to t weak and subvertible.

A note on interoperability: Values of PkzipWeak and None are specified in PKWARE's zip specification, and are considered to be "standard". Zip archives produced using these options will be interoperable with many other zip tools and libraries, including Windows

Explorer.

Values of WinZipAes128 and WinZipAes256 are not part of the Zip specification, but rather imply the use of a vendor-specific extension from WinZip. If you want to produce interoperable Zip archives, do not use these values. For example, if you produce a zip archive using WinZipAes256, you will be able to open it in Windows Explorer on Windows XP and Vista, but you will not be able to extract entries; trying this will lead to an "unspecified error". For this reason, some people has said that a zip archive that uses WinZip's AES encryption is not actually zip archive at all. A zip archive produced this way will be readable with the WinZip tool (Version 11 and beyond).

There are other third-party tools and libraries, both commercial and otherwise, that support WinZip's AES encryption. These will be able to read AES-encrypted zip archives produced by DotNetZip, and converse applications that use DotNetZip to read zip archives will be able to read AES-encrypted archives produced by those tools or libraries. Consult th documentation for those other tools and libraries to find out if WinZip's AES encryption is supported.

In case you care: According to the WinZip specification, the actual AES key used is derived from the Password via an algorithm that complies with RFC 2898, using an iteration count of 1000. The algorithm is sometimes referred to as PBKDF2, which stands for "Password Based Key Derivation Function #2".

A word about password strength and length: The AES encryption technology is very good, but any system is only as secure as the weake link. If you want to secure your data, be sure to use a password that is hard to guess. To make it harder to guess (increase its "entropy"), you should make it longer. If you use normal characters from an ASCII keyboard, a password of length 20 will be strong enough that it will be impossible to guess. For more information on that, I'd encourage you to read this article.

The WinZip AES algorithms are not supported with the version of DotNetZip that runs on the .NET Compact Framework. This is because .NET CF lacks the HMACSHA1 class that is required for producing the

archive.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractExistingFileAction

Enumeration

Reference ► <u>lonic.Zip</u> ► ExtractExistingFileAction

C# -

An enum for the options when extracting an entry would overwrite an existing file.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enum Extra	actExistingFileAct	ion

Public Enumeration ExtractExistingFileAction

public enum class ExtractExistingFileAction

■ Members

Member	Description
Throw	Throw an exception when extraction we overwrite an existing file. (For COM clie 0 (zero).)
OverwriteSilently	When extraction would overwrite an ex overwrite the file silently. The overwrite even if the target file is marked as reac COM clients, this is a 1.)
DoNotOverwrite	When extraction would overwrite an ex don't overwrite the file, silently. (For CC this is a 2.)
InvokeExtractProgressEvent	When extraction would overwrite an exinvoke the ExtractProgress event, usin type of ExtractEntryWouldC this way, the application can decide, just whether to overwrite the file. For exam application may wish to pop up a dialog

user to choose. You may want to exam ExtractLocation property before making decision. If, after your processing in the progress event, you want to NOT extra set ExtractExistingFile on the ZipProgressEventArgs.Curr to DoNotOverwrite. If you do war the file, set ZipEntry.ExtractExistingF OverwriteSilently. If you want the Extraction, set ZipProgressEventArgs.Canc Cancelling differs from using DoNotOv that a cancel will not extract any furthe there are any. (For COM clients, the va enum is a 3.)

■ Remarks

This enum describes the actions that the library can take when an <code>Extract()</code> or <code>ExtractWithPassword()</code> method is called to extract an entry to a filesystem, and the extraction would overwrite an existing filesystem file.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractProgressEventArgs Class

Reference ► Ionic.Zip ► ExtractProgressEventArgs

C# ______

Provides information about the progress of the extract operation.

■ Declaration Syntax

C# Visual Basic Visual C++

public class ExtractProgressEventArgs : ZipProgressE

Public Class ExtractProgressEventArgs _

Inherits <u>ZipProgressEventArgs</u>

public ref class ExtractProgressEventArgs : public ∠

■ Members

All M	Members Methods		Properties		
✓ Public✓ Protected			✓ Instance✓ Static	Declare Inherite	
Icon	Member		Description		
	<u>ArchiveName</u>		Returns the archive name associate to this event. (Inherited from ZipProgressEventArgs .)		sociated
	BytesTransfer	<u>red</u>	far for th (Inherite	umber of bytes read or written this entry. ited from ogressEventArgs.)	
	Cancel			ent handler, set this to e or extract operation s.	

		(Inherited from ZipProgressEventArgs.)
	CurrentEntry	The name of the last entry saved or extracted.
		(Inherited from ZipProgressEventArgs.)
	EntriesExtracted	Number of entries extracted so far. This is set only if the EventType is Extracting_BeforeExtractEntry or Extracting_AfterExtractEntry, and the Extract() is occurring witin the scope (a call to ExtractAll().
	EntriesTotal	The total number of entries to be saved or extracted. (Inherited from ZipProgressEventArgs .)
≅	Equals(Object)	Determines whether the specified Object is equal to the current Object . (Inherited from Object .)
	EventType	The type of event being reported. (Inherited from ZipProgressEventArgs.)
	ExtractLocation	Returns the extraction target location, a filesystem path.
्रृं	Finalize()	Allows an <u>Object</u> to attempt to free resources and perform other cleanup operations before the <u>Object</u> is reclaimed by garbage collection.

		(Inherited from Object.)
=0	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
= Q	GetType()	Gets the <u>Type</u> of the current instance. (Inherited from <u>Object</u> .)
Ģ [©]	MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object .)
=0	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
	<u>TotalBytesToTransfer</u>	Total number of bytes that will be reac or written for this entry. This number will be -1 if the value cannot be determined. (Inherited from ZipProgressEventArgs.)

■ Inheritance Hierarchy

Object

EventArgs

ZipProgressEventArgs

ExtractProgressEventArgs

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Entries Extracted Property

Reference ► Ionic.Zip ► ExtractProgressEventArgs ►

EntriesExtracted

Number of entries extracted so far. This is set only if the EventType is Extracting_BeforeExtractEntry or Extracting_AfterExtractEntry, and the Extract() is occurring witin the scope of a call to ExtractAll().

■ Declaration Syntax

```
C# Visual Basic Visual C++
public int EntriesExtracted { get; }

Public ReadOnly Property EntriesExtracted As Integer
    Get

public:
property int EntriesExtracted {
    int get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractLocation Property

Reference ► Ionic.Zip ► ExtractProgressEventArgs ►

ExtractLocation

Returns the extraction target location, a filesystem path.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++
public string ExtractLocation { get; }

Public ReadOnly Property ExtractLocation As String Get

public:
property String^ ExtractLocation {
    String^ get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 OpenDelegate Delegate

Reference ► Ionic.Zip ► OpenDelegate

C#

Delegate in which the application opens the stream, just-in-time, for the named entry.

■ Declaration Syntax

Parameters

entryName (String)

The name of the ZipEntry that the application should open the stream for.

■ Remarks

When you add an entry via <u>AddEntry(String, OpenDelegate</u>, <u>CloseDelegate</u>), the application code provides the logic that opens and closes the stream for the given ZipEntry.

See Also

AddEntry(String, OpenDelegate, CloseDelegate)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **ReadOptions Class**

Reference ► Ionic.Zip ► ReadOptions

C#

A class for collecting the various options that can be used when Readin zip files for extraction or update.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public class Read	dOptions	
Public Class Reac	dOptions	
	- Io	
public ref class	ReadOptions	

Members

All M	All Members Constructors		Methods Properties		
✓ Public✓ Protected			✓ Instance✓ Static	Declare Inherite	
Icon	Member		Description		
≅	ReadOp	tions()	Initializes a new instance of the ReadOptions class		е
	Encoding	g	use when reac careful specify value you use the Encoding u archive was cr different archiv	Text.Encoding in the zip are ing the encoding here is not the sused when the zeroeated (possibly ver) you will get sults and possible.	chive. B g. If the same as ip by a

≡	Equals(Object)	Determines whether the specified Object is equal to the current Object . (Inherited from Object .)
ā	Finaliza ()	, ,
4.	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.
		(Inherited from Object.)
=	GetHashCode()	Serves as a hash function for a particular type.
		(Inherited from Object.)
≅	GetType()	Gets the <u>Type</u> of the current instance.
		(Inherited from Object.)
Ģ [™]	MemberwiseClone()	Creates a shallow copy of the current Object.
		(Inherited from Object.)
	ReadProgress	An event handler for Read operations. When opening large zip archives, you may want to display a progress bar or other indicator of status progress while reading. This parameter allows you to specify a ReadProgress Event Handle directly. When you call Read(), the progress event is invoked as necessary.
	StatusMessageWriter	The System. IO. TextWriter to use for writing verbose status messages during operations on the zig

		archive. A console application may wish to pass System. Console. Out to get messages on the Console. A graphica or headless application may wish to capture the messages in a different TextWriter, such as a System. IO. StringWriter.
≅	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)

■ Remarks

When reading a zip file, there are several options an application can set to modify how the file is read, or what the library does while reading. Th class collects those options into one container.

Pass an instance of the ReadOptions class into the ZipFile.Read() method.

. .

■ Inheritance Hierarchy

Object

ReadOptions

■ See Also

Read(String, ReadOptions)
Read(Stream, ReadOptions)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ReadOptions Constructor

Reference ► Ionic.Zip ► ReadOptions ► ReadOptions()

Initializes a new instance of the **ReadOptions** class

C# -

■ Declaration Syntax

```
C# Visual Basic Visual C++

public ReadOptions()

Public Sub New

public:
ReadOptions()
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 Encoding Property | Reference ► Ionic.Zip ► ReadOptions ► Encoding | C#

The System.Text.Encoding to use when reading in the zip archive. Be careful specifying the encoding. If the value you use here is not the same as the Encoding used when the zip archive was created (possibly by a different archiver) you will get unexpected results and possibly exceptions.

Declaration Syntax

See Also

ProvisionalAlternateEncoding

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ReadProgress Property

Reference ► Ionic.Zip ► ReadOptions ► ReadProgress

An event handler for Read operations. When opening large zip archives you may want to display a progress bar or other indicator of status progress while reading. This parameter allows you to specify a ReadProgress Event Handler directly. When you call Read(), the progress event is invoked as necessary.

Declaration Syntax

```
C# Visual Basic Visual C++

public EventHandler<ReadProgressEventArgs> ReadProgr

Public Property ReadProgress As EventHandler(Of Read Get Set

public:

property EventHandler<ReadProgressEventArgs^>^ ReadProgressEventArgs^>^ get ()

void set (EventHandler<ReadProgressEventArgs)

}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Status Message Writer Property

Reference ► <u>Ionic.Zip</u> ► <u>ReadOptions</u> ► **StatusMessageWriter**

The System. IO. TextWriter to use for writing verbose status messages during operations on the zip archive. A console application may wish to pass System. Console. Out to get messages on the Console. A graphical or headless application may wish to capture the messages in a different TextWriter, such as a System. IO. StringWriter.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ReadProgressEventArgs Class

Reference ► Ionic.Zip ► ReadProgressEventArgs

C#

Provides information about the progress of a Read operation.

■ Declaration Syntax

C# Visual Basic Visual C++

public class ReadProgressEventArgs : ZipProgressEven

Public Class ReadProgressEventArgs _ Inherits ZipProgressEventArgs

public ref class ReadProgressEventArgs : public ZipP

■ Members

All M	lembers Methods			Properties		
✓ Public ✓ Protected			✓ Instance✓ Static	Declare Inherite		
Icon	Member		Description			
	to this e			is the archive name associated event. ted from gressEventArgs.)		
	so (In		The number of bytes read or writter so far for this entry. (Inherited from ZipProgressEventArgs .)		written	
	the sa		In an event handler, set this to cance the save or extract operation that is in progress.			

	(Inherited from ZipProgressEventArgs .)
CurrentEntry	The name of the last entry saved or extracted.
	(Inherited from ZipProgressEventArgs.)
<u>EntriesTotal</u>	The total number of entries to be saved or extracted.
	(Inherited from ZipProgressEventArgs .)
Equals(Object)	Determines whether the specified Object is equal to the current Object .
	(Inherited from Object.)
<u>EventType</u>	The type of event being reported.
	(Inherited from ZipProgressEventArgs.)
Finalize()	Allows an <u>Object</u> to attempt to free resources and perform other cleanup operations before the <u>Object</u> is reclaimed by garbage collection.
	(Inherited from Object.)
GetHashCode()	Serves as a hash function for a particular type.
	(Inherited from Object.)
GetType()	Gets the Type of the current instance
	(Inherited from Object.)
MemberwiseClone()	Creates a shallow copy of the current Object.
	EntriesTotal Equals(Object) EventType Finalize() GetHashCode() GetType()

		(Inherited from Object.)
∃	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
	<u>TotalBytesToTransfer</u>	Total number of bytes that will be read or written for this entry. This number will be -1 if the value cannot be determined. (Inherited from ZipProgressEventArgs.)

■ Inheritance Hierarchy

Object

EventArgs

ZipProgressEventArgs

ReadProgressEventArgs

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 Save Progress Event Args Class

Reference ► Ionic.Zip ► SaveProgressEventArgs

C# ,

Provides information about the progress of a save operation.

■ Declaration Syntax

C# Visual Basic Visual C++

public class SaveProgressEventArgs : ZipProgressEven

Public Class SaveProgressEventArgs _ Inherits ZipProgressEventArgs

public ref class SaveProgressEventArgs : public ZipP

■ Members

All M	All Members Methods			Properties	
✓ Public✓ Protected			✓ Instance✓ Static	Declare Inherite	
Icon	Member		Description		
	<u>ArchiveName</u>		Returns the archive name associato this event. (Inherited from ZipProgressEventArgs.)		
	so far (Inher		The number of bytes read or written so far for this entry. (Inherited from ZipProgressEventArgs .)		written
	the sa		In an event handler, set this to cance the save or extract operation that is in progress.		

		(Inherited from ZipProgressEventArgs.)
	CurrentEntry	The name of the last entry saved or extracted.
		(Inherited from ZipProgressEventArgs.)
	EntriesSaved	Number of entries saved so far.
	EntriesTotal	The total number of entries to be saved or extracted. (Inherited from ZipProgressEventArgs .)
≅	Equals(Object)	Determines whether the specified Object is equal to the current Object . (Inherited from Object .)
	EventType	The type of event being reported. (Inherited from ZipProgressEventArgs.)
Ģ ₩	Finalize()	Allows an <u>Object</u> to attempt to free resources and perform other cleanup operations before the <u>Object</u> is reclaimed by garbage collection. (Inherited from <u>Object</u> .)
≅	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
≅	GetType()	Gets the <u>Type</u> of the current instance (Inherited from <u>Object</u> .)

9	MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object .)
∄	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
	<u>TotalBytesToTransfer</u>	Total number of bytes that will be read or written for this entry. This number will be -1 if the value cannot be determined. (Inherited from ZipProgressEventArgs .)

■ Inheritance Hierarchy

Object

EventArgs

ZipProgressEventArgs

SaveProgressEventArgs

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

lacksquare

4

Ionic Zip Library v1.9.1.6 Entries Saved Property

Reference ► Ionic.Zip ► SaveProgressEventArgs ► EntriesSaved

C#

Number of entries saved so far.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public int EntriesSaved { get; }

Public ReadOnly Property EntriesSaved As Integer
    Get

public:
property int EntriesSaved {
    int get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **SelfExtractorFlavor Enumeration**

Reference ► Ionic.Zip ► SelfExtractorFlavor

C#

An enum that provides the different self-extractor flavors

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enum Self	ExtractorFlavor	
Public Enumeration	on SelfExtractorFl	avor
public enum class	SelfExtractorFla	vor

■ Members

Member	Description
ConsoleApplication	A self-extracting zip archive that runs fron the console or command line.
WinFormsApplication	A self-extracting zip archive that presents a graphical user interface when it is executed.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **SelfExtractorSaveOptions Class**

Reference ► Ionic.Zip ► SelfExtractorSaveOptions

C# -

The options for generating a self-extracting archive.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public class Self	ExtractorSaveOption	ons
Public Class Self	ExtractorSaveOpti	ons
nublic ref class	SelfExtractorSave	Ontions

■ Members

All M	Members Constructors		Methods Properties			
✓ Public✓ Protected			✓ Instance✓ Static		Declare Inherite	
Icon	Member	•		Des	cription	
≡ ₩	SelfExtractorSaveOptions()		Initializes a new instance of the SelfExtractorSaveOptions class			
	Copyright		The copyright notice, to embed into the ge EXE.		•	
	DefaultExtractDirectory			the ı	default extract user will see wh ning the self-ext	en

		archive.
	Description	The description to embed into the generated EXE.
ĕ₩	Equals(Object)	Determines whether the specified <u>Object</u> is equal to the current <u>Object</u> . (Inherited from <u>Object</u> .)
	ExtractExistingFile	Specify what the self- extractor will do when extracting an entry would overwrite an existing file.
	FileVersion	The file version number to embed into the generated EXE. It will show up, for example, during a mouseover in Windows Explorer.
ē	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
	Flavor	The type of SFX to create.

± ♦	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
≡	GetType()	Gets the <u>Type</u> of the curren instance. (Inherited from <u>Object</u> .)
in the second	IconFile	The name of an .ico file in the filesystem to use for the application icon for the generated SFX.
Ģ	MemberwiseClone()	Creates a shallow copy of the current <u>Object</u> . (Inherited from <u>Object</u> .)
	PostExtractCommandLine	The command to run after extraction.
	<u>ProductName</u>	The product name to ember into the generated EXE.
	ProductVersion	The product version to embed into the generated EXE. It will show up, for example, during a mouseover in Windows Explorer.
	Quiet	Whether the ConsoleApplication SFX wi

		be quiet during extraction.
	RemoveUnpackedFilesAfterExecute	Whether to remove the files that have been unpacked, after executing the PostExtractCommandLine.
	<u>SfxExeWindowTitle</u>	The title to display in the Window of a GUI SFX, whil it extracts.
≅©	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)

■ Inheritance Hierarchy

Object

SelfExtractorSaveOptions

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Self Extractor Save Options

Constructor

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ►

SelfExtractorSaveOptions()

Initializes a new instance of the SelfExtractorSaveOptions class

■ Declaration Syntax

Visual C++ Visual Basic C# public SelfExtractorSaveOptions() Public Sub New public: SelfExtractorSaveOptions()

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Copyright Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ► Copyright

The copyright notice, if any, to embed into the generated $\stackrel{\smile}{\mathsf{EXE}}$.

■ Declaration Syntax

Remarks

It will show up, for example, while viewing properties of the file in Windows Explorer. You can use any arbitrary string, but typically you want something like "Copyright © Dino Chiesa 2011".

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 DefaultExtractDirectory Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ►

DefaultExtractDirectory

The default extract directory the user will see when running the self-extracting archive.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public string DefaultExtractDirectory { get; set; }

Public Property DefaultExtractDirectory As String
    Get
    Set

public:
property String^ DefaultExtractDirectory {
    String^ get ();
    void set (String^ value);
}
```

- Remarks

Passing null (or Nothing in VB) here will cause the Self Extractor to use the the user's personal directory (<u>Personal</u>) for the default extract location.

This is only a default location. The actual extract location will be settable on the command line when the SFX is executed.

You can specify environment variables within this string, with %NAME%. The value of these variables will be expanded at the time the SFX is rur Example: $\%USERPROFILE\%\Documents\unpack\ may\ expand\ \epsilon$ runtime to c:\users\melvin\Documents\unpack.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Description Property

<u>Reference</u> ► <u>Ionic.Zip</u> ► <u>SelfExtractorSaveOptions</u> ► **Description**

C#

The description to embed into the generated EXE.

■ Declaration Syntax

Remarks

Use any arbitrary string. This text will be displayed during a mouseover Windows Explorer. If you specify nothing, then the string "DotNetZip SF Archive" is embedded into the EXE as the description.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractExistingFile Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ►

ExtractExistingFile

Specify what the self-extractor will do when extracting an entry would overwrite an existing file.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public ExtractExistingFileAction ExtractExistingFile

Public Property ExtractExistingFile As ExtractExisti
    Get
    Set

public:

property ExtractExistingFileAction ExtractExistingFileAction get ();
    void set (ExtractExistingFileAction value);
}
```

Remarks

The default behavior is to Throw.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 File Version Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ► FileVersion

The file version number to embed into the generated EXE. It will show up, for example, during a mouseover in Windows Explorer.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flavor Property

<u>Reference</u> ► <u>Ionic.Zip</u> ► <u>SelfExtractorSaveOptions</u> ► **Flavor**

The type of SFX to create.

■ Declaration Syntax

C#

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ConFile Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ► IconFile

The name of an .ico file in the filesystem to use for the application icon for the generated SFX.

■ Declaration Syntax

■ Remarks

Normally, DotNetZip will embed an "zipped folder" icon into the generate SFX. If you prefer to use a different icon, you can specify it here. It shows be a .ico file. This file is passed as the /win32icon option to the csc.exe compiler when constructing the SFX file.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 PostExtractCommandLine Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ►

PostExtractCommandLine

The command to run after extraction.



■ Declaration Syntax

```
C# Visual Basic Visual C++
public string PostExtractCommandLine { get; set; }

Public Property PostExtractCommandLine As String
    Get
    Set

public:
property String^ PostExtractCommandLine {
    String^ get ();
    void set (String^ value);
}
```

■ Remarks

This is optional. Leave it empty (null in C# or Nothing in VB) to rur no command after extraction.

If it is non-empty, the SFX will execute the command specified in this string on the user's machine, and using the extract directory as the working directory for the process, after unpacking the archive. The program to execute can include a path, if you like. If you want to execute a program that accepts arguments, specify the program name, followed by a space, and then the arguments for the program, each separated by a space, just as you would on a normal command line. Example: program.exe arg1 arg2. The string prior to the first space will taken as the program name, and the string following the first space specifies the arguments to the program.

If you want to execute a program that has a space in the name or path of the file, surround the program name in double-quotes. The first character of the command line should be a double-quote character, and there must be a matching double-quote following the end of the program file name. Any optional arguments to the program follow that, separated by spaces Example: "c:\project files\program name.exe" arg: arg2.

If the flavor of the SFX is

SelfExtractorFlavor.ConsoleApplication, then the SFX starts a new process, using this string as the post-extract comman line. The SFX waits for the process to exit. The exit code of the post-extract command line is returned as the exit code of the command-line self-extractor exe. A non-zero exit code is typically used to indicated a failure by the program. In the case of an SFX, a non-zero exit code may indicate a failure during extraction, OR, it may indicate a failure of the run-after-extract program if specified, OR, it may indicate the run-after-extract program could not be fuond. There is no way to distinguish these conditions from the calling shell, aside from parsing the output of the SFX. If you have Quiet set to true, you may not see error messages, a problem occurs.

If the flavor of the SFX is

SelfExtractorFlavor.WinFormsApplication, then the SFX starts a new process, using this string as the post-extract comman line, and using the extract directory as the working directory for the process. The SFX does not wait for the command to complete, and doe not check the exit code of the program. If the run-after-extract program cannot be fuond, a message box is displayed indicating that fact.

You can specify environment variables within this string, with a format lil %NAME%. The value of these variables will be expanded at the time the SFX is run. Example: %WINDIR%\system32\xcopy.exe may expand at runtime to c:\Windows\System32\xcopy.exe.

By combining this with the

RemoveUnpackedFilesAfterExecute flag, you can create ar SFX that extracts itself, runs a file that was extracted, then deletes all the

files that were extracted. If you want it to run "invisibly" then set Flavo to SelfExtractorFlavor.ConsoleApplication, and set Quiet to true. The user running such an EXE will see a console windc appear, then disappear quickly. You may also want to specify the defaul extract location, with DefaultExtractDirectory.

If you set Flavor to SelfExtractorFlavor.WinFormsApplication, and set Quiet to true, then a GUI with progressbars is displayed, but it is "non interactive" - it accepts no input from the user. Instead the SFX just automatically unpacks and exits.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ProductName Property

<u>Reference</u> ► <u>Ionic.Zip</u> ► <u>SelfExtractorSaveOptions</u> ► **ProductName**

C#

The product name to embed into the generated EXE.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public string ProductName { get; set; }

Public Property ProductName As String
    Get
    Set

public:
property String^ ProductName {
    String^ get ();
    void set (String^ value);
}
```

Remarks

Use any arbitrary string. This text will be displayed while viewing properties of the EXE file in Windows Explorer.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ProductVersion Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ►

ProductVersion

The product version to embed into the generated EXE. It will show up, f example, during a mouseover in Windows Explorer.

Declaration Syntax

- Remarks

You can use any arbitrary string, but a human-readable version number is recommended. For example "v1.2 alpha" or "v4.2 RC2". If you specify nothing, then there is no product version embedded into the EXE.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Quiet Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ► Quiet

Whether the ConsoleApplication SFX will be quiet during extraction.

■ Declaration Syntax

Remarks

This option affects the way the generated SFX runs. By default it is false When you set it to true,...

ConsoleApplication no messages will be emitted during successful operation.	Flavor	Behavior
Double-clicking the SFX in Windows Explorer or as an attachment in an ema will cause a console window to appear briefly, before it disappears. If you run the ConsoleApplication SFX from the cmd.exe prompt, it runs as a normal	ConsoleApplication	successful operation. Double-clicking the SFX in Windows Explorer or as an attachment in an ema will cause a console window to appear briefly, before it disappears. If you run the ConsoleApplication SFX from the

	console app; by default, because it is quiet, it displays no messages to the console. If you pass the -v+ command line argument to the Console SFX when you run it, you will get verbose messages to the console.
WinFormsApplication	the SFX extracts automatically when the application is launched, with no additional user input.

When you set it to false,...

Flavor	Behavior
ConsoleApplication	the extractor will emit a message to the console for each entry extracted. When double-clicking to launch the SFX the console window will remain, and the SFX will emit a message for each file as it extracts. The messages fly by quickly, they won't be easily readable, unless the extracted files are fairly large.
WinFormsApplication	the SFX presents a forms UI and allows the user to select options before extracting.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 RemoveUnpackedFilesAfterExecute

Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ►

RemoveUnpackedFilesAfterExecute

C.#

Whether to remove the files that have been unpacked, after executing the PostExtractCommandLine.

Declaration Syntax

```
Visual Basic
                                     Visual C++
C#
public bool RemoveUnpackedFilesAfterExecute { get; s
Public Property RemoveUnpackedFilesAfterExecute As B
        Get
        Set
public:
property bool RemoveUnpackedFilesAfterExecute {
        bool get ();
        void set (bool value);
}
```

Remarks

If true, and if there is a PostExtractCommandLine, and if the command runs successfully, then the files that the SFX unpacked will be removed afterwards. If the command does not complete successfully (non-zero return code), that is interpreted as a failure, and the extracted files will n be removed.

Setting this flag, and setting Flavor to SelfExtractorFlavor.ConsoleApplication, and setting Quiet to true, results in an SFX that extracts itself, runs a file that was extracted, then deletes all the files that were extracted, with no intervention by the user. You may also want to specify the default extrac

location, with DefaultExtractDirectory.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Sfx ExeWindow Title Property

Reference ► Ionic.Zip ► SelfExtractorSaveOptions ►

SfxExeWindowTitle

The title to display in the Window of a GUI SFX, while it extracts.

■ Declaration Syntax

■ Remarks

By default the title show in the GUI window of a self-extractor is "DotNetZip Self-extractor (http://DotNetZip.codeplex.com/)". You can change that by setting this property before saving the SFX.

This property has an effect only when producing a Self-extractor of flavour SelfExtractorFlavor.WinFormsApplication.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Set Compression Callback Delegate

Reference ► Ionic.Zip ► SetCompressionCallback

C#

Delegate for the callback by which the application tells the library the CompressionLevel to use for a file.

Declaration Syntax

Parameters

localFileName (String)
fileNameInArchive (String)

■ Remarks

Using this callback, the application can, for example, specify that previously-compressed files (.mp3, .png, .docx, etc) should use a CompressionLevel of None, or can set the compression level based on any other factor.

■ See Also

SetCompression

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Þ

Reference ► Ionic.Zip ► SfxGenerationException

C#

Issued when errors occur saving a self-extracting archive.

■ Declaration Syntax

C# Visual Basic Visual C++

[SerializableAttribute]

public class SfxGenerationException : ZipException

<<u>SerializableAttribute</u>> _ Public Class SfxGenerationException _ Inherits <u>ZipException</u>

[SerializableAttribute]

public ref class SfxGenerationException : public Zip

Members

All M	embers	Constructors	Methods	Properties	
		✓ Instance✓ Static		Declare Inherite	
Icon	Member			Description	
≡	SfxGene	erationException()		Default ctor.	
≅	SfxGenerationException(String)		Come on, you how exception Why are you I this document	ns work. ooking a	
9	SfxGene	erationException(S	SerializationInfo	Come on, you	know

	StreamingContext)	how exceptions work. Why are you looking a this documentation?
	Data	Gets a collection of key/value pairs that provide additional user-defined information about the exception. (Inherited from Exception.)
€₩	Equals(Object)	Determines whether the specified <u>Object</u> is equal to the current <u>Object</u> . (Inherited from <u>Object</u>
<u>~</u>	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object
₫	GetBaseException()	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from

		Exception.)
≅	GetHashCode()	Serves as a hash function for a particula type.
		(Inherited from Object
€ 🚱	GetObjectData(SerializationInfo, StreamingContext)	When overridden in a derived class, sets the SerializationInfo with information about the exception. (Inherited from
		Exception.)
€ 🔷	GetType()	Gets the runtime type of the current instance
		(Inherited from Exception.)
==	HelpLink	Gets or sets a link to the help file associate with this exception.
		(Inherited from Exception.)
	HResult	Gets or sets HRESULT, a coded numerical value that is assigned to a specific exception. (Inherited from
		Exception.)
	InnerException	Gets the Exception instance that caused the current exception.

		(Inherited from Exception.)
*	MemberwiseClone()	Creates a shallow copy of the current Object.
		(Inherited from Object
	<u>Message</u>	Gets a message that describes the current exception.
		(Inherited from Exception .)
	Source	Gets or sets the name of the application or the object that causes the error.
		(Inherited from Exception .)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception .)
		<u>Exocption</u> .)
	<u>TargetSite</u>	Gets the method that throws the current exception.
		(Inherited from Exception.)
≡	ToString()	Creates and returns a

string representation of the current exception.

(Inherited from Exception.)

■ Inheritance Hierarchy

Object

- Exception
 - **ZipException**
 - **SfxGenerationException**

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

|

Constructor

Reference ► Ionic.Zip ► SfxGenerationException ►

SfxGenerationException()

■ Members



C#

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Constructor

Reference ► Ionic.Zip ► SfxGenerationException ►

SfxGenerationException()

Default ctor.



■ Declaration Syntax

C# Visual Basic Visual C++

public SfxGenerationException()

Public Sub New

public:
SfxGenerationException()

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Constructor (info, context)

Reference ► Ionic.Zip ► SfxGenerationException ►

SfxGenerationException(SerializationInfo, StreamingContext)

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

Parameters

info (SerializationInfo)

The serialization info for the exception.

context (StreamingContext)

The streaming context from which to deserialize.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Constructor (message)

Reference ► Ionic.Zip ► SfxGenerationException ►

SfxGenerationException(String)

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

```
Visual C++
C#
                   Visual Basic
public SfxGenerationException(
         string message
)
Public Sub New ( _
         message As <u>String</u> _
)
public:
SfxGenerationException(
         String^ message
```

Parameters

message (String)

The message in the exception.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 WriteDelegate Delegate

Reference ► Ionic.Zip ► WriteDelegate

C#

Delegate in which the application writes the ZipEntry content for the named entry.

Declaration Syntax

Parameters

entryName (String)

The name of the entry that must be written.

stream (Stream)

The stream to which the entry data should be written.

Remarks

When you add an entry and specify a WriteDelegate, via AddEntry(String, WriteDelegate), the application code provides the logic that writes the entry data directly into the zip file.

Examples

This example shows how to define a WriteDelegate that obtains a DataSet, and then writes the XML for the DataSet into the zip archive. There's no need to save the XML to a disk file first.

```
private void WriteEntry (String filename, Stream out
{
    DataSet ds1 = ObtainDataSet();
    ds1.WriteXml(output);
}

private void Run()
{
    using (var zip = new ZipFile())
    {
        zip.AddEntry(zipEntryName, WriteEntry);
        zip.Save(zipFileName);
    }
}
```

VB.NET □ Cop

```
Private Sub WriteEntry (ByVal filename As String, By
    DataSet ds1 = ObtainDataSet()
    ds1.WriteXml(stream)
End Sub

Public Sub Run()
    Using zip = New ZipFile
        zip.AddEntry(zipEntryName, New WriteDelegate
        zip.Save(zipFileName)
    End Using
End Sub
```

See Also

AddEntry(String, WriteDelegate)

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

lonic Zip Library v1.9.1.6 **Zip64Option Enumeration**Reference ► lonic.Zip ► Zip64Option

C#

Options for using ZIP64 extensions when saving zip archives.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enum Zip64	1Option	
Public Enumeration	on Zip64Option	
public enum class	Zip64Option	

■ Members

Member	Description	
Default	The default behavior, which is "Never". (For COM clients, this is a 0 (zero).)	
Never	Do not use ZIP64 extensions when writing zip archives. (For COM clients, this is a 0 (zero).) Use ZIP64 extensions when writing zip archives, as necessary. For example, when a single entry exceeds 0xFFFFFF in size, or when the archive as a whole exceeds 0xFFFFFFF in size, or when there are more than 65535 entries in an archive. (For COM clients, this is a 1.)	
AsNecessary		
Always	Always use ZIP64 extensions when writin zip archives, even when unnecessary. (For COM clients, this is a 2.)	

■ Remarks

Designed many years ago, the <u>original zip specification from PKWARE</u> allowed for 32-bit quantities for the compressed and uncompressed size of zip entries, as well as a 32-bit quantity for specifying the length of the zip archive itself, and a maximum of 65535 entries. These limits are now regularly exceeded in many backup and archival scenarios. Recently, PKWare added extensions to the original zip spec, called "ZIP64 extensions", to raise those limitations. This property governs whether DotNetZip will use those extensions when writing zip archives. The use these extensions is optional and explicit in DotNetZip because, despite the status of ZIP64 as a bona fide standard, many other zip tools and libraries do not support ZIP64, and therefore a zip file with ZIP64 extensions may be unreadable by some of those other tools.

Set this property to **Always** to always use ZIP64 extensions when saving, regardless of whether your zip archive needs it. Suppose you at 5 files, each under 100k, to a ZipFile. If you specify Always for this flag, you will get a ZIP64 archive, though the archive does not need to use ZIP64 because none of the original zip limits had been exceeded.

Set this property to **Never** to tell the DotNetZip library to never use ZIP6 extensions. This is useful for maximum compatibility and interoperability at the expense of the capability of handling large files or large archives. NB: Windows Explorer in Windows XP and Windows Vista cannot currently extract files from a zip64 archive, so if you want to guarantee that a zip archive produced by this library will work in Windows Explorer use Never. If you set this property to **Never**, and your application creates a zip that would exceed one of the Zip limits, the library will thro an exception while saving the zip file.

Set this property to **AsNecessary** to tell the DotNetZip library to use the ZIP64 extensions when required by the entry. After the file is compressed, the original and compressed sizes are checked, and if the exceed the limits described above, then zip64 can be used. That is the general idea, but there is an additional wrinkle when saving to a non-seekable device, like the ASP.NET Response.OutputStream, or Console.Out. When using non-seekable streams for output, the ent header - which indicates whether zip64 is in use - is emitted before it is known if zip64 is necessary. It is only after all entries have been saved that it can be known if ZIP64 will be required. On seekable output

streams, after saving all entries, the library can seek backward and reemit the zip file header to be consistent with the actual ZIP64 requirement. But using a non-seekable output stream, the library cannot seek backward, so the header can never be changed. In other words, the archive's use of ZIP64 extensions is not alterable after the header is emitted. Therefore, when saving to non-seekable streams, using **AsNecessary** is the same as using **Always**: it will always produce a zip archive that uses ZIP64 extensions.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Entry Class

Reference ► Ionic.Zip ► ZipEntry

C# -

Represents a single entry in a ZipFile. Typically, applications get a ZipEntry by enumerating the entries within a ZipFile, or by adding an entry to a ZipFile.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public class ZipE	Entry	
Public Class ZipE	Entry	
public ref class	ZipEntry	

■ Members

All M	embers	Constructors	Methods	Prope	rties	
✓ Public✓ Protected		✓ Instance✓ Static			Declare Inherite	
Icon	Member				Descrip	tion
∃	ZipEntry	Q			Default	construc
	Accesse	<u>dTime</u>			Last Acc the file r the entry	epreser
	Alternate	Encoding			Specifie text enc this ZipE	oding us

AlternateEncodingUsage	Describes if and this instance sho apply AlternateEncodi encode the Filel and Comment, v saving.
Attributes	The file attribute the entry.
BitField	The bitfield for the entry as defined zip spec. You probably never to look at this.
Comment	The comment attached to the ZipEntry.
CompressedSize	The compressed of the file, in byt within the zip are
CompressionLevel	Sets the compressive level to be used the entry when some the zip archive. applies only for

		CompressionMe = DEFLATE.
***	CompressionMethod	The compressio method employe this ZipEntry.
	CompressionRatio	The ratio of compressed size uncompressed sthe ZipEntry.
	Crc	The 32-bit CRC (Cyclic Redunda Check) on the contents of the ZipEntry.
	CreationTime	The file creation for the file represented by entry.
	EmitTimesInUnixFormatWhenSaving	Specifies wheth Creation, Acces Modified times f given entry will I emitted in "Unix format" when the archive is saved

	EmitTimesInWindowsFormatWhenSaving	Specifies wheth Creation, Acces Modified times f given entry will I emitted in "Winc format" when th archive is saved
	Encryption	Set this to speci which encryption algorithm to use the entry when s it to a zip archiv
₫ ۞	Equals(Object)	Determines whe the specified Ot equal to the currobject. (Inherited from Object.)
Ξ₩	Extract()	Extract the entry the filesystem, s at the current we directory.
₫₩	Extract(ExtractExistingFileAction)	Extract the entry file in the filesys using the specif behavior when extraction would overwrite an exi file.

₫♠	Extract(Stream)	Extracts the ent the specified str
≓ ₩	Extract(String)	Extract the entry the filesystem, s at the specified directory.
⊒ 🍑	Extract(String, ExtractExistingFileAction)	Extract the entry the filesystem, s at the specified directory, and us the specified be when extraction overwrite an exi file.
	<u>ExtractExistingFile</u>	The action the li should take whe extracting a file already exists.
Ξ₩	ExtractWithPassword(String)	Extract the entry the filesystem, the current work directory and the specified passw
=0	ExtractWithPassword(String, String)	Extract the entry the filesystem, s

		at the specified directory, and us the specified password.
Ξ₩	ExtractWithPassword(ExtractExistingFileAction, String)	Extract the entry file in the filesys relative to the cudirectory, using specified behav when extraction overwrite an exifile.
Ξ₩	ExtractWithPassword(String, ExtractExistingFileAction, String)	Extract the entry the filesystem, s at the specified directory, and us the specified be when extraction overwrite an exi file.
₫ 🍑	ExtractWithPassword(Stream, String)	Extracts the ent the specified str using the specif Password. For example, the ca could extract to Console.Out, or MemoryStream.
	<u>FileName</u>	The name of the

		contained in the ZipEntry.
ij [®]	Finalize()	Allows an Object attempt to free resources and perform other cloperations before Object is reclain garbage collecti
		(Inherited from Object.)
€	GetHashCode()	Serves as a has function for a particular type.
		(Inherited from Object.)
≡	GetType()	Gets the <u>Type</u> o current instance
		(Inherited from Object.)
	IncludedInMostRecentSave	Indicates wheth entry was includ the most recent
	Info	Provides a hum readable string information about ZipEntry.
	<u>InputStream</u>	The stream that

		provides conten the ZipEntry.
	<u>InputStreamWasJitProvided</u>	A flag indicating whether the InputStream wa provided Just-in
	IsDirectory	True if the entry directory (not a This is a readon property on the
	IsText	Indicates whether entry is marked text file. Be care when using on the property. Unless have a good readyou should probignore this property.
	LastModified	The time and day which the file indicated by the ZipEntry was modified.
Ģ [¯] ©	MemberwiseClone()	Creates a shallo copy of the curre Object.

		(Inherited from Object.)
	ModifiedTime	Last Modified tir the file represen the entry.
∄ 🍑	OpenReader()	Opens a readab stream correspo to the zip entry i archive. The stre decompresses a decrypts as necessary, as it read.
≓	OpenReader(String)	Opens a readab stream for an encrypted zip er the archive. The stream decompt and decrypts as necessary, as it read.
	OutputUsedZip64	Indicates whethe entry actually us ZIP64 extensior it was most rece written to the ou file or stream.
	Password	The Password to used when encr

		a ZipEntry uzipFile. Sar or when decrypt entry upon Extra
	ProvisionalAlternateEncoding	Obsolete. The text encoding use for the FileN and Comment of ZipEntry, when the default encoding insufficient.
	RequiresZip64	Indicates wheth entry requires Z extensions.
	SetCompression	A callback that a the application t specify the compression to for a given entry is about to be a to the zip archiv
€	SetEntryTimes(DateTime, DateTime, DateTime)	Sets the NTFS Creation, Acces Modified times f given entry.
	Source	An enum indicate the source of the

		ZipEntry.
	Timestamp	The type of time attached to the ZipEntry.
Ξ₩	ToString()	Provides a string representation coinstance. (Overrides Object. To String)
	UncompressedSize	The size of the 1 bytes, before compression, or extraction.
	<u>UsesEncryption</u>	A derived prope that is true if t entry uses encry
	UseUnicodeAsNecessary	Obsolete. Set to indicate whether to use I encoding for filenames and comments.
	VersionNeeded	The version of the engine needed the ZipEnti



■ Inheritance Hierarchy

Object

ZipEntry

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Þ

Ionic Zip Library v1.9.1.6 Zip Entry Constructor

Reference ► Ionic.Zip ► ZipEntry ► ZipEntry()

C#

Default constructor.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public ZipEntry()

Public Sub New

public:
ZipEntry()
```

■ Remarks

Applications should never need to call this directly. It is exposed to support COM Automation environments.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Accessed Time Property

Reference ► Ionic.Zip ► ZipEntry ► AccessedTime

C# _-

Last Access time for the file represented by the entry.

■ Declaration Syntax

■ Remarks

This value may or may not be meaningful. If the ZipEntry was read from an existing Zip archive, this information may not be available. For a explanation of why, see ModifiedTime.

See Also

ModifiedTime
CreationTime
SetEntryTimes(DateTime, DateTime, DateTime)

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Alternate Encoding Property

Reference ► Ionic.Zip ► ZipEntry ► AlternateEncoding

Specifies the alternate text encoding used by this ZipEntry

Declaration Syntax

Remarks

The default text encoding used in Zip files for encoding filenames and comments is IBM437, which is something like a superset of ASCII. In cases where this is insufficient, applications can specify an alternate encoding.

When creating a zip file, the usage of the alternate encoding is governe by the <u>AlternateEncodingUsage</u> property. Typically you would set both properties to tell DotNetZip to employ an encoding that is not IBM437 in the zipfile you are creating.

Keep in mind that because the ZIP specification states that the only vali encodings to use are IBM437 and UTF-8, if you use something other than that, then zip tools and libraries may not be able to successfully reather the zip archive you generate. The zip specification states that applications should presume that IBM437 is in use, except when a special bit is set, which indicates UTF-There is no way to specify an arbitrary code page, within the zip file itse When you create a zip file encoded with gb2312 or ibm861 or anything other than IBM437 or UTF-8, then the application that reads the zip file needs to "know" which code page to use. In some cases, the code page used when reading is chosen implicitly. For example, WinRar uses the ambient code page for the host desktop operating system. The pitfall here is that if you create a zip in Copenhagen and send it to Tokyo, the reader of the zipfile may not be able to decode successfully.

Examples

This example shows how to create a zipfile encoded with a languagespecific encoding:

```
Using (var zip = new ZipFile())
{
   zip.AlternateEnoding = System.Text.Encoding.GetEn
   zip.AlternateEnodingUsage = ZipOption.Always;
   zip.AddFileS(arrayOfFiles);
   zip.Save("Myarchive-Encoded-in-IBM861.zip");
}
```

See Also

<u>AlternateEncodingUsage</u>

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Jonic Zip (Module: Jonic Zip) Version: 1.0.1.8 (1.0.1)

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

Ionic Zip Library v1.9.1.6 Alternate Encoding Usage Property

Reference ► Ionic.Zip ► ZipEntry ► AlternateEncodingUsage

Describes if and when this instance should apply AlternateEncoding to encode the FileName and Comment, when saving.

■ Declaration Syntax

See Also

AlternateEncoding

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Attributes Property Reference ► Ionic.Zip ► ZipEntry ► Attributes C#

The file attributes for the entry.

■ Declaration Syntax

■ Remarks

The <u>attributes</u> in NTFS include ReadOnly, Archive, Hidden, System, and Indexed. When adding a ZipEntry to a ZipFile, these attributes are set implicitly when adding an entry from the filesystem. When adding ar entry from a stream or string, the Attributes are not set implicitly. Regardless of the way an entry was added to a ZipFile, you can set the attributes explicitly if you like.

When reading a ZipEntry from a ZipFile, the attributes are set according to the data stored in the ZipFile. If you extract the entry from the archive to a filesystem file, DotNetZip will set the attributes on the resulting file accordingly.

The attributes can be set explicitly by the application. For example the application may wish to set the FileAttributes. ReadOnly bit

for all entries added to an archive, so that on unpack, this attribute will t set on the extracted file. Any changes you make to this property are made permanent only when you call a Save() method on the ZipFile instance that contains the ZipEntry.

For example, an application may wish to zip up a directory and set the ReadOnly bit on every file in the archive, so that upon later extraction, the resulting files will be marked as ReadOnly. Not every extraction tool respects these attributes, but if you unpack with DotNetZip, as for example in a self-extracting archive, then the attributes will be set as the are stored in the ZipFile.

These attributes may not be interesting or useful if the resulting archive extracted on a non-Windows platform. How these attributes get used upon extraction depends on the platform and tool used.

This property is only partially supported in the Silverlight version of the library: applications can read attributes on entries within ZipFiles. But extracting entries within Silverlight will not set the attributes on the extracted files.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

The bitfield for the entry as defined in the zip spec. You probably never need to look at this.

Declaration Syntax

```
C# Visual Basic Visual C++

public short BitField { get; }

Public ReadOnly Property BitField As Short
    Get

public:
property short BitField {
        short get ();
}
```

■ Remarks

You probably do not need to concern yourself with the contents of this property, but in case you do:

bit	meaning
0	set if encryption is used.
1- 2	set to determine whether normal, max, fast deflation. DotNetZip library always leaves these bits unset when writing (indicating "normal" deflation"), but can read an entry with any value here.
3	Indicates that the Crc32, Compressed and Uncompressed sizes ar zero in the local header. This bit gets set on an entry during writing a zip file, when it is saved to a non-seekable output stream.
4	reserved for "enhanced deflating". This library doesn't do enhanced deflating.

- 5 set to indicate the zip is compressed patched data. This library doesn't do that.
- 6 set if PKWare's strong encryption is used (must also set bit 1 if bit (is set). This bit is not set if WinZip's AES encryption is set.
- 7 not used
- 8 not used
- 9 not used
- 10 not used
- 11 Language encoding flag (EFS). If this bit is set, the filename and comment fields for this file must be encoded using UTF-8. This library currently does not support UTF-8.
- 12 Reserved by PKWARE for enhanced compression.
- 13 Used when encrypting the Central Directory to indicate selected data values in the Local Header are masked to hide their actual values. See the section in the Zip specification describing the Strong Encryption Specification for details.
- 14 Reserved by PKWARE.
- 15 Reserved by PKWARE.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Comment Property Reference ► Ionic.Zip ► ZipEntry ► Comment C#

The comment attached to the ZipEntry.

Declaration Syntax

■ Remarks

Each entry in a zip file can optionally have a comment associated to it. The comment might be displayed by a zip tool during extraction, for example.

By default, the **Comment** is encoded in IBM437 code page. You can specify an alternative with <u>AlternateEncoding</u> and <u>AlternateEncodingUsage</u>.

See Also

AlternateEncoding
AlternateEncodingUsage

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

The compressed size of the file, in bytes, within the zip archive.

Declaration Syntax

```
C# Visual Basic Visual C++
public long CompressedSize { get; }

Public ReadOnly Property CompressedSize As Long
    Get

public:
property long long CompressedSize {
    long long get ();
}
```

Remarks

When reading a ZipFile, this value is read in from the existing zip file. When creating or updating a ZipFile, the compressed size is computed during compression. Therefore the value on a ZipEntry is valid after a call to Save() (or one of its overloads) in that case.

See Also

<u>UncompressedSize</u>

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Appropriate Tip (Madula: Ispia Tip) Version: 1, 0, 1, 0, (1, 0, 1, 0)

Ionic Zip Library v1.9.1.6 Compression Level Property

Reference ► Ionic.Zip ► ZipEntry ► CompressionLevel

C# •

Sets the compression level to be used for the entry when saving the zip archive. This applies only for CompressionMethod = DEFLATE.

Declaration Syntax

■ Remarks

When using the DEFLATE compression method, Varying the compression level used on entries can affect the size-vs-speed tradeoff when compression and decompressing data streams or files.

If you do not set this property, the default compression level is used, which normally gives a good balance of compression efficiency and compression speed. In some tests, using <code>BestCompression</code> can double the time it takes to compress, while delivering just a small increase in compression efficiency. This behavior will vary with the type of data you compress. If you are in doubt, just leave this setting alone, and accept the default.

When setting this property on a ZipEntry that is read from an existin

zip file, calling ZipFile.Save() will cause the new CompressionLevel to be used on the entry in the newly saved zip file.

Setting this property may have the side effect of modifying the CompressionMethod property. If you set the CompressionLevel to a value other than None, CompressionMethod will be set to Deflate, if it was previously None.

Setting this property has no effect if the CompressionMethod is something other than Deflate or None.

■ See Also

CompressionMethod

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Compression Method Property

Reference ► Ionic.Zip ► ZipEntry ► CompressionMethod

The compression method employed for this ZipEntry.

Declaration Syntax

C#

■ Remarks

<u>The Zip specification</u> allows a variety of compression methods. This library supports just two: 0x08 = Deflate. 0x00 = Store (no compression) for reading or writing.

When reading an entry from an existing zipfile, the value you retrieve here indicates the compression method used on the entry by the original creator of the zip. When writing a zipfile, you can specify either 0x08 (Deflate) or 0x00 (None). If you try setting something else, you will get a exception.

You may wish to set CompressionMethod to CompressionMethod. None (0) when zipping already-compresse data like a jpg, png, or mp3 file. This can save time and cpu cycles.

When setting this property on a ZipEntry that is read from an existin

zip file, calling ZipFile.Save() will cause the new CompressionMethod to be used on the entry in the newly saved zip file.

Setting this property may have the side effect of modifying the CompressionLevel property. If you set the CompressionMethod to a value other than None, and CompressionLevel is previously set to None, then CompressionLevel will be set to Default.

Examples

In this example, the first entry added to the zip archive uses the default behavior - compression is used where it makes sense. The second entr the MP3 file, is added to the archive without being compressed.

```
C# □ Cop
```

```
using (ZipFile zip = new ZipFile(ZipFileToCreate))
{
   ZipEntry e1= zip.AddFile(@"notes\Readme.txt");
   ZipEntry e2= zip.AddFile(@"music\StopThisTrain.mp3
   e2.CompressionMethod = CompressionMethod.None;
   zip.Save();
}
```

```
VB.NET
```

```
Using zip As New ZipFile(ZipFileToCreate)
  zip.AddFile("notes\Readme.txt")
  Dim e2 as ZipEntry = zip.AddFile("music\StopThisTr
  e2.CompressionMethod = CompressionMethod.None
  zip.Save
End Using
```

See Also

CompressionMethod

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compression Ratio Property

Reference ► Ionic.Zip ► ZipEntry ► CompressionRatio

The ratio of compressed size to uncompressed size of the ZipEntry.

Declaration Syntax

```
C# Visual Basic Visual C++
public double CompressionRatio { get; }

Public ReadOnly Property CompressionRatio As Double Get

public:
property double CompressionRatio {
    double get ();
}
```

■ Remarks

This is a ratio of the compressed size to the uncompressed size of the entry, expressed as a double in the range of 0 to 100+. A value of 100 indicates no compression at all. It could be higher than 100 when the compression algorithm actually inflates the data, as may occur for small files, or uncompressible data that is encrypted.

You could format it for presentation to a user via a format string of " {3,5:F0}%" to see it as a percentage.

If the size of the original uncompressed file is 0, implying a denominator of 0, the return value will be zero.

This property is valid after reading in an existing zip file, or after saving the ZipFile that contains the ZipEntry. You cannot know the effect of compression transform until you try it.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Crc Property

Reference ► Ionic.Zip ► ZipEntry ► Crc
C#
```

The 32-bit CRC (Cyclic Redundancy Check) on the contents of the ZipEntry.

Declaration Syntax

```
C# Visual Basic Visual C++

public int Crc { get; }

Public ReadOnly Property Crc As Integer
    Get

public:
property int Crc {
    int get ();
}
```

■ Remarks

You probably don't need to concern yourself with this. It is used internal by DotNetZip to verify files or streams upon extraction.

The value is a <u>32-bit CRC</u> using 0xEDB88320 for the polynomial. This is the same CRC-32 used in PNG, MPEG-2, and other protocols and formats. It is a read-only property; when creating a Zip archive, the CRC for each entry is set only after a call to <u>Save()</u> on the containing ZipFile. When reading an existing zip file, the value of this property reflects the stored CRC for the entry.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Creation Time Property

Reference ► Ionic.Zip ► ZipEntry ► CreationTime

C#

The file creation time for the file represented by the entry.

■ Declaration Syntax

■ Remarks

This value may or may not be meaningful. If the ZipEntry was read from an existing zip archive, and the creation time was not set on the entry when the zip file was created, then this property may be meaningless. For an explanation of why, see ModifiedTime.

See Also

ModifiedTime
AccessedTime
SetEntryTimes(DateTime, DateTime, DateTime)

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library

v1.9.1.6 EmitTimesInUnixFormatWhenSaving Property

Reference ► Ionic.Zip ► ZipEntry ►

EmitTimesInUnixFormatWhenSaving

Specifies whether the Creation, Access, and Modified times for the give entry will be emitted in "Unix(tm) format" when the zip archive is saved.

Declaration Syntax

■ Remarks

An application creating a zip archive can use this flag to explicitly specif that the file times for the entry should or should not be stored in the zip archive in the format used by Unix. By default this flag is false, meaning the Unix-format times are not stored in the zip archive.

When adding an entry from a file or directory, the Creation (CreationTime), Access (AccessedTime), and Modified (ModifiedTime) times for the given entry are automatically set from the filesystem value: When adding an entry from a stream or string, all three values are implicitly set to DateTime.Now. Applications can also explicitly set those

times by calling <u>SetEntryTimes(DateTime, DateTime, DateTime)</u>.

PKWARE's zip specification describes multiple ways to format these times in a zip file. One is the format Windows applications normally use 100ns ticks since Jan 1, 1601 UTC. The other is a format Unix applications typically use: seconds since Jan 1, 1970 UTC. Each format can be stored in an "extra field" in the zip entry when saving the zip archive. The former uses an extra field with a Header Id of 0x000A, while latter uses a header ID of 0x5455.

Not all tools and libraries can interpret these fields. Windows compresse folders is one that can read the Windows Format timestamps, while I believe the Infozip tools can read the Unix format timestamps. Although the time values are easily convertible, subject to a loss of precision, son tools and libraries may be able to read only one or the other. DotNetZip can read or write times in either or both formats.

The times stored are taken from <u>ModifiedTime</u>, <u>AccessedTime</u>, and <u>CreationTime</u>.

This property is not mutually exclusive from the EmitTimesInWindowsFormatWhenSaving property. It is possible that a zip entry can embed the timestamps in both forms, one form, or neither. But, there are no guarantees that a program running on Mac or Linux w gracefully handle NTFS Formatted times, or that a non-DotNetZip-powered application running on Windows will be able to handle file time in Unix format. When in doubt, test.

Normally you will use the <u>ZipFile.EmitTimesInUnixFormatWhenSaving</u> property, to specify the behavior for all entries, rather than the property on each individual entry.

■ See Also

SetEntryTimes(DateTime, DateTime, DateTime)
EmitTimesInWindowsFormatWhenSaving
EmitTimesInUnixFormatWhenSaving
CreationTime
AccessedTime
ModifiedTime

Send comments on this topic to ${\underline{\sf dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

•

Ionic Zip Library

v1.9.1.6EmitTimesInWindowsFormatWhenSaving **Property**

Reference ► Ionic.Zip ► ZipEntry ►

EmitTimesInWindowsFormatWhenSaving

Specifies whether the Creation, Access, and Modified times for the give entry will be emitted in "Windows format" when the zip archive is saved.

Declaration Syntax

```
C#
                  Visual Basic
                                     Visual C++
public bool EmitTimesInWindowsFormatWhenSaving { get
Public Property EmitTimesInWindowsFormatWhenSaving A
        Get
        Set
public:
property bool EmitTimesInWindowsFormatWhenSaving {
        bool get ();
        void set (bool value);
}
```

Remarks

An application creating a zip archive can use this flag to explicitly specif that the file times for the entry should or should not be stored in the zip archive in the format used by Windows. The default value of this proper is true.

When adding an entry from a file or directory, the Creation (<u>CreationTime</u>), Access (<u>AccessedTime</u>), and Modified (<u>ModifiedTime</u>) times for the given entry are automatically set from the filesystem value: When adding an entry from a stream or string, all three values are implicitly set to DateTime.Now. Applications can also explicitly set those times by calling <u>SetEntryTimes(DateTime, DateTime, DateTime)</u>.

PKWARE's zip specification describes multiple ways to format these times in a zip file. One is the format Windows applications normally use 100ns ticks since Jan 1, 1601 UTC. The other is a format Unix applications typically use: seconds since January 1, 1970 UTC. Each format can be stored in an "extra field" in the zip entry when saving the zip archive. The former uses an extra field with a Header Id of 0x000A, while the latter uses a header ID of 0x5455.

Not all zip tools and libraries can interpret these fields. Windows compressed folders is one that can read the Windows Format timestamps, while I believe the Infozip tools can read the Unix format timestamps. Although the time values are easily convertible, subject to loss of precision, some tools and libraries may be able to read only one or the other. DotNetZip can read or write times in either or both formats.

The times stored are taken from <u>ModifiedTime</u>, <u>AccessedTime</u>, and <u>CreationTime</u>.

This property is not mutually exclusive from the EmitTimesInUnixFormatWhenSaving property. It is possible that a zip entry can embed the timestamps in both forms, one form, or neither. Bu there are no guarantees that a program running on Mac or Linux will gracefully handle NTFS Formatted times, or that a non-DotNetZip-powered application running on Windows will be able to handle file time in Unix format. When in doubt, test.

Normally you will use the

<u>ZipFile.EmitTimesInWindowsFormatWhenSaving</u> property, to specify the behavior for all entries in a zip, rather than the property on each individual entry.

See Also

SetEntryTimes(DateTime, DateTime, DateTime)
EmitTimesInUnixFormatWhenSaving
CreationTime
AccessedTime
ModifiedTime

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Encryption Property Reference ► Ionic.Zip ► ZipEntry ► Encryption C#

Set this to specify which encryption algorithm to use for the entry when saving it to a zip archive.

Declaration Syntax

Remarks

Set this property in order to encrypt the entry when the ZipFile is saved. When setting this property, you must also set a <u>Password</u> on the entry. If you set a value other than <u>None</u> on this property and do not set <u>Password</u> then the entry will not be encrypted. The <u>ZipEntry</u> data is encrypted as the <u>ZipFile</u> is saved, when you call <u>Save()</u> or one of its cousins on the containing <u>ZipFile</u> instance. You do not need to specify the <u>Encryption</u> when extracting entries from an archive.

The Zip specification from PKWare defines a set of encryption algorithm and the data formats for the zip archive that support them, and PKWare supports those algorithms in the tools it produces. Other vendors of tool and libraries, such as WinZip or Xceed, typically support a *subset* of the algorithms specified by PKWare. These tools can sometimes support

additional different encryption algorithms and data formats, not specified by PKWare. The AES Encryption specified and supported by WinZip is the most popular example. This library supports a subset of the complet set of algorithms specified by PKWare and other vendors.

There is no common, ubiquitous multi-vendor standard for strong encryption within zip files. There is broad support for so-called "traditional" Zip encryption, sometimes called Zip 2.0 encryption, as specified by PKWare, but this encryption is considered weak and breakable. This library currently supports the Zip 2.0 "weak" encryption, and also a stronger WinZip-compatible AES encryption, using either 121 bit or 256-bit key strength. If you want DotNetZip to support an algorithn that is not currently supported, call the author of this library and maybe we can talk business.

The ZipFile class also has a Encryption property. In most cases you will use *that* property when setting encryption. This property takes precedence over any Encryption set on the ZipFile itself. Typically, you would use the per-entry Encryption when most entries in the zip archive use one encryption algorithm, and a few entries use a different one. If all entries in the zip file use the same Encryption, then it is simpler to just set this property on the ZipFile itself, when creating a z archive.

Some comments on updating archives: If you read a ZipFile, you came modify the Encryption on an encrypted entry: you can remove encryption from an entry that was encrypted; you can encrypt an entry that was not encrypted previously; or, you can change the encryption algorithm. The changes in encryption are not made permanent until you call Save() on the ZipFile. To effect changes in encryption, the entry content is streamed through several transformations, depending on the modification the application has requested. For example if the entry is not encrypted and the application sets Encryption to PkzipWeak, then at the time of Save(), the original entry is read and decompressed, then recompressed and encrypted. Conversely, if the original entry is encrypted with PkzipWeak encryption, and the application sets the Encryption property to WinZipAes128, then at the time of Save(), the original entry is decrypted via PKZIP encryption and

decompressed, then re-compressed and re-encrypted with AES. This all happens automatically within the library, but it can be time-consuming for large entries.

Additionally, when updating archives, it is not possible to change the password when changing the encryption algorithm. To change both the algorithm and the password, you need to Save() the zipfile twice. First s the Encryption to None, then call Save(). Then set the Encryption to the new value (not "None"), then call Save() once again.

The WinZip AES encryption algorithms are not supported on the .NET Compact Framework.

Examples

This example creates a zip archive that uses encryption, and then extracts entries from the archive. When creating the zip archive, the ReadMe.txt file is zipped without using a password or encryption. The other file uses encryption.

C#

```
// Create a zip archive with AES Encryption.
using (ZipFile zip = new ZipFile())
{
    zip.AddFile("ReadMe.txt")
    ZipEntry e1= zip.AddFile("2008-Regional-Sales-Re e1.Encryption= EncryptionAlgorithm.WinZipAes256; e1.Password= "Top.Secret.No.Peeking!"; zip.Save("EncryptedArchive.zip");
}

// Extract a zip archive that uses AES Encryption.
// You do not need to specify the algorithm during e using (ZipFile zip = ZipFile.Read("EncryptedArchive.
{
    // Specify the password that is used during extr // all entries that require a password:
```

```
zip.Password= "Top.Secret.No.Peeking!";
zip.ExtractAll("extractDirectory");
}
```

VB.NET

```
' Create a zip that uses Encryption.
Using zip As New ZipFile()
    zip.AddFile("ReadMe.txt")
    Dim e1 as ZipEntry
    e1= zip.AddFile("2008-Regional-Sales-Report.pdf"
    e1.Encryption= EncryptionAlgorithm.WinZipAes256
    e1.Password= "Top.Secret.No.Peeking!"
    zip.Save("EncryptedArchive.zip")
End Using
' Extract a zip archive that uses AES Encryption.
' You do not need to specify the algorithm during ex
Using (zip as ZipFile = ZipFile.Read("EncryptedArchi
      Specify the password that is used during extra
    ' all entries that require a password:
    zip.Password= "Top.Secret.No.Peeking!"
    zip.ExtractAll("extractDirectory")
End Using
```

Exceptions

Exception	Condition
InvalidOperationException	Thrown in the setter if EncryptionAlgorithm.Unsupported is specified.

■ See Also

ZipEntry.Password ZipFile.Encryption

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa



This method has a bunch of overloads! One of them is sure to be the right one for you... If you don't like these, check out the ExtractWithPassword() methods.

■ Members

Icon	Member	Description
₫ 🚱	Extract()	Extract the entry to the filesystem, starting at the current working directory.
∄	Extract(ExtractExistingFileAction)	Extract the entry to a file in the filesystem, using the specified behavior when extraction woul overwrite an existing file.
≅	Extract(Stream)	Extracts the entry to the specified stream.
∃	Extract(String)	Extract the entry to the filesystem, starting at the specified base directory.
ۥ	Extract(String, ExtractExistingFileAction)	Extract the entry to the filesystem, starting at the specified base directory, and using the specified behavior when extraction would overwrite an existing file.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Extract Method Reference ► Ionic.Zip ► ZipEntry ► Extract() C#

Extract the entry to the filesystem, starting at the current working directory.

Declaration Syntax

```
C# Visual Basic Visual C++

public void Extract()

Public Sub Extract

public:
void Extract()
```

■ Remarks

This method extracts an entry from a zip file into the current working directory. The path of the entry as extracted is the full path as specified the zip archive, relative to the current working directory. After the file is extracted successfully, the file attributes and timestamps are set.

The action taken when extraction an entry would overwrite an existing fi is determined by the ExtractExistingFile property.

Within the call to Extract(), the content for the entry is written into a filesystem file, and then the last modified time of the file is set according to the LastModified property on the entry. See the remarks the LastModified property for some details about the last modified time.

See Also

<u>ExtractExistingFile</u> <u>Extract(ExtractExistingFileAction)</u>

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Extract Method (extractExistingFile)

Reference ► Ionic.Zip ► ZipEntry ►

Extract(ExtractExistingFileAction)

Extract the entry to a file in the filesystem, using the specified behavior when extraction would overwrite an existing file.

Declaration Syntax

Parameters

extractExistingFile (ExtractExistingFileAction)

The action to take if extraction would overwrite an existing file.

Remarks

See the remarks on the <u>LastModified</u> property, for some details about how the last modified time of the file is set after extraction.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | Extract Method (stream) | Reference | Ionic.Zip | ZipEntry | Extract(Stream) | C#

Extracts the entry to the specified stream.

■ Declaration Syntax

Parameters

stream (Stream)

the stream to which the entry should be extracted.

Remarks

The caller can specify any write-able stream, for example a <u>FileStream</u>, <u>MemoryStream</u>, or ASP.NET's <u>Response</u>. <u>OutputStream</u>. The content will be decrypted and decompressed as necessary. If the entry encrypted and no password is provided, this method will throw.

The position on the stream is not reset by this method before it extracts You may want to call stream. Seek() before calling ZipEntry. Extract().

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Extract Method (baseDirectory)

Reference ► Ionic.Zip ► ZipEntry ► Extract(String)

C#

Extract the entry to the filesystem, starting at the specified base director

■ Declaration Syntax

Parameters

baseDirectory (String)

the pathname of the base directory

Remarks

Using this method, existing entries in the filesystem will not be overwritten. If you would like to force the overwrite of existing files, see the ExtractExistingFile property, or call ExtractExistingFile Action).

See the remarks on the <u>LastModified</u> property, for some details about how the last modified time of the created file is set.

Examples

This example extracts only the entries in a zip file that are .txt files, into directory called "textfiles".

C# □ Cop

```
using (ZipFile zip = ZipFile.Read("PackedDocuments.z
{
   foreach (string s1 in zip.EntryFilenames)
   {
     if (s1.EndsWith(".txt"))
      {
       zip[s1].Extract("textfiles");
      }
   }
}
```

VB.NET

■ See Also

ExtractExistingFile
Extract(String, ExtractExistingFileAction)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

4

Ionic Zip Library v1.9.1.6 Extract Method (baseDirectory, extractExistingFile)

Reference ► Ionic.Zip ► ZipEntry ► Extract(String,

ExtractExistingFileAction)

Extract the entry to the filesystem, starting at the specified base director and using the specified behavior when extraction would overwrite an existing file.

Declaration Syntax

ExtractExistingFileAction extractExistingFil

Parameters

baseDirectory (String)

the pathname of the base directory

extractExistingFile (ExtractExistingFileAction)

The action to take if extraction would overwrite an existing file.

■ Remarks

See the remarks on the <u>LastModified</u> property, for some details about how the last modified time of the created file is set.

Examples

C# □ Cop

VB.NET

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Extract Existing File Property

Reference ► Ionic.Zip ► ZipEntry ► ExtractExistingFile

The action the library should take when extracting a file that already exists.

Declaration Syntax

```
C# Visual Basic Visual C++

public ExtractExistingFileAction ExtractExistingFile

Public Property ExtractExistingFile As ExtractExisti
    Get
    Set

public:

property ExtractExistingFileAction ExtractExistingFileAction get ();
    void set (ExtractExistingFileAction value);
}
```

■ Remarks

This property affects the behavior of the Extract methods (one of the Extract() or ExtractWithPassword() overloads), when extraction would would overwrite an existing filesystem file. If you do no set this property, the library throws an exception when extracting an ent would overwrite an existing file.

This property has no effect when extracting to a stream, or when the file to be extracted does not already exist.

Examples

This example shows how to set the ExtractExistingFile property in an ExtractProgress event, in response to user input. The ExtractProgress event is invoked if and only if the

ExtractExistingFile property was previously set to ExtractExistingFileAction.InvokeExtractProgres

C# Cop public static void ExtractProgress(object sender, Ex { if (e.EventType == ZipProgressEventType.Extracti Console.WriteLine("extract {0} ", e.CurrentE else if (e.EventType == ZipProgressEventType.Ext { ZipEntry entry = e.CurrentEntry; string response = null; // Ask the user if he wants overwrite the fi do { Console.Write("Overwrite {0} in {1} ? (y response = Console.ReadLine(); Console.WriteLine(); } while (response != null && response[0]!='Y response[0]!='N' && response[0]!='C if (response[0]=='C') e.Cancel = true; else if (response[0]=='Y') entry.ExtractExistingFile = ExtractExist else entry.ExtractExistingFile= ExtractExisti

See Also

}

}

ExtractExistingFile

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa **▶**

Ionic Zip Library v1.9.1.6 ExtractWithPassword Method

Reference ► Ionic.Zip ► ZipEntry ► ExtractWithPassword()

This method has a bunch of overloads! One of them is sure to be the right one for you...

■ Members

lcon	Member	Description
€	ExtractWithPassword(String)	Extract the entry to the filesystem using the currer working director and the specifie password.
=₩	ExtractWithPassword(String, String)	Extract the entry to the filesystem starting at the specified base directory, and using the specified password.
≐₩	ExtractWithPassword(ExtractExistingFileAction, String)	Extract the entry to a file in the filesystem, relative to the current directory using the specified behavior when extraction would overwrite an

		existing file.
⊒₩	ExtractWithPassword(String, ExtractExistingFileAction, String)	Extract the entry to the filesystem starting at the specified base directory, and using the specified behavior when extraction would overwrite an existing file.
∃	ExtractWithPassword(Stream, String)	Extracts the entry to the specified stream using the specified Password. For example, the caller could extract to Console.Out, or to a MemoryStream.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractWithPassword Method (extractExistingFile, password)

Reference ► Ionic.Zip ► ZipEntry ►

ExtractWithPassword(ExtractExistingFileAction, String)

Extract the entry to a file in the filesystem, relative to the current director using the specified behavior when extraction would overwrite an existing file.

Declaration Syntax

```
C#
                  Visual Basic
                                     Visual C++
public void ExtractWithPassword(
        ExtractExistingFileAction extractExistingFil
        string password
)
```

```
Public Sub ExtractWithPassword (
           extractExistingFile As <a href="ExtractExistingFileAc">ExtractExistingFileAc</a>
           password As <u>String</u> _
```

```
public:
void ExtractWithPassword(
        ExtractExistingFileAction extractExistingFil
        String^ password
```

Parameters

extractExistingFile (ExtractExistingFileAction)

The action to take if extraction would overwrite an existing file.

password (String)

The Password to use for decrypting the entry.

Remarks

See the remarks on the <u>LastModified</u> property, for some details about how the last modified time of the created file is set.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

F

. 4

Ionic Zip Library v1.9.1.6 ExtractWithPassword Method (stream, password)

Reference ► Ionic.Zip ► ZipEntry ► ExtractWithPassword(Stream, String)

Extracts the entry to the specified stream, using the specified Password For example, the caller could extract to Console.Out, or to a MemoryStream.

Declaration Syntax

```
C# Visual Basic Visual C++

public void ExtractWithPassword(
    Stream stream,
    string password
)
```

Parameters

stream (Stream)

the stream to which the entry should be extracted.

password (String)

The password to use for decrypting the entry.

Remarks

The caller can specify any write-able stream, for example a <u>FileStream</u>, <u>MemoryStream</u>, or ASP.NET's <u>Response</u>. <u>OutputStream</u>. The content will be decrypted and decompressed as necessary. If the entry i encrypted and no password is provided, this method will throw.

The position on the stream is not reset by this method before it extracts You may want to call stream. Seek() before calling ZipEntry. Extract().

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractWithPassword Method

(password)

Reference ► <u>Ionic.Zip</u> ► <u>ZipEntry</u> ► ExtractWithPassword(String)

Extract the entry to the filesystem, using the current working directory and the specified password.

Declaration Syntax

Parameters

password (String)

The Password to use for decrypting the entry.

■ Remarks

Existing entries in the filesystem will not be overwritten. If you would like to force the overwrite of existing files, see the ExtractExistingFileproperl or call ExtractWithPassword(ExtractExistingFileAction, String).

See the remarks on the <u>LastModified</u> property for some details about he the "last modified" time of the created file is set.

Examples

In this example, entries that use encryption are extracted using a particular password.

C# □ Cop

```
using (var zip = ZipFile.Read(FilePath))
{
    foreach (ZipEntry e in zip)
    {
        if (e.UsesEncryption)
            e.ExtractWithPassword("Secret!");
        else
            e.Extract();
    }
}
```

VB.NET

```
Using zip As ZipFile = ZipFile.Read(FilePath)
    Dim e As ZipEntry
    For Each e In zip
        If (e.UsesEncryption)
            e.ExtractWithPassword("Secret!")
        Else
            e.Extract
        End If
    Next
End Using
```

■ See Also

ExtractExistingFile

ExtractWithPassword(ExtractExistingFileAction, String)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

lonic Zip Library v1.9.1.6 ExtractWithPassword Method (baseDirectory, extractExistingFile, password)

Reference ► <u>Ionic.Zip</u> ► <u>ZipEntry</u> ► ExtractWithPassword(String, ExtractExistingFileAction, String)

Extract the entry to the filesystem, starting at the specified base director and using the specified behavior when extraction would overwrite an existing file.

Declaration Syntax

```
C#
                    Visual Basic
                                        Visual C++
public void ExtractWithPassword(
         string baseDirectory,
         ExtractExistingFileAction extractExistingFil
         string password
Public Sub ExtractWithPassword ( __
         baseDirectory As String, _
         extractExistingFile As <a href="ExtractExistingFileAc">ExtractExistingFileAc</a>
         password As <u>String</u> _
)
public:
void ExtractWithPassword(
         String^ baseDirectory,
         ExtractExistingFileAction extractExistingFil
```

Parameters

```
baseDirectory (String)
```

the pathname of the base directory

String^ password

extractExistingFile (ExtractExistingFileAction)

The action to take if extraction would overwrite an existing file.

password (String)

The Password to use for decrypting the entry.

■ Remarks

See the remarks on the <u>LastModified</u> property, for some details about how the last modified time of the created file is set.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa



Ionic Zip Library v1.9.1.6 ExtractWithPassword Method (baseDirectory, password)

Reference ► Ionic.Zip ► ZipEntry ► ExtractWithPassword(String, String)

Extract the entry to the filesystem, starting at the specified base director and using the specified password.

Declaration Syntax

Parameters

baseDirectory (String)

The pathname of the base directory.

password (String)

The Password to use for decrypting the entry.

■ Remarks

Existing entries in the filesystem will not be overwritten. If you would like to force the overwrite of existing files, see the ExtractExistingFileproperl or call ExtractWithPassword(ExtractExistingFileAction, String).

See the remarks on the <u>LastModified</u> property, for some details about how the last modified time of the created file is set.

■ See Also

<u>ExtractExistingFile</u> <u>ExtractWithPassword(String, ExtractExistingFileAction, String)</u>

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 FileName Property Reference ► Ionic.Zip ► ZipEntry ► FileName C#

The name of the file contained in the ZipEntry.

Declaration Syntax

```
C# Visual Basic Visual C++
public string FileName { get; set; }

Public Property FileName As String
    Get
    Set

public:
property String^ FileName {
    String^ get ();
    void set (String^ value);
}
```

■ Remarks

This is the name of the entry in the ZipFile itself. When creating a zi archive, if the ZipEntry has been created from a filesystem file, via a call to AddFile(String, String) or AddItem(String, String), or a related overload, the value of this property is derived from the name of that file. The FileName property does not include drive letters, and may include a different directory path, depending on the value of the directoryPathInArchive parameter used when adding the entry into the ZipFile.

In some cases there is no related filesystem file - for example when a ZipEntry is created using AddEntry(String, String) or one of the similar overloads. In this case, the value of this property is derived from the fileName and the directory path passed to that method.

When reading a zip file, this property takes the value of the entry name as stored in the zip file. If you extract such an entry, the extracted file wi take the name given by this property.

Applications can set this property when creating new zip archives or when reading existing archives. When setting this property, the actual value that is set will replace backslashes with forward slashes, in accordance with the Zip specification, for compatibility with Unix(tm) and ... get this.... Amiga!

If an application reads a ZipFile via Read(String) or a related overload, and then explicitly sets the FileName on an entry contained within the ZipFile, and then calls Save(), the application will effectively rename the entry within the zip archive.

If an application sets the value of FileName, then calls Extract() on the entry, the entry is extracted to a file using the newly set value as the filename. The FileName value is made permanent in the zip archive only *after* a call to one of the ZipFile.Save() methods on the ZipFile that contains the ZipEntry.

If an application attempts to set the FileName to a value that would result in a duplicate entry in the ZipFile, an exception is thrown.

When a ZipEntry is contained within a ZipFile, applications cannot rename the entry within the context of a foreach (For Eacl in VB) loop, because of the way the ZipFile stores entries. If you need to enumerate through all the entries and rename one or more of them, use ZipFile.EntriesSorted as the collection. See also, ZipFile.GetEnumerator().

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Included In Most Recent Save

Property

Reference ► Ionic.Zip ► ZipEntry ► IncludedInMostRecentSave

Indicates whether the entry was included in the most recent save.

Declaration Syntax

```
C# Visual Basic Visual C++
public bool IncludedInMostRecentSave { get; }

Public ReadOnly Property IncludedInMostRecentSave As Get

public:
property bool IncludedInMostRecentSave {
    bool get ();
}
```

Remarks

An entry can be excluded or skipped from a save if there is an error opening or reading the entry.

See Also

ZipErrorAction

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

4

```
Ionic Zip Library v1.9.1.6
Info Property

Reference ► Ionic.Zip ► ZipEntry ► Info
C#
```

Provides a human-readable string with information about the ZipEntry.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public string Info { get; }

Public ReadOnly Property Info As String
    Get

public:
property String^ Info {
    String^ get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 InputStream Property Reference ► Ionic.Zip ► ZipEntry ► InputStream C#

The stream that provides content for the ZipEntry.

Declaration Syntax

■ Remarks

The application can use this property to set the input stream for an entry on a just-in-time basis. Imagine a scenario where the application create a ZipFile comprised of content obtained from hundreds of files, via calls to AddFile(). The DotNetZip library opens streams on these files on a just-in-time basis, only when writing the entry out to an externatore within the scope of a ZipFile.Save() call. Only one input stream is opened at a time, as each entry is being written out.

Now imagine a different application that creates a ZipFile with content obtained from hundreds of streams, added through AddEntry(String, Stream). Normally the application would supply an ope stream to that call. But when large numbers of streams are being added this can mean many open streams at one time, unnecessarily.

To avoid this, call <u>AddEntry(String, OpenDelegate, CloseDelegate)</u> and specify delegates that open and close the stream at the time of Save.

Setting the value of this property when the entry was not added from a stream (for example, when the ZipEntry was added with AddFile(String) or AddDirectory(String), or when the entry was added by reading an existing zip archive) will throw an exception.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 InputStreamWasJitProvided

Property

Reference ► Ionic.Zip ► ZipEntry ► InputStreamWasJitProvided

A flag indicating whether the InputStream was provided Just-in-time.

Declaration Syntax

```
C# Visual Basic Visual C++
public bool InputStreamWasJitProvided { get; }

Public ReadOnly Property InputStreamWasJitProvided A Get

public:
property bool InputStreamWasJitProvided {
    bool get ();
}
```

Remarks

When creating a zip archive, an application can obtain content for one c more of the ZipEntry instances from streams, using the AddEntry(String, Stream) method. At the time of calling that method, the application can supply null as the value of the stream parameter. By doing so, the application indicates to the library that it will provide a stream for the entry on a just-in-time basis, at the time one of the ZipFile.Save() methods is called and the data for the various entries are being compressed and written out.

In this case, the application can set the <u>InputStream</u> property, typically within the SaveProgress event (event type: <u>Saving_BeforeWriteEntry</u>) for that entry.

The application will later want to call Close() and Dispose() on that stream. In the SaveProgress event, when the event type is

<u>Saving_AfterWriteEntry</u>, the application can do so. This flag indicates the stream has been provided by the application on a just-in-time basis and that it is the application's responsibility to call Close/Dispose on tha stream.

■ See Also

InputStream

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

•

Ionic Zip Library v1.9.1.6 SDirectory Property Reference ► Ionic.Zip ► ZipEntry ► IsDirectory C#

True if the entry is a directory (not a file). This is a readonly property on the entry.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public bool IsDirectory { get; }

Public ReadOnly Property IsDirectory As Boolean Get

public:
property bool IsDirectory {
    bool get ();
}
```

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
ISTEXT Property

Reference ► Ionic.Zip ► ZipEntry ► IsText
C#
```

Indicates whether an entry is marked as a text file. Be careful when usir on this property. Unless you have a good reason, you should probably ignore this property.

Declaration Syntax

■ Remarks

The ZIP format includes a provision for specifying whether an entry in the zip archive is a text or binary file. This property exposes that metadata item. Be careful when using this property: It's not clear that this property as a firm meaning, across tools and libraries.

To be clear, when reading a zip file, the property value may or may not I set, and its value may or may not be valid. Not all entries that you may think of as "text" entries will be so marked, and entries marked as "text" are not guaranteed in any way to be text entries. Whether the value is s and set correctly depends entirely on the application that produced the zip file.

There are many zip tools available, and when creating zip files, some of

them "respect" the IsText metadata field, and some of them do not. Unfortunately, even when an application tries to do "the right thing", it's not always clear what "the right thing" is.

There's no firm definition of just what it means to be "a text file", and the zip specification does not help in this regard. Twenty years ago, text wa ASCII, each byte was less than 127. IsText meant, all bytes in the file were less than 127. These days, it is not the case that all text files have all bytes less than 127. Any unicode file may have bytes that are above 0x7f. The zip specification has nothing to say on this topic. Therefore, it not clear what IsText really means.

This property merely tells a reading application what is stored in the metadata for an entry, without guaranteeing its validity or its meaning.

When DotNetZip is used to create a zipfile, it attempts to set this field "correctly." For example, if a file ends in ".txt", this field will be set. Your application may override that default setting. When writing a zip file, you must set the property before calling Save() on the ZipFile.

When reading a zip file, a more general way to decide just what kind of file is contained in a particular entry is to use the file type database store in the operating system. The operating system stores a table that says, file with .jpg extension is a JPG image file, a file with a .xml extension is an XML document, a file with a .txt is a pure ASCII text document, and son. To get this information on Windows, you need to read and parse the registry.

Examples

```
c#
using (var zip = new ZipFile())
{
   var e = zip.UpdateFile("Descriptions.mme", "");
   e.IsText = true;
   zip.Save(zipPath);
}
```

VB.NET

```
Using zip As New ZipFile
    Dim e2 as ZipEntry = zip.AddFile("Descriptions.m
    e.IsText= True
    zip.Save(zipPath)
End Using
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 LastModified Property Reference ► Ionic.Zip ► ZipEntry ► LastModified C#

The time and date at which the file indicated by the ZipEntry was lamodified.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public DateTime LastModified { get; set; }

Public Property LastModified As DateTime
    Get
    Set

public:
property DateTime LastModified {
    DateTime get ();
    void set (DateTime value);
}
```

Remarks

The DotNetZip library sets the LastModified value for an entry, equal to the Last Modified time of the file in the filesystem. If an entry is added from a stream, the library uses <code>System.DateTime.Now</code> for this value, for the given entry.

This property allows the application to retrieve and possibly set the LastModified value on an entry, to an arbitrary value. DateTime values with a DateTimeKind. Unspecified ar taken to be expressed as DateTimeKind. Local.

Be aware that because of the way <u>PKWare's Zip specification</u> describes how times are stored in the zip file, the full precision of the System.DateTime datatype is not stored for the last modified time when saving zip files. For more information on how times are formatted, see the PKZip specification.

The actual last modified time of a file can be stored in multiple ways in the zip file, and they are not mutually exclusive:

- In the so-called "DOS" format, which has a 2-second precision. Values are rounded to the nearest even second. For example, if the time on the file is 12:34:43, then it will be stored as 12:34:44. This first value is accessible via the LastModified property. This value is always present in the metadata for each zip entry. In some cases the value is invalid, or zero.
- In the so-called "Windows" or "NTFS" format, as an 8-byte integer quantity expressed as the number of 1/10 milliseconds (in other words the number of 100 nanosecond units) since January 1, 1601 (UTC). This format is how Windows represents file times. This time is accessible via the ModifiedTime property.
- In the "Unix" format, a 4-byte quantity specifying the number of seconds since January 1, 1970 UTC.
- In an older format, now deprecated but still used by some current tools. This format is also a 4-byte quantity specifying the number of seconds since January 1, 1970 UTC.

Zip tools and libraries will always at least handle (read or write) the DOS time, and may also handle the other time formats. Keep in mind that while the names refer to particular operating systems, there is nothing it the time formats themselves that prevents their use on other operating systems.

When reading ZIP files, the DotNetZip library reads the Windowsformatted time, if it is stored in the entry, and sets both
LastModified and ModifiedTime to that value. When writing
ZIP files, the DotNetZip library by default will write both time quantities.
can also emit the Unix-formatted time if desired (See
EmitTimesInUnixFormatWhenSaving.)

The last modified time of the file created upon a call to

ZipEntry.Extract() may be adjusted during extraction to compensate for differences in how the .NET Base Class Library deals with daylight saving time (DST) versus how the Windows filesystem deawith daylight saving time. Raymond Chen provides some good context.

In a nutshell: Daylight savings time rules change regularly. In 2007, for example, the inception week of DST changed. In 1977, DST was in plac all year round. In 1945, likewise. And so on. Win32 does not attempt to guess which time zone rules were in effect at the time in question. It will render a time as "standard time" and allow the app to change to DST as necessary. .NET makes a different choice.

Compare the output of FileInfo.LastWriteTime.ToString("f") with what yo see in the Windows Explorer property sheet for a file that was last writte to on the other side of the DST transition. For example, suppose the file was last modified on October 17, 2003, during DST but DST is not currently in effect. Explorer's file properties reports Thursday, October 1 2003, 8:45:38 AM, but .NETs FileInfo reports Thursday, October 17, 2003, 9:45 AM.

Win32 says, "Thursday, October 17, 2002 8:45:38 AM PST". Note: Pacific STANDARD Time. Even though October 17 of that year occurred during Pacific Daylight Time, Win32 displays the time as standard time because that's what time it is NOW.

.NET BCL assumes that the current DST rules were in place at the time in question. So, .NET says, "Well, if the rules in effect now were also in effect on October 17, 2003, then that would be daylight time" so it displays "Thursday, October 17, 2003, 9:45 AM PDT" - daylight time.

So .NET gives a value which is more intuitively correct, but is also potentially incorrect, and which is not invertible. Win32 gives a value which is intuitively incorrect, but is strictly correct.

Because of this funkiness, this library adds one hour to the LastModified time on the extracted file, if necessary. That is to say, if the time in question had occurred in what the .NET Base Class Library assumed to be DST. This assumption may be wrong given the constantly changing DST rules, but it is the best we can do.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ModifiedTime Property Reference ► Ionic.Zip ► ZipEntry ► ModifiedTime C#

Last Modified time for the file represented by the entry.

Declaration Syntax

■ Remarks

This value corresponds to the "last modified" time in the NTFS file times as described in <u>the Zip specification</u>. When getting this property, the value may be different from <u>LastModified</u>. When setting the property, the <u>LastModified</u> property also gets set, but with a lower precision.

Let me explain. It's going to take a while, so get comfortable. Originally, waaaaay back in 1989 when the ZIP specification was originally described by the esteemed Mr. Phil Katz, the dominant operating syster of the time was MS-DOS. MSDOS stored file times with a 2-second precision, because, c'mon, who is ever going to need better resolution than THAT? And so ZIP files, regardless of the platform on which the zil file was created, store file times in exactly the same format that DOS used in 1989.

Since then, the ZIP spec has evolved, but the internal format for file

timestamps remains the same. Despite the fact that the way times are stored in a zip file is rooted in DOS heritage, any program on any operating system can format a time in this way, and most zip tools and libraries DO - they round file times to the nearest even second and store it just like DOS did 25+ years ago.

PKWare extended the ZIP specification to allow a zip file to store what are called "NTFS Times" and "Unix(tm) times" for a file. These are the *last write*, *last access*, and *file creation* times of a particular file. These metadata are not actually specific to NTFS or Unix. They are tracked for each file by NTFS and by various Unix filesystems, but they are also tracked by other filesystems, too. The key point is that the times are *formatted in the zip file* in the same way that NTFS formats the time (tick since win32 epoch), or in the same way that Unix formats the time (seconds since Unix epoch). As with the DOS time, any tool or library running on any operating system is capable of formatting a time in one of these ways and embedding it into the zip file.

These extended times are higher precision quantities than the DOS time. As described above, the (DOS) LastModified has a precision of 2 seconds. The Unix time is stored with a precision of 1 second. The NTF time is stored with a precision of 0.0000001 seconds. The quantities are easily convertible, except for the loss of precision you may incur.

A zip archive can store the {C,A,M} times in NTFS format, in Unix forma or not at all. Often a tool running on Unix or Mac will embed the times in Unix format (1 second precision), while WinZip running on Windows might embed the times in NTFS format (precision of of 0.0000001 seconds). When reading a zip file with these "extended" times, in either format, DotNetZip represents the values with the ModifiedTime, AccessedTime and CreationTime properties on the ZipEntry.

While any zip application or library, regardless of the platform it runs on, could use any of the time formats allowed by the ZIP specification, not a zip tools or libraries do support all these formats. Storing the higher-precision times for each entry is optional for zip files, and many tools an libraries don't use the higher precision quantities at all. The old DOS time, represented by <u>LastModified</u>, is guaranteed to be present, though

sometimes unset.

Ok, getting back to the question about how the LastModified property relates to this ModifiedTime property... LastModified is always set, while ModifiedTime is not. (The other times stored in the NTFS times extension. CreationTime and AccessedTime also may not be set on an entry that is read from an existing zip file.) When reading a zip file, then LastModified takes the DOS time the is stored with the file. If the DOS time has been stored as zero in the zipfile, then this library will use DateTime. Now for the LastModified value. If the ZIP file was created by an evolved tool, then there will also be higher precision NTFS or Unix times in the zip file In that case, this library will read those times, and set LastModified and ModifiedTime to the same value, the one corresponding to the last write time of the file. If there are no higher precision times stored for the entry, then ModifiedTime remains unset (likewise AccessedTime and CreationTime), and LastModified keeps its DOS time.

When creating zip files with this library, by default the extended time properties (ModifiedTime, AccessedTime, and CreationTime) are set on the ZipEntry instance, and these data are stored in the zip archive for each entry, in NTFS format. If you add an entry from an actual filesystem file, then the entry gets the actual file times for that file, to NTFS-level precision. If you add an entry from a stream, or a string, then the times get the value DateTime. Now. In this case LastModified and ModifiedTime will be identical, to seconds of precision. You can explicitly set the CreationTime, AccessedTime, and ModifiedTime of an entry using the proper setters. If you want to set all of those quantities, it's more efficient to use the SetEntryTimes(DateTime, DateTime, DateTime) method. Those changes are not made permanent in the zip file until you call Save() or one of its cousins.

When creating a zip file, you can override the default behavior of this library for formatting times in the zip file, disabling the embedding of file

times in NTFS format or enabling the storage of file times in Unix formation both. You may want to do this, for example, when creating a zip file of Windows, that will be consumed on a Mac, by an application that is not hip to the "NTFS times" format. To do this, use the EmitTimesInWindowsFormatWhenSaving and EmitTimesInUnixFormatWhenSaving properties. A valid zip file may sto

EmitTimesInUnixFormatWhenSaving properties. A valid zip file may sto the file times in both formats. But, there are no guarantees that a program running on Mac or Linux will gracefully handle the NTFS-formatted times when Unix times are present, or that a non-DotNetZip-powered application running on Windows will be able to handle file time in Unix format. DotNetZip will always do something reasonable; other libraries or tools may not. When in doubt, test.

I'll bet you didn't think one person could type so much about time, eh? And reading it was so enjoyable, too! Well, in appreciation, <u>maybe you</u> should donate?

See Also

AccessedTime
CreationTime
LastModified
SetEntryTimes(DateTime, DateTime, DateTime)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 OpenReader Method

Reference ► Ionic.Zip ► ZipEntry ► OpenReader()

C#

■ Members

Icon	Member	Description
€₩	OpenReader()	Opens a readable stream corresponding to the zip entry in the archive. The stream decompresses and decrypts as necessary, as it is read.
≡ ₩	OpenReader(String)	Opens a readable stream for an encrypted zip entry in the archive. Th stream decompresses and decrypts as necessary, as it is read.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Opens a readable stream corresponding to the zip entry in the archive. The stream decompresses and decrypts as necessary, as it is read.

Declaration Syntax

```
C# Visual Basic Visual C++

public CrcCalculatorStream OpenReader()

Public Function OpenReader As CrcCalculatorStream

public:
CrcCalculatorStream^ OpenReader()
```

Return Value

The Stream for reading.

■ Remarks

DotNetZip offers a variety of ways to extract entries from a zip file. This method allows an application to extract an entry by reading a <u>Stream</u>.

The return value is of type <u>CrcCalculatorStream</u>. Use it as you would ar stream for reading. When an application calls <u>Read(Byte[], Int32, Int32)</u> on that stream, it will receive data from the zip entry that is decrypted ar decompressed as necessary.

CrcCalculatorStream adds one additional feature: it keeps a CRC32 checksum on the bytes of the stream as it is read. The CRC value is available in the Crc property on the CrcCalculatorStream. When the read is complete, your application should check this CRC against the Crc property on the ZipEntry to validate the content of the ZipEntry. You don't have to validate the entry using the CRC, but you should, to verify integrity.

Check the example for how to do this.

If the entry is protected with a password, then you need to provide a password prior to calling **OpenReader()**, either by setting the <u>Password</u> property on the entry, or the <u>Password</u> property on the <u>ZipFile</u> itself. Or, you can use <u>OpenReader(String)</u>, the overload of OpenReader that accepts a password parameter.

If you want to extract entry data into a write-able stream that is already opened, like a <u>FileStream</u>, do not use this method. Instead, use <u>Extract(Stream)</u>.

Your application may use only one stream created by OpenReader() at time, and you should not call other Extract methods before completing your reads on a stream obtained from OpenReader(). This is because there is really only one source stream for the compressed content. A ca to OpenReader() seeks in the source stream, to the beginning of the compressed content. A subsequent call to OpenReader() on a different entry will seek to a different position in the source stream, as will a call t Extract() or one of its overloads. This will corrupt the state for the decompressing stream from the original call to OpenReader().

The OpenReader () method works only when the ZipEntry is obtained from an instance of ZipFile. This method will throw an exception if the ZipEntry is obtained from a ZipInputStream.

Examples

This example shows how to open a zip archive, then read in a named entry via a stream. After the read loop is complete, the code compares the calculated during the read loop with the expected CRC on the ZipEntry, to verify the extraction.

```
Using (ZipFile zip = new ZipFile(ZipFileToRead))
{
   ZipEntry e1= zip["Elevation.mp3"];
   using (Ionic.Zlib.CrcCalculatorStream s = e1.OpenR
   {
```

```
byte[] buffer = new byte[4096];
int n, totalBytesRead= 0;
do {
    n = s.Read(buffer,0, buffer.Length);
    totalBytesRead+=n;
} while (n>0);
if (s.Crc32 != e1.Crc32)
    throw new Exception(string.Format("The Zip Ent
    if (totalBytesRead != e1.UncompressedSize)
        throw new Exception(string.Format("We read an
    }
}
```

VB.NET

```
Using zip As New ZipFile(ZipFileToRead)
    Dim e1 As ZipEntry = zip.Item("Elevation.mp3")
    Using s As Ionic.Zlib.CrcCalculatorStream = e1.0
        Dim n As Integer
        Dim buffer As Byte() = New Byte(4096) {}
        Dim totalBytesRead As Integer = 0
        Do
            n = s.Read(buffer, 0, buffer.Length)
            totalBytesRead = (totalBytesRead + n)
        Loop While (n > 0)
        If (s.Crc32 <> e1.Crc32) Then
            Throw New Exception(String.Format("The Z
        End If
        If (totalBytesRead <> e1.UncompressedSize) T
            Throw New Exception(String.Format("We re
        End If
    End Using
End Using
```

See Also

Extract(Stream)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 OpenReader Method (password)

Reference ► Ionic.Zip ► ZipEntry ► OpenReader(String)

Opens a readable stream for an encrypted zip entry in the archive. The stream decompresses and decrypts as necessary, as it is read.

Declaration Syntax

Parameters

password (String)

The password to use for decrypting the entry.

■ Return Value

The Stream for reading.

■ Remarks

See the documentation on the OpenReader() method for full details. Th overload allows the application to specify a password for the ZipEntr to be read.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Indicates whether the entry actually used ZIP64 extensions, as it was most recently written to the output file or stream.

Declaration Syntax

```
C# Visual Basic Visual C++
public Nullable<bool> OutputUsedZip64 { get; }

Public ReadOnly Property OutputUsedZip64 As Nullable Get

public:
property Nullable<bool> OutputUsedZip64 {
    Nullable<bool> get ();
}
```

■ Remarks

This Nullable property is null (Nothing in VB) until a Save() method of the containing ZipFile instance has been called. HasValue is true only after a Save() method has been called.

The value of this property for a particular ZipEntry may change over successive calls to Save() methods on the containing ZipFile, even if the file that corresponds to the ZipEntry does not. This may happen other entries contained in the ZipFile expand, causing the offset for this particular entry to exceed 0xFFFFFFF.

See Also

RequiresZip64

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

|4|

The Password to be used when encrypting a ZipEntry upon ZipFile.Save(), or when decrypting an entry upon Extract().

Declaration Syntax

Remarks

This is a write-only property on the entry. Set this to request that the ent be encrypted when writing the zip archive, or set it to specify the password to be used when extracting an existing entry that is encrypted

The password set here is implicitly used to encrypt the entry during the Save() operation, or to decrypt during the Extract() or OpenReader() operation. If you set the Password on a ZipEntry after calling Save(), there is no effect.

Consider setting the <u>Encryption</u> property when using a password. Answering concerns that the standard password protection supported b all zip tools is weak, WinZip has extended the ZIP specification with a way to use AES Encryption to protect entries in the Zip file. Unlike the

"PKZIP 2.0" encryption specified in the PKZIP specification, AES Encryption uses a standard, strong, tested, encryption algorithm. DotNetZip can create zip archives that use WinZip-compatible AES encryption, if you set the Encryption property. But, archives created that use AES encryption may not be readable by all other tools and libraries For example, Windows Explorer cannot read a "compressed folder" (a z file) that uses AES encryption, though it can read a zip file that uses "PKZIP encryption."

The ZipFile class also has a Password property. This property takes precedence over any password set on the ZipFile itself. Typically, you would use the per-entry Password when most entries in the zip archive use one password, and a few entries use a different password. If all entries in the zip file use the same password, then it is simpler to just se this property on the ZipFile itself, whether creating a zip archive or extracting a zip archive.

Some comments on updating archives: If you read a ZipFile, you cannot modify the password on any encrypted entry, except by extractir the entry with the original password (if any), removing the original entry via RemoveEntry(ZipEntry), and then adding a new entry with a new Password.

For example, suppose you read a ZipFile, and there is an encrypted entry. Setting the Password property on that ZipEntry and then callin Save() on the ZipFile does not update the password on that entry in the archive. Neither is an exception thrown. Instead, what happens during the Save() is the existing entry is copied through to the new zi archive, in its original encrypted form. Upon re-reading that archive, the entry can be decrypted with its original password.

If you read a ZipFile, and there is an un-encrypted entry, you can set the Password on the entry and then call Save() on the ZipFile, and get encryption on that entry.

Examples

This example creates a zip file with two entries, and then extracts the

entries from the zip file. When creating the zip file, the two files are adde to the zip file using password protection. Each entry uses a different password. During extraction, each file is extracted with the appropriate password.

C#

```
// create a file with encryption
using (ZipFile zip = new ZipFile())
{
    ZipEntry entry;
    entry= zip.AddFile("Declaration.txt");
    entry.Password= "123456!";
    entry = zip.AddFile("Report.xls");
    entry.Password= "1Secret!";
    zip.Save("EncryptedArchive.zip");
}
// extract entries that use encryption
using (ZipFile zip = ZipFile.Read("EncryptedArchive.
{
    ZipEntry entry;
    entry = zip["Declaration.txt"];
    entry.Password = "123456!";
    entry.Extract("extractDir");
    entry = zip["Report.xls"];
    entry.Password = "1Secret!";
    entry.Extract("extractDir");
}
```

VB.NET

```
Using zip As New ZipFile
   Dim entry as ZipEntry
   entry= zip.AddFile("Declaration.txt")
   entry.Password= "123456!"
   entry = zip.AddFile("Report.xls")
   entry.Password= "1Secret!"
   zip.Save("EncryptedArchive.zip")
```

```
' extract entries that use encryption
Using (zip as ZipFile = ZipFile.Read("EncryptedArchi
    Dim entry as ZipEntry
    entry = zip("Declaration.txt")
    entry.Password = "123456!"
    entry.Extract("extractDir")
    entry = zip("Report.xls")
    entry.Password = "1Secret!"
    entry.Extract("extractDir")
End Using
```

■ See Also

Encryption
ZipFile.Password

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 Provisional Alternate Encoding

Property

Reference ► <u>Ionic.Zip</u> ► <u>ZipEntry</u> ► **ProvisionalAlternateEncoding**

The text encoding to use for the FileName and Comment on this ZipEntry, when the default encoding is insufficient.

■ Declaration Syntax

```
C#
                  Visual Basic
                                     Visual C++
[ObsoleteAttribute("This property is obsolete since
        true)1
public Encoding ProvisionalAlternateEncoding { get;
< ObsoleteAttribute("This property is obsolete since</pre>
        True)>
Public Property Provisional Alternate Encoding As Enco
        Get
        Set
[ObsoleteAttribute(L"This property is obsolete since
        true)1
public:
property Encoding^ ProvisionalAlternateEncoding {
        Encoding^ get ();
        void set (Encoding^ value);
```

Remarks

}

Don't use this property. See <u>AlternateEncoding</u>.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

4	
•	

Indicates whether the entry requires ZIP64 extensions.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public Nullable<bool> RequiresZip64 { get; }

Public ReadOnly Property RequiresZip64 As Nullable(O Get

public:
property Nullable<bool> RequiresZip64 {
         Nullable<bool> get ();
}
```

■ Remarks

This property is null (Nothing in VB) until a Save() method on the containing ZipFile instance has been called. The property is non-null (HasValue is true) only after a Save() method has been called.

After the containing ZipFile has been saved, the Value of this property is true if any of the following three conditions holds: the uncompressed size of the entry is larger than 0xFFFFFFF; the compressed size of the entry is larger than 0xFFFFFFFF; the relative offset of the entry within the zip archive is larger than 0xFFFFFFFF. These quantities are not known until a Save() is attempted on the zip archive and the compression is applied.

If none of the three conditions holds, then the Value is false.

A Value of false does not indicate that the entry, as saved in the zip archive, does not use ZIP64. It merely indicates that ZIP64 is *not*

required. An entry may use ZIP64 even when not required if the UseZip64WhenSaving property on the containing ZipFile instance is set to Always, or if the UseZip64WhenSaving property on the containing ZipFile instance is set to AsNecessary and the output stream was n seekable.

■ See Also

OutputUsedZip64

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

_

Ionic Zip Library v1.9.1.6 SetCompression Property

Reference ► Ionic.Zip ► ZipEntry ► SetCompression C#

A callback that allows the application to specify the compression to use for a given entry that is about to be added to the zip archive.

Declaration Syntax

```
C# Visual Basic Visual C++
public SetCompressionCallback SetCompression { get;

Public Property SetCompression As SetCompressionCall Get Set

public:
property SetCompressionCallback^ SetCompression {
    SetCompressionCallback^ get ();
    void set (SetCompressionCallback^ value);
}
```

Remarks

See <u>SetCompression</u>

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

F

1

lonic Zip Library v1.9.1.6 SetEntryTimes Method (created, accessed, modified)

Reference ► <u>lonic.Zip</u> ► <u>ZipEntry</u> ► SetEntryTimes(DateTime, DateTime, DateTime)

Sets the NTFS Creation, Access, and Modified times for the given entry

■ Declaration Syntax

```
Public Sub SetEntryTimes ( _ created As DateTime, _ accessed As DateTime, _ modified As DateTime _ )
```

Parameters

```
created (<a href="DateTime">DateTime</a>)
```

the creation time of the entry.

accessed (DateTime)

the last access time of the entry.

modified (DateTime)

the last modified time of the entry.

Remarks

When adding an entry from a file or directory, the Creation, Access, and Modified times for the given entry are automatically set from the filesystem values. When adding an entry from a stream or string, the values are implicitly set to DateTime.Now. The application may wish to set these values to some arbitrary value, before saving the archive, and can do so using the various setters. If you want to set all of the times, th method is more efficient.

The values you set here will be retrievable with the <u>ModifiedTime</u>, <u>CreationTime</u> and <u>AccessedTime</u> properties.

When this method is called, if both EmitTimesInWindowsFormatWhenSaving are false, then the EmitTimesInWindowsFormatWhenSaving flag is automatica set.

DateTime values provided here without a DateTimeKind are assumed to be Local Time.

■ See Also

EmitTimesInWindowsFormatWhenSaving
EmitTimesInUnixFormatWhenSaving
AccessedTime
CreationTime
ModifiedTime

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Source Property

Reference ► Ionic.Zip ► ZipEntry ► Source

C#

An enum indicating the source of the ZipEntry.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public ZipEntrySource Source { get; }

Public ReadOnly Property Source As ZipEntrySource Get

public:
property ZipEntrySource Source {
    ZipEntrySource get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 Timestamp Property | Reference | Ionic.Zip | ZipEntry | Timestamp | C# | C#

The type of timestamp attached to the ZipEntry.

Declaration Syntax

```
C# Visual Basic Visual C++
public ZipEntryTimestamp Timestamp { get; }

Public ReadOnly Property Timestamp As ZipEntryTimest
    Get

public:
property ZipEntryTimestamp Timestamp {
    ZipEntryTimestamp get ();
}
```

Remarks

This property is valid only for a ZipEntry that was read from a zip archivit indicates the type of timestamp attached to the entry.

■ See Also

<u>EmitTimesInWindowsFormatWhenSaving</u> <u>EmitTimesInUnixFormatWhenSaving</u>

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ToString Method

Reference ► Ionic.Zip ► ZipEntry ► ToString()

C# -

Provides a string representation of the instance.

■ Declaration Syntax

C# Visual Basic Visual C++

public override string ToString()

Public Overrides Function ToString As String

public:

virtual String^ ToString() override

Return Value

a string representation of the instance.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Uncompressed Size Property

Reference ► Ionic.Zip ► ZipEntry ► UncompressedSize

The size of the file, in bytes, before compression, or after extraction.

Declaration Syntax

```
C# Visual Basic Visual C++
public long UncompressedSize { get; }

Public ReadOnly Property UncompressedSize As Long Get

public:
property long long UncompressedSize {
    long long get ();
}
```

■ Remarks

When reading a ZipFile, this value is read in from the existing zip file. When creating or updating a ZipFile, the uncompressed size is computed during compression. Therefore the value on a ZipEntry is valid after a call to Save() (or one of its overloads) in that case.

See Also

CompressedSize

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Uses Encryption Property

Reference ► Ionic.Zip ► ZipEntry ► UsesEncryption

C#

A derived property that is true if the entry uses encryption.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public bool UsesEncryption { get; }

Public ReadOnly Property UsesEncryption As Boolean Get

public:
property bool UsesEncryption {
    bool get ();
}
```

Remarks

This is a readonly property on the entry. When reading a zip file, the value for the <code>ZipEntry</code> is determined by the data read from the zip fil After saving a ZipFile, the value of this property for each <code>ZipEntry</code> indicates whether encryption was actually used (which will have been true if the <code>Password</code> was set and the <code>Encryption</code> property was somethin other than <code>None</code>.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UseUnicodeAsNecessary Property

Reference ► Ionic.Zip ► ZipEntry ► UseUnicodeAsNecessary

;#

Set to indicate whether to use UTF-8 encoding for filenames and comments.

■ Declaration Syntax

```
C# Visual Basic Visual C++

[ObsoleteAttribute("Beginning with v1.9.1.6 of DotNe public bool UseUnicodeAsNecessary { get; set; }

<ObsoleteAttribute("Beginning with v1.9.1.6 of DotNe Public Property UseUnicodeAsNecessary As Boolean Get Set

[ObsoleteAttribute(L"Beginning with v1.9.1.6 of DotNe public:
```

```
[ObsoleteAttribute(L"Beginning with v1.9.1.6 of DotN
public:
property bool UseUnicodeAsNecessary {
        bool get ();
        void set (bool value);
}
```

■ Remarks

If this flag is set, the comment and filename for the entry will be encode with UTF-8, as described in the Zip specification, if necessary. "Necessary" means, the filename or entry comment (if any) cannot be reflexively encoded and decoded using the default code page, IBM437.

Setting this flag to true is equivalent to setting ProvisionalAlternateEncoding to System. Text. Encoding. UTF8

This flag has no effect or relation to the text encoding used within the file itself.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa



The version of the zip engine needed to read the ZipEntry.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public short VersionNeeded { get; }

Public ReadOnly Property VersionNeeded As Short
    Get

public:
property short VersionNeeded {
        short get ();
}
```

Remarks

This is a readonly property, indicating the version of the Zip specification that the extracting tool or library must support to extract the given entry. Generally higher versions indicate newer features. Older zip engines obviously won't know about new features, and won't be able to extract entries that depend on those newer features.

value	Features
20	a basic Zip Entry, potentially using PKZIP encryption.
45	The ZIP64 extension is used on the entry.
46	File is compressed using BZIP2 compression*
50	File is encrypted using PkWare's DES, 3DES, (broken) RC2 or RC4
51	File is encrypted using PKWare's AES encryption or corrected RC2 encryption.

52	File is encrypted using corrected RC2-64 encryption**
61	File is encrypted using non-OAEP key wrapping***
63	File is compressed using LZMA, PPMd+, Blowfish, or Twofish

There are other values possible, not listed here. DotNetZip supports regular PKZip encryption, and ZIP64 extensions. DotNetZip cannot extract entries that require a zip engine higher than 45.

This value is set upon reading an existing zip file, or after saving a zip archive.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZipErrorAction Property Reference ► Ionic.Zip ► ZipEntry ► ZipErrorAction C#

The action to take when an error is encountered while opening or readir files as they are saved into a zip archive.

Declaration Syntax

■ Remarks

Errors can occur within a call to <u>ZipFile.Save</u>, as the various files contained in a ZipFile are being saved into the zip archive. During the Save, DotNetZip will perform a File.Open on the file associated to the ZipEntry, and then will read the entire contents of the file as it is zipped. Either the open or the Read may fail, because of lock conflicts c other reasons. Using this property, you can specify the action to take when such errors occur.

Typically you will NOT set this property on individual ZipEntry instances Instead, you will set the <u>ZipFile.ZipErrorAction</u> property on the ZipFile instance, before adding any entries to the **ZipFile**. If you do this, errors encountered on behalf of any of the entries in the ZipFile will be handled the same way.

But, if you use a <u>ZipError</u> handler, you will want to set this property on the <u>ZipEntry</u> within the handler, to communicate back to DotNetZip what you would like to do with the particular error.

■ See Also

ZipErrorAction ZipError

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 **Zip Entry Source Enumeration**

Reference ► Ionic.Zip ► ZipEntrySource

C#

An enum that specifies the source of the ZipEntry.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enum ZipEntrySource		
Public Enumeration	on ZipEntrySource	

public enum class ZipEntrySource

Members

Member	Description	
None	Default value. Invalid on a bonafide ZipEntry.	
FileSystem	The entry was instantiated by calling AddFile() or another method that added a entry from the filesystem.	
Stream	The entry was instantiated via AddEntry(String, String) or AddEntry(String, Stream).	
ZipFile	The ZipEntry was instantiated by reading a zipfile.	
WriteDelegate	The content for the ZipEntry will be or wa provided by the WriteDelegate.	
JitStream	The content for the ZipEntry will be obtained from the stream dispensed by the OpenDelegate. The entry was instantiated via AddEntry(String, OpenDelegate, CloseDelegate).	

ZipOutputStream

The content for the ZipEntry will be or wa obtained from a ZipOutputStream.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Entry Timestamp Enumeration

Reference ► Ionic.Zip ► ZipEntryTimestamp

C#

An enum that specifies the type of timestamp available on the ZipEntry.

■ Declaration Syntax

C# Visual Basic Visual C++

[FlagsAttribute]
public enum ZipEntryTimestamp

<<u>FlagsAttribute</u>> _ Public Enumeration ZipEntryTimestamp

[FlagsAttribute]
public enum class ZipEntryTimestamp

Members

Member	Description
None	Default value.
DOS	A DOS timestamp with 2-second precision.
Windows	A Windows timestamp with 100-ns precision.
Unix	A Unix timestamp with 1-second precisior
InfoZip1	A Unix timestamp with 1-second precisior stored in InfoZip v1 format. This format is outdated and is supported for reading archives only.

■ Remarks

The last modified time of a file can be stored in multiple ways in a zip file and they are not mutually exclusive:

- In the so-called "DOS" format, which has a 2-second precision. Values are rounded to the nearest even second. For example, if the time on the file is 12:34:43, then it will be stored as 12:34:44. This first value is accessible via the LastModified property. This value is always present in the metadata for each zip entry. In some cases the value is invalid, or zero.
- In the so-called "Windows" or "NTFS" format, as an 8-byte integer quantity expressed as the number of 1/10 milliseconds (in other words the number of 100 nanosecond units) since January 1, 1601 (UTC). This format is how Windows represents file times. This time is accessible via the ModifiedTime property.
- In the "Unix" format, a 4-byte quantity specifying the number of seconds since January 1, 1970 UTC.
- In an older format, now deprecated but still used by some current tools. This format is also a 4-byte quantity specifying the number of seconds since January 1, 1970 UTC.

This bit field describes which of the formats were found in a ZipEntry that was read.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **Zip Error Action Enumeration**

Reference ► Ionic.Zip ► ZipErrorAction

C#

An enum providing the options when an error occurs during opening or reading of a file or directory that is being saved to a zip file.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enum ZipEr	rorAction	
Public Enumeration	on ZipErrorAction	
public enum class	ZipErrorAction	

■ Members

Member	Description
Throw	Throw an exception when an error occurs while zipping. This is the default behavior. (For COM clients, this is a 0 (zero).)
Skip	When an error occurs during zipping, for example file cannot be opened, skip the file causing the error, and continue zipping. (For COM clients, this is a 1.)
Retry	When an error occurs during zipping, for example file cannot be opened, retry the operation that caused the error. Be careful with this option. If the error is not temporary, the library will retry forever. (For COM clients, this is a 2.)
InvokeErrorEvent	When an error occurs, invoke the zipError event. The event type used is Error_Saving . A typical use of this option: a GUI application may wish to pop u a dialog to allow the user to view the error that occurred, and choose an appropriate action. After

your processing in the error event, if you want to skip the file, set ZipErrorAction on the ZipProgressEventArgs.CurrentEntr to Skip. If you want the exception to be thrown, set ZipErrorAction on the CurrentEntry to Throw. If you want to cance the zip, set ZipProgressEventArgs.Cancel to true. Cancelling differs from using Skip in that a cancel will not save any further entries, if there are any. (For COM clients, the value of this enum is a 3.)

■ Remarks

This enum describes the actions that the library can take when an error occurs opening or reading a file, as it is being saved into a Zip archive.

In some cases an error will occur when DotNetZip tries to open a file to be added to the zip archive. In other cases, an error might occur after the file has been successfully opened, while DotNetZip is reading the file.

The first problem might occur when calling AddDirectory() on a directory that contains a Clipper .dbf file; the file is locked by Clipper and cannot lopened by another process. An example of the second problem is the ERROR_LOCK_VIOLATION that results when a file is opened by another process, but not locked, and a range lock has been taken on the file. Microsoft Outlook takes range locks on .PST files.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Error Event Args Class

Reference ► Ionic.Zip ► ZipErrorEventArgs

C# -

Provides information about the an error that occurred while zipping.

■ Declaration Syntax

C# Visual Basic Visual C++
public class ZipErrorEventArgs : ZipProgressEventArg

Public Class ZipErrorEventArgs _ Inherits ZipProgressEventArgs

public ref class ZipErrorEventArgs : public ZipProgr

■ Members

All M	All Members Methods		Properties		
PublicProtected			✓ Instance✓ Static	Declare Inherite	
lcon	Member		Descrip	tion	
	ArchiveName		Returns the archive name associate to this event. (Inherited from ZipProgressEventArgs .)		
	BytesTransferr	r <u>ed</u>	so far fo	nber of bytes read or this entry. ed from ressEventArgs.)	written
	<u>Cancer</u> III an c			ent handler, set this to e or extract operation s.	

		(Inherited from ZipProgressEventArgs.)
	CurrentEntry	The name of the last entry saved or extracted.
		(Inherited from ZipProgressEventArgs.)
	<u>EntriesTotal</u>	The total number of entries to be saved or extracted.
		(Inherited from ZipProgressEventArgs.)
≅	Equals(Object)	Determines whether the specified Object is equal to the current Object .
		(Inherited from Object.)
	<u>EventType</u>	The type of event being reported. (Inherited from ZipProgressEventArgs .)
	Exception	Returns the exception that occurred, any.
	<u>FileName</u>	Returns the name of the file that caused the exception, if any.
ÿ ©	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.
		(Inherited from Object.)
= 📦	GetHashCode()	Serves as a hash function for a particular type.

		(Inherited from Object.)
=0	GetType()	Gets the <u>Type</u> of the current instance (Inherited from <u>Object</u> .)
Ģ ₩	MemberwiseClone()	Creates a shallow copy of the current Object . (Inherited from Object .)
=0	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
	<u>TotalBytesToTransfer</u>	Total number of bytes that will be read or written for this entry. This number will be -1 if the value cannot be determined. (Inherited from ZipProgressEventArgs .)

■ Inheritance Hierarchy

Object

EventArgs

L. ZipProgressEventArgs

ZipErrorEventArgs

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Exception Property

Reference ► Ionic.Zip ► ZipErrorEventArgs ► Exception

Returns the exception that occurred, if any.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++
public Exception Exception { get; }

Public ReadOnly Property Exception As Exception
    Get

public:
property Exception^ Exception {
        Exception^ get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 FileName Property

Reference ► Ionic.Zip ► ZipErrorEventArgs ► FileName

Returns the name of the file that caused the exception, if any.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public string FileName { get; }

Public ReadOnly Property FileName As String
    Get

public:
property String^ FileName {
    String^ get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Exception Class

Reference ► Ionic.Zip ► ZipException

C#

Base class for all exceptions defined by and throw by the Zip library.

■ Declaration Syntax

C# Visual Basic Visual C++

[SerializableAttribute]

public class ZipException : Exception

<<u>SerializableAttribute</u>> _ Public Class ZipException _ Inherits <u>Exception</u>

[SerializableAttribute]

public ref class ZipException : public Exception

■ Members

All M	Members Constructors Metho		Metho	nods Properties		
		✓ Instance✓ Static			Declare Inherite	
Icon	Member			Description		
≡	ZipException()			Default ctor.		
€ 🚱	ZipException(String)		Come on, you know how exceptions work. Why are you looking at this documentation?			
=	ZipExce	ption(String, Exce	ption)		on, you know ho ons work. Why a	

oking at this documentation?
ome on, you know how ceptions work. Why are you oking at this documentation?
ets a collection of key/value hirs that provide additional her-defined information about e exception.
etermines whether the ecified Object is equal to the rrent Object. The object of the object of the object of the object.
lows an Object to attempt to ee resources and perform her cleanup operations before Object is reclaimed by arbage collection. The Object is reclaimed by arbage collection.
hen overridden in a derived ass, returns the Exception that the root cause of one or morbsequent exceptions. Therited from Exception.)
erves as a hash function for a articular type. The inherited from Object.)
h

	<u>GetObjectData(SerializationInfo, StreamingContext)</u>	When overridden in a derived class, sets the <u>SerializationInfo</u> with information about the exception. (Inherited from <u>Exception</u> .)
=0	GetType()	Gets the runtime type of the current instance. (Inherited from Exception .)
	HelpLink	Gets or sets a link to the help file associated with this exception. (Inherited from Exception.)
	HResult	Gets or sets HRESULT, a code numerical value that is assigned to a specific exception. (Inherited from Exception.)
==	InnerException	Gets the <u>Exception</u> instance the caused the current exception. (Inherited from <u>Exception</u> .)
Ģ ♥	MemberwiseClone()	Creates a shallow copy of the current <u>Object</u> . (Inherited from <u>Object</u> .)
	<u>Message</u>	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error.

		(Inherited from Exception.)
	StackTrace	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	<u>TargetSite</u>	Gets the method that throws the current exception. (Inherited from Exception .)
₫ڼ	ToString()	Creates and returns a string representation of the current exception.
		(Inherited from Exception.)

■ Inheritance Hierarchy

Object

Exception

ZipException

BadCrcException

BadPasswordException

BadReadException

BadStateException

SfxGenerationException

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Reference ► Ionic.Zip ► ZipException ► ZipException()

■ Members

Icon	Member	Description
≟∳	ZipException()	Default ctor.
€ 🖗	ZipException(String)	Come on, you know how exceptions work. Why are you looking at this documentation?
≓	ZipException(String, Exception)	Come on, you know how exceptions work. Why are you looking at this documentation?
Ģ	ZipException(SerializationInfo, StreamingContext)	Come on, you know how exceptions work. Why are you looking at this documentation?

C#

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Reference ► Ionic.Zip ► ZipException ► ZipException()

Default ctor.

■ Declaration Syntax

C# Visual Basic Visual C++

public ZipException()

Public Sub New

public:
ZipException()

C#

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

context)

Reference ► Ionic.Zip ► ZipException ►

ZipException(SerializationInfo, StreamingContext)

C#

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

Parameters

info (SerializationInfo)

The serialization info for the exception.

context (StreamingContext)

The streaming context from which to deserialize.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

(message)

Reference ► Ionic.Zip ► ZipException ► ZipException(String)

Come on, you know how exceptions work. Why are you looking at this documentation?

■ Declaration Syntax

```
C# Visual Basic Visual C++

public ZipException(
    string message
)

Public Sub New ( _
    message As String _
)
```

Parameters

message (String)

The message in the exception.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

(message, innerException)

Reference ► Ionic.Zip ► ZipException ► ZipException(String, Exception)

Come on, you know how exceptions work. Why are you looking at this documentation?

Declaration Syntax

```
C# Visual Basic Visual C++

public ZipException(
    string message,
    Exception innerException
)
```

```
Public Sub New ( _ message As String, _ innerException As Exception _ )
```

Parameters

message (String)

The message in the exception.

innerException (Exception)

The innerException for this exception.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip File Class

Reference ► Ionic.Zip ► ZipFile

C#

The ZipFile type represents a zip archive file.

■ Declaration Syntax

Members

All Mem	bers	Constructors	Methods	Properties		Fields	Events
✓ Public✓ Protected		✓ Instance✓ Static		DeclaredInherited			
Icon	Memb	per			Des	cription	
€	ZipFile(String)				Creates a new ZipFile instance, using the specified filename.		
≅©	ZipFile(String, Encoding)				Zi	ates a neworile in a graph of the spenne of	stance, cified

		and the specified Encoding.
∃ ₩	ZipFile()	Create a zip file, without specifying a target filename or stream to save to.
≅ 🍑	ZipFile(Encoding)	Create a zip file, specifying a text Encoding, but without specifying a target filename or stream to save to.
€	ZipFile(String, TextWriter)	Creates a new ZipFile instance, using the specified name for the filename and the specified status message writer
	ZipFile(String, TextWriter, Encoding)	Creates a new ZipFile instance, using the specified name for the filename the specified status message writer, and the specified Encoding
∃	AddDirectory(String)	Adds the contents of a filesystem directory to

		a Zip file archive.
₫ 🚱	AddDirectory(String, String)	Adds the contents of a filesystem directory to a Zip file archive, overriding the path to be used for entries in the archive.
₫•	AddDirectoryByName(String)	Creates a directory in the zip archive.
	AddDirectoryWillTraverseReparsePoints	Indicates whether NTFS Reparse Points like junctions, should be traversed during calls to AddDirectory()
₫₩	AddEntry(String, String)	Adds a named entry into the zip archive, taking content for the entry from a string.
₫ 心	AddEntry(String, String, Encoding)	Adds a named entry into the zip archive, taking content for the entry from a string, an using the specified texencoding.

≓ ₩	AddEntry(String, Stream)	Create an entry in the ZipFile using the given Stream as input. The entry will have the given filename.
∃	AddEntry(String, WriteDelegate)	Add a ZipEntry for which content is writte directly by the application.
≓ ₩	AddEntry(String, OpenDelegate, CloseDelegate)	Add an entry, for which the application will provide a stream, just-in-time.
±₩	AddEntry(String, Byte[])	Add an entry into the zip archive using the given filename and directory path within the archive, and the given content for the file. No file is created i the filesystem.
≅©	AddFile(String)	Adds a File to a Zip file archive.
≡	AddFile(String, String)	Adds a File to a Zip file

		archive, potentially overriding the path to be used within the zip archive.
ۥ	AddFiles(IEnumerable <string>)</string>	This method adds a se of files to the ZipFile.
=0	AddFiles(IEnumerable <string>, String)</string>	Adds a set of files to the ZipFile, using the specified directory path in the archive.
= 🗳	AddFiles(IEnumerable <string>, Boolean, String)</string>	Adds a set of files to the ZipFile, using the specified directory path in the archive, an preserving the full directory structure in the filenames.
€₩	AddItem(String)	Adds an item, either a file or a directory, to a zip file archive.
∃	AddItem(String, String)	Adds an item, either a file or a directory, to a zip file archive, explicitly specifying the directory path to be

		used in the archive.
<i>\$</i>	<u>AddProgress</u>	An event handler invoked before, during and after Adding entries to a zip archive
₫ 🚱	AddSelectedFiles(String)	Adds to the ZipFile a set of files from the current working directory on disk, that conform to the specified criteria.
₫ 🚱	AddSelectedFiles(String, Boolean)	Adds to the ZipFile a set of files from the disk that conform to th specified criteria, optionally recursing into subdirectories.
₫₩	AddSelectedFiles(String, String)	Adds to the ZipFile a set of files from a specified directory in the filesystem, that conform to the specified criteria.
≡	AddSelectedFiles(String, String, Boolean)	Adds to the ZipFile a set of files from the specified directory on disk, that conform to

		the specified criteria.
≓	AddSelectedFiles(String, String, String)	Adds to the ZipFile a selection of files from the specified directory on disk, that conform the specified criteria, and using a specified root path for entries added to the zip archive.
≓	AddSelectedFiles(String, String, String, Boolean)	Adds to the ZipFile a selection of files from the specified directory on disk, that conform the specified criteria, optionally recursing through subdirectories and using a specified root path for entries added to the zip archive.
	AlternateEncoding	A Text Encoding to use when encoding the filenames and comments for all the ZipEntry items, during a ZipFile.Save() operation.
	<u>AlternateEncodingUsage</u>	A flag that tells if and

		when this instance should apply AlternateEncoding to encode the filenames and comments associated to of ZipEntry objects contained within this instance.
	BufferSize	Size of the IO buffer used while saving.
• \$	BufferSizeDefault	Default size of the buffer used for IO.
	CaseSensitiveRetrieval	Indicates whether to perform case-sensitive matching on the filename when retrieving entries in the zipfile via the stringbased indexer.
= \S	CheckZip(String)	Checks a zip file to se if its directory is consistent.
=\$ \$	CheckZip(String, Boolean, TextWriter)	Checks a zip file to se if its directory is consistent, and

		optionally fixes the directory if necessary.
=\$ \$	CheckZipPassword(String, String)	Verify the password or a zip file.
	CodecBufferSize	Size of the work buffer to use for the ZLIB codec during compression.
	Comment	A comment attached the zip archive.
E	CompressionLevel	Sets the compression level to be used for entries subsequently added to the zip archive.
	CompressionMethod	The compression method for the zipfile.
Ξ ψ	ContainsEntry(String)	Returns true if an entry by the given name exists in the ZipFile.
	Count	Returns the number of entries in the Zip

		archive.
₹ 3	<u>DefaultEncoding</u>	The default text encoding used in zip archives. It is numeric 437, also known as IBM437.
Ξ ◊	Dispose()	Closes the read and write streams associated to the ZipFile, if necessary.
₹	Dispose(Boolean)	Disposes any managed resources, if the flag is set, then marks the instance disposed. This method is typically not called explicitly from application code.
	EmitTimesInUnixFormatWhenSaving	Specifies whether the Creation, Access, and Modified times for entries added to the zifile will be emitted in "Unix(tm) format" whe the zip archive is saved.

	EmitTimesInWindowsFormatWhenSaving	Specifies whether the Creation, Access, and Modified times for entries added to the zifile will be emitted in Windows format when the zip archive is saved.
	Encryption	The Encryption to use for entries added to th ZipFile.
	Entries	Returns the readonly collection of entries in the Zip archive.
	EntriesSorted	Returns a readonly collection of entries in the Zip archive, sorted by FileName.
	<u>EntryFileNames</u>	The list of filenames for the entries contained within the zip archive.
∄	Equals(Object)	Determines whether the specified Object is equal to the current Object. (Inherited from Object

∃ 🗳	ExtractAll(String)	Extracts all of the item in the zip archive, to the specified path in the filesystem. The path can be relative or fully-qualified.
∃ 🍑	ExtractAll(String, ExtractExistingFileAction)	Extracts all of the item in the zip archive, to the specified path in the filesystem, using the specified behavior when extraction would overwrite an existing file.
	ExtractExistingFile	The action the library should take when extracting a file that already exists.
4	ExtractProgress	An event handler invoked before, during and after extraction of entries in the zip archive.
∃	ExtractSelectedEntries(String)	Selects and Extracts a set of Entries from the ZipFile.
=		

	ExtractSelectedEntries(String, ExtractExistingFileAction)	Selects and Extracts a set of Entries from the ZipFile.
≟∳	ExtractSelectedEntries(String, String)	Selects and Extracts a set of Entries from the ZipFile.
≅♠	ExtractSelectedEntries(String, String, String)	Selects and Extracts a set of Entries from the ZipFile.
≅	ExtractSelectedEntries(String, String, String, String, ExtractExistingFileAction)	Selects and Extracts a set of Entries from the ZipFile.
<i>₹</i>	Finalize()	Allows an Object to attempt to free resources and perforn other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object
=♦5	FixZipDirectory(String)	Rewrite the directory within a zipfile.
	FlattenFoldersOnExtract	Indicates whether extracted files should keep their paths as stored in the zip

		archive.
	FullScan	Indicates whether to perform a full scan of the zip file when reading it.
≅ 🍑	GetEnumerator()	Generic IEnumerator support, for use of a ZipFile in an enumeration.
≅ 🍑	GetHashCode()	Serves as a hash function for a particula type. (Inherited from Object
≅©	GetNewEnum()	An IEnumerator, for use of a ZipFile in a foreach construct.
≡	GetType()	Gets the <u>Type</u> of the current instance. (Inherited from <u>Object</u>
	Info	Provides a human-readable string with information about the ZipFile.
€ 🔷	Initialize(String)	Initialize a ZipFile

		instance by reading in a zip file.
	InputUsesZip64	Indicates whether the most recent Read() operation read a zip fil that uses ZIP64 extensions.
₽	IsZipFile(String)	Checks the given file t see if it appears to be valid zip file.
=♦S	IsZipFile(String, Boolean)	Checks a file to see if is a valid zip file.
=♦S	IsZipFile(Stream, Boolean)	Checks a stream to see if it contains a vali zip archive.
	ltem[Int32]	This is an integer indexer into the Zip archive.
	Item[String]	This is a name-based indexer into the Zip archive.
S	LibraryVersion	Returns the version

		number on the DotNetZip assembly.
	<u>MaxOutputSegmentSize</u>	The maximum size of an output segment, when saving a split Zij file.
ġ ©	MemberwiseClone()	Creates a shallow cop of the current <u>Object</u> . (Inherited from <u>Object</u>
	Name	The name of the ZipFile, on disk.
	NumberOfSegmentsForMostRecentSave	Returns the number of segments used in the most recent Save() operation.
	OutputUsedZip64	Indicates whether the most recent Save() operation used ZIP64 extensions.
	ParallelDeflateMaxBufferPairs	The maximum numbe of buffer pairs to use when performing parallel compression.

	ParallelDeflateThreshold	The size threshold for an entry, above which a parallel deflate is used.
	Password	Sets the password to be used on the ZipFile instance.
	ProvisionalAlternateEncoding	Obsolete. The text encoding to use when writing new entries to the ZipFile, for those entries that cannot be encoded with the default (IBM437) encoding; or, the text encoding that was used when reading the entries from the ZipFile.
₽\$	Read(String)	Reads a zip file archiv and returns the instance.
=♦S	Read(String, ReadOptions)	Reads a zip file archiv from the named filesystem file using th specified options.

=\$ \$	Read(Stream)	Reads a zip archive from a stream.
□ \	Read(Stream, ReadOptions)	Reads a zip file archiv from the given stream using the specified options.
4	ReadProgress	An event handler invoked before, during and after the reading (a zip archive.
∄	RemoveEntries(ICollection <zipentry>)</zipentry>	This method removes a collection of entries from the ZipFile.
Ξ Φ	RemoveEntries(ICollection <string>)</string>	This method removes a collection of entries from the ZipFile, to name.
± ©	RemoveEntry(ZipEntry)	Removes the given ZipEntry from the zip archive.
=0	RemoveEntry(String)	Removes the ZipEntry with the

		given filename from th zip archive.
∃	RemoveSelectedEntries(String)	Remove entries from the zipfile by specified criteria.
₫••	RemoveSelectedEntries(String, String)	Remove entries from the zipfile by specified criteria, and within the specified path in the archive.
	RequiresZip64	Indicates whether the archive requires ZIP64 extensions.
∃	Save()	Saves the Zip archive to a file, specified by the Name property of the ZipFile.
=₩	Save(String)	Save the file to a new zipfile, with the given name.
≡	Save(Stream)	Save the zip archive to the specified stream.
3		

	SaveProgress	An event handler invoked when a Save(starts, before and afte each entry has been written to the archive, when a Save() completes, and during other Save events.
Ξ₩	SaveSelfExtractor(String, SelfExtractorFlavor)	Saves the ZipFile instance to a self-extracting zip archive.
€Ŵ	SaveSelfExtractor(String, SelfExtractorSaveOptions)	Saves the ZipFile instance to a self-extracting zip archive, using the specified save options.
Ξ₩	SelectEntries(String)	Retrieve entries from the zipfile by specified criteria.
Ξ Ϣ	SelectEntries(String, String)	Retrieve entries from the zipfile by specified criteria.
E	SetCompression	A callback that allows the application to specify the compression level to use for entries

		subsequently added to the zip archive.
	SortEntriesBeforeSaving	Whether to sort the ZipEntries before saving the file.
in the second	<u>StatusMessageTextWriter</u>	Gets or sets the TextWriter to which status message are delivered for the instance.
	Strategy	The compression strategy to use for all entries.
	TempFileFolder	Gets or sets the name for the folder to store the temporary file this library writes when saving a zip archive.
ĕ₩	ToString()	Provides a string representation of the instance. (Overrides Object. ToString().)
≟∳	<u>UpdateDirectory(String)</u>	Add or update a directory in a zip

		archive.
₫ 🚱	UpdateDirectory(String, String)	Add or update a directory in the zip archive at the specific root directory in the archive.
₫ 🗳	UpdateEntry(String, String)	Updates the given entry in the ZipFile using the given string as content for the ZipEntry.
∃	UpdateEntry(String, String, Encoding)	Updates the given entry in the ZipFile using the given string as content for the ZipEntry.
≡₩	UpdateEntry(String, WriteDelegate)	Updates the given entry in the ZipFile using the given delegate as the source for content for the ZipEntry.
≓	<u>UpdateEntry(String, OpenDelegate, CloseDelegate)</u>	Updates the given entry in the ZipFile using the given

		delegates to open and close the stream that provides the content for the ZipEntry.
≓	UpdateEntry(String, Stream)	Updates the given entry in the ZipFile using the given stream as input, and the giver filename and given directory Path.
∃	<pre>UpdateEntry(String, Byte[])</pre>	Updates the given entry in the ZipFile using the given byte array as content for th entry.
≡	<u>UpdateFile(String)</u>	Adds or Updates a Filin a Zip file archive.
≡	UpdateFile(String, String)	Adds or Updates a Filin a Zip file archive.
∃	<u>UpdateFiles(IEnumerable<string>)</string></u>	Adds or updates a set of files in the ZipFile.
=	<u>UpdateFiles(IEnumerable<string>,</string></u> <u>String)</u>	Adds or updates a set of files to the

		ZipFile, using the specified directory pat in the archive.
∃	<u>UpdateItem(String)</u>	Add or update a file or directory in the zip archive.
=	UpdateItem(String, String)	Add or update a file or directory.
∃	UpdateSelectedFiles(String, String, String, Boolean)	Updates the ZipFile with a selection of files from the disk that conform to the specified criteria.
	UseUnicodeAsNecessary	Obsolete. Indicates whether to encode entry filename and entry comments using Unicode (UTF-8
	<u>UseZip64WhenSaving</u>	Specify whether to use ZIP64 extensions whe saving a zip archive.
<i>§</i>	ZipError	An event that is raised when an error occurs during open or read of

	files while saving a zip archive.
ZipErrorAction	The action the library should take when an error is encountered while opening or reading files as they are saved into a zip archive.

■ Remarks

This is the main type in the DotNetZip class library. This class reads and writes zip files, as defined in the <u>specification for zip files described by PKWare</u>. The compression for this implementation is provided by a managed-code version of Zlib, included with DotNetZip in the classes in the Ionic.Zlib namespace.

This class provides a general purpose zip file capability. Use it to read, create, or update zip files. When you want to create zip files using a Stream type to write the zip file, you may want to consider the ZipOutputStream class.

Both the ZipOutputStream class and the ZipFile class can be used to create zip files. Both of them support many of the common zip features, including Unicode, different compression methods and levels, and ZIP64. They provide very similar performance when creating zip files.

The ZipFile class is generally easier to use than ZipOutputStream and should be considered a higher-level interface. For example, when creating a zip file via calls to the PutNextEntry() and Write() methods on the ZipOutputStream class, the caller is responsible for opening the

file, reading the bytes from the file, writing those bytes into the ZipOutputStream, setting the attributes on the ZipEntry, and setting the created, last modified, and last accessed timestamps on the zip entry. All of these things are done automatically by a call to ZipFile.AddFile(). For this reason, the ZipOutputStream is genera recommended for use only when your application emits arbitrary data, not necessarily data from a filesystem file, directly into a zip file, and do so using a Stream metaphor.

Aside from the differences in programming model, there are other differences in capability between the two classes.

- ZipFile can be used to read and extract zip files, in addition to creating zip files. ZipOutputStream cannot read zip files. If you want to use a stream to read zip files, check out the ZipInputStream class.
- ZipOutputStream does not support the creation of segmenter or spanned zip files.
- ZipOutputStream cannot produce a self-extracting archive.

Be aware that the <code>ZipFile</code> class implements the <code>IDisposable</code> interfac In order for <code>ZipFile</code> to produce a valid zip file, you use use it within a using clause (<code>Using</code> in VB), or call the <code>Dispose()</code> method explicitly See the examples for how to employ a using clause.

■ Inheritance Hierarchy

Object

ZipFile

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

Ionic Zip Library v1.9.1.6 **ZipFile Constructor**Reference ► Ionic.Zip ► ZipFile ► ZipFile()

C#

■ Members

Icon	Member	Description
≘	ZipFile(String)	Creates a new ZipFile instance, using the specified filename.
∃	ZipFile(String, Encoding)	Creates a new ZipFile instance, using the specified name for the filename, and the specified Encoding
≅	ZipFile()	Create a zip file, without specifying a target filename or stream to save to.
=0	ZipFile(Encoding)	Create a zip file, specifying a text Encoding, but without specifying a target filename or stream to save to.
€	ZipFile(String, TextWriter)	Creates a new ZipFile instance, using the specified name for the filename, and the specified status message writer.
⊒₩	ZipFile(String, TextWriter, Encoding)	Creates a new ZipFile instance, using the specified name for the filename, the specified status message writer, and the specified Encoding.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | ZipFile Constructor | Reference ► Ionic.Zip ► ZipFile ► ZipFile() | C#

Create a zip file, without specifying a target filename or stream to save t

Declaration Syntax

```
C# Visual Basic Visual C++

public ZipFile()

Public Sub New

public:
ZipFile()
```

Remarks

See the documentation on the <u>ZipFile constructor that accepts a single string argument</u> for basic information on all the <u>ZipFile constructors</u>.

After instantiating with this constructor and adding entries to the archive the application should call Save(String) or Save(Stream) to save to a file or a stream, respectively. The application can also set the Name proper and then call the no-argument Save() method. (This is the preferred approach for applications that use the library through COM interop.) If you call the no-argument Save() method without having set the Name c the ZipFile, either through the parameterized constructor or through the explicit property, the Save() will throw, because there is no place to save the file.

Instances of the ZipFile class are not multi-thread safe. You may have multiple threads that each use a distinct ZipFile instance, or you can synchronize multi-thread access to a single instance.

Examples

This example creates a Zip archive called Backup.zip, containing all the

files in the directory DirectoryToZip. Files within subdirectories are not zipped up.

C# □ Cop

```
using (ZipFile zip = new ZipFile())
{
    // Store all files found in the top level director
    // note: this code does not recurse subdirectories
    String[] filenames = System.IO.Directory.GetFiles(
    zip.AddFiles(filenames, "files");
    zip.Save("Backup.zip");
}
```

VB.NET

```
Using zip As New ZipFile
   ' Store all files found in the top level directo
   ' note: this code does not recurse subdirectorie
   Dim filenames As String() = System.IO.Directory.
   zip.AddFiles(filenames, "files")
   zip.Save("Backup.zip")
End Using
```

Cop

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | ZipFile Constructor (fileName) | Reference | Ionic.Zip | ZipFile | ZipFile (String) | C#

Creates a new ZipFile instance, using the specified filename.

Declaration Syntax

Parameters

fileName (String)

The filename to use for the new zip archive.

■ Remarks

Applications can use this constructor to create a new ZipFile for writing, or to slurp in an existing zip archive for read and update purposes.

To create a new zip archive, an application can call this constructor, passing the name of a file that does not exist. The name may be a fully qualified path. Then the application can add directories or files to the <code>ZipFile</code> via <code>AddDirectory()</code>, <code>AddFile()</code>, <code>AddItem()</code> and then write the zip archive to the disk by calling <code>Save()</code>. The zip file is not actually opened and written to the disk until the application calls

ZipFile.Save(). At that point the new zip file with the given name is created.

If you won't know the name of the Zipfile until the time you call ZipFile.Save(), or if you plan to save to a stream (which has no name), then you should use the no-argument constructor.

The application can also call this constructor to read an existing zip archive. passing the name of a valid zip file that does exist. But, it's bett form to use the static <code>Read(String)</code> method, passing the name of the zip file, because using <code>ZipFile.Read()</code> in your code communicates very clearly what you are doing. In either case, the file is then read into the <code>ZipFile</code> instance. The app can then enumerate the entries or can modify the zip file, for example adding entries, removing entries, changing comments, and so on.

One advantage to this parameterized constructor: it allows applications use the same code to add items to a zip archive, regardless of whether the zip file exists.

Instances of the ZipFile class are not multi-thread safe. You may no party on a single instance with multiple threads. You may have multiple threads that each use a distinct ZipFile instance, or you can synchronize multi-thread access to a single instance.

By the way, since DotNetZip is so easy to use, don't you think <u>you shou</u> donate \$5 or \$10?

Examples

This example shows how to create a zipfile, and add a few files into it.

C# □ Cop

```
String ZipFileToCreate = "archive1.zip";
String DirectoryToZip = "c:\\reports";
using (ZipFile zip = new ZipFile())
{
   // Store all files found in the top level director
```

```
String[] filenames = System.IO.Directory.GetFiles(
  zip.AddFiles(filenames, "files");
  zip.Save(ZipFileToCreate);
}
```

VB.NET

```
Dim ZipFileToCreate As String = "archive1.zip"
Dim DirectoryToZip As String = "c:\reports"
Using zip As ZipFile = New ZipFile()
    Dim filenames As String() = System.IO.Directory.
    zip.AddFiles(filenames, "files")
    zip.Save(ZipFileToCreate)
End Using
```

Exceptions

Exception	Condition
ZipException	Thrown if name refers to an existing file that is not a valid zip file.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

■ ►

Ionic Zip Library v1.9.1.6 ZipFile Constructor (fileName, statusMessageWriter)

Reference ► Ionic.Zip ► ZipFile ► ZipFile(String, TextWriter)

Creates a new ZipFile instance, using the specified name for the filename, and the specified status message writer.

Declaration Syntax

```
C# Visual Basic Visual C++

public ZipFile(
    string fileName,
    TextWriter statusMessageWriter
)
```

Parameters

fileName (<u>String</u>)

The filename to use for the new zip archive.

statusMessageWriter (<u>TextWriter</u>)

A TextWriter to use for writing verbose status messages.

Remarks

See the documentation on the **ZipFile** constructor that accepts a single

string argument for basic information on all the ZipFile constructors.

This version of the constructor allows the caller to pass in a TextWriter, which verbose messages will be written during extraction or creation of the zip archive. A console application may wish to pass System. Console. Out to get messages on the Console. A graphical or headless application may wish to capture the messages in a different TextWriter, for example, a StringWriter, and then display the messages in a TextBox, or generate an audit log of ZipFile operations.

To encrypt the data for the files added to the ZipFile instance, set the Password property after creating the ZipFile instance.

Instances of the ZipFile class are not multi-thread safe. You may no party on a single instance with multiple threads. You may have multiple threads that each use a distinct ZipFile instance, or you can synchronize multi-thread access to a single instance.

Examples

```
using (ZipFile zip = new ZipFile("Backup.zip", Conso
{
    // Store all files found in the top level director
    // note: this code does not recurse subdirectories
    // Status messages will be written to Console.Out
    String[] filenames = System.IO.Directory.GetFiles(
    zip.AddFiles(filenames);
    zip.Save();
}
```

VB.NET
□ Cop

```
Using zip As New ZipFile("Backup.zip", Console.Out)
  ' Store all files found in the top level directo
  ' note: this code does not recurse subdirectorie
  ' Status messages will be written to Console.Out
  Dim filenames As String() = System.IO.Directory.
```

```
zip.AddFiles(filenames)
zip.Save()
End Using
```

Exceptions

Exception	Condition
<u>ZipException</u>	Thrown if name refers to an existing file that is not a valid zip file.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

F

Ionic Zip Library v1.9.1.6 ZipFile Constructor (fileName, statusMessageWriter, encoding)

Reference ► <u>lonic.Zip</u> ► <u>ZipFile</u> ► **ZipFile(String, TextWriter, Encoding)**

Creates a new ZipFile instance, using the specified name for the filename, the specified status message writer, and the specified Encoding.

Declaration Syntax

```
C# Visual Basic Visual C++

public ZipFile(
    string fileName,
    TextWriter statusMessageWriter,
    Encoding encoding
)
```

Parameters

fileName (String)

The filename to use for the new zip archive.

statusMessageWriter (TextWriter)

A TextWriter to use for writing verbose status messages.

encoding (Encoding)

The Encoding is used as the default alternate encoding for entries with filenames or comments that cannot be encoded with the IBM437 code page.

■ Remarks

This constructor works like the <u>ZipFile constructor that accepts a single string argument</u>. See that reference for detail on what this constructor does.

This version of the constructor allows the caller to pass in a TextWriter, and an Encoding. The TextWriter will collect verbose messages that are generated by the library during extraction or creation of the zip archive. A console application may wish to pass System.Console.Out to get messages on the Console. A graphical or headless application may wish to capture the messages in different TextWriter, for example, a StringWriter, and then display the messages in a TextBox, or generate an audit log of ZipFile operations.

The Encoding is used as the default alternate encoding for entries with filenames or comments that cannot be encoded with the IBM437 code page. This is a equivalent to setting the ProvisionalAlternateEncoding property on the ZipFile instance after construction.

To encrypt the data for the files added to the ZipFile instance, set the Password property after creating the ZipFile instance.

Instances of the ZipFile class are not multi-thread safe. You may no party on a single instance with multiple threads. You may have multiple threads that each use a distinct ZipFile instance, or you can synchronize multi-thread access to a single instance.

Exceptions

Exception	Condition
ZipException	Thrown if fileName refers to an existin file that is not a valid zip file.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZipFile Constructor (fileName, encoding)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► **ZipFile**(String, Encoding)

Creates a new ZipFile instance, using the specified hame for the filename, and the specified Encoding.

Declaration Syntax

```
C# Visual Basic Visual C++

public ZipFile(
    string fileName,
    Encoding encoding
)
```

Parameters

fileName (String)

The filename to use for the new zip archive.

encoding (Encoding)

The Encoding is used as the default alternate encoding for entries with filenames or comments that cannot be encoded with the IBM437 code page.

Remarks

See the documentation on the <u>ZipFile constructor that accepts a single</u> <u>string argument</u> for basic information on all the **ZipFile** constructors.

The Encoding is used as the default alternate encoding for entries with filenames or comments that cannot be encoded with the IBM437 code page. This is equivalent to setting the ProvisionalAlternateEncoding property on the ZipFile instance after construction.

Instances of the ZipFile class are not multi-thread safe. You may no party on a single instance with multiple threads. You may have multiple threads that each use a distinct ZipFile instance, or you can synchronize multi-thread access to a single instance.

Exceptions

Exception	Condition
ZipException	Thrown if name refers to an existing file that is not a valid zip file.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip File Constructor (encoding)

Reference ► Ionic.Zip ► ZipFile ► ZipFile(Encoding)

Create a zip file, specifying a text Encoding, but without specifying a target filename or stream to save to.

Declaration Syntax

Parameters

encoding (Encoding)

The Encoding is used as the default alternate encoding for entries with filenames or comments that cannot be encoded with the IBM437 code page.

Remarks

See the documentation on the <u>ZipFile constructor that accepts a single string argument</u> for basic information on all the <u>ZipFile constructors</u>.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 Add Directory Method | Reference ► Ionic.Zip ► ZipFile ► Add Directory() | C#

This method has 2 overloads.

■ Members

lcon	Member	Description
∃	AddDirectory(String)	Adds the contents of a filesystem directory to a Zip file archive.
∃ ₩	AddDirectory(String, String)	Adds the contents of a filesystem directory to a Zip file archive, overriding the path to be used for entries in the archive.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 AddDirectory Method (directoryName)

Reference ► Ionic.Zip ► ZipFile ► AddDirectory(String)

Adds the contents of a filesystem directory to a Zip file archive.

■ Declaration Syntax

Parameters

directoryName (String)

The name of the directory to add.

■ Return Value

The ZipEntry added.

Remarks

The name of the directory may be a relative path or a fully-qualified path Any files within the named directory are added to the archive. Any subdirectories within the named directory are also added to the archive, recursively.

Top-level entries in the named directory will appear as top-level entries the zip archive. Entries in subdirectories in the named directory will result in entries in subdirectories in the zip archive.

If you want the entries to appear in a containing directory in the zip archive itself, then you should call the AddDirectory() overload that allow you to explicitly specify a directory path for use in the archive.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to each ZipEntry added.

■ See Also

AddItem(String)
AddFile(String)
UpdateDirectory(String)
AddDirectory(String, String)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

lonic Zip Library v1.9.1.6 Add Directory Method (directory Name, directory PathInArchive)

Reference ► Ionic.Zip ► ZipFile ► AddDirectory(String, String)

Adds the contents of a filesystem directory to a Zip file archive, overriding the path to be used for entries in the archive.

■ Declaration Syntax

Parameters

directoryName (String)

The name of the directory to add.

directoryPathInArchive (String)

Specifies a directory path to use to override any path in the DirectoryName. This path may, or may not, correspond to a real directory in the current filesystem. If the zip is later extracted, this is the path used for the extracted file or directory. Passing null (Nothing in VB) or the empty string ("") will insert the items at the

root path within the archive.

Return Value

The ZipEntry added.

■ Remarks

The name of the directory may be a relative path or a fully-qualified path. The add operation is recursive, so that any files or subdirectories within the name directory are also added to the archive.

Top-level entries in the named directory will appear as top-level entries the zip archive. Entries in subdirectories in the named directory will result in entries in subdirectories in the zip archive.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to each ZipEntry added.

Examples

In this code, calling the ZipUp() method with a value of "c:\reports" for the directory parameter will result in a zip file structure in which all entries a contained in a toplevel "reports" directory.

```
public void ZipUp(string targetZip, string directory
{
   using (var zip = new ZipFile())
   {
     zip.AddDirectory(directory, System.IO.Path.GetFizip.Save(targetZip);
   }
}
```

■ See Also

AddItem(String, String)
AddFile(String, String)
UpdateDirectory(String, String)

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 AddDirectoryByName Method (directoryNameInArchive)

Reference ► Ionic.Zip ► ZipFile ► AddDirectoryByName(String)

Creates a directory in the zip archive.

C#

Declaration Syntax

Parameters

directoryNameInArchive (String)

The name of the directory to create in the archive.

■ Return Value

The ZipEntry added.

Remarks

Use this when you want to create a directory in the archive but there is a corresponding filesystem representation for that directory.

You will probably not need to do this in your code. One of the only times

you will want to do this is if you want an empty directory in the zip archive. The reason: if you add a file to a zip archive that is stored within a multi-level directory, all of the directory tree is implicitly created in the zip archive.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library

v1.9.1.6AddDirectoryWillTraverseReparsePoints Property

Reference ► Ionic.Zip ► ZipFile ►

AddDirectoryWillTraverseReparsePoints

Indicates whether NTFS Reparse Points, like junctions, should be traversed during calls to AddDirectory().

Declaration Syntax

```
C# Visual Basic Visual C++
public bool AddDirectoryWillTraverseReparsePoints {
Public Property AddDirectoryWillTraverseReparsePoint
    Get
    Set

public:
property bool AddDirectoryWillTraverseReparsePoints
    bool get ();
    void set (bool value);
}
```

Remarks

By default, calls to AddDirectory() will traverse NTFS reparse points, like mounted volumes, and directory junctions. An example of a junction is the "My Music" directory in Windows Vista. In some cases you may not want DotNetZip to traverse those directories. In that case, set this property to false.

Examples

C# □ Cop

```
using (var zip = new ZipFile())
```

```
{
    zip.AddDirectoryWillTraverseReparsePoints = fals
    zip.AddDirectory(dirToZip, "fodder");
    zip.Save(zipFileToCreate);
}
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 AddEntry Method Reference ► Ionic.Zip ► ZipFile ► AddEntry()

C#

■ Members

Icon	Member	Description
Ξ₩	AddEntry(String, String)	Adds a named entry into the zip archive, taking content for the entry from a string.
€ 🖟	AddEntry(String, String, Encoding)	Adds a named entry into the zip archive, taking content for the entry from a string, and using the specified text encoding.
Ξ Φ	AddEntry(String, Stream)	Create an entry in the ZipFile using the given Stream as input. The entry will have the given filename
∃	AddEntry(String, WriteDelegate)	Add a ZipEntry for which content is written directly by the application.
∃∳	AddEntry(String, OpenDelegate, CloseDelegate)	Add an entry, for which the application will provide a stream, just-in-time.
Ξ₩	AddEntry(String, Byte[])	Add an entry into the zip archive usin the given filename and directory path within the archive, and the given content for the file. No file is created i the filesystem.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 AddEntry Method (entryName, opener, closer)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► AddEntry(String, OpenDelegate, CloseDelegate)

Add an entry, for which the application will provide a stream, just-in-time

Declaration Syntax

Parameters

```
entryName (String)
    the name of the entry to add

opener (OpenDelegate)
    the delegate that will be invoked to open the stream
closer (CloseDelegate)
```

the delegate that will be invoked to close the stream

Return Value

the ZipEntry added

Remarks

In cases where the application wishes to open the stream that holds the content for the ZipEntry, on a just-in-time basis, the application can use this method and provide delegates to open and close the stream.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

Examples

This example uses anonymous methods in C# to open and close the source stream for the content for a zip entry. In a real application, the logic for the OpenDelegate would probably be more involved.

Examples

This example uses delegates in VB.NET to open and close the the source stream for the content for a zip entry. VB 9.0 lacks support for "Sub" lambda expressions, and so the CloseDelegate must be an actua named Sub.

VB.NET

```
Function MyStreamOpener(ByVal entryName As String) A
    '' This simply opens a file. You probably want
    '' more involved here: open a stream to read fro
       open a stream on an HTTP connection, and so o
    Return File.OpenRead(entryName)
End Function
Sub MyStreamCloser(entryName As String, stream As St
    stream.Close()
End Sub
Public Sub Run()
    Dim dirToZip As String = "fodder"
    Dim zipFileToCreate As String = "Archive.zip"
    Dim opener As OpenDelegate = AddressOf MyStreamO
    Dim closer As CloseDelegate = AddressOf MyStream
    Dim numFilestoAdd As Int32 = 4
    Using zip As ZipFile = New ZipFile
        Dim i As Integer
        For i = 0 To numFilesToAdd - 1
            zip.AddEntry(String.Format("content-{0:0
        Next i
        zip.Save(zipFileToCreate)
    End Using
End Sub
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 AddEntry Method (entryName, writer)

Reference ► Ionic.Zip ► ZipFile ► AddEntry(String, WriteDelegate)

Add a ZipEntry for which content is written directly by the application.

Declaration Syntax

```
C# Visual Basic Visual C++

public ZipEntry AddEntry(
    string entryName,
    WriteDelegate writer
)
```

Parameters

entryName (String)

the name of the entry to add

writer (WriteDelegate)

the delegate which will write the entry content

Return Value

the ZipEntry added

Remarks

When the application needs to write the zip entry data, use this method add the ZipEntry. For example, in the case that the application wishes to write the XML representation of a DataSet into a ZipEntry, the application can use this method to do so.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

About progress events: When using the WriteDelegate, DotNetZip does not issue any SaveProgress events with EventType = Saving_EntryBytesRead. (This is because it is the application's code the runs in WriteDelegate - there's no way for DotNetZip to know when to issue a EntryBytesRead event.) Applications that want to update a progress bar or similar status indicator should do so from within the WriteDelegate itself. DotNetZip will issue the other SaveProgress event including Saving_Started, Saving_BeforeWriteEntry, and Saving_AfterWriteEntry.

Note: When you use PKZip encryption, it's normally necessary to compute the CRC of the content to be encrypted, before compressing o encrypting it. Therefore, when using PKZip encryption with a WriteDelegate, the WriteDelegate CAN BE called twice: once to compute the CRC, and the second time to potentially compress and encrypt. Surprising, but true. This is because PKWARE specified that the encryption initialization data depends on the CRC. If this happens, for each call of the delegate, your application must stream the same entry data in its entirety. If your application writes different data during the second call, it will result in a corrupt zip file.

The double-read behavior happens with all types of entries, not only those that use WriteDelegate. It happens if you add an entry from a filesystem file, or using a string, or a stream, or an opener/closer pair. B in those cases, DotNetZip takes care of reading twice; in the case of the WriteDelegate, the application code gets invoked twice. Be aware.

As you can imagine, this can cause performance problems for large streams, and it can lead to correctness problems when you use a WriteDelegate. This is a pretty big pitfall. There are two ways to avoid it. First, and most preferred: don't use PKZIP encryption. If you us the WinZip AES encryption, this problem doesn't occur, because the encryption protocol doesn't require the CRC up front. Second: if you do choose to use PKZIP encryption, write out to a non-seekable stream (lik standard output, or the Response. OutputStream in an ASP.NET application). In this case, DotNetZip will use an alternative encryption protocol that does not rely on the CRC of the content. This also implies setting bit 3 in the zip entry, which still presents problems for some zip tools.

In the future I may modify DotNetZip to *always* use bit 3 when PKZIP encryption is in use. This seems like a win overall, but there will be som work involved. If you feel strongly about it, visit the DotNetZip forums ar vote up the Workitem tracking this issue.

Examples

C#

This example shows an application filling a DataSet, then saving the contents of that DataSet as XML, into a ZipEntry in a ZipFile, using an anonymous delegate in C#. The DataSet XML is never saved to a disk file.

Cop var c1= new System.Data.SqlClient.SqlConnection(conn var da = new System.Data.SqlClient.SqlDataAdapter() { SelectCommand= new System.Data.SqlClient.Sq **}**; DataSet ds1 = new DataSet(); da.Fill(ds1, "Invoices"); using(Ionic.Zip.ZipFile zip = new Ionic.Zip.ZipFile({ zip.AddEntry(zipEntryName, (name, stream) => ds1. zip.Save(zipFileName); }

Examples

This example uses an anonymous method in C# as the WriteDelegate t provide the data for the ZipEntry. The example is a bit contrived - the AddFile() method is a simpler way to insert the contents of a file int an entry in a zip file. On the other hand, if there is some sort of processing or transformation of the file contents required before writing, the application could use the WriteDelegate to do it, in this way.

```
C#
                                                 Cop
using (var input = File.Open(filename, FileMode.Open
    using(Ionic.Zip.ZipFile zip = new Ionic.Zip.ZipF
        zip.AddEntry(zipEntryName, (name,output) =>
                 byte[] buffer = new byte[BufferSize]
                 int n;
                 while ((n = input.Read(buffer, 0, bu
                 {
                     // could transform the data here
                     output.Write(buffer, 0, n);
                     // could update a progress bar h
                 }
            });
        zip.Save(zipFileName);
    }
```

Examples

}

This example uses a named delegate in VB to write data for the given ZipEntry (VB9 does not have anonymous delegates). The example here is a bit contrived - a simpler way to add the contents of a file to a ZipEnt is to simply use the appropriate AddFile() method. The key scenari for which the WriteDelegate makes sense is saving a DataSet, in XML format, to the zip file. The DataSet can write XML to a stream, and the WriteDelegate is the perfect place to write into the zip file. There ma

be other data structures that can write to a stream, but cannot be read ϵ a stream. The WriteDelegate would be appropriate for those case as well.

VB.NET

```
Private Sub WriteEntry (ByVal name As String, ByVal
    Using input As FileStream = File.Open(filename,
        Dim n As Integer = -1
        Dim buffer As Byte() = New Byte(BufferSize){
        Do While n <> 0
            n = input.Read(buffer, 0, buffer.Length)
            output.Write(buffer, 0, n)
        Loop
    End Using
End Sub
Public Sub Run()
    Using zip = New ZipFile
        zip.AddEntry(zipEntryName, New WriteDelegate
        zip.Save(zipFileName)
    End Using
End Sub
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

Ionic Zip Library v1.9.1.6 AddEntry Method (entryName,

byteContent)

Reference ► Ionic.Zip ► ZipFile ► AddEntry(String, Byte[])

C#

Add an entry into the zip archive using the given filename and directory path within the archive, and the given content for the file. No file is created in the filesystem.

Declaration Syntax

```
C# Visual Basic Visual C++

public ZipEntry AddEntry(
    string entryName,
    byte[] byteContent
)
```

Parameters

entryName (String)

The name, including any path, to use within the archive for the entr

byteContent (Byte[])

The data to use for the entry.

■ Return Value

The ZipEntry added.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 AddEntry Method (entryName, stream)

Reference ► Ionic.Zip ► ZipFile ► AddEntry(String, Stream)

Create an entry in the ZipFile using the given Stream as input. The entry will have the given filename.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public ZipEntry AddEntry(
    string entryName,
    Stream stream
)
```

Parameters

entryName (String)

The name, including any path, which is shown in the zip file for the added entry.

stream (Stream)

The input stream from which to grab content for the file

■ Return Value

The ZipEntry added.

Remarks

The application should provide an open, readable stream; in this case it will be read during the call to <u>Save()</u> or one of its overloads.

The passed stream will be read from its current position. If necessary, callers should set the position in the stream before calling AddEntry(). This might be appropriate when using this method with a MemoryStrear for example.

In cases where a large number of streams will be added to the ZipFile, the application may wish to avoid maintaining all of the streams open simultaneously. To handle this situation, the application should use the AddEntry(String, OpenDelegate, CloseDelegate) overload.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

Examples

This example adds a single entry to a ZipFile via a Stream.

C# Cop

```
String zipToCreate = "Content.zip";
String fileNameInArchive = "Content-From-Stream.bin"
using (System.IO.Stream streamToRead = MyStreamOpene
{
   using (ZipFile zip = new ZipFile())
   {
     ZipEntry entry= zip.AddEntry(fileNameInArchive,
     zip.AddFile("Readme.txt");
     zip.Save(zipToCreate); // the stream is read im
   }
}
```

VB.NET

```
Dim zipToCreate As String = "Content.zip"
Dim fileNameInArchive As String = "Content-From-Stre
Using streamToRead as System.IO.Stream = MyStreamOpe
   Using zip As ZipFile = New ZipFile()
        Dim entry as ZipEntry = zip.AddEntry(fileNameInA zip.AddFile("Readme.txt")
        zip.Save(zipToCreate) '' the stream is read imp
        End Using
End Using
```

See Also

UpdateEntry(String, Stream)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

lonic Zip Library v1.9.1.6 AddEntry Method (entryName, content)

Reference ► Ionic.Zip ► ZipFile ► AddEntry(String, String)

Adds a named entry into the zip archive, taking content for the entry from a string.

Declaration Syntax

```
C# Visual Basic Visual C++

public ZipEntry AddEntry(
    string entryName,
    string content
)
```

Parameters

entryName (String)

The name, including any path, to use for the entry within the archive content (String)

The content of the file, should it be extracted from the zip.

■ Return Value

The ZipEntry added.

Remarks

Calling this method creates an entry using the given fileName and directory path within the archive. There is no need for a file by the given name to exist in the filesystem; the name is used within the zip archive only. The content for the entry is encoded using the default text encodin for the machine, or on Silverlight, using UTF-8.

Examples

This example shows how to add an entry to the zipfile, using a string as content for that entry.

C# □ Cop

```
string Content = "This string will be the content of
using (ZipFile zip1 = new ZipFile())
{
   zip1.AddFile("MyDocuments\\Resume.doc", "files");
   zip1.AddEntry("Readme.txt", Content);
   zip1.Comment = "This zip file was created at " + S
   zip1.Save("Content.zip");
}
```

VB.NET

```
Public Sub Run()
  Dim Content As String = "This string will be the c
  Using zip1 As ZipFile = New ZipFile
    zip1.AddEntry("Readme.txt", Content)
    zip1.AddFile("MyDocuments\Resume.doc", "files")
    zip1.Comment = ("This zip file was created at "
    zip1.Save("Content.zip")
  End Using
End Sub
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

)

Ionic Zip Library v1.9.1.6 AddEntry Method (entryName, content, encoding)

Reference ► Ionic.Zip ► ZipFile ► AddEntry(String, String, **Encoding**)

C#

Adds a named entry into the zip archive, taking content for the entry from a string, and using the specified text encoding.

Declaration Syntax

```
Visual C++
C#
                   Visual Basic
public ZipEntry AddEntry(
        string entryName,
        string content,
        Encoding encoding
)
```

```
Public Function AddEntry ( _
          entryName As String, _
         content As String, _
         encoding As <a href="Encoding">Encoding</a> _
) As ZipEntry
```

```
public:
ZipEntry^ AddEntry(
        String^ entryName,
        String^ content,
        Encoding ^ encoding
```

Parameters

entryName (String)

The name, including any path, to use within the archive for the entr

content (String)

The content of the file, should it be extracted from the zip.

encoding (Encoding)

The text encoding to use when encoding the string. Be aware: This is distinct from the text encoding used to encode the fileName, as specified in ProvisionalAlternateEncoding.

Return Value

The ZipEntry added.

Remarks

Calling this method creates an entry using the given fileName and directory path within the archive. There is no need for a file by the given name to exist in the filesystem; the name is used within the zip archive only.

The content for the entry, a string value, is encoded using the given text encoding. A BOM (byte-order-mark) is emitted into the file, if the Encoding parameter is set for that.

Most Encoding classes support a constructor that accepts a boolean, indicating whether to emit a BOM or not. For example see UTF8Encoding(Boolean).

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

| Ionic Zip Library v1.9.1.6 AddFile Method | Reference | Ionic.Zip | ZipFile | AddFile() | C#

This method has two overloads.

■ Members

Icon	Member	Description
=	AddFile(String)	Adds a File to a Zip file archive.
≓	AddFile(String, String)	Adds a File to a Zip file archive, potentially overriding the path to be used within the zip archive.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

| Ionic Zip Library v1.9.1.6 AddFile Method (fileName) | Reference ► Ionic.Zip ► ZipFile ► AddFile(String) | C#

Adds a File to a Zip file archive.

Declaration Syntax

Parameters

fileName (String)

The name of the file to add. It should refer to a file in the filesystem The name of the file may be a relative path or a fully-qualified path.

■ Return Value

The ZipEntry corresponding to the File added.

■ Remarks

This call collects metadata for the named file in the filesystem, including the file attributes and the timestamp, and inserts that metadata into the resulting ZipEntry. Only when the application calls Save() on the ZipFile, does DotNetZip read the file from the filesystem and then write the content to the zip file archive.

This method will throw an exception if an entry with the same name already exists in the ZipFile.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

Examples

In this example, three files are added to a Zip archive. The ReadMe.txt file will be placed in the root of the archive. The .png file will be placed in a folder within the zip called photos\personal. The pdf file will be include into a folder within the zip called Desktop.

```
try
{
   using (ZipFile zip = new ZipFile())
   {
      zip.AddFile("c:\\photos\\personal\\7440-N49th.pn
      zip.AddFile("c:\\Desktop\\2008-Regional-Sales-Re
      zip.AddFile("ReadMe.txt");

      zip.Save("Package.zip");
   }
} catch (System.Exception ex1)
{
   System.Console.Error.WriteLine("exception: " + ex1
}
```

VB.NET

```
Try
Using zip As ZipFile = New ZipFile
    zip.AddFile("c:\photos\personal\7440-N49th.
    zip.AddFile("c:\Desktop\2008-Regional-Sales
```

```
zip.AddFile("ReadMe.txt")
    zip.Save("Package.zip")
    End Using
Catch ex1 As Exception
    Console.Error.WriteLine("exception: {0}", ex1.T
End Try
```

■ See Also

AddItem(String)
AddDirectory(String)
UpdateFile(String)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 AddFile Method (fileName, directoryPathInArchive)

Reference ► Ionic.Zip ► ZipFile ► AddFile(String, String)

Adds a File to a Zip file archive, potentially overriding the path to be use within the zip archive.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public ZipEntry AddFile(
    string fileName,
    string directoryPathInArchive
)
```

Parameters

fileName (String)

The name of the file to add. The name of the file may be a relative path or a fully-qualified path.

directoryPathInArchive (String)

Specifies a directory path to use to override any path in the fileName. This path may, or may not, correspond to a real directory in the current filesystem. If the files within the zip are later extracted this is the path used for the extracted file. Passing null

(Nothing in VB) will use the path on the fileName, if any. Passing the empty string ("") will insert the item at the root path within the archive.

■ Return Value

The ZipEntry corresponding to the file added.

Remarks

The file added by this call to the ZipFile is not written to the zip file archive until the application calls Save() on the ZipFile.

This method will throw an exception if an entry with the same name already exists in the ZipFile.

This version of the method allows the caller to explicitly specify the directory path to be used in the archive.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

Examples

In this example, three files are added to a Zip archive. The ReadMe.txt file will be placed in the root of the archive. The .png file will be placed in a folder within the zip called images. The pdf file will be included into a folder within the zip called files\docs, and will be encrypted with the give password.

```
try
{
   using (ZipFile zip = new ZipFile())
   {
     // the following entry will be inserted at the r
```

```
zip.AddFile("c:\\datafiles\\ReadMe.txt", "");
  // this image file will be inserted into the "im
  zip.AddFile("c:\\photos\\personal\\7440-N49th.pn
  // the following will result in a password-prote
  // files\\docs\\2008-Regional-Sales-Report.pdf
  zip.Password = "EncryptMe!";
  zip.AddFile("c:\\Desktop\\2008-Regional-Sales-Re
  zip.Save("Archive.zip");
  }
}
catch (System.Exception ex1)
{
  System.Console.Error.WriteLine("exception: {0}", e
}
```

VB.NET

```
Using zip As ZipFile = New ZipFile
    ' the following entry will be inserted at th
    zip.AddFile("c:\datafiles\ReadMe.txt", "")
    ' this image file will be inserted into the
    zip.AddFile("c:\photos\personal\7440-N49th.p
    ' the following will result in a password-pr
    ' files\\docs\\2008-Regional-Sales-Report.pd
    zip.Password = "EncryptMe!"
    zip.AddFile("c:\Desktop\2008-Regional-Sales-
    zip.Save("Archive.zip")
    End Using
Catch ex1 As Exception
    Console.Error.WriteLine("exception: {0}", ex1)
End Try
```

See Also

AddItem(String, String)
AddDirectory(String, String)
UpdateFile(String, String)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 AddFiles Method

Reference ► Ionic.Zip ► ZipFile ► AddFiles()

C#

■ Members

lcon	Member	Description
∄ 🍑	AddFiles(IEnumerable <string>)</string>	This method adds a set of files t the ZipFile.
€	AddFiles(IEnumerable <string>, String)</string>	Adds a set of files to the ZipFile, using the specified directory path in the archive.
₫ 🍑	AddFiles(IEnumerable <string>, Boolean, String)</string>	Adds a set of files to the ZipFile, using the specified directory path in the archive, and preserving the full directory structure in the filenames.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 AddFiles Method (fileNames)

Reference ► Ionic.Zip ► ZipFile ► AddFiles(IEnumerable<String>)

C#

This method adds a set of files to the ZipFile.

■ Declaration Syntax

Parameters

fileNames (<u>IEnumerable</u><<u>String</u>>)

The collection of names of the files to add. Each string should refer to a file in the filesystem. The name of the file may be a relative pat or a fully-qualified path.

Remarks

Use this method to add a set of files to the zip archive, in one call. For example, a list of files received from

System. IO. Directory. GetFiles() can be added to a zip archive in one call.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>,

<u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to each ZipEntry added.

Examples

This example shows how to create a zip file, and add a few files into it.

C# □ Cop

```
String ZipFileToCreate = "archive1.zip";
String DirectoryToZip = "c:\\reports";
using (ZipFile zip = new ZipFile())
{
    // Store all files found in the top level director
    String[] filenames = System.IO.Directory.GetFiles(
    zip.AddFiles(filenames);
    zip.Save(ZipFileToCreate);
}
```

VB.NET

```
Dim ZipFileToCreate As String = "archive1.zip"
Dim DirectoryToZip As String = "c:\reports"
Using zip As ZipFile = New ZipFile
    ' Store all files found in the top level directo
    Dim filenames As String() = System.IO.Directory.
    zip.AddFiles(filenames)
    zip.Save(ZipFileToCreate)
End Using
```

See Also

AddSelectedFiles(String, String)

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 AddFiles Method (fileNames, preserveDirHierarchy, directoryPathInArchive)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► AddFiles(IEnumerable<String>, Boolean, String)

Adds a set of files to the ZipFile, using the specified directory path if the archive, and preserving the full directory structure in the filenames.

■ Declaration Syntax

Parameters

fileNames (<u>lEnumerable</u><<u>String</u>>)

The names of the files to add. Each string should refer to a file in th filesystem. The name of the file may be a relative path or a fully-qualified path.

preserveDirHierarchy (Boolean)

whether the entries in the zip archive will reflect the directory hierarchy that is present in the various filenames. For example, if fileNames includes two paths, \Animalia\Chordata\Mammalia\Info.t and \Plantae\Magnoliophyta\Dicotyledon\Info.txt, then calling this method with preserveDirHierarchy = false will result in an exception because of a duplicate entry name, while calling this method with preserveDirHierarchy = true will result in the full direcory paths being included in the entries added to the ZipFile.

directoryPathInArchive (String)

Specifies a directory path to use to override any path in the file name. This path may, or may not, correspond to a real directory in the current filesystem. If the files within the zip are later extracted, this is the path used for the extracted file. Passing null (Nothing in VB) will use the path on each of the fileNames, i any. Passing the empty string ("") will insert the item at the root patl within the archive.

Remarks

If preserveDirHierarchy is true, any directory structure present in the filenames contained in the list is preserved in the archive. On the other hand, if preserveDirHierarchy is false, any directory structure that may k present in the filenames contained in the list is "flattened" in the archive Each file in the list is added to the archive in the specified top-level directory.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to each ZipEntry added.

See Also

AddSelectedFiles(String, String)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 AddFiles Method (fileNames, directoryPathInArchive)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► AddFiles(IEnumerable<String>, String)

Adds a set of files to the **ZipFile**, using the specified directory path if the archive.

Declaration Syntax

Parameters

fileNames (<u>IEnumerable</u><<u>String</u>>)

The names of the files to add. Each string should refer to a file in th filesystem. The name of the file may be a relative path or a fully-qualified path.

directoryPathInArchive (String)

Specifies a directory path to use to override any path in the file name. This path may, or may not, correspond to a real directory in

the current filesystem. If the files within the zip are later extracted, this is the path used for the extracted file. Passing null (Nothing in VB) will use the path on each of the fileNames, i any. Passing the empty string ("") will insert the item at the root path within the archive.

■ Remarks

Any directory structure that may be present in the filenames contained i the list is "flattened" in the archive. Each file in the list is added to the archive in the specified top-level directory.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to each ZipEntry added.

See Also

AddSelectedFiles(String, String)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 AddItem Method | Reference | Ionic.Zip | ZipFile | AddItem() | C#

This method has two overloads.

■ Members

Icon	Member	Description
∃	AddItem(String)	Adds an item, either a file or a directory, to a zip file archive.
≟	AddItem(String, String)	Adds an item, either a file or a directory, to a zip file archive, explicitly specifying the directory path to be used in the archive.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Adds an item, either a file or a directory, to a zip file archive.

Declaration Syntax

Parameters

fileOrDirectoryName (String)

the name of the file or directory to add.

■ Return Value

The ZipEntry added.

Remarks

This method is handy if you are adding things to zip archive and don't want to bother distinguishing between directories or files. Any files are added as single entries. A directory added through this method is added recursively: all files and subdirectories contained within the directory are

added to the ZipFile.

The name of the item may be a relative path or a fully-qualified path. Remember, the items contained in **ZipFile** instance get written to the disk only when you call **Save()** or a similar save method.

The directory name used for the file within the archive is the same as th directory name (potentially a relative path) specified in the *fileOrDirectoryName*.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

See Also

AddFile(String)
AddDirectory(String)
UpdateItem(String)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 AddItem Method (fileOrDirectoryName, directoryPathInArchive)

Reference ► Ionic.Zip ► ZipFile ► AddItem(String, String)

Adds an item, either a file or a directory, to a zip file archive, explicitly specifying the directory path to be used in the archive.

■ Declaration Syntax

Parameters

)

fileOrDirectoryName (String)

the name of the file or directory to add.

directoryPathInArchive (String)

The name of the directory path to use within the zip archive. This path need not refer to an extant directory in the current filesystem. the files within the zip are later extracted, this is the path used for the extracted file. Passing null (Nothing in VB) will use the path of the fileOrDirectoryName. Passing the empty string ("") will insert the

item at the root path within the archive.

Return Value

The ZipEntry added.

■ Remarks

If adding a directory, the add is recursive on all files and subdirectories contained within it.

The name of the item may be a relative path or a fully-qualified path. Th item added by this call to the ZipFile is not read from the disk nor written to the zip file archive until the application calls Save() on the ZipFile.

This version of the method allows the caller to explicitly specify the directory path to be used in the archive, which would override the "natural" path of the filesystem file.

Encryption will be used on the file data if the Password has been set on the ZipFile object, prior to calling this method.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

Examples

This example shows how to zip up a set of files into a flat hierarchy, regardless of where in the filesystem the files originated. The resulting z archive will contain a toplevel directory named "flat", which itself will contain files Readme.txt, MyProposal.docx, and Image1.jpg. A subdirectory under "flat" called SupportFiles will contain all the files in the "c:\SupportFiles" directory on disk.

C#
String[] itemnames= {

```
"c:\\fixedContent\\Readme.txt",
  "MyProposal.docx",
  "c:\\SupportFiles", // a directory
  "images\\Image1.jpg"
};
try
{
 using (ZipFile zip = new ZipFile())
  {
    for (int i = 1; i < itemnames.Length; i++)</pre>
      // will add Files or Dirs, recurses and flatte
      zip.AddItem(itemnames[i], "flat");
    zip.Save(ZipToCreate);
 }
catch (System.Exception ex1)
  System.Console.Error.WriteLine("exception: {0}", e
```

VB.NET

Catch ex1 As Exception
 Console.Error.WriteLine("exception: {0}", ex1.To
End Try

Exceptions

Exception	Condition
FileNotFoundException	Thrown if the file or directory passed in does not exist.

■ See Also

AddFile(String, String)
AddDirectory(String, String)
UpdateItem(String, String)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

▶1

An event handler invoked before, during, and after Adding entries to a z archive.

Declaration Syntax

```
C# Visual Basic Visual C++

public event EventHandler<AddProgressEventArgs> AddP

Public Event AddProgress As EventHandler(Of AddProgr

public:
    event EventHandler<AddProgressEventArgs^>^ AddProgr
    void add (EventHandler<AddProgressEventArgs^void remove (EventHandler)
```

■ Remarks

Adding a large number of entries to a zip file can take a long time. For example, when calling <u>AddDirectory(String)</u> on a directory that contains 50,000 files, it could take 3 minutes or so. This event handler allws an application to track the progress of the Add operation, and to optionally cancel a lengthy Add operation.

Examples

```
c#
int _numEntriesToAdd= 0;
int _numEntriesAdded= 0;
void AddProgressHandler(object sender, AddProgressEv
{
    switch (e.EventType)
    {
        case ZipProgressEventType.Adding_Started:
```

```
Console.WriteLine("Adding files to the z
            break;
        case ZipProgressEventType.Adding_AfterAddEnt
            _numEntriesAdded++;
            Console.WriteLine(String.Format("Adding
                                      _numEntriesAdde
            break;
        case ZipProgressEventType.Adding_Completed:
            Console.WriteLine("Added all files");
            break;
}
void CreateTheZip()
    using (ZipFile zip = new ZipFile())
    {
        zip.AddProgress += AddProgressHandler;
        zip.AddDirectory(System.IO.Path.GetFileName(
        zip.Save(ZipFileToCreate);
    }
}
```

VB.NET
□ Cop

```
Private Sub AddProgressHandler(ByVal sender As Object Select Case e.EventType

Case ZipProgressEventType.Adding_Started

Console.WriteLine("Adding files to the zexit Select

Case ZipProgressEventType.Adding_AfterAddEnt

Console.WriteLine(String.Format("Adding Exit Select

Case ZipProgressEventType.Adding_Completed

Console.WriteLine("Added all files")

Exit Select

End Select

End Select

End Sub
```

```
Sub CreateTheZip()
   Using zip as ZipFile = New ZipFile
        AddHandler zip.AddProgress, AddressOf AddPro
        zip.AddDirectory(System.IO.Path.GetFileName(
        zip.Save(ZipFileToCreate);
   End Using
End Sub
```

■ See Also

SaveProgress
ReadProgress
ExtractProgress

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Add Selected Files Method

Reference ► Ionic.Zip ► ZipFile ► AddSelectedFiles() C#

■ Members

Icon	Member	Description
₫	AddSelectedFiles(String)	Adds to the ZipFile a set of files from the current working directory on disk that conform to the specified criteria.
≡	AddSelectedFiles(String, Boolean)	Adds to the ZipFile a set of files from the disk that conform to the specified criteria, optionally recursing into subdirectories.
≅	AddSelectedFiles(String, String)	Adds to the ZipFile a set of files from a specified directory in the filesysten that conform to the specified criteria.
∃	AddSelectedFiles(String, String, Boolean)	Adds to the ZipFile a set of files from the specified directory on disk, that conform to the specified criteria.
∄	AddSelectedFiles(String, String, String)	Adds to the ZipFile a selection of file from the specified directory on disk, that conform to the specified criteria, and using a specified root path for entries added to the zip archive.
≡	AddSelectedFiles(String, String, Boolean)	Adds to the ZipFile a selection of file from the specified directory on disk,

that conform to the specified criteria, optionally recursing through subdirectories, and using a specified root path for entries added to the zip archive.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Add Selected Files Method

(selectionCriteria)

Reference ► Ionic.Zip ► ZipFile ► AddSelectedFiles(String)

Adds to the ZipFile a set of files from the current working directory on disk, that conform to the specified criteria.

■ Declaration Syntax

Parameters

selectionCriteria (String)

The criteria for file selection

Remarks

This method selects files from the the current working directory matchin the specified criteria, and adds them to the ZipFile.

Specify the criteria in statements of 3 elements: a noun, an operator, an a value. Consider the string "name != *.doc" . The noun is "name". The operator is "!=", implying "Not Equal". The value is "*.doc". That criterior in English, says "all files with a name that does not end in the .doc

extension."

Supported nouns include "name" (or "filename") for the filename; "atime "mtime", and "ctime" for last access time, last modfied time, and created time of the file, respectively; "attributes" (or "attrs") for the file attributes; "size" (or "length") for the file length (uncompressed), and "type" for the type of object, either a file or a directory. The "attributes", "name" and "type" nouns both support = and != as operators. The "size", "atime", "mtime", and "ctime" nouns support = and !=, and >, >=, <, <= as well. The times are taken to be expressed in local time.

Specify values for the file attributes as a string with one or more of the characters H,R,S,A,I,L in any order, implying file attributes of Hidden, ReadOnly, System, Archive, NotContextIndexed, and ReparsePoint (symbolic link) respectively.

To specify a time, use YYYY-MM-DD-HH:mm:ss or YYYY/MM/DD-HH:mm:ss as the format. If you omit the HH:mm:ss portion, it is assume to be 00:00:00 (midnight).

The value for a size criterion is expressed in integer quantities of bytes, kilobytes (use k or kb after the number), megabytes (m or mb), or gigabytes (g or gb).

The value for a name is a pattern to match against the filename, potentially including wildcards. The pattern follows CMD.exe glob rules: implies one or more of any character, while ? implies one character. If the name pattern contains any slashes, it is matched to the entire filename, including the path; otherwise, it is matched against only the filename without the path. This means a pattern of "**.*" matches all files one directory level deep, while a pattern of "*.*" matches all files in all directories.

To specify a name pattern that includes spaces, use single quotes arounthe pattern. A pattern of "'* *.*" will match all files that have spaces in the filename. The full criteria string for that would be "name = '* *.*".

The value for a type criterion is either F (implying a file) or D (implying a directory).

Some examples:

criteria	Files retrieved
name != *.xls	any file with an extension that is not .xls
name = *.mp3	any file with a .mp3 extension.
*.mp3	(same as above) any file with a .mp3 extension.
attributes = A	all files whose attributes include the Archive bit.
attributes != H	all files whose attributes do not include the Hidden bit.
mtime > 2009-01- 01	all files with a last modified time after January 1st, 2009.
size > 2gb	all files whose uncompressed size is greater than 2gb.
type = D	all directories in the filesystem.

You can combine criteria with the conjunctions AND or OR. Using a stril like "name = *.txt AND size >= 100k" for the selectionCriteria retrieves entries whose names end in .txt, and whose uncompressed size is greater than or equal to 100 kilobytes.

For more complex combinations of criteria, you can use parenthesis to group clauses in the boolean logic. Without parenthesis, the precedence of the criterion atoms is determined by order of appearance. Unlike the C# language, the AND conjunction does not take precendence over the logical OR. This is important only in strings that contain 3 or more criterion atoms. In other words, "name = *.txt and size > 1000 or attributes = H" implies "((name = *.txt AND size > 1000) OR attributes = H)" while "attributes = H OR name = *.txt and size > 1000" evaluates to ((attributes = H OR name = *.txt) AND size > 1000)". When in doubt, us parenthesis.

Using time properties requires some extra care. If you want to retrieve ϵ entries that were last updated on 2009 February 14, specify a time rang like so:"mtime >= 2009-02-14 AND mtime < 2009-02-15". Read this to say: all files updated after 12:00am on February 14th, until 12:00am on

February 15th. You can use the same bracketing approach to specify ar time period - a year, a month, a week, and so on.

The syntax allows one special case: if you provide a string with no spaces, it is treated as a pattern to match for the filename. Therefore a string like "*.xls" will be equivalent to specifying "name = *.xls".

There is no logic in this method that insures that the file inclusion criteria are internally consistent. For example, it's possible to specify criteria that says the file must have a size of less than 100 bytes, as well as a size that is greater than 1000 bytes. Obviously no file will ever satisfy such criteria, but this method does not detect such logical inconsistencies. The caller is responsible for insuring the criteria are sensible.

Using this method, the file selection does not recurse into subdirectories and the full path of the selected files is included in the entries added into the zip archive. If you don't like these behaviors, see the other overload of this method.

Examples

This example zips up all *.csv files in the current working directory.

```
using (ZipFile zip = new ZipFile())
{
    // To just match on filename wildcards,
    // use the shorthand form of the selectionCriter
    zip.AddSelectedFiles("*.csv");
    zip.Save(PathToZipArchive);
}
```

```
VB.NET © Cop
```

```
Using zip As ZipFile = New ZipFile()
    zip.AddSelectedFiles("*.csv")
    zip.Save(PathToZipArchive)
End Using
```

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

√]

Ionic Zip Library v1.9.1.6 Add Selected Files Method (selectionCriteria, recurseDirectories)

Reference ► Ionic.Zip ► ZipFile ► AddSelectedFiles(String, **Boolean**)

Adds to the ZipFile a set of files from the disk that conform to the specified criteria, optionally recursing into subdirectories.

Declaration Syntax

```
Visual C++
C#
                  Visual Basic
public void AddSelectedFiles(
        string selectionCriteria,
        bool recurseDirectories
Public Sub AddSelectedFiles ( _
        selectionCriteria As String, _
        recurseDirectories As Boolean
```

```
public:
void AddSelectedFiles(
        String^ selectionCriteria,
        bool recurseDirectories
```

Parameters

selectionCriteria (String)

The criteria for file selection

recurseDirectories (Boolean)

If true, the file selection will recurse into subdirectories.

Remarks

This method selects files from the the current working directory matchin the specified criteria, and adds them to the ZipFile. If recurseDirectories is true, files are also selected from subdirectories, and the directory structure in the filesystem is reproduce in the zip archive, rooted at the current working directory.

Using this method, the full path of the selected files is included in the entries added into the zip archive. If you don't want this behavior, use or of the overloads of this method that allows the specification of a directoryInArchive.

For details on the syntax for the selectionCriteria parameter, see <u>AddSelectedFiles(String)</u>.

Examples

This example zips up all *.xml files in the current working directory, or a subdirectory, that are larger than 1mb.

```
C# □ Cop
```

```
using (ZipFile zip = new ZipFile())
{
    // Use a compound expression in the selectionCri
    zip.AddSelectedFiles("name = *.xml and size >
        zip.Save(PathToZipArchive);
}
```

VB.NET

```
Using zip As ZipFile = New ZipFile()
   ' Use a compound expression in the selectionCrit
   zip.AddSelectedFiles("name = *.xml and size >
   zip.Save(PathToZipArchive)
End Using
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

<u>|</u>

Ionic Zip Library v1.9.1.6 Add Selected Files Method (selection Criteria, directory On Disk)

Reference ► Ionic.Zip ► ZipFile ► AddSelectedFiles(String, String)

Adds to the ZipFile a set of files from a specified directory in the filesystem, that conform to the specified criteria.

■ Declaration Syntax

Parameters

)

selectionCriteria (String)

The criteria for file selection

directoryOnDisk (String)

The name of the directory on the disk from which to select files.

Remarks

This method selects files that conform to the specified criteria, from the

the specified directory on disk, and adds them to the ZipFile. The search does not recurse into subdirectores.

Using this method, the full filesystem path of the files on disk is reproduced on the entries added to the zip file. If you don't want this behavior, use one of the other overloads of this method.

For details on the syntax for the selectionCriteria parameter, see <u>AddSelectedFiles(String)</u>.

Examples

}

This example zips up all *.xml files larger than 1mb in the directory give by "d:\rawdata".

```
Using (ZipFile zip = new ZipFile())
{
    // Use a compound expression in the selectionCri
    zip.AddSelectedFiles("name = *.xml and size >
    zip.Save(PathToZipArchive);
```

VB.NET

```
Using zip As ZipFile = New ZipFile()
   ' Use a compound expression in the selectionCrit
   zip.AddSelectedFiles("name = *.xml and size >
   zip.Save(PathToZipArchive)
End Using
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Add Selected Files Method (selectionCriteria, directoryOnDisk, recurseDirectories)

Reference ► Ionic.Zip ► ZipFile ► AddSelectedFiles(String, String, **Boolean**)

Adds to the ZipFile a set of files from the specified directory on disk, that conform to the specified criteria.

Declaration Syntax

```
Visual C++
                  Visual Basic
C#
public void AddSelectedFiles(
        string selectionCriteria,
        string directoryOnDisk,
        bool recurseDirectories
```

```
Public Sub AddSelectedFiles (
        selectionCriteria As String, _
        directoryOnDisk As String, _
        recurseDirectories As Boolean
```

```
public:
void AddSelectedFiles(
        String^ selectionCriteria,
        String^ directoryOnDisk,
        bool recurseDirectories
```

Parameters

```
selectionCriteria (String)
   The criteria for file selection
directoryOnDisk (String)
```

The filesystem path from which to select files.

recurseDirectories (Boolean)

If true, the file selection will recurse into subdirectories.

Remarks

This method selects files from the the specified disk directory matching the specified selection criteria, and adds them to the ZipFile. If recurseDirectories is true, files are also selected from subdirectories.

The full directory structure in the filesystem is reproduced on the entries added to the zip archive. If you don't want this behavior, use one of the overloads of this method that allows the specification of a directoryInArchive.

For details on the syntax for the selectionCriteria parameter, see <u>AddSelectedFiles(String)</u>.

Examples

This example zips up all *.csv files in the "files" directory, or any subdirectory, that have been saved since 2009 February 14th.

```
using (ZipFile zip = new ZipFile())
{
    // Use a compound expression in the selectionCri
    zip.AddSelectedFiles("name = *.csv and mtime >
    zip.Save(PathToZipArchive);
}
```

VB.NET

```
Using zip As ZipFile = New ZipFile()
   ' Use a compound expression in the selectionCrit
   zip.AddSelectedFiles("name = *.csv and mtime >
   zip.Save(PathToZipArchive)
End Using
```

Examples

This example zips up all files in the current working directory, and all its child directories, except those in the excludethis subdirectory.

VB.NET

```
Using Zip As ZipFile = New ZipFile(zipfile)
  Zip.AddSelectedFfiles("name != 'excludethis\*.*'",
  Zip.Save()
End Using
```

|

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 Add Selected Files Method (selection Criteria, directory On Disk, directory Path In Archive)

Reference ► Ionic.Zip ► ZipFile ► AddSelectedFiles(String, String, String)

Adds to the ZipFile a selection of files from the specified directory on disk, that conform to the specified criteria, and using a specified root pa for entries added to the zip archive.

Declaration Syntax

Parameters

selectionCriteria (String)

The criteria for selection of files to add to the ZipFile.

directoryOnDisk (String)

The path to the directory in the filesystem from which to select files

directoryPathInArchive (String)

Specifies a directory path to use to in place of the directoryOnDisk. This path may, or may not, correspond to a real directory in the current filesystem. If the files within the zip are later extracted, this is the path used for the extracted file. Passing null (nothing in VB) will use the path on the file name, if any; in othe words it would use directoryOnDisk, plus any subdirectory. Passing the empty string ("") will insert the item at the root path within the archive.

Remarks

This method selects files from the specified disk directory matching the specified selection criteria, and adds those files to the ZipFile, using the specified directory path in the archive. The search does not recurse into subdirectories. For details on the syntax for the selectionCriteria parameter, see AddSelectedFiles(String).

Examples

This example zips up all *.psd files in the "photos" directory that have been saved since 2009 February 14th, and puts them all in a zip file, using the directory name of "content" in the zip archive itself. When the zip archive is unzipped, the folder containing the .psd files will be name "content".

```
Using (ZipFile zip = new ZipFile())
{
    // Use a compound expression in the selectionCri
    zip.AddSelectedFiles("name = *.psd and mtime >
    zip.Save(PathToZipArchive);
}
```

Cop

VB.NET

```
Using zip As ZipFile = New ZipFile
   zip.AddSelectedFiles("name = *.psd and mtime >
   zip.Save(PathToZipArchive)
End Using
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 AddSelectedFiles Method (selectionCriteria, directoryOnDisk, directoryPathInArchive, recurseDirectories)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► AddSelectedFiles(String, String, String, Boolean)

Adds to the ZipFile a selection of files from the specified directory on disk, that conform to the specified criteria, optionally recursing through subdirectories, and using a specified root path for entries added to the z archive.

Declaration Syntax

Parameters

selectionCriteria (String)

The criteria for selection of files to add to the ZipFile.

directoryOnDisk (String)

The path to the directory in the filesystem from which to select files

directoryPathInArchive (String)

Specifies a directory path to use to in place of the directoryOnDisk. This path may, or may not, correspond to a real directory in the current filesystem. If the files within the zip are later extracted, this is the path used for the extracted file. Passing null (nothing in VB) will use the path on the file name, if any; in othe words it would use directoryOnDisk, plus any subdirectory. Passing the empty string ("") will insert the item at the root path within the archive.

recurseDirectories (Boolean)

If true, the method also scans subdirectories for files matching the criteria.

Remarks

This method selects files from the specified disk directory that match the specified selection criteria, and adds those files to the ZipFile, using the specified directory path in the archive. If recurseDirectories is true, files are also selected from subdirectories, and the directory structure in the filesystem is reproduced in the zip archive, rooted at the directory specified by directoryOnDisk. For details on the syntax for the selectionCriteria parameter, see AddSelectedFiles(String).

Examples

This example zips up all files that are NOT *.pst files, in the current working directory and any subdirectories.

```
C#
using (ZipFile zip = new ZipFile())
{
   zip.AddSelectedFiles("name != *.pst", SourceDire
```

```
zip.Save(PathToZipArchive);
}
```

VB.NET

Cop^{*}

```
Using zip As ZipFile = New ZipFile
    zip.AddSelectedFiles("name != *.pst", SourceDire
    zip.Save(PathToZipArchive)
End Using
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

•

A Text Encoding to use when encoding the filenames and comments for all the ZipEntry items, during a ZipFile.Save() operation.

■ Declaration Syntax

Remarks

Whether the encoding specified here is used during the save depends calternateEncodingUsage.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Alternate Encoding Usage Property

Reference ► Ionic.Zip ► ZipFile ► AlternateEncodingUsage

A flag that tells if and when this instance should apply AlternateEncodin to encode the filenames and comments associated to of ZipEntry object contained within this instance.

Declaration Syntax

```
Visual Basic
                                    Visual C++
C#
public ZipOption AlternateEncodingUsage { get; set;
Public Property AlternateEncodingUsage As ZipOption
        Get
        Set
public:
property ZipOption AlternateEncodingUsage {
        ZipOption get ();
        void set (ZipOption value);
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BufferSize Property Reference ► Ionic.Zip ► ZipFile ► BufferSize C#

Size of the IO buffer used while saving.

■ Declaration Syntax

■ Remarks

First, let me say that you really don't need to bother with this. It is here t allow for optimizations that you probably won't make! It will work fine if you don't set or get this property at all. Ok?

Now that we have *that* out of the way, the fine print: This property affect the size of the buffer that is used for I/O for each entry contained in the zip file. When a file is read in to be compressed, it uses a buffer given b the size here. When you update a zip file, the data for unmodified entrie is copied from the first zip file to the other, through a buffer given by the size here.

Changing the buffer size affects a few things: first, for larger buffer sizes the memory used by the ZipFile, obviously, will be larger during I/O operations. This may make operations faster for very much larger files. Last, for any given entry, when you use a larger buffer there will be fewer

progress events during I/O operations, because there's one progress event generated for each time the buffer is filled and then emptied.

The default buffer size is 8k. Increasing the buffer size may speed thing up as you compress larger files. But there are no hard-and-fast rules here, eh? You won't know til you test it. And there will be a limit where ever larger buffers actually slow things down. So as I said in the beginning, it's probably best if you don't set or get this property at all.

Examples

This example shows how you might set a large buffer size for efficiency when dealing with zip entries that are larger than 1gb.

```
C# □ Cop
```

```
using (ZipFile zip = new ZipFile())
{
    zip.SaveProgress += this.zip1_SaveProgress;
    zip.AddDirectory(directoryToZip, "");
    zip.UseZip64WhenSaving = Zip64Option.Always;
    zip.BufferSize = 65536*8; // 65536 * 8 = 512k
    zip.Save(ZipFileToCreate);
}
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Buffer Size Default Field

Reference ► Ionic.Zip ► ZipFile ► BufferSizeDefault

C#

Default size of the buffer used for IO.

■ Declaration Syntax

C# Visual Basic Visual C++

public static readonly int BufferSizeDefault

Public Shared ReadOnly BufferSizeDefault As Integer

public:

static initonly int BufferSizeDefault

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CaseSensitiveRetrieval Property

Reference ► Ionic.Zip ► ZipFile ► CaseSensitiveRetrieval

Indicates whether to perform case-sensitive matching on the filename when retrieving entries in the zipfile via the string-based indexer.

■ Declaration Syntax

■ Remarks

The default value is false, which means don't do case-sensitive matching. In other words, retrieving zip["ReadMe.Txt"] is the same as zip["readme.txt"]. It really makes sense to set this to true only if you a not running on Windows, which has case-insensitive filenames. But since this library is not built for non-Windows platforms, in most cases you should just leave this property alone.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CheckZip Method

Reference ► Ionic.Zip ► ZipFile ► CheckZip()

C#

■ Members

Icon	Member	Description
=\$S	CheckZip(String)	Checks a zip file to see if its directory is consistent.
= ♦ 5	CheckZip(String, Boolean, TextWriter)	Checks a zip file to see if its directory is consistent, and optionally fixes the directory if necessary.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CheckZip Method (zipFileName)

C#

Reference ► Ionic.Zip ► ZipFile ► CheckZip(String)

Checks a zip file to see if its directory is consistent.

Declaration Syntax

Parameters

zipFileName (String)

The filename to of the zip file to check.

■ Return Value

true if the named zip file checks OK. Otherwise, false.

Remarks

In cases of data error, the directory within a zip file can get out of synch with the entries in the zip file. This method checks the given zip file and returns true if this has occurred.

This method may take a long time to run for large zip files.

This method is not supported in the Reduced or Compact Framework versions of DotNetZip.

Developers using COM can use the ComHelper.CheckZip(String) method.

■ See Also

<u>FixZipDirectory(String)</u>
CheckZip(String, Boolean, TextWriter)

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 CheckZip Method (zipFileName, fixIfNecessary, writer)

Reference ► Ionic.Zip ► ZipFile ► CheckZip(String, Boolean, TextWriter)

Checks a zip file to see if its directory is consistent, and optionally fixes the directory if necessary.

Declaration Syntax

```
C# Visual Basic Visual C++

public static bool CheckZip(
    string zipFileName,
    bool fixIfNecessary,
    TextWriter writer
)
```

Parameters

zipFileName (String)

The filename to of the zip file to check.

fixIfNecessary (Boolean)

If true, the method will fix the zip file if necessary.

writer (TextWriter)

a TextWriter in which messages generated while checking will be written.

Return Value

true if the named zip is OK; false if the file needs to be fixed.

Remarks

In cases of data error, the directory within a zip file can get out of synch with the entries in the zip file. This method checks the given zip file, and returns true if this has occurred. It also optionally fixes the zipfile, saving the fixed copy in *Name_*Fixed.zip.

This method may take a long time to run for large zip files. It will take even longer if the file actually needs to be fixed, and if fixIfNecessary is true.

This method is not supported in the Reduced or Compact Framework versions of DotNetZip.

■ See Also

CheckZip(String)
FixZipDirectory(String)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CheckZipPassword Method (zipFileName, password)

Reference ► Ionic.Zip ► ZipFile ► CheckZipPassword(String, String)

Verify the password on a zip file.

C#

■ Declaration Syntax

Parameters

)

zipFileName (String)

The filename to of the zip file to fix.

password (String)

The password to check.

■ Return Value

a bool indicating whether the password matches.

■ Remarks

Keep in mind that passwords in zipfiles are applied to zip entries, not to the entire zip file. So testing a zipfile for a particular password doesn't work in the general case. On the other hand, it's often the case that a single password will be used on all entries in a zip file. This method works for that case.

There is no way to check a password without doing the decryption. So this code decrypts and extracts the given zipfile into Null

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Size of the work buffer to use for the ZLIB codec during compression.

■ Declaration Syntax

■ Remarks

When doing ZLIB or Deflate compression, the library fills a buffer, then passes it to the compressor for compression. Then the library reads out the compressed bytes. This happens repeatedly until there is no more uncompressed data to compress. This property sets the size of the buffer that will be used for chunk-wise compression. In order for the setting to take effect, your application needs to set this property before calling one of the ZipFile.Save() overloads.

Setting this affects the performance and memory efficiency of compression and decompression. For larger files, setting this to a larger size may improve compression performance, but the exact numbers val depending on available memory, the size of the streams you are compressing, and a bunch of other variables. I don't have good firm recommendations on how to set it. You'll have to test it yourself. Or just leave it alone and accept the default.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Comment Property Reference ► Ionic.Zip ► ZipFile ► Comment C#

A comment attached to the zip archive.

Declaration Syntax

■ Remarks

This property is read/write. It allows the application to specify a commer for the ZipFile, or read the comment for the ZipFile. After setting this property, changes are only made permanent when you call a Save() method.

According to <u>PKWARE's zip specification</u>, the comment is not encrypted even if there is a password set on the zip file.

The specification does not describe how to indicate the encoding used a comment string. Many "compliant" zip tools and libraries use IBM437 as the code page for comments; DotNetZip, too, follows that practice. C the other hand, there are situations where you want a Comment to be encoded with something else, for example using code page 950 "Big-5 Chinese". To fill that need, DotNetZip will encode the comment following the same procedure it follows for encoding filenames: (a) if

AlternateEncodingUsage is Never, it uses the default encoding (IBM437). (b) if AlternateEncodingUsage is Always, it always uses the alternate encoding (AlternateEncoding). (c) if AlternateEncodingUsage AsNecessary, it uses the alternate encoding only if the default encoding is not sufficient for encoding the comment - in other words if decoding the result does not produce the original string. This decision is taken at the time of the call to ZipFile.Save().

When creating a zip archive using this library, it is possible to change th value of <u>AlternateEncoding</u> between each entry you add, and between adding entries and the call to Save(). Don't do this. It will likely result a zip file that is not readable by any tool or application. For best interoperability, leave <u>AlternateEncoding</u> alone, or specify it only once, before adding any entries to the ZipFile instance.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | CompressionLevel Property | | Reference ► Ionic.Zip ► ZipFile ► CompressionLevel | C#

Sets the compression level to be used for entries subsequently added to the zip archive.

Declaration Syntax

Remarks

Varying the compression level used on entries can affect the size-vsspeed tradeoff when compression and decompressing data streams or files.

As with some other properties on the ZipFile class, like Password, Encryption, and ZipErrorAction, setting this property on a ZipFile instance will cause the specified CompressionLevel to be used or all ZipEntry items that are subsequently added to the ZipFile instance. If you set this property after you have added items to the ZipFile, but before you have called Save(), those items will not us the specified compression level.

If you do not set this property, the default compression level is used,

which normally gives a good balance of compression efficiency and compression speed. In some tests, using <code>BestCompression</code> can double the time it takes to compress, while delivering just a small increase in compression efficiency. This behavior will vary with the type of data you compress. If you are in doubt, just leave this setting alone, and accept the default.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compression Method Property

Reference ► Ionic.Zip ► ZipFile ► CompressionMethod

The compression method for the zipfile.

C#

Declaration Syntax

Remarks

By default, the compression method is CompressionMethod.Deflate.

See Also

CompressionMethod

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Contains Entry Method (name)

Reference ► Ionic.Zip ► ZipFile ► ContainsEntry(String)

Returns true if an entry by the given name exists in the ZipFile.

Declaration Syntax

Parameters

name (String)

the name of the entry to find

■ Return Value

true if an entry with the given name exists; otherwise false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Count Property

Reference ► Ionic.Zip ► ZipFile ► Count
C#
```

Returns the number of entries in the Zip archive.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public int Count { get; }

Public ReadOnly Property Count As Integer
    Get

public:
property int Count {
    int get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **DefaultEncoding Property**

Reference ► Ionic.Zip ► ZipFile ► DefaultEncoding

C#

|

The default text encoding used in zip archives. It is numeric 437, also known as IBM437.

Declaration Syntax

■ See Also

ProvisionalAlternateEncoding

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Dispose Method

Reference ► Ionic.Zip ► ZipFile ► Dispose()

C#

■ Members

Icon	Member	Description
∄	Dispose()	Closes the read and write streams associated to the ZipFile, if necessary.
Ģ ¯ ₩	Dispose(Boolean)	Disposes any managed resources, if the flag is set, then marks the instanc disposed. This method is typically no called explicitly from application code

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Closes the read and write streams associated to the **ZipFile**, if necessary.

Declaration Syntax

```
C# Visual Basic Visual C++

public void Dispose()

Public Sub Dispose

public:
virtual void Dispose() sealed
```

■ Remarks

The Dispose() method is generally employed implicitly, via a using(...) {...} statement. (Using...End Using in VB) If you do not employ a using statement, insure that your application calls Dispose() explicitly. For example, in a Powershell application, or an application that uses the COM interop interface, you must call Dispose(explicitly.

Examples

This example extracts an entry selected by name, from the Zip file to the Console.

```
C# □ Cop
```

```
using (ZipFile zip = ZipFile.Read(zipfile))
{
  foreach (ZipEntry e in zip)
  {
   if (WantThisEntry(e.FileName))
      zip.Extract(e.FileName, Console.OpenStandardOu
```

```
}
} // Dispose() is called implicitly here.
```

VB.NET

Cop

```
Using zip As ZipFile = ZipFile.Read(zipfile)
    Dim e As ZipEntry
    For Each e In zip
        If WantThisEntry(e.FileName) Then
            zip.Extract(e.FileName, Console.OpenStanda
        End If
    Next
End Using ' Dispose is implicity called here
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

▶1

Disposes any managed resources, if the flag is set, then marks the instance disposed. This method is typically not called explicitly from application code.

Declaration Syntax

Parameters

disposeManagedResources (Boolean)

indicates whether the method should dispose streams or not.

■ Remarks

Applications should call the no-arg Dispose method.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library

v1.9.1.6 EmitTimesInUnixFormatWhenSaving Property

Reference ► Ionic.Zip ► ZipFile ►

EmitTimesInUnixFormatWhenSaving

Specifies whether the Creation, Access, and Modified times for entries added to the zip file will be emitted in "Unix(tm) format" when the zip archive is saved.

Declaration Syntax

■ Remarks

An application creating a zip archive can use this flag to explicitly specif that the file times for the entries should or should not be stored in the zip archive in the format used by Unix. By default this flag is false, meaning the Unix-format times are not stored in the zip archive.

When adding an entry from a file or directory, the Creation (CreationTime), Access (AccessedTime), and Modified (ModifiedTime) times for the given entry are automatically set from the filesystem value. When adding an entry from a stream or string, all three values are

implicitly set to DateTime.Now. Applications can also explicitly set those times by calling <u>SetEntryTimes(DateTime, DateTime, DateTime)</u>.

PKWARE's zip specification describes multiple ways to format these times in a zip file. One is the format Windows applications normally use 100ns ticks since January 1, 1601 UTC. The other is a format Unix applications typically use: seconds since January 1, 1970 UTC. Each format can be stored in an "extra field" in the zip entry when saving the zip archive. The former uses an extra field with a Header Id of 0x000A, while the latter uses a header ID of 0x5455, although you probably don' need to know that.

Not all tools and libraries can interpret these fields. Windows compresse folders is one that can read the Windows Format timestamps, while I believe the Infozip tools can read the Unix format timestamps. Some tools and libraries may be able to read only one or the other. DotNetZip can read or write times in either or both formats.

The times stored are taken from <u>ModifiedTime</u>, <u>AccessedTime</u>, and <u>CreationTime</u>.

This property is not mutually exclusive of the EmitTimesInWindowsFormatWhenSaving property. It is possible and legal and valid to produce a zip file that contains timestamps encoded in the Unix format as well as in the Windows format, in addition to the LastModified time attached to each entry in the zip archive, a time that i always stored in "DOS format". And, notwithstanding the names PKWar uses for these time formats, any of them can be read and written by any computer, on any operating system. But, there are no guarantees that a program running on Mac or Linux will gracefully handle a zip file with "Windows" formatted times, or that an application that does not use DotNetZip but runs on Windows will be able to handle file times in Unix format.

When in doubt, test. Sorry, I haven't got a complete list of tools and whice sort of timestamps they can use and will tolerate. If you get any good information and would like to pass it on, please do so and I will include that information in this documentation.

■ See Also

<u>EmitTimesInUnixFormatWhenSaving</u> <u>EmitTimesInWindowsFormatWhenSaving</u>

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

▶ 1

Ionic Zip Library

v1.9.1.6EmitTimesInWindowsFormatWhenSaving Property

Reference ► Ionic.Zip ► ZipFile ►

EmitTimesInWindowsFormatWhenSaving

Specifies whether the Creation, Access, and Modified times for entries added to the zip file will be emitted in Windows format when the zip archive is saved.

Declaration Syntax

■ Remarks

An application creating a zip archive can use this flag to explicitly specif that the file times for the entries should or should not be stored in the zip archive in the format used by Windows. By default this flag is true, meaning the Windows-format times are stored in the zip archive.

When adding an entry from a file or directory, the Creation (CreationTime), Access (AccessedTime), and Modified (ModifiedTime) times for the given entry are automatically set from the filesystem value. When adding an entry from a stream or string, all three values are

implicitly set to <code>DateTime.Now</code>. Applications can also explicitly set those times by calling <code>SetEntryTimes(DateTime, DateTime)</code>.

PKWARE's zip specification describes multiple ways to format these times in a zip file. One is the format Windows applications normally use 100ns ticks since January 1, 1601 UTC. The other is a format Unix applications typically use: seconds since January 1, 1970 UTC. Each format can be stored in an "extra field" in the zip entry when saving the zip archive. The former uses an extra field with a Header Id of 0x000A, while the latter uses a header ID of 0x5455, although you probably don' need to know that.

Not all tools and libraries can interpret these fields. Windows compresse folders is one that can read the Windows Format timestamps, while I believe the Infozip tools can read the Unix format timestamps. Some tools and libraries may be able to read only one or the other. DotNetZip can read or write times in either or both formats.

The times stored are taken from <u>ModifiedTime</u>, <u>AccessedTime</u>, and <u>CreationTime</u>.

The value set here applies to all entries subsequently added to the ZipFile.

This property is not mutually exclusive of the EmitTimesInUnixFormatWhenSaving property. It is possible and legal and valid to produce a zip file that contains timestamps encoded in the Unix format as well as in the Windows format, in addition to the LastModified time attached to each entry in the archive, a time that is always stored in "DOS format". And, notwithstanding the names PKWar uses for these time formats, any of them can be read and written by any computer, on any operating system. But, there are no guarantees that a program running on Mac or Linux will gracefully handle a zip file with "Windows" formatted times, or that an application that does not use DotNetZip but runs on Windows will be able to handle file times in Unix format.

When in doubt, test. Sorry, I haven't got a complete list of tools and whice sort of timestamps they can use and will tolerate. If you get any good

information and would like to pass it on, please do so and I will include that information in this documentation.

Examples

This example shows how to save a zip file that contains file timestamps in a format normally used by Unix.

C#

```
using (var zip = new ZipFile())
{
    // produce a zip file the Mac will like
    zip.EmitTimesInWindowsFormatWhenSaving = false;
    zip.EmitTimesInUnixFormatWhenSaving = true;
    zip.AddDirectory(directoryToZip, "files");
    zip.Save(outputFile);
}
```

VB.NET

```
Using zip As New ZipFile
   '' produce a zip file the Mac will like
   zip.EmitTimesInWindowsFormatWhenSaving = False
   zip.EmitTimesInUnixFormatWhenSaving = True
   zip.AddDirectory(directoryToZip, "files")
   zip.Save(outputFile)
End Using
```

See Also

<u>EmitTimesInWindowsFormatWhenSaving</u> <u>EmitTimesInUnixFormatWhenSaving</u>

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Encryption Property Reference ► Ionic.Zip ► ZipFile ► Encryption C#

The Encryption to use for entries added to the ZipFile.

Declaration Syntax

■ Remarks

Set this when creating a zip archive, or when updating a zip archive. Th specified Encryption is applied to the entries subsequently added to the ZipFile instance. Applications do not need to set the Encryption property when reading or extracting a zip archive.

If you set this to something other than EncryptionAlgorithm.None, you walso need to set the <u>Password</u>.

As with some other properties on the ZipFile class, like Password and CompressionLevel, setting this property on a ZipFile instance v cause the specified EncryptionAlgorithm to be used on all ZipEntry items that are subsequently added to the ZipFile instance. In other words, if you set this property after you have added items to the ZipFile, but before you have called Save(), those items will not be

encrypted or protected with a password in the resulting zip archive. To get a zip archive with encrypted entries, set this property, along with the Password property, before calling AddFile, AddItem, or AddDirectory (etc.) on the ZipFile instance.

If you read a ZipFile, you can modify the Encryption on an encrypted entry, only by setting the Encryption property on the ZipEntry itself. Setting the Encryption property on the ZipFile, once it has been created via a call to ZipFile. Read() does not affect entries that were previously read.

For example, suppose you read a ZipFile, and there is an encrypted entry. Setting the Encryption property on that ZipFile and then calling Save() on the ZipFile does not update the Encryption used for the entries in the archive. Neither is an exception thrown. Instead, what happens during the Save() is that all previously existing entries are copied through to the new zip archive, with whatever encryption and password that was used when originally creating the zip archive. Upon re-reading that archive, to extract entries, applications should use the original password or passwords, if any.

Suppose an application reads a ZipFile, and there is an encrypted entry. Setting the Encryption property on that ZipFile and then adding new entries (via AddFile(), AddEntry(), etc) and then calling Save() on the ZipFile does not update the Encryption on any of the entries that had previously been in the ZipFile. The Encryption property applies only to the newly-added entries.

Examples

This example creates a zip archive that uses encryption, and then extracts entries from the archive. When creating the zip archive, the ReadMe.txt file is zipped without using a password or encryption. The other files use encryption.

C#

```
// Create a zip archive with AES Encryption.
using (ZipFile zip = new ZipFile())
{
    zip.AddFile("ReadMe.txt");
    zip.Encryption= EncryptionAlgorithm.WinZipAes256
    zip.Password= "Top.Secret.No.Peeking!";
    zip.AddFile("7440-N49th.png");
    zip.AddFile("2008-Regional-Sales-Report.pdf");
    zip.Save("EncryptedArchive.zip");
}
// Extract a zip archive that uses AES Encryption.
// You do not need to specify the algorithm during e
using (ZipFile zip = ZipFile.Read("EncryptedArchive.
{
    zip.Password= "Top.Secret.No.Peeking!";
    zip.ExtractAll("extractDirectory");
}
```

VB.NET

```
' Create a zip that uses Encryption.
Using zip As New ZipFile()
    zip.Encryption= EncryptionAlgorithm.WinZipAes256
    zip.Password= "Top.Secret.No.Peeking!"
    zip.AddFile("ReadMe.txt")
    zip.AddFile("7440-N49th.png")
    zip.AddFile("2008-Regional-Sales-Report.pdf")
    zip.Save("EncryptedArchive.zip")
End Using
' Extract a zip archive that uses AES Encryption.
' You do not need to specify the algorithm during ex
Using (zip as ZipFile = ZipFile.Read("EncryptedArchizip.Password= "Top.Secret.No.Peeking!"
    zip.ExtractAll("extractDirectory")
End Using
```

See Also

ZipFile.Password ZipEntry.Encryption

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

```
Ionic Zip Library v1.9.1.6
Entries Property

Reference ► Ionic.Zip ► ZipFile ► Entries
C#
```

Returns the readonly collection of entries in the Zip archive.

■ Declaration Syntax

Remarks

If there are no entries in the current **ZipFile**, the value returned is a non-null zero-element collection. If there are entries in the zip file, the elements are returned in no particular order.

This is the implied enumerator on the ZipFile class. If you use a ZipFile instance in a context that expects an enumerator, you will ge this collection.

See Also

EntriesSorted

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

•

Ionic Zip Library v1.9.1.6 Entries Sorted Property

Reference ► Ionic.Zip ► ZipFile ► EntriesSorted

C# _

Returns a readonly collection of entries in the Zip archive, sorted by FileName.

Declaration Syntax

■ Remarks

If there are no entries in the current ZipFile, the value returned is a non-null zero-element collection. If there are entries in the zip file, the elements are returned sorted by the name of the entry.

Examples

This example fills a Windows Forms ListView with the entries in a zip file

C# Cop

```
using (ZipFile zip = ZipFile.Read(zipFile))
{
    foreach (ZipEntry entry in zip.EntriesSorted)
    {
        ListViewItem item = new ListViewItem(n.ToStr n++;
        string[] subitems = new string[] {
            entry.FileName.Replace("/","\\"),
```

```
entry.LastModified.ToString("yyyy-MM-dd
entry.UncompressedSize.ToString(),
    String.Format("{0,5:F0}%", entry.Compres
    entry.CompressedSize.ToString(),
        (entry.UsesEncryption) ? "Y" : "N",
        String.Format("{0:X8}", entry.Crc)};

foreach (String s in subitems)
{
    ListViewItem.ListViewSubItem subitem = n
        subitem.Text = s;
        item.SubItems.Add(subitem);
}

this.listView1.Items.Add(item);
}
```

■ See Also

Entries

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

1

The list of filenames for the entries contained within the zip archive.

Declaration Syntax

```
C# Visual Basic Visual C++
public ICollection<string> EntryFileNames { get; }

Public ReadOnly Property EntryFileNames As ICollecti
    Get

public:
property ICollection<String^>^ EntryFileNames {
    ICollection<String^>^ get ();
}
```

Return Value

The list of strings for the filenames contained within the Zip archive.

Remarks

According to the ZIP specification, the names of the entries use forward slashes in pathnames. If you are scanning through the list, you may have to swap forward slashes for backslashes.

Examples

This example shows one way to test if a filename is already contained within a zip archive.

C# □ Cop

```
String zipFileToRead= "PackedDocuments.zip";
string candidate = "DatedMaterial.xps";
using (ZipFile zip = new ZipFile(zipFileToRead))
{
```

VB.NET □ Cop

■ See Also

Item[String]

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

•

Ionic Zip Library v1.9.1.6 ExtractAll Method

Reference ► Ionic.Zip ► ZipFile ► ExtractAll()

C#

■ Members

Icon	Member	Description
∃	ExtractAll(String)	Extracts all of the items in the zip archive, to the specified path in the filesystem. The path can be relative or fully-qualified.
∃	ExtractAll(String, ExtractExistingFileAction)	Extracts all of the items in the zip archive, to the specified path in the filesystem, using the specified behavior when extraction would overwrite an existing file.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | ExtractAll Method (path) | Reference ► Ionic.Zip ► ZipFile ► ExtractAll(String) | C#

Extracts all of the items in the zip archive, to the specified path in the filesystem. The path can be relative or fully-qualified.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public void ExtractAll(
    string path
)

Public Sub ExtractAll ( _
    path As String _
)

public:
void ExtractAll(
    String^ path
)
```

Parameters

path (String)

The path to which the contents of the zipfile will be extracted. The path can be relative or fully-qualified.

Remarks

This method will extract all entries in the ZipFile to the specified pat

If an extraction of a file from the zip archive would overwrite an existing file in the filesystem, the action taken is dictated by the ExtractExistingFile property, which overrides any setting you may have made on individual ZipEntry instances. By default, if you have not set th property on the ZipFile instance, the entry will not be extracted, the

existing file will not be overwritten and an exception will be thrown. To change this, set the property, or use the ExtractExistingFileAction) overload that allows you to specify an ExtractExistingFileAction parameter.

The action to take when an extract would overwrite an existing file applies to all entries. If you want to set this on a per-entry basis, then yo must use one of the <u>ZipEntry.Extract</u> methods.

This method will send verbose output messages to the StatusMessageTextWriter, if it is set on the ZipFile instance.

You may wish to take advantage of the ExtractProgress event.

About timestamps: When extracting a file entry from a zip archive, the extracted file gets the last modified time of the entry as stored in the archive. The archive may also store extended file timestamp informatior including last accessed and created times. If these are present in the ZipEntry, then the extracted file will also get these times.

A Directory entry is somewhat different. It will get the times as described for a file entry, but, if there are file entries in the zip archive that, when extracted, appear in the just-created directory, then when those file entries are extracted, the last modified and last accessed times of the directory will change, as a side effect. The result is that after an extraction of a directory and a number of files within the directory, the la modified and last accessed timestamps on the directory will reflect the time that the last file was extracted into the directory, rather than the tim stored in the zip archive for the directory.

To compensate, when extracting an archive with ExtractAll, DotNetZip will extract all the file and directory entries as described above but it will then make a second pass on the directories, and reset the time on the directories to reflect what is stored in the zip archive.

This compensation is performed only within the context of an ExtractAll. If you call ZipEntry. Extract on a directory entry the timestamps on directory in the filesystem will reflect the times stored in the zip. If you then call ZipEntry. Extract on a file entry, which

is extracted into the directory, the timestamps on the directory will be updated to the current time.

Examples

This example extracts all the entries in a zip archive file, to the specifiec target directory. The extraction will overwrite any existing files silently.

```
C#

String TargetDirectory= "uppack":
```

```
String TargetDirectory= "unpack";
using(ZipFile zip= ZipFile.Read(ZipFileToExtract))
{
    zip.ExtractExistingFile= ExtractExistingFileActi
    zip.ExtractAll(TargetDirectory);
}
```

VB.NET

```
Dim TargetDirectory As String = "unpack"
Using zip As ZipFile = ZipFile.Read(ZipFileToExtract
    zip.ExtractExistingFile= ExtractExistingFileActi
    zip.ExtractAll(TargetDirectory)
End Using
```

■ See Also

<u>ExtractProgress</u> <u>ExtractExistingFile</u>

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

F

Ionic Zip Library v1.9.1.6 ExtractAll Method (path, extractExistingFile)

Reference ► Ionic.Zip ► ZipFile ► ExtractAll(String,

ExtractExistingFileAction)

Extracts all of the items in the zip archive, to the specified path in the filesystem, using the specified behavior when extraction would overwrite an existing file.

Declaration Syntax

Parameters

path (String)

The path to which the contents of the zipfile will be extracted. The path can be relative or fully-qualified.

extractExistingFile (ExtractExistingFileAction)

The action to take if extraction would overwrite an existing file.

Remarks

This method will extract all entries in the ZipFile to the specified pat For an extraction that would overwrite an existing file, the behavior is dictated by *extractExistingFile*, which overrides any setting you may have made on individual ZipEntry instances.

The action to take when an extract would overwrite an existing file applies to all entries. If you want to set this on a per-entry basis, then you must use ExtractExistingFileAction) or one of the similar methods.

Calling this method is equivalent to setting the <u>ExtractExistingFile</u> property and then calling <u>ExtractAll(String)</u>.

This method will send verbose output messages to the StatusMessageTextWriter, if it is set on the ZipFile instance.

Examples

This example extracts all the entries in a zip archive file, to the specifiec target directory. It does not overwrite any existing files.

```
C#
String TargetDirectory= "c:\\unpack";
using(ZipFile zip= ZipFile.Read(ZipFileToExtract))
{
   zip.ExtractAll(TargetDirectory, ExtractExistingFil
}
```

```
VB.NET
```

```
Dim TargetDirectory As String = "c:\unpack"
Using zip As ZipFile = ZipFile.Read(ZipFileToExtract
    zip.ExtractAll(TargetDirectory, ExtractExistingF
End Using
```

See Also

ExtractSelectedEntries(String, ExtractExistingFileAction)

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

The action the library should take when extracting a file that already exists.

Declaration Syntax

```
C# Visual Basic Visual C++
public ExtractExistingFileAction ExtractExistingFile
Public Property ExtractExistingFile As ExtractExisti
    Get
    Set

public:
property ExtractExistingFileAction ExtractExistingFileAction get ();
    void set (ExtractExistingFileAction value);
}
```

Remarks

This property affects the behavior of the Extract methods (one of the Extract() or ExtractWithPassword() overloads), when extraction would would overwrite an existing filesystem file. If you do no set this property, the library throws an exception when extracting an ent would overwrite an existing file.

This property has no effect when extracting to a stream, or when the file to be extracted does not already exist.

See Also

ExtractExistingFile

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa



Ionic Zip Library v1.9.1.6 ExtractProgress Event

Reference ► Ionic.Zip ► ZipFile ► ExtractProgress

C#

An event handler invoked before, during, and after extraction of entries the zip archive.

Declaration Syntax

```
C# Visual Basic Visual C++

public event EventHandler<ExtractProgressEventArgs>

Public Event ExtractProgress As EventHandler(Of Extr

public:
    event EventHandler<ExtractProgressEventArgs^>^ Extr
        void add (EventHandler<ExtractProgressEventArgs)

void remove (EventHandler<ExtractProgressEventArgs)

}
```

■ Remarks

Depending on the particular event, different properties on the ExtractProgressEventArgs parameter are set. The following table summarizes the available EventTypes and the conditions under which this event handler is invoked with a ExtractProgressEventArg with the given EventType.

value of EntryType	Meani
ZipProgressEventType.Extracting_BeforeExtractAll	Set wł Archiv Extrac meani
ZipProgressEventType.Extracting_AfterExtractAll	Set who The Alextrac

	meani
ZipProgressEventType.Extracting_BeforeExtractEntry	Set whe ZipFile meani Currer Entries
ZipProgressEventType.Extracting_AfterExtractEntry	Set wh ZipFile are me Entries Extrac
ZipProgressEventType.Extracting_EntryBytesWritten	Set wi in the entry. Archiv Bytes
ZipProgressEventType.Extracting_ExtractEntryWouldOverwrite	Set win the overwhele used Extron the to Invo

Examples

```
C#

private static bool justHadByteUpdate = false;
public static void ExtractProgress(object sender, Ex
{
   if(e.EventType == ZipProgressEventType.Extracting_
   {
     if (justHadByteUpdate)
        Console.SetCursorPosition(0, Console.CursorTop)
```

```
Console.Write(" \{0\}/\{1\} (\{2:N0\}\%)", e.BytesTra
                  e.BytesTransferred / (0.01 * e.Tot
    justHadByteUpdate = true;
  }
  else if(e.EventType == ZipProgressEventType.Extrac
    if (justHadByteUpdate)
      Console.WriteLine();
    Console.WriteLine("Extracting: {0}", e.CurrentEn
    justHadByteUpdate= false;
 }
}
public static ExtractZip(string zipToExtract, string
  string TargetDirectory= "extract";
  using (var zip = ZipFile.Read(zipToExtract)) {
    zip.ExtractProgress += ExtractProgress;
    foreach (var e in zip1)
      e.Extract(TargetDirectory, true);
 }
}
```

VB.NET

```
End Sub
Private Shared justHadByteUpdate As Boolean = False
Public Shared Sub MyExtractProgress(ByVal sender As
    If (e.EventType = ZipProgressEventType.Extractin
        If ExtractTest.justHadByteUpdate Then
            Console.SetCursorPosition(0, Console.Cur
        End If
                          \{0\}/\{1\} (\{2:N0\}\%)", e.Byte
        Console.Write("
        ExtractTest.justHadByteUpdate = True
    ElseIf (e.EventType = ZipProgressEventType.Extra
        If ExtractTest.justHadByteUpdate Then
            Console.WriteLine
        End If
        Console.WriteLine("Extracting: {0}", e.Curre
        ExtractTest.justHadByteUpdate = False
    End If
End Sub
```

See Also

SaveProgress ReadProgress AddProgress

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

<u>|</u>

Ionic Zip Library v1.9.1.6 ExtractSelectedEntries Method

Reference ► Ionic.Zip ► ZipFile ► ExtractSelectedEntries()

C# -

■ Members

Icon	Member	Description
≡	ExtractSelectedEntries(String)	Selects and Extracts a set of Entries from the ZipFile.
€ 🖗	ExtractSelectedEntries(String, ExtractExistingFileAction)	Selects and Extracts a set of Entries from the ZipFile.
≅ 🚱	ExtractSelectedEntries(String, String)	Selects and Extracts a set of Entries from the ZipFile.
≡	ExtractSelectedEntries(String, String, String)	Selects and Extracts a set of Entries from the ZipFile.
≅	ExtractSelectedEntries(String, String, String, ExtractExistingFileAction)	Selects and Extracts a set of Entries from the ZipFile.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractSelectedEntries Method (selectionCriteria)

Reference ► Ionic.Zip ► ZipFile ► ExtractSelectedEntries(String)

Selects and Extracts a set of Entries from the ZipFile.

C#

■ Declaration Syntax

Parameters

selectionCriteria (String)

the selection criteria for entries to extract.

■ Remarks

The entries are extracted into the current working directory.

If any of the files to be extracted already exist, then the action taken is ϵ specified in the <code>ExtractExistingFile</code> property on the corresponding ZipEntry instance. By default, the action taken in this case is to throw ar exception.

For information on the syntax of the selectionCriteria string, see

AddSelectedFiles(String).

Examples

This example shows how extract all XML files modified after 15 January 2009.

```
c#
using (ZipFile zip = ZipFile.Read(zipArchiveName))
{
  zip.ExtractSelectedEntries("name = *.xml and mti
}
```

■ See Also

ExtractSelectedEntries(String, ExtractExistingFileAction)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 ExtractSelectedEntries Method (selectionCriteria, extractExistingFile)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► ExtractSelectedEntries(String, ExtractExistingFileAction)

Selects and Extracts a set of Entries from the ZipFile.



Declaration Syntax

Parameters

selectionCriteria (String)

the selection criteria for entries to extract.

extractExistingFile (ExtractExistingFileAction)

The action to take if extraction would overwrite an existing file.

■ Remarks

The entries are extracted into the current working directory. When

extraction would would overwrite an existing filesystem file, the action taken is as specified in the *extractExistingFile* parameter.

For information on the syntax of the string describing the entry selection criteria, see AddSelectedFiles(String).

Examples

This example shows how extract all XML files modified after 15 January 2009, overwriting any existing files.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractSelectedEntries Method (selectionCriteria, directoryPathInArchive)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► ExtractSelectedEntries(String, String)

Selects and Extracts a set of Entries from the ZipFile.



Declaration Syntax

Parameters

selectionCriteria (String)

the selection criteria for entries to extract.

directoryPathInArchive (String)

the directory in the archive from which to select entries. If null, then all directories in the archive are used.

■ Remarks

The entries are selected from the specified directory within the archive, and then extracted into the current working directory.

If any of the files to be extracted already exist, then the action taken is ε specified in the <code>ExtractExistingFile</code> property on the corresponding ZipEntry instance. By default, the action taken in this case is to throw ar exception.

For information on the syntax of the string describing the entry selection criteria, see AddSelectedFiles(String).

Examples

This example shows how extract all XML files modified after 15 January 2009, and writes them to the "unpack" directory.

```
c#
using (ZipFile zip = ZipFile.Read(zipArchiveName))
{
  zip.ExtractSelectedEntries("name = *.xml and mti
}
```

See Also

ExtractSelectedEntries(String, String, String, ExtractExistingFileAction)

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractSelectedEntries Method (selectionCriteria, directoryInArchive, extractDirectory)

Reference ► Ionic.Zip ► ZipFile ► ExtractSelectedEntries(String, String, String)

Selects and Extracts a set of Entries from the ZipFile.

C#

Declaration Syntax

```
C# Visual Basic Visual C++

public void ExtractSelectedEntries(
    string selectionCriteria,
    string directoryInArchive,
    string extractDirectory
)
```

```
Public Sub ExtractSelectedEntries ( _ selectionCriteria As String, _ directoryInArchive As String, _ extractDirectory As String _ )
```

Parameters

selectionCriteria (String)

the selection criteria for entries to extract.

directoryInArchive (String)

the directory in the archive from which to select entries. If null, then

all directories in the archive are used.

extractDirectory (String)

the directory on the disk into which to extract. It will be created if it does not exist.

■ Remarks

The entries are extracted into the specified directory. If any of the files to be extracted already exist, an exception will be thrown.

For information on the syntax of the string describing the entry selection criteria, see AddSelectedFiles(String).

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ExtractSelectedEntries Method (selectionCriteria, directoryPathInArchive, extractDirectory, extractExistingFile)

Reference ► <u>lonic.Zip</u> ► <u>ZipFile</u> ► ExtractSelectedEntries(String, String, String, ExtractExistingFileAction)

Selects and Extracts a set of Entries from the ZipFile.

C#

■ Declaration Syntax

```
Public Sub ExtractSelectedEntries ( _ selectionCriteria As String, _ directoryPathInArchive As String, _ extractDirectory As String, _ extractExistingFile As ExtractExistingFileAc
)
```

Parameters

selectionCriteria (String)

the selection criteria for entries to extract.

directoryPathInArchive (String)

The directory in the archive from which to select entries. If null, ther all directories in the archive are used.

extractDirectory (String)

The directory on the disk into which to extract. It will be created if it does not exist.

extractExistingFile (ExtractExistingFileAction)

The action to take if extraction would overwrite an existing file.

Remarks

The entries are extracted into the specified directory. When extraction would would overwrite an existing filesystem file, the action taken is as specified in the *extractExistingFile* parameter.

For information on the syntax of the string describing the entry selection criteria, see AddSelectedFiles(String).

Examples

This example shows how extract all files with an XML extension or with size larger than 100,000 bytes, and puts them in the unpack directory. For any files that already exist in that destination directory, they will not be overwritten.

```
C# Cop
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Fix Zip Directory Method

(zipFileName)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► FixZipDirectory(String)

Rewrite the directory within a zipfile.

■ Declaration Syntax

C#

Parameters

zipFileName (String)

The filename to of the zip file to fix.

■ Remarks

In cases of data error, the directory in a zip file can get out of synch with the entries in the zip file. This method attempts to fix the zip file if this had occurred.

This can take a long time for large zip files.

This won't work if the zip file uses a non-standard code page - neither IBM437 nor UTF-8.

This method is not supported in the Reduced or Compact Framework versions of DotNetZip.

Developers using COM can use the <u>ComHelper.FixZipDirectory(String)</u> method.

■ See Also

CheckZip(String)
CheckZip(String, Boolean, TextWriter)

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Flatten Folders On Extract Property

Reference ► Ionic.Zip ► ZipFile ► FlattenFoldersOnExtract

Indicates whether extracted files should keep their paths as stored in the zip archive.

Declaration Syntax

■ Remarks

This property affects Extraction. It is not used when creating zip archive

With this property set to false, the default, extracting entries from a z file will create files in the filesystem that have the full path associated to the entry within the zip file. With this property set to true, extracting entries from the zip file results in files with no path: the folders are "flattened."

An example: suppose the zip file contains entries /directory1/file1.txt an /directory2/file2.txt. With FlattenFoldersOnExtract set to false the files created will be \directory1\file1.txt and \directory2\file2.txt. With the property set to true, the files created are file1.txt and file2.txt.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
FullScan Property

Reference ► Ionic.Zip ► ZipFile ► FullScan
C#
```

Indicates whether to perform a full scan of the zip file when reading it.

Declaration Syntax

■ Remarks

You almost never want to use this property.

When reading a zip file, if this flag is true (True in VB), the entire zip archive will be scanned and searched for entries. For large archives, thi can take a very, long time. The much more efficient default behavior is t read the zip directory, which is stored at the end of the zip file. But, in some cases the directory is corrupted and you need to perform a full scan of the zip file to determine the contents of the zip file. This property lets you do that, when necessary.

This flag is effective only when calling Initialize(String). Normally you would read a ZipFile with the static ZipFile.Read method. But you can't set the Fullscan property on the ZipFile instance when you use static factory method like ZipFile. Read.

Examples

This example shows how to read a zip file using the full scan approach, and then save it, thereby producing a corrected zip file.

C# □ Cop

```
using (var zip = new ZipFile())
{
    zip.FullScan = true;
    zip.Initialize(zipFileName);
    zip.Save(newName);
}
```

VB.NET

```
Using zip As New ZipFile
    zip.FullScan = True
    zip.Initialize(zipFileName)
    zip.Save(newName)
End Using
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Generic IEnumerator support, for use of a ZipFile in an enumeration.

Declaration Syntax

```
C# Visual Basic Visual C++
public IEnumerator<ZipEntry> GetEnumerator()

Public Function GetEnumerator As IEnumerator(Of ZipE

public:
virtual IEnumerator<ZipEntry^>^ GetEnumerator() seal
```

Return Value

A generic enumerator suitable for use within a foreach loop.

Remarks

You probably do not want to call **GetEnumerator** explicitly. Instead is implicitly called when you use a foreach loop in C#, or a **For Each** loop in VB.NET.

Examples

This example reads a zipfile of a given name, then enumerates the entries in that zip file, and displays the information about each entry on the Console.

```
Using (ZipFile zip = ZipFile.Read(zipfile))
{
  bool header = true;
  foreach (ZipEntry e in zip)
  {
    if (header)
```

```
{
       System.Console.WriteLine("Zipfile: {0}", zip.
       System.Console.WriteLine("Version Needed: 0x{
       System.Console.WriteLine("BitField: 0x{0:X2}"
       System.Console.WriteLine("Compression Method:
       System.Console.WriteLine("\n{1, -22} {2, -6} {3
                     "Filename", "Modified", "Size"
       System.Console.WriteLine(new System.String('-
       header = false;
    }
    System.Console.WriteLine("\{1, -22\} \{2, -6\} \{3, 4: F0\}
                e.FileName,
                 e.LastModified.ToString("yyyy-MM-dd
                 e.UncompressedSize,
                 e.CompressionRatio,
                 e.CompressedSize);
    e.Extract();
}
```

VB.NET

```
Console.WriteLine("{1,-22} {2,-6} {3,4:F0}%
e.FileName, _
e.LastModified.ToString("yyyy-MM-dd HH:mm:
e.UncompressedSize, _
e.CompressionRatio, _
e.CompressedSize )
e.Extract
Next
End Using
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

|

4

Ionic Zip Library v1.9.1.6 **GetNewEnum Method**

Reference ► Ionic.Zip ► ZipFile ► GetNewEnum()

C# -

An IEnumerator, for use of a ZipFile in a foreach construct.

Declaration Syntax

```
C# Visual Basic Visual C++

public IEnumerator GetNewEnum()

Public Function GetNewEnum As IEnumerator

public:
IEnumerator^ GetNewEnum()
```

■ Return Value

The IEnumerator over the entries in the ZipFile.

Remarks

This method is included for COM support. An application generally does not call this method directly. It is called implicitly by COM clients when enumerating the entries in the ZipFile instance. In VBScript, this is done with a For Each statement. In Javascript, this is done with new Enumerator(zipfile).

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Info
Property

Reference
► Ionic.Zip
► ZipFile
► Info

C#
```

Provides a human-readable string with information about the ZipFile.

Declaration Syntax

```
C# Visual Basic Visual C++

public string Info { get; }

Public ReadOnly Property Info As String
    Get

public:
property String^ Info {
    String^ get ();
}
```

Remarks

The information string contains 10 lines or so, about each ZipEntry, describing whether encryption is in use, the compressed and uncompressed length of the entry, the offset of the entry, and so on. As result the information string can be very long for zip files that contain many entries.

This information is mostly useful for diagnostic purposes.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

| Ionic Zip Library v1.9.1.6 | Initialize Method (fileName) | Reference ► Ionic.Zip ► ZipFile ► Initialize(String) | C#

Initialize a ZipFile instance by reading in a zip file.

Declaration Syntax

Parameters

fileName (String)

the name of the existing zip file to read in.

Remarks

This method is primarily useful from COM Automation environments, when reading or extracting zip files. In COM, it is not possible to invoke parameterized constructors for a class. A COM Automation application can update a zip file by using the <u>default (no argument) constructor</u>, the calling Initialize() to read the contents of an on-disk zip archive into the ZipFile instance.

.NET applications are encouraged to use the ZipFile.Read() methods for better clarity.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 InputUsesZip64 Property

Reference ► Ionic.Zip ► ZipFile ► InputUsesZip64

C#

▶

Indicates whether the most recent Read () operation read a zip file the uses ZIP64 extensions.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public Nullable<bool> InputUsesZip64 { get; }

Public ReadOnly Property InputUsesZip64 As Nullable(
    Get

public:
property Nullable<bool> InputUsesZip64 {
    Nullable<bool> get ();
}
```

■ Remarks

This property will return null (Nothing in VB) if you've added an entry aft reading the zip file.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Is Zip File Method

Reference ► Ionic.Zip ► ZipFile ► IsZipFile()

C#

■ Members

Icon	Member	Description
= \S	IsZipFile(String)	Checks the given file to see if it appears to be a valid zip file.
= \S	IsZipFile(String, Boolean)	Checks a file to see if it is a valid zip file.
= \$	IsZipFile(Stream, Boolean)	Checks a stream to see if it contains valid zip archive.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Is Zip File Method (stream,

testExtract)

Reference ► Ionic.Zip ► ZipFile ► IsZipFile(Stream, Boolean)

Checks a stream to see if it contains a valid zip archive.

C# <u>▼</u>

Declaration Syntax

```
C# Visual Basic Visual C++

public static bool IsZipFile(
    Stream stream,
    bool testExtract
)
```

```
Public Shared Function IsZipFile ( _ stream As Stream, _ testExtract As Boolean _ ) As Boolean
```

Parameters

stream (Stream)

The stream to check.

testExtract (Boolean)

true if the caller wants to extract each entry.

■ Return Value

true if the stream contains a valid zip archive.

Remarks

This method reads the zip archive contained in the specified stream, verifying the ZIP metadata as it reads. If testExtract is true, this method also extracts each entry in the archive, dumping all the bits into Null.

If everything succeeds, then the method returns true. If anything fails - f example if an incorrect signature or CRC is found, indicating a corrupt file, the the method returns false. This method also returns false for a fil that does not exist.

If testExtract is true, this method reads in the content for each entry, expands it, and checks CRCs. This provides an additional check beyond verifying the zip header data.

If testExtract is true, and if any of the zip entries are protected wit a password, this method will return false. If you want to verify a ZipFile that has entries which are protected with a password, you will need to d that manually.

■ See Also

IsZipFile(String, Boolean)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | SZipFile Method (fileName) | Reference | Ionic.Zip | ZipFile | IsZipFile(String) | C# | □

Checks the given file to see if it appears to be a valid zip file.

Declaration Syntax

Parameters

fileName (String)

The file to check.

■ Return Value

true if the file appears to be a zip file.

■ Remarks

Calling this method is equivalent to calling <u>IsZipFile(String, Boolean)</u> wit the testExtract parameter set to false.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Is Zip File Method (file Name,

testExtract)

Reference ► Ionic.Zip ► ZipFile ► IsZipFile(String, Boolean)

Checks a file to see if it is a valid zip file.

C# -

Declaration Syntax

```
C# Visual Basic Visual C++

public static bool IsZipFile(
    string fileName,
    bool testExtract
)
```

Parameters

fileName (String)

The zip file to check.

testExtract (Boolean)

true if the caller wants to extract each entry.

Return Value

true if the file contains a valid zip file.

■ Remarks

This method opens the specified zip file, reads in the zip archive, verifying the ZIP metadata as it reads.

If everything succeeds, then the method returns true. If anything fails - f example if an incorrect signature or CRC is found, indicating a corrupt file, the the method returns false. This method also returns false for a fil that does not exist.

If *testExtract* is true, as part of its check, this method reads in the content for each entry, expands it, and checks CRCs. This provides an additional check beyond verifying the zip header and directory data.

If *testExtract* is true, and if any of the zip entries are protected with a password, this method will return false. If you want to verify a ZipFilt that has entries which are protected with a password, you will need to d that manually.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

■ Members

Icon	Member	Description
	Item[Int32]	This is an integer indexer into the Zip archive.
	Item[String]	This is a name-based indexer into the Zip archive.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

```
| Ionic Zip Library v1.9.1.6 | Item Property (ix)
| Reference ► | Ionic.Zip ► | ZipFile ► | Item[Int32] | C#
```

This is an integer indexer into the Zip archive.

Declaration Syntax

```
C# Visual Basic Visual C++

public ZipEntry this[
    int ix
] { get; }

Public ReadOnly Default Property Item ( _
    ix As Integer _
) As ZipEntry
    Get

public:
property ZipEntry^ default[int ix] {
    ZipEntry^ get (int ix);
}
```

Parameters

ix (<u>Int32</u>)

Return Value

The ZipEntry within the Zip archive at the specified index. If the entr does not exist in the archive, this indexer throws.

■ Remarks

This property is read-only.

Internally, the ZipEntry instances that belong to the ZipFile are stored in a Dictionary. When you use this indexer the first time, it create a read-only List<ZipEntry> from the Dictionary. Values Collection

If at any time you modify the set of entries in the ZipFile, either by adding an entry, removing an entry, or renaming an entry, a new List wil be created, and the numeric indexes for the remaining entries may be different.

This means you cannot rename any ZipEntry from inside an enumeratic of the zip file.

The index value.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | Item Property (fileName) | Reference ► Ionic.Zip ► ZipFile ► Item[String] | C#

This is a name-based indexer into the Zip archive.

Declaration Syntax

Parameters

fileName (String)

The name of the file, including any directory path, to retrieve from the zip. The filename match is not case-sensitive by default; you can use the <u>CaseSensitiveRetrieval</u> property to change this behavior. The pathname can use forward-slashes or backward slashes.

■ Return Value

The ZipEntry within the Zip archive, given by the specified filename. If the named entry does not exist in the archive, this indexer returns null (Nothing in VB).

Remarks

This property is read-only.

The <u>CaseSensitiveRetrieval</u> property on the <u>ZipFile</u> determines whether retrieval via this indexer is done via case-sensitive comparison. By default, retrieval is not case sensitive. This makes sense on Window in which filesystems are not case sensitive.

Regardless of case-sensitivity, it is not always the case that this[value].FileName == value. In other words, the FileName property of the ZipEntry retrieved with this indexer, ma or may not be equal to the index value.

This is because DotNetZip performs a normalization of filenames passe to this indexer, before attempting to retrieve the item. That normalizatior includes: removal of a volume letter and colon, swapping backward slashes for forward slashes. So,

```
zip["dir1\\entry1.txt"].FileName ==
"dir1/entry.txt".
```

Directory entries in the zip file may be retrieved via this indexer only with names that have a trailing slash. DotNetZip automatically appends a trailing slash to the names of any directory entries added to a zip.

Examples

This example extracts only the entries in a zip file that are .txt files.

```
using (ZipFile zip = ZipFile.Read("PackedDocuments.z
{
  foreach (string s1 in zip.EntryFilenames)
  {
    if (s1.EndsWith(".txt"))
        zip[s1].Extract("textfiles");
    }
}
```

VB.NET

Exceptions

Exception	Condition
ArgumentException	Thrown if the caller attempts to assign a non-null value to the indexer.

See Also

RemoveEntry(String)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Library Version Property

Reference ► Ionic.Zip ► ZipFile ► LibraryVersion

C#

Returns the version number on the DotNetZip assembly.

■ Declaration Syntax

Remarks

This property is exposed as a convenience. Callers could also get the version value by retrieving GetName(). Version on the System. Reflection. Assembly object pointing to the DotNetZip assembly. But sometimes it is not clear which assembly is being loaded. This property makes it clear.

This static property is primarily useful for diagnostic purposes.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 MaxOutputSegmentSize Property

Reference ► Ionic.Zip ► ZipFile ► MaxOutputSegmentSize

The maximum size of an output segment, when saving a split Zip file.

Declaration Syntax

```
C# Visual Basic Visual C++
public int MaxOutputSegmentSize { get; set; }

Public Property MaxOutputSegmentSize As Integer
    Get
    Set

public:
property int MaxOutputSegmentSize {
    int get ();
    void set (int value);
}
```

■ Remarks

Set this to a non-zero value before calling <u>Save()</u> or <u>Save(String)</u> to specify that the ZipFile should be saved as a split archive, also sometimes called a spanned archive. Some also call them multi-file archives.

A split zip archive is saved in a set of discrete filesystem files, rather that in a single file. This is handy when transmitting the archive in email or some other mechanism that has a limit to the size of each file. The first file in a split archive will be named <code>basename.z01</code>, the second will be named <code>basename.z02</code>, and so on. The final file is named <code>basename.zip</code>. According to the zip specification from PKWare, the minimum value is 65536, for a 64k segment size. The maximum numbe of segments allows in a split archive is 99.

The value of this property determines the maximum size of a split segment when writing a split archive. For example, suppose you have a ZipFile that would save to a single file of 200k. If you set the MaxOutputSegmentSize to 65536 before calling Save(), you w get four distinct output files. On the other hand if you set this property to 256k, then you will get a single-file archive for that ZipFile.

The size of each split output file will be as large as possible, up to the maximum size set here. The zip specification requires that some data fields in a zip archive may not span a split boundary, and an output segment may be smaller than the maximum if necessary to avoid that problem. Also, obviously the final segment of the archive may be smalle than the maximum segment size. Segments will never be larger than the value set with this property.

You can save a split Zip file only when saving to a regular filesystem file It's not possible to save a split zip file as a self-extracting archive, nor is possible to save a split zip file to a stream. When saving to a SFX or to Stream, this property is ignored.

About interoperability: Split or spanned zip files produced by DotNetZip can be read by WinZip or PKZip, and vice-versa. Segmented zip files may not be readable by other tools, if those other tools don't support zip spanning or splitting. When in doubt, test. I don't believe Windows Explorer can extract a split archive.

This property has no effect when reading a split archive. You can read ε split archive in the normal way with DotNetZip.

When saving a zip file, if you want a regular zip file rather than a split zil file, don't set this property, or set it to Zero.

If you read a split archive, with Read(String) and then subsequently call ZipFile.Save(), unless you set this property before calling Save(), you will get a normal, single-file archive.

See Also

NumberOfSegmentsForMostRecentSave

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Name Property

Reference ► Ionic.Zip ► ZipFile ► Name
C#
```

The name of the ZipFile, on disk.

Declaration Syntax

Remarks

When the ZipFile instance was created by reading an archive using one of the ZipFile. Read methods, this property represents the name of the zip file that was read. When the ZipFile instance was created by using the no-argument constructor, this value is null (Nothing in VB).

If you use the no-argument constructor, and you then explicitly set this property, when you call <u>Save()</u>, this name will specify the name of the zi file created. Doing so is equivalent to calling <u>Save(String)</u>. When instantiating a <u>ZipFile</u> by reading from a stream or byte array, the <u>Name</u> property remains <u>null</u>. When saving to a stream, the <u>Name</u> property is implicitly set to <u>null</u>.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library

v1.9.1.6 Number Of Segments For Most Recent Save Property

Reference ► Ionic.Zip ► ZipFile ►

NumberOfSegmentsForMostRecentSave

Returns the number of segments used in the most recent Save() operation.

Declaration Syntax

```
C# Visual Basic Visual C++
public int NumberOfSegmentsForMostRecentSave { get;

Public ReadOnly Property NumberOfSegmentsForMostRece Get

public:
property int NumberOfSegmentsForMostRecentSave {
    int get ();
}
```

■ Remarks

This is normally zero, unless you have set the <u>MaxOutputSegmentSize</u> property. If you have set <u>MaxOutputSegmentSize</u>, and then you save a file, after the call to Save() completes, you can read this value to learn the number of segments that were created.

If you call Save("Archive.zip"), and it creates 5 segments, then you will have filesystem files named Archive.z01, Archive.z02, Archive.z03, Archive.z04, and Archive.zip, and the value of this property will be 5.

■ See Also

MaxOutputSegmentSize

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

√

Indicates whether the most recent Save () operation used ZIP64 extensions.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public Nullable < bool > OutputUsedZip64 { get; }

Public ReadOnly Property OutputUsedZip64 As Nullable Get

public:
property Nullable < bool > OutputUsedZip64 {
    Nullable < bool > get ();
}
```

■ Remarks

The use of ZIP64 extensions within an archive is not always necessary, and for interoperability concerns, it may be desired to NOT use ZIP64 if possible. The UseZip64WhenSaving property can be set to use ZIP64 extensions only when necessary. In those cases, Sometimes application want to know whether a Save() actually used ZIP64 extensions. Applications can query this read-only property to learn whether ZIP64 has been used in a just-saved ZipFile.

The value is null (or Nothing in VB) if the archive has not been saved.

Non-null values (HasValue is true) indicate whether ZIP64 extensions were used during the most recent Save() operation. The ZIP64 extensions may have been used as required by any particular entry because of its uncompressed or compressed size, or because the

archive is larger than 4294967295 bytes, or because there are more the 65534 entries in the archive, or because the UseZip64WhenSavin property was set to Always, or because the UseZip64WhenSaving property was set to AsNecessary and the output stream was not seekable. The value of this property does not indicate the reason the ZIP64 extensions were used.

See Also

<u>UseZip64WhenSaving</u> <u>RequiresZip64</u>

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel Deflate Max Buffer Pairs

Property

Reference ► Ionic.Zip ► ZipFile ► ParallelDeflateMaxBufferPairs

The maximum number of buffer pairs to use when performing parallel compression.

Declaration Syntax

Remarks

This property sets an upper limit on the number of memory buffer pairs create when performing parallel compression. The implementation of th parallel compression stream allocates multiple buffers to facilitate parall compression. As each buffer fills up, the stream uses ThreadPool.QueueUserWorkItem() to compress those buffers in a background threadpool thread. After a buffer is compressed, it is reordered and written to the output stream.

A higher number of buffer pairs enables a higher degree of parallelism, which tends to increase the speed of compression on multi-cpu computers. On the other hand, a higher number of buffer pairs also implies a larger memory consumption, more active worker threads, and

higher cpu utilization for any compression. This property enables the application to limit its memory consumption and CPU utilization behavio depending on requirements.

For each compression "task" that occurs in parallel, there are 2 buffers allocated: one for input and one for output. This property sets a limit for the number of pairs. The total amount of storage space allocated for buffering will then be (N*S*2), where N is the number of buffer pairs, S i the size of each buffer (BufferSize). By default, DotNetZip allocates 4 buffer pairs per CPU core, so if your machine has 4 cores, and you reta the default buffer size of 128k, then the ParallelDeflateOutputStream wi use 4 * 4 * 2 * 128kb of buffer memory in total, or 4mb, in blocks of 128kb. If you then set this property to 8, then the number will be 8 * 2 * 128kb of buffer memory, or 2mb.

CPU utilization will also go up with additional buffers, because a larger number of buffer pairs allows a larger number of background threads to compress in parallel. If you find that parallel compression is consuming too much memory or CPU, you can adjust this value downward.

The default value is 16. Different values may deliver better or worse results, depending on your priorities and the dynamic performance characteristics of your storage and compute resources.

This property is not the number of buffer pairs to use; it is an upper limit An illustration: Suppose you have an application that uses the default value of this property (which is 16), and it runs on a machine with 2 CPL cores. In that case, DotNetZip will allocate 4 buffer pairs per CPU core, for a total of 8 pairs. The upper limit specified by this property has no effect.

The application can set this value at any time before calling ZipFile.Save().

See Also

ParallelDeflateThreshold

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel Deflate Threshold Property

Reference ► Ionic.Zip ► ZipFile ► ParallelDeflateThreshold

The size threshold for an entry, above which a parallel deflate is used.

Declaration Syntax

Remarks

DotNetZip will use multiple threads to compress any ZipEntry, if the entries larger than the given size. Zero means "always use parallel deflate", while -1 means "never use parallel deflate". The default value for this property is 512k. Aside from the special values of 0 and 1, the minimum value is 65536.

If the entry size cannot be known before compression, as with a readforward stream, then Parallel deflate will never be performed, unless the value of this property is zero.

A parallel deflate operations will speed up the compression of large files on computers with multiple CPUs or multiple CPU cores. For files above 1mb, on a dual core or dual-cpu (2p) machine, the time required to compress the file can be 70% of the single-threaded deflate. For very large files on 4p machines the compression can be done in 30% of the

normal time. The downside is that parallel deflate consumes extra memory during the deflate, and the deflation is not as effective.

Parallel deflate tends to yield slightly less compression when compared to as single-threaded deflate; this is because the original data stream is split into multiple independent buffers, each of which is compressed in parallel. But because they are treated independently, there is no opportunity to share compression dictionaries. For that reason, a deflate stream may be slightly larger when compressed using parallel deflate, ϵ compared to a traditional single-threaded deflate. Sometimes the increase over the normal deflate is as much as 5% of the total compressed size. For larger files it can be as small as 0.1%.

Multi-threaded compression does not give as much an advantage when using Encryption. This is primarily because encryption tends to slow down the entire pipeline. Also, multi-threaded compression gives less of an advantage when using lower compression levels, for example BestSpeed. You may have to perform some tests to determine the best approach for your situation.

See Also

ParallelDeflateMaxBufferPairs

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Password Property

Reference
► Ionic.Zip
► ZipFile
► Password

C#
```

Sets the password to be used on the **ZipFile** instance.

Declaration Syntax

■ Remarks

When writing a zip archive, this password is applied to the entries, not to the zip archive itself. It applies to any ZipEntry subsequently added the ZipFile, using one of the AddFile, AddDirectory, AddEntry, or AddItem methods, etc. When reading a zip archive, this property applies to any entry subsequently extracted from the ZipFile using one of the Extract methods on the ZipFile class.

When writing a zip archive, keep this in mind: though the password is so on the ZipFile object, according to the Zip spec, the "directory" of the archive - in other words the list of entries or files contained in the archive - is not encrypted with the password, or protected in any way. If you set the Password property, the password actually applies to individual entrie that are added to the archive, subsequent to the setting of this property. The list of filenames in the archive that is eventually created will appear

in clear text, but the contents of the individual files are encrypted. This is how Zip encryption works.

One simple way around this limitation is to simply double-wrap sensitive filenames: Store the files in a zip file, and then store that zip file within a second, "outer" zip file. If you apply a password to the outer zip file, ther readers will be able to see that the outer zip file contains an inner zip file But readers will not be able to read the directory or file list of the inner z file.

If you set the password on the <code>ZipFile</code>, and then add a set of files to the archive, then each entry is encrypted with that password. You may also want to change the password between adding different entries. If you set the password, add an entry, then set the password to <code>null</code> (<code>Nothing</code> in VB), and add another entry, the first entry is encrypted ar the second is not. If you call <code>AddFile()</code>, then set the <code>Password</code> property, then call <code>ZipFile.Save</code>, the file added will not be password-protected, and no warning will be generated.

When setting the Password, you may also want to explicitly set the Encryption property, to specify how to encrypt the entries added to the ZipFile. If you set the Password to a non-null value and do not set Encryption, then PKZip 2.0 ("Weak") encryption is used. This encryptior is relatively weak but is very interoperable. If you set the password to a null value (Nothing in VB), Encryption is reset to None.

All of the preceding applies to writing zip archives, in other words when you use one of the Save methods. To use this property when reading or an existing ZipFile, do the following: set the Password property on the ZipFile, then call one of the Extract() overloads on the ZipEntry. In this case, the entry is extracted using the Password that is specified the ZipFile instance. If you have not set the Password property, then the password is null, and the entry is extracted with no password.

If you set the Password property on the ZipFile, then call Extract() an entry that has not been encrypted with a password, the password is not used for that entry, and the ZipEntry is extracted as

normal. In other words, the password is used only if necessary.

The ZipEntry class also has a Password property. It takes precedence over this property on the ZipFile. Typically, you would use the perentry Password when most entries in the zip archive use one password and a few entries use a different password. If all entries in the zip file us the same password, then it is simpler to just set this property on the ZipFile itself, whether creating a zip archive or extracting a zip archive.

Examples

This example creates a zip file, using password protection for the entrie and then extracts the entries from the zip file. When creating the zip file the Readme.txt file is not protected with a password, but the other two are password-protected as they are saved. During extraction, each file i extracted with the appropriate password.

C#

```
// create a file with encryption
using (ZipFile zip = new ZipFile())
{
    zip.AddFile("ReadMe.txt");
    zip.Password= "!Secret1";
    zip.AddFile("MapToTheSite-7440-N49th.png");
    zip.AddFile("2008-Regional-Sales-Report.pdf");
    zip.Save("EncryptedArchive.zip");
}

// extract entries that use encryption
using (ZipFile zip = ZipFile.Read("EncryptedArchive.
{
    zip.Password= "!Secret1";
    zip.ExtractAll("extractDir");
}
```

VB.NET

```
Using zip As New ZipFile
    zip.AddFile("ReadMe.txt")
    zip.Password = "123456!"
    zip.AddFile("MapToTheSite-7440-N49th.png")
    zip.Password= "!Secret1";
    zip.AddFile("2008-Regional-Sales-Report.pdf")
    zip.Save("EncryptedArchive.zip")
End Using

' extract entries that use encryption
Using (zip as ZipFile = ZipFile.Read("EncryptedArchizip.Password= "!Secret1"
    zip.Password= "!Secret1"
    zip.ExtractAll("extractDir")
End Using
```

■ See Also

ZipFile.Encryption ZipEntry.Password

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 Provisional Alternate Encoding

Property

Reference ► Ionic.Zip ► ZipFile ► ProvisionalAlternateEncoding

The text encoding to use when writing new entries to the ZipFile, for those entries that cannot be encoded with the default (IBM437) encoding or, the text encoding that was used when reading the entries from the ZipFile.

Declaration Syntax

```
C# Visual Basic Visual C++

[ObsoleteAttribute("use AlternateEncoding instead.")
public Encoding ProvisionalAlternateEncoding { get;

<ObsoleteAttribute("use AlternateEncoding instead.")
Public Property ProvisionalAlternateEncoding As Encodet
Set
```

■ Remarks

In <u>its zip specification</u>, PKWare describes two options for encoding filenames and comments: using IBM437 or UTF-8. But, some archiving tools or libraries do not follow the specification, and instead encode characters using the system default code page. For example, WinRAR when run on a machine in Shanghai may encode filenames with the Big 5 Chinese (950) code page. This behavior is contrary to the Zip specification, but it occurs anyway.

When using DotNetZip to write zip archives that will be read by one of these other archivers, set this property to specify the code page to use when encoding the FileName and Comment for each ZipEntry in the zip file, for values that cannot be encoded with the default codepage for zip files, IBM437. This is why this property is "provisional". In all cases, IBM437 is used where possible, in other words, where no loss of data would result. It is possible, therefore, to have a given entry with a Comment encoded in IBM437 and a FileName encoded with the specified "provisional" codepage.

Be aware that a zip file created after you've explicitly set the **ProvisionalAlternateEncoding** property to a value other than IBM437 may not be compliant to the PKWare specification, and may not be readable by compliant archivers. On the other hand, many (most?) archivers are non-compliant and can read zip files created in arbitrary code pages. The trick is to use or specify the proper codepage when reading the zip.

When creating a zip archive using this library, it is possible to change th value of **ProvisionalAlternateEncoding** between each entry you add, and between adding entries and the call to Save(). Don't do this. It wi likely result in a zipfile that is not readable. For best interoperability, eith leave **ProvisionalAlternateEncoding** alone, or specify it only once, before adding any entries to the ZipFile instance. There is one exception to this recommendation, described later.

When using an arbitrary, non-UTF8 code page for encoding, there is no standard way for the creator application - whether DotNetZip, WinZip, WinRar, or something else - to formally specify in the zip file which codepage has been used for the entries. As a result, readers of zip files are not able to inspect the zip file and determine the codepage that was used for the entries contained within it. It is left to the application or user to determine the necessary codepage when reading zip files encoded this way. In other words, if you explicitly specify the codepage when you create the zipfile, you must explicitly specify the same codepage when reading the zipfile.

The way you specify the code page to use when reading a zip file varies

depending on the tool or library you use to read the zip. In DotNetZip, you use a ZipFile.Read() method that accepts an encoding parameter. It isn possible with Windows Explorer, as far as I know, to specify an explicit codepage to use when reading a zip. If you use an incorrect codepage when reading a zipfile, you will get entries with filenames that are incorrect, and the incorrect filenames may even contain characters that are not legal for use within filenames in Windows. Extracting entries with illegal characters in the filenames will lead to exceptions. It's too bad, but this is just the way things are with code pages in zip files. Caveat Empto

Example: Suppose you create a zipfile that contains entries with filenames that have Danish characters. If you use **ProvisionalAlternateEncoding** equal to "iso-8859-1" (cp 28591), the filenames will be correctly encoded in the zip. But, to read that zipfile correctly, you have to specify the same codepage at the time you read if try to read that zip file with Windows Explorer or another application that is not flexible with respect to the codepage used to decode filename in zipfiles, you will get a filename like "Inf°.txt".

When using DotNetZip to read a zip archive, and the zip archive uses a arbitrary code page, you must specify the encoding to use before or when the Zipfile is READ. This means you must use a ZipFile.Read() method that allows you to specify a System.Text.Encoding parameter. Setting the ProvisionalAlternateEncoding property after your application has read ir the zip archive will not affect the entry names of entries that have alread been read in.

And now, the exception to the rule described above. One strategy for specifying the code page for a given zip file is to describe the code page in a human-readable form in the Zip comment. For example, the comment may read "Entries in this archive are encoded in the Big5 code page". For maximum interoperability, the zip comment in this case shou be encoded in the default, IBM437 code page. In this case, the zip comment is encoded using a different page than the filenames. To do this, Specify ProvisionalAlternateEncoding to your desired region-specific code page, once before adding any entries, and then reset ProvisionalAlternateEncoding to IBM437 before

setting the **Comment** property and calling Save().

Examples

This example shows how to read a zip file using the Big-5 Chinese code page (950), and extract each entry in the zip file. For this code to work a desired, the <code>Zipfile</code> must have been created using the big5 code page (CP950). This is typical, for example, when using WinRar on a machine with CP950 set as the default code page. In that case, the names of entries within the Zip archive will be stored in that code page, and reading the zip archive must be done using that code page. If the application did not use the correct code page in <code>ZipFile.Read()</code>, then names of entries within the zip archive would not be correctly retrieved.

```
using (var zip = ZipFile.Read(zipFileName, System.Te
{
    // retrieve and extract an entry using a name en
    zip[MyDesiredEntry].Extract("unpack");
}
```

VB.NET

See Also

DefaultEncoding

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Read Method

Reference ► Ionic.Zip ► ZipFile ► Read()

C#

■ Members

Icon	Member	Description
= \S	Read(String)	Reads a zip file archive and returns the instance.
=♦5	Read(String, ReadOptions)	Reads a zip file archive from the named filesystem file using the specified options.
= \S	Read(Stream)	Reads a zip archive from a stream.
= \\$	Read(Stream, ReadOptions)	Reads a zip file archive from the give stream using the specified options.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 Read Method (zipStream) | Reference ► Ionic.Zip ► ZipFile ► Read(Stream) | C#

Reads a zip archive from a stream.

Declaration Syntax

Parameters

zipStream (Stream)

the stream containing the zip data.

■ Return Value

The ZipFile instance read from the stream

■ Remarks

When reading from a file, it's probably easier to just use ZipFile.Read(String, ReadOptions). This overload is useful when when the zip archive content is available from an already-open stream. The stream must be open and readable and seekable when calling this method. The stream is left open when the reading is completed.

Using this overload, the stream is read using the default System.Text.Encoding, which is the IBM437 codepage. If you want to specify the encoding to use when reading the zipfile content, se ZipFile.Read(Stream, ReadOptions). This

Reading of zip content begins at the current position in the stream. This means if you have a stream that concatenates regular data and zip data if you position the open, readable stream at the start of the zip data, you will be able to read the zip archive using this constructor, or any of the ZipFile constructors that accept a <u>Stream</u> as input. Some examples of where this might be useful: the zip content is concatenated at the end o a regular EXE file, as some self-extracting archives do. (Note: SFX files produced by DotNetZip do not work this way; they can be read as norm ZIP files). Another example might be a stream being read from a database, where the zip content is embedded within an aggregate strea of data.

Examples

This example shows how to Read zip content from a stream, and extrace one entry into a different stream. In this example, the filename "NameOfEntryInArchive.doc", refers only to the name of the entry within the zip archive. A file by that name is not created in the filesystem. The I/O is done strictly with the given streams.

```
c#
using (ZipFile zip = ZipFile.Read(InputStream))
{
   zip.Extract("NameOfEntryInArchive.doc", OutputStr
}
```

VB.NET

```
Using zip as ZipFile = ZipFile.Read(InputStream)
  zip.Extract("NameOfEntryInArchive.doc", OutputStr
End Using
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 Read Method (zipStream, options)

Reference ► Ionic.Zip ► ZipFile ► Read(Stream, ReadOptions)

Reads a zip file archive from the given stream using the specified options.

Declaration Syntax

```
C# Visual Basic Visual C++

public static ZipFile Read(
    Stream zipStream,
    ReadOptions options
)
```

```
Public Shared Function Read ( _ zipStream As Stream, _ options As ReadOptions _ ) As ZipFile
```

Parameters

zipStream (Stream)

the stream containing the zip data.

options (ReadOptions)

The set of options to use when reading the zip file.

Return Value

The ZipFile instance read from the stream.

■ Remarks

When reading from a file, it's probably easier to just use ZipFile.Read(String, ReadOptions). This overload is useful when when the zip archive content is available from an already-open stream. The stream must be open and readable and seekable when calling this method. The stream is left open when the reading is completed.

Reading of zip content begins at the current position in the stream. This means if you have a stream that concatenates regular data and zip data if you position the open, readable stream at the start of the zip data, you will be able to read the zip archive using this constructor, or any of the ZipFile constructors that accept a <u>Stream</u> as input. Some examples of where this might be useful: the zip content is concatenated at the end o a regular EXE file, as some self-extracting archives do. (Note: SFX files produced by DotNetZip do not work this way; they can be read as norm ZIP files). Another example might be a stream being read from a database, where the zip content is embedded within an aggregate strea of data.

Exceptions

Exception	Condition
Exception	Thrown if the zip archive cannot be read.

■ See Also

Read(String, ReadOptions)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 Read Method (fileName) | Reference ► Ionic.Zip ► ZipFile ► Read(String) | C#

Reads a zip file archive and returns the instance.

Declaration Syntax

Parameters

fileName (String)

The name of the zip archive to open. This can be a fully-qualified or relative pathname.

■ Return Value

The instance read from the zip archive.

■ Remarks

The stream is read using the default System. Text. Encoding, which is the IBM437 codepage.

Exceptions

Exception	Condition
Exception	Thrown if the ZipFile cannot be read. The implementation of this method relies on System.IO.File.OpenRead, which can throw a variety of exceptions, including specific exceptions if a file is no found, an unauthorized access exception exceptions for poorly formatted filenames and so on.

See Also

Read(String, ReadOptions)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Read Method (fileName, options)

Reference ► Ionic.Zip ► ZipFile ► Read(String, ReadOptions)

Reads a zip file archive from the named filesystem file using the specific options.

Declaration Syntax

```
C# Visual Basic Visual C++

public static ZipFile Read(
    string fileName,
    ReadOptions options
)
```

Parameters

fileName (String)

The name of the zip archive to open. This can be a fully-qualified or relative pathname.

options (ReadOptions)

The set of options to use when reading the zip file.

■ Return Value

The ZipFile instance read from the zip archive.

■ Remarks

This version of the Read() method allows the caller to pass in a TextWriter an Encoding, via an instance of the ReadOptions class. The ZipFile is read in using the specified encoding for entries where UTF-8 encoding is not explicitly specified.

Examples

This example shows how to read a zip file using the Big-5 Chinese code page (950), and extract each entry in the zip file, while sending status messages out to the Console.

For this code to work as intended, the zipfile must have been created using the big5 code page (CP950). This is typical, for example, when using WinRar on a machine with CP950 set as the default code page. It that case, the names of entries within the Zip archive will be stored in the code page, and reading the zip archive must be done using that code page. If the application did not use the correct code page in ZipFile.Read(), then names of entries within the zip archive would not be correctly retrieved.

C# □ Cop

```
string zipToExtract = "MyArchive.zip";
string extractDirectory = "extract";
var options = new ReadOptions
{
   StatusMessageWriter = System.Console.Out,
   Encoding = System.Text.Encoding.GetEncoding(950)
};
using (ZipFile zip = ZipFile.Read(zipToExtract, opti
{
   foreach (ZipEntry e in zip)
   {
      e.Extract(extractDirectory);
   }
}
```

VB.NET

Examples

This example shows how to read a zip file using the default code page, remove entries that have a modified date before a given threshold, sending status messages out to a StringWriter.

C#

```
var options = new ReadOptions
{
  StatusMessageWriter = new System.IO.StringWriter()
};
using (ZipFile zip = ZipFile.Read("PackedDocuments.")
 var Threshold = new DateTime(2007,7,4);
  // We cannot remove the entry from the list, withi
  // an enumeration of said list.
  // So we add the doomed entry to a list to be remo
  // pass 1: mark the entries for removal
 var MarkedEntries = new System.Collections.Generic
  foreach (ZipEntry e in zip)
  {
    if (e.LastModified < Threshold)</pre>
      MarkedEntries.Add(e);
  }
```

```
// pass 2: actually remove the entry.
foreach (ZipEntry zombie in MarkedEntries)
    zip.RemoveEntry(zombie);
zip.Comment = "This archive has been updated.";
zip.Save();
}
// can now use contents of sw, eg store in an audit
```

VB.NET
□ Cop

```
Dim options as New ReadOptions
options.StatusMessageWriter = New System.IO.StringWr
Using zip As ZipFile = ZipFile.Read("PackedDocuments
    Dim Threshold As New DateTime(2007, 7, 4)
    ' We cannot remove the entry from the list, with
    ' an enumeration of said list.
    ' So we add the doomed entry to a list to be rem
    ' pass 1: mark the entries for removal
    Dim MarkedEntries As New System.Collections.Gene
    Dim e As ZipEntry
    For Each e In zip
        If (e.LastModified < Threshold) Then</pre>
            MarkedEntries.Add(e)
        End If
    Next
     pass 2: actually remove the entry.
    Dim zombie As ZipEntry
    For Each zombie In MarkedEntries
        zip.RemoveEntry(zombie)
    Next
    zip.Comment = "This archive has been updated."
    zip.Save
End Using
 can now use contents of sw, eg store in an audit l
```

Exceptions

Exception	Condition
Exception	Thrown if the zipfile cannot be read. The

implementation of this method relies on System. IO.File.OpenRead, which can throw a variety of exceptions, including specific exceptions if a file is not found, an unauthorized access exception exceptions for poorly formatted filenames and so on.

F

See Also

Read(Stream, ReadOptions)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ReadProgress Event

Reference ► Ionic.Zip ► ZipFile ► ReadProgress

C#

An event handler invoked before, during, and after the reading of a zip archive.

Declaration Syntax

```
C# Visual Basic Visual C++

public event EventHandler<ReadProgressEventArgs> Rea

Public Event ReadProgress As EventHandler(Of ReadPro

public:
    event EventHandler<ReadProgressEventArgs^>^ ReadPro
        void add (EventHandler<ReadProgressEventArgs
        void remove (EventHandler<ReadProgressEventArgs)

}
```

■ Remarks

Depending on the particular event being signaled, different properties of the <u>ReadProgressEventArgs</u> parameter are set. The following table summarizes the available EventTypes and the conditions under which this event handler is invoked with a <u>ReadProgressEventArgs</u> with given EventType.

value of EntryType	Meaning and conditions
ZipProgressEventType.Reading_Started	Fired just as ZipFile.Read() begins. Meaningfu properties: ArchiveName.
ZipProgressEventType.Reading_Completed	Fired when

	ZipFile.Read() has completed. Meaningful properties: ArchiveName.
ZipProgressEventType.Reading_ArchiveBytesRead	Fired while readin updates the number of bytes read for the entire archive. Meaningful properties: ArchiveName, CurrentEntry, BytesTransferred, TotalBytesToTrans
ZipProgressEventType.Reading_BeforeReadEntry	Indicates an entry about to be read f the archive. Meaningful properties: ArchiveName, EntriesTotal.
ZipProgressEventType.Reading_AfterReadEntry	Indicates an entry has just been rear from the archive. Meaningful properties: ArchiveName, EntriesTotal, CurrentEntry.

■ See Also

SaveProgress
AddProgress
ExtractProgress

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

√

Ionic Zip Library v1.9.1.6 RemoveEntries Method

Reference ► Ionic.Zip ► ZipFile ► RemoveEntries()

C#

■ Members

Icon	Member	Description
∄	RemoveEntries(ICollection <zipentry>)</zipentry>	This method removes a collection of entries from the ZipFile.
≘	RemoveEntries(ICollection <string>)</string>	This method removes a collection of entries from the ZipFile, by name.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 RemoveEntries Method

(entriesToRemove)

Reference ► Ionic.Zip ► ZipFile ►

RemoveEntries(ICollection<ZipEntry>)

This method removes a collection of entries from the ZipFile.

■ Declaration Syntax

Parameters

entriesToRemove (ICollection<ZipEntry>)

A collection of ZipEntry instances from this zip file to be removed. For example, you can pass in an array of ZipEntry instances; or you can call SelectEntries(), and then add or remove entries from that ICollection
ZipEntry> (ICollection(Of ZipEntry) in VB), and pass the ICollection to this method.

See Also

<u>SelectEntries(String)</u> <u>RemoveSelectedEntries(String)</u> Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 RemoveEntries Method

(entriesToRemove)

Reference ► Ionic.Zip ► ZipFile ►

RemoveEntries(ICollection<String>)

This method removes a collection of entries from the ZipFile, by name.

Declaration Syntax

Parameters

entriesToRemove (ICollection<String>)

A collection of strings that refer to names of entries to be removed from the ZipFile. For example, you can pass in an array or a List of Strings that provide the names of entries to be removed.

See Also

<u>SelectEntries(String)</u> <u>RemoveSelectedEntries(String)</u>

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 RemoveEntry Method

Reference ► Ionic.Zip ► ZipFile ► RemoveEntry()

C#

■ Members

Icon	Member	Description
∃	RemoveEntry(ZipEntry)	Removes the given ZipEntry from the zip archive.
≅ 🚱	RemoveEntry(String)	Removes the ZipEntry with the given filename from the zip archive.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 RemoveEntry Method (entry)

Reference ► Ionic.Zip ► ZipFile ► RemoveEntry(ZipEntry)

Removes the given ZipEntry from the zip archive.

Declaration Syntax

Parameters

```
entry (ZipEntry)
```

The ZipEntry to remove from the zip.

Remarks

After calling RemoveEntry, the application must call Save to make the changes permanent.

Examples

In this example, all entries in the zip archive dating from before December 31st, 2007, are removed from the archive. This is actually much easier if you use the RemoveSelectedEntries method. But I need an example for RemoveEntry, so here it is.

C# □ Cop

```
String ZipFileToRead = "ArchiveToModify.zip";
System.DateTime Threshold = new System.DateTime(2007
using (ZipFile zip = ZipFile.Read(ZipFileToRead))
{
 var EntriesToRemove = new System.Collections.Gener
  foreach (ZipEntry e in zip)
    if (e.LastModified < Threshold)</pre>
    {
      // We cannot remove the entry from the list, w
      // an enumeration of said list.
      // So we add the doomed entry to a list to be
      EntriesToRemove.Add(e);
 }
  // actually remove the doomed entries.
  foreach (ZipEntry zombie in EntriesToRemove)
    zip.RemoveEntry(zombie);
  zip.Comment= String.Format("This zip archive was u
                              System.DateTime.Now.ToS
  // save with a different name
 zip.Save("Archive-Updated.zip");
```

VB.NET

```
Dim ZipFileToRead As String = "ArchiveToModify.zip"
Dim Threshold As New DateTime(2007, 12, 31)
Using zip As ZipFile = ZipFile.Read(ZipFileToRead)
    Dim EntriesToRemove As New System.Collections.Ge
    Dim e As ZipEntry
    For Each e In zip
        If (e.LastModified < Threshold) Then
        ' We cannot remove the entry from the li</pre>
```

```
' an enumeration of said list.
' So we add the doomed entry to a list to EntriesToRemove.Add(e)
End If
Next

' actually remove the doomed entries.
Dim zombie As ZipEntry
For Each zombie In EntriesToRemove
    zip.RemoveEntry(zombie)
Next
zip.Comment = String.Format("This zip archive was 'save as a different name
zip.Save("Archive-Updated.zip")
End Using
```

Exceptions

Exception	Condition
ArgumentException	Thrown if the specified ZipEntry does not exist in the ZipFile.

See Also

RemoveSelectedEntries(String)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 RemoveEntry Method (fileName)

Reference ► Ionic.Zip ► ZipFile ► RemoveEntry(String)

Removes the ZipEntry with the given filename from the zip archive.

Declaration Syntax

Parameters

fileName (String)

The name of the file, including any directory path, to remove from the zip. The filename match is not case-sensitive by default; you can use the CaseSensitiveRetrieval property to change this behavior. The pathname can use forward-slashes or backward slashes.

■ Remarks

After calling RemoveEntry, the application must call Save to make the changes permanent.

Examples

This example shows one way to remove an entry with a given filename from an existing zip archive.

C# □ Cop

VB.NET

```
Dim zipFileToRead As String = "PackedDocuments.zip"
Dim candidate As String = "DatedMaterial.xps"
Using zip As ZipFile = ZipFile.Read(zipFileToRead)
    If zip.EntryFilenames.Contains(candidate) Then
        zip.RemoveEntry(candidate)
        zip.Comment = String.Format("The file '{0}'
        zip.Save
    End If
End Using
```

Exceptions

Exception	Condition
InvalidOperationException	Thrown if the ZipFile is not updatable
ArgumentException	Thrown if a ZipEntry with the specified filename does not exist in the ZipFile.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Remove Selected Entries Method

Reference ► Ionic.Zip ► ZipFile ► RemoveSelectedEntries()

C# _

■ Members

Icon	Member	Description
≅	RemoveSelectedEntries(String)	Remove entries from the zipfile by specified criteria.
€ 🖗	RemoveSelectedEntries(String, String)	Remove entries from the zipfile by specified criteria, and within the specified path in the archive

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Remove Selected Entries Method (selection Criteria)

Reference ► Ionic.Zip ► ZipFile ► RemoveSelectedEntries(String)

Remove entries from the zipfile by specified criteria.

C#

Declaration Syntax

Parameters

selectionCriteria (String)

the string that specifies which entries to select

■ Return Value

the number of entries removed

Remarks

This method allows callers to remove the collection of entries from the zipfile that fit the specified criteria. The criteria are described in a string format, and can include patterns for the filename; constraints on the size of the entry; constraints on the last modified, created, or last accessed time for the file described by the entry; or the attributes of the entry.

For details on the syntax for the selectionCriteria parameter, see AddSelectedFiles(String).

This method is intended for use with a ZipFile that has been read from storage. When creating a new ZipFile, this method will work only after the ZipArchive has been Saved to the disk (the ZipFile class subsequently and implicitly reads the Zip archive from storage.) Calling SelectEntries on a ZipFile that has not yet been saved will deliver undefined results.

Examples

This example removes all entries in a zip file that were modified prior to January 1st, 2008.

```
Using (ZipFile zip1 = ZipFile.Read(ZipFileName))
{
    // remove all entries from prior to Jan 1, 2008
    zip1.RemoveEntries("mtime < 2008-01-01");
    // don't forget to save the archive!
    zip1.Save();
}</pre>
```

VB.NET

```
Using zip As ZipFile = ZipFile.Read(ZipFileName)
   ' remove all entries from prior to Jan 1, 2008
   zip1.RemoveEntries("mtime < 2008-01-01")
   ' do not forget to save the archive!
   zip1.Save
End Using</pre>
```

Exceptions

Exception	Condition
Exception	Thrown if selectionCriteria has an invalid syntax.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Remove Selected Entries Method (selection Criteria, directory Path In Archive)

Reference ► Ionic.Zip ► ZipFile ► RemoveSelectedEntries(String, String)

Remove entries from the zipfile by specified criteria, and within the specified path in the archive.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public int RemoveSelectedEntries(
    string selectionCriteria,
    string directoryPathInArchive
)
```

Parameters

selectionCriteria (String)

the string that specifies which entries to select

directoryPathInArchive (String)

the directory in the archive from which to select entries. If null, then all directories in the archive are used.

■ Return Value

the number of entries removed

Remarks

This method allows callers to remove the collection of entries from the zipfile that fit the specified criteria. The criteria are described in a string format, and can include patterns for the filename; constraints on the size of the entry; constraints on the last modified, created, or last accessed time for the file described by the entry; or the attributes of the entry.

For details on the syntax for the selectionCriteria parameter, see <u>AddSelectedFiles(String)</u>.

This method is intended for use with a ZipFile that has been read from storage. When creating a new ZipFile, this method will work only after the ZipArchive has been Saved to the disk (the ZipFile class subsequently and implicitly reads the Zip archive from storage.) Calling SelectEntries on a ZipFile that has not yet been saved will deliver undefined results.

Examples

```
Using (ZipFile zip1 = ZipFile.Read(ZipFileName))
{
    // remove all entries from prior to Jan 1, 2008
    zip1.RemoveEntries("mtime < 2008-01-01", "docume
    // a call to ZipFile.Save will make the modifica
    zip1.Save();
}</pre>
```

VB.NET

```
Using zip As ZipFile = ZipFile.Read(ZipFileName)
   ' remove all entries from prior to Jan 1, 2008
   zip1.RemoveEntries("mtime < 2008-01-01", "docume
   ' a call to ZipFile.Save will make the modificat
   zip1.Save
End Using</pre>
```

Exceptions

Exception	Condition
Exception	Thrown if selectionCriteria has an invalid syntax.

F

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Indicates whether the archive requires ZIP64 extensions.

Declaration Syntax

```
C# Visual Basic Visual C++
public Nullable<bool> RequiresZip64 { get; }

Public ReadOnly Property RequiresZip64 As Nullable(O Get

public:
property Nullable<bool> RequiresZip64 {
         Nullable<bool> get ();
}
```

Remarks

This property is null (or Nothing in VB) if the archive has not been saved, and there are fewer than 65334 ZipEntry items contained in the archive.

The Value is true if any of the following four conditions holds: the uncompressed size of any entry is larger than 0xFFFFFFF; the compressed size of any entry is larger than 0xFFFFFFFF; the relative offset of any entry within the zip archive is larger than 0xFFFFFFFF; or there are more than 65534 entries in the archive. (0xFFFFFFFF = 4,294,967,295). The result may not be known until a Save() is attempted on the zip archive. The Value of this Nullable property may be set only AFTER one of the Save() methods has been called.

If none of the four conditions holds, and the archive has been saved, the the Value is false.

A Value of false does not indicate that the zip archive, as saved, does not use ZIP64. It merely indicates that ZIP64 is not required. An archive may use ZIP64 even when not required if the UseZip64WhenSaving property is set to Always, or if the UseZip64WhenSaving property is set to AsNecessary and the output stream was not seekable. Use the OutputUsedZip64 property to determine if the most recent Save () method resulted in an archive that utilized the ZIP64 extensions.

See Also

<u>UseZip64WhenSaving</u> <u>OutputUsedZip64</u>

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

Ionic Zip Library v1.9.1.6 Save Method

Reference ► Ionic.Zip ► ZipFile ► Save()

C#

■ Members

Icon	Member	Description
≡	Save()	Saves the Zip archive to a file, specified by the Name property of the ZipFile.
€ 🖗	Save(String)	Save the file to a new zipfile, with the given name.
≡	Save(Stream)	Save the zip archive to the specified stream.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

```
Ionic Zip Library v1.9.1.6
Save Method

Reference ► Ionic.Zip ► ZipFile ► Save()
C#
```

Saves the Zip archive to a file, specified by the Name property of the ZipFile.

Declaration Syntax

```
C# Visual Basic Visual C++

public void Save()

Public Sub Save

public:
void Save()
```

■ Remarks

The ZipFile instance is written to storage, typically a zip file in a filesystem, only when the caller calls Save. In the typical case, the Sav operation writes the zip content to a temporary file, and then renames the temporary file to the desired name. If necessary, this method will delete pre-existing file before the rename.

The <u>Name</u> property is specified either explicitly, or implicitly using one o the parameterized ZipFile constructors. For COM Automation clients, th <u>Name</u> property must be set explicitly, because COM Automation clients cannot call parameterized constructors.

When using a filesystem file for the Zip output, it is possible to call Sav multiple times on the ZipFile instance. With each call the zip conten is re-written to the same output file.

Data for entries that have been added to the ZipFile instance is written to the output when the Save method is called. This means that the input streams for those entries must be available at the time the

application calls Save. If, for example, the application adds entries with AddEntry using a dynamically-allocated MemoryStream, the memory stream must not have been disposed before the call to Save. See the InputStream property for more discussion of the availability requirements of the input stream for an entry, and an approach for providing just-in-time stream lifecycle management.

Exceptions

Exception	Condition
BadStateException	Thrown if you haven't specified a location or stream for saving the zip, either in the constructor or by setting the Name property, or if you try to save a regular zip archive to a filename with a .exe extension.
OverflowException	Thrown if MaxOutputSegmentSize is non zero, and the number of segments that would be generated for the spanned zip file during the save operation exceeds 99 If this happens, you need to increase the segment size.

■ See Also

AddEntry(String, Stream)

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

| Ionic Zip Library v1.9.1.6 Save Method (outputStream) | Reference ► Ionic.Zip ► ZipFile ► Save(Stream) | C#

Save the zip archive to the specified stream.

■ Declaration Syntax

Parameters

outputStream (Stream)

The System. IO. Stream to write to. It must be writable. If you created the ZipFile instanct by calling ZipFile.Read(), this stream must not be the same stream you passed to ZipFile.Read().

■ Remarks

The ZipFile instance is written to storage - typically a zip file in a filesystem, but using this overload, the storage can be anything accessible via a writable stream - only when the caller calls Save.

Use this method to save the zip content to a stream directly. A common scenario is an ASP.NET application that dynamically generates a zip file and allows the browser to download it. The application can call

Save (Response.OutputStream) to write a zipfile directly to the output stream, without creating a zip file on the disk on the ASP.NET server.

Be careful when saving a file to a non-seekable stream, including Response.OutputStream. When DotNetZip writes to a non-seekable stream, the zip archive is formatted in such a way that may not be compatible with all zip tools on all platforms. It's a perfectly legal and compliant zip file, but some people have reported problems opening file produced this way using the Mac OS archive utility.

Examples

This example saves the zipfile content into a MemoryStream, and then gets the array of bytes from that MemoryStream.

```
using (var zip = new Ionic.Zip.ZipFile())
{
    zip.CompressionLevel= Ionic.Zlib.CompressionLeve
    zip.Password = "VerySecret.";
    zip.Encryption = EncryptionAlgorithm.WinZipAes12
    zip.AddFile(sourceFileName);
    MemoryStream output = new MemoryStream();
    zip.Save(output);

byte[] zipbytes = output.ToArray();
}
```

Examples

This example shows a pitfall you should avoid. DO NOT read from a stream, then try to save to the same stream. DO NOT DO THIS:

```
C#
using (var fs = new FileSteeam(filename, FileMode.Op
{
  using (var zip = Ionic.Zip.ZipFile.Read(inputStrea))
```

```
{
    zip.AddEntry("Name1.txt", "this is the content")
    zip.Save(inputStream); // NO NO NO!!
}
```

Better like this:

```
c#
using (var zip = Ionic.Zip.ZipFile.Read(filename))
{
    zip.AddEntry("Name1.txt", "this is the content")
    zip.Save(); // YES!
}
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

| Ionic Zip Library v1.9.1.6 Save Method (fileName) | Reference ► Ionic.Zip ► ZipFile ► Save(String) | C#

Save the file to a new zipfile, with the given name.

■ Declaration Syntax

Parameters

fileName (String)

The name of the zip archive to save to. Existing files will be overwritten with great prejudice.

■ Remarks

This method allows the application to explicitly specify the name of the zip file when saving. Use this when creating a new zip file, or when updating a zip archive.

An application can also save a zip archive in several places by calling this method multiple times in succession, with different filenames.

The ZipFile instance is written to storage, typically a zip file in a

filesystem, only when the caller calls Save. The Save operation writes the zip content to a temporary file, and then renames the temporary file the desired name. If necessary, this method will delete a pre-existing file before the rename.

Examples

This example shows how to create and Save a zip file.

```
c#
using (ZipFile zip = new ZipFile())
{
  zip.AddDirectory(@"c:\reports\January");
  zip.Save("January.zip");
}
```

VB.NET

```
Using zip As New ZipFile()
  zip.AddDirectory("c:\reports\January")
  zip.Save("January.zip")
End Using
```

Examples

This example shows how to update a zip file.

```
C#
using (ZipFile zip = ZipFile.Read("ExistingArchive.z
{
   zip.AddFile("NewData.csv");
   zip.Save("UpdatedArchive.zip");
}
```

```
VB.NET
```

```
Using zip As ZipFile = ZipFile.Read("ExistingArchive
zip.AddFile("NewData.csv")
zip.Save("UpdatedArchive.zip")
```

End Using

Exceptions

Exception	Condition
ArgumentException	Thrown if you specify a directory for the filename.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Save Progress Event

Reference ► Ionic.Zip ► ZipFile ► SaveProgress

C#

An event handler invoked when a Save() starts, before and after each entry has been written to the archive, when a Save() completes, and during other Save events.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public event <u>EventHandler</u><<u>SaveProgressEventArgs</u>> Sav

Public Event SaveProgress As <u>EventHandler</u>(Of <u>SaveProgressEventArgs</u>)

public:
    event <u>EventHandler</u><<u>SaveProgressEventArgs</u>^> SaveProgred add (<u>EventHandler</u><<u>SaveProgressEventArgs</u>)

void remove (<u>EventHandler</u><<u>SaveProgressEventArgs</u>)
```

■ Remarks

Depending on the particular event, different properties on the <u>SaveProgressEventArgs</u> parameter are set. The following table summarizes the available EventTypes and the conditions under which this event handler is invoked with a <u>SaveProgressEventArgs</u> with given EventType.

value of EntryType	Meaning condition
ZipProgressEventType.Saving_Started	Fired whe ZipFile.Sa begins.
ZipProgressEventType.Saving_BeforeSaveEntry	Fired with ZipFile.Sa

	before wr for each p entry.
ZipProgressEventType.Saving_AfterSaveEntry	Fired with ZipFile.Sa after havi writing da particular
ZipProgressEventType.Saving_Completed	Fired whe ZipFile.Sa completed
ZipProgressEventType.Saving_AfterSaveTempArchive	Fired afte temporary been crea happens saving to This even invoked w to a strea
ZipProgressEventType.Saving_BeforeRenameTempArchive	Fired just renaming temporary permaner This happ when sav disk file. I will not be when sav stream.
ZipProgressEventType.Saving_AfterRenameTempArchive	Fired just renaming temporary permaner This happ when sav disk file. 1

	will not be when sav stream.
ZipProgressEventType.Saving_AfterCompileSelfExtractor	Fired afte extracting has finish compiling EventTyp only within SaveSelff
ZipProgressEventType.Saving_BytesRead	Set during of a partic to update of the Save this Event set, the BytesTrand the number that have from the sistream. TotalBytes is the numbytes in the uncompression of the sistem of the

Examples

This example uses an anonymous method to handle the SaveProgress event, by updating a progress bar.

C#

```
progressBar1.Value = 0;
progressBar1.Max = listbox1.Items.Count;
using (ZipFile zip = new ZipFile())
{
    // listbox1 contains a list of filenames
    zip.AddFiles(listbox1.Items);
```

```
// do the progress bar:
zip.SaveProgress += (sender, e) => {
   if (e.EventType == ZipProgressEventType.Saving
        progressBar1.PerformStep();
   }
};
zip.Save(fs);
}
```

Examples

This example uses a named method as the SaveProgress event handler, to update the user, in a console-based application.

C# □ Cop

```
static bool justHadByteUpdate= false;
public static void SaveProgress(object sender, SaveP
{
    if (e.EventType == ZipProgressEventType.Saving_S
        Console.WriteLine("Saving: {0}", e.ArchiveNa
    else if (e.EventType == ZipProgressEventType.Sav
        justHadByteUpdate= false;
        Console.WriteLine();
        Console.WriteLine("Done: {0}", e.ArchiveName
    }
    else if (e.EventType == ZipProgressEventType.Sav
    {
        if (justHadByteUpdate)
            Console.WriteLine();
        Console.WriteLine(" Writing: {0} ({1}/{2})"
                          e.CurrentEntry.FileName, e
        justHadByteUpdate= false;
    }
```

```
else if (e.EventType == ZipProgressEventType.Sav
    {
        if (justHadByteUpdate)
            Console.SetCursorPosition(0, Console.Cur
                              \{0\}/\{1\} (\{2:N0\}\%)", e.B
         Console.Write("
                      e.BytesTransferred / (0.01 * e
        justHadByteUpdate= true;
    }
}
public static ZipUp(string targetZip, string directo
{
  using (var zip = new ZipFile()) {
    zip.SaveProgress += SaveProgress;
    zip.AddDirectory(directory);
    zip.Save(targetZip);
 }
}
```

VB.NET □ Cop

```
Public Sub ZipUp(ByVal targetZip As String, ByVal di
    Using zip As ZipFile = New ZipFile
        AddHandler zip.SaveProgress, AddressOf MySav
        zip.AddDirectory(directory)
        zip.Save(targetZip)
    End Using
End Sub

Private Shared justHadByteUpdate As Boolean = False

Public Shared Sub MySaveProgress(ByVal sender As Obj
    If (e.EventType Is ZipProgressEventType.Saving_S
        Console.WriteLine("Saving: {0}", e.ArchiveNa

ElseIf (e.EventType Is ZipProgressEventType.Savi
        justHadByteUpdate = False
        Console.WriteLine
```

```
Console.WriteLine("Done: {0}", e.ArchiveName
    ElseIf (e.EventType Is ZipProgressEventType.Savi
        If justHadByteUpdate Then
            Console.WriteLine
        End If
        Console.WriteLine(" Writing: {0} ({1}/{2})"
        justHadByteUpdate = False
    ElseIf (e.EventType Is ZipProgressEventType.Savi
        If justHadByteUpdate Then
            Console.SetCursorPosition(0, Console.Cur
        End If
        Console.Write(" \{0\}/\{1\}\ (\{2:N0\}\%)", e.By
                      e.TotalBytesToTransfer, _
                      (CDbl(e.BytesTransferred) / (0
        justHadByteUpdate = True
    End If
End Sub
```

Examples

This is a more complete example of using the SaveProgress events in a Windows Forms application, with a Thread object.

C# □ Cop

```
delegate void SaveEntryProgress(SaveProgressEventArg
delegate void ButtonClick(object sender, EventArgs e

public class WorkerOptions
{
    public string ZipName;
    public string Folder;
    public string Encoding;
    public string Comment;
    public int ZipFlavor;
    public Zip64Option Zip64;
}
```

```
private int _progress2MaxFactor;
private bool _saveCanceled;
private long _totalBytesBeforeCompress;
private long _totalBytesAfterCompress;
private Thread _workerThread;
private void btnZipup_Click(object sender, EventArgs
{
    KickoffZipup();
}
private void btnCancel_Click(object sender, EventArg
    if (this.lblStatus.InvokeRequired)
        this.lblStatus.Invoke(new ButtonClick(this.b
    else
        _saveCanceled = true;
        lblStatus.Text = "Canceled...";
        ResetState();
    }
}
private void KickoffZipup()
{
   _folderName = tbDirName.Text;
    if (_folderName == null || _folderName == "") re
    if (this.tbZipName.Text == null || this.tbZipNam
    // check for existence of the zip file:
    if (System.IO.File.Exists(this.tbZipName.Text))
    {
        var dlgResult = MessageBox.Show(String.Forma
                                         "Confirmatio
```

```
if (dlgResult != DialogResult.Yes) return;
    System.IO.File.Delete(this.tbZipName.Text);
}
_saveCanceled = false;
_nFilesCompleted = 0;
_totalBytesAfterCompress = 0;
_totalBytesBeforeCompress = 0;
this.btn0k.Enabled = false;
this.btnOk.Text = "Zipping...";
this.btnCancel.Enabled = true;
lblStatus.Text = "Zipping...";
var options = new WorkerOptions
    ZipName = this.tbZipName.Text,
    Folder = _folderName,
    Encoding = "ibm437"
};
if (this.comboBox1.SelectedIndex != 0)
{
    options.Encoding = this.comboBox1.SelectedIt
}
if (this.radioFlavorSfxCmd.Checked)
    options.ZipFlavor = 2;
else if (this.radioFlavorSfxGui.Checked)
    options.ZipFlavor = 1;
else options.ZipFlavor = 0;
if (this.radioZip64AsNecessary.Checked)
    options.Zip64 = Zip64Option.AsNecessary;
else if (this.radioZip64Always.Checked)
    options.Zip64 = Zip64Option.Always;
else options.Zip64 = Zip64Option.Never;
options.Comment = String.Format("Encoding: {0}
            options. Encoding,
```

```
FlavorToString(options.ZipFlavor),
                options.Zip64.ToString(),
                System.DateTime.Now.ToString("yyyy-M
                this.Text);
    if (this.tbComment.Text != TB_COMMENT_NOTE)
        options.Comment += this.tbComment.Text;
    _workerThread = new Thread(this.DoSave);
    _workerThread.Name = "Zip Saver thread";
    _workerThread.Start(options);
    this.Cursor = Cursors.WaitCursor;
}
private void DoSave(Object p)
{
   WorkerOptions options = p as WorkerOptions;
    try
    {
        using (var zip1 = new ZipFile())
        {
            zip1.ProvisionalAlternateEncoding = Syst
            zip1.Comment = options.Comment;
            zip1.AddDirectory(options.Folder);
            _entriesToZip = zip1.EntryFileNames.Coun
            SetProgressBars();
            zip1.SaveProgress += this.zip1_SaveProgr
            zip1.UseZip64WhenSaving = options.Zip64;
            if (options.ZipFlavor == 1)
                zip1.SaveSelfExtractor(options.ZipNa
            else if (options.ZipFlavor == 2)
                zip1.SaveSelfExtractor(options.ZipNa
            else
                zip1.Save(options.ZipName);
        }
```

```
catch (System.Exception exc1)
    {
        MessageBox.Show(String.Format("Exception whi
        btnCancel_Click(null, null);
    }
}
void zip1_SaveProgress(object sender, SaveProgressEv
    switch (e.EventType)
    {
        case ZipProgressEventType.Saving_AfterWriteE
            StepArchiveProgress(e);
            break;
        case ZipProgressEventType.Saving_EntryBytesR
            StepEntryProgress(e);
            break;
        case ZipProgressEventType.Saving_Completed:
            SaveCompleted();
            break;
        case ZipProgressEventType.Saving_AfterSaveTe
            // this event only occurs when saving an
            TempArchiveSaved();
            break;
    if (_saveCanceled)
        e.Cancel = true;
}
private void StepArchiveProgress(SaveProgressEventAr
{
    if (this.progressBar1.InvokeRequired)
    {
        this.progressBar1.Invoke(new SaveEntryProgre
```

```
else
    {
        if (!_saveCanceled)
            _nFilesCompleted++;
            this.progressBar1.PerformStep();
            _totalBytesAfterCompress += e.CurrentEnt
            _totalBytesBeforeCompress += e.CurrentEn
            // reset the progress bar for the entry:
            this.progressBar2.Value = this.progressB
            this.Update();
        }
    }
}
private void StepEntryProgress(SaveProgressEventArgs
    if (this.progressBar2.InvokeRequired)
    {
        this.progressBar2.Invoke(new SaveEntryProgre
    else
    {
        if (!_saveCanceled)
            if (this.progressBar2.Maximum == 1)
            {
                // reset
                Int64 max = e.TotalBytesToTransfer;
                _{progress2MaxFactor} = 0;
                while (max > System.Int32.MaxValue)
                {
                    max /= 2;
                    _progress2MaxFactor++;
                this.progressBar2.Maximum = (int)max
```

```
lblStatus.Text = String.Format("{0}
                    _nFilesCompleted + 1, _entriesTo
            }
             int xferred = e.BytesTransferred >> _pr
             this.progressBar2.Value = (xferred >= t
                ? this.progressBar2.Maximum
                : xferred;
             this.Update();
        }
   }
}
private void SaveCompleted()
{
    if (this.lblStatus.InvokeRequired)
    {
        this.lblStatus.Invoke(new MethodInvoker(this
    else
        lblStatus.Text = String.Format("Done, Compre
            _nFilesCompleted, (100.00 * _totalBytesA
         ResetState();
    }
}
private void ResetState()
{
    this.btnCancel.Enabled = false;
    this.btnOk.Enabled = true;
    this.btn0k.Text = "Zip it!";
    this.progressBar1.Value = 0;
    this.progressBar2.Value = 0;
    this.Cursor = Cursors.Default;
    if (!_workerThread.IsAlive)
        _workerThread.Join();
```

}

See Also

ReadProgress
AddProgress
ExtractProgress

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Save Self Extractor Method

Reference ► Ionic.Zip ► ZipFile ► SaveSelfExtractor() C#

■ Members

Icon	Member	Description
≅	SaveSelfExtractor(String, SelfExtractorFlavor)	Saves the ZipFile instance to a self-extracting zip archive.
∃	SaveSelfExtractor(String, SelfExtractorSaveOptions)	Saves the ZipFile instance to a self- extracting zip archive, using the specified save options.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Save Self Extractor Method (avo To Gonorato, flavor)

(exeToGenerate, flavor)

Reference ► <u>lonic.Zip</u> ► <u>ZipFile</u> ► SaveSelfExtractor(String, SelfExtractorFlavor)

Saves the ZipFile instance to a self-extracting zip archive.

■ Declaration Syntax

Parameters

exeToGenerate (String)

a pathname, possibly fully qualified, to be created. Typically it will end in an .exe extension.

flavor (SelfExtractorFlavor)

Indicates whether a Winforms or Console self-extractor is desired.

■ Remarks

The generated exe image will execute on any machine that has the .NE Framework 2.0 installed on it. The generated exe image is also a valid ZIP file, readable with DotNetZip or another Zip library or tool such as WinZip.

There are two "flavors" of self-extracting archive. The WinFormsApplication version will pop up a GUI and allow the user to select a target directory into which to extract. There's also a checkbox allowing the user to specify to overwrite existing files, and another checkbox to allow the user to request that Explorer be opened to see the extracted files after extraction. The other flavor is ConsoleApplication. A self-extractor generated with that flavor setting will run from the command line. It accepts command-line options to set the overwrite behavior, and to specify the target extraction directory.

There are a few temporary files created during the saving to a self-extracting zip. These files are created in the directory pointed to by TempFileFolder, which defaults to GetTempPath(). These temporary file are removed upon successful completion of this method.

When a user runs the WinForms SFX, the user's personal directory (Environment.SpecialFolder.Personal) will be used as the default extract location. If you want to set the default extract location, you should use the other overload of SaveSelfExtractor()/ The user who runs the SFX will have the opportunity to change the extract directory before extracting. When the user runs the Command-Line SFX, the user must explicitly specify the directory to which to extract. The .NET Framework 2.0 is required on the computer when the self-extracting archive is run.

NB: This method is not available in the version of DotNetZip build for the .NET Compact Framework, nor in the "Reduced" DotNetZip library.

Examples

```
C# □ Cop
```

```
string DirectoryPath = "c:\\Documents\\Project7";
using (ZipFile zip = new ZipFile())
```

```
{
    zip.AddDirectory(DirectoryPath, System.IO.Path.G
    zip.Comment = "This will be embedded into a self
    zip.SaveSelfExtractor("archive.exe", SelfExtract
}
```

VB.NET

```
Dim DirectoryPath As String = "c:\Documents\Project7
Using zip As New ZipFile()
    zip.AddDirectory(DirectoryPath, System.IO.Path.G
    zip.Comment = "This will be embedded into a self
    zip.SaveSelfExtractor("archive.exe", SelfExtract
End Using
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Save Self Extractor Method (exe To Generate, options)

Reference ► Ionic.Zip ► ZipFile ► SaveSelfExtractor(String,

SelfExtractorSaveOptions)

Saves the ZipFile instance to a self-extracting zip archive, using the specified save options. \mathbb{C}^{π}

■ Declaration Syntax

Parameters

exeToGenerate (String)

The name of the EXE to generate.

options (SelfExtractorSaveOptions)

provides the options for creating the Self-extracting archive.

Remarks

This method saves a self extracting archive, using the specified save options. These options include the flavor of the SFX, the default extract directory, the icon file, and so on. See the documentation for SaveSelfExtractor(String, SelfExtractorFlavor) for more details.

The user who runs the SFX will have the opportunity to change the extract directory before extracting. If at the time of extraction, the specified directory does not exist, the SFX will create the directory before extracting the files.

Examples

This example saves a WinForms-based self-extracting archive EXE tha will use c:\ExtractHere as the default extract location. The C# code shows syntax for .NET 3.0, which uses an object initializer for the SelfExtractorOptions object.

C# □ Cop

```
string DirectoryPath = "c:\\Documents\\Project7";
using (ZipFile zip = new ZipFile())
{
    zip.AddDirectory(DirectoryPath, System.IO.Path.G
    zip.Comment = "This will be embedded into a self
    var options = new SelfExtractorOptions
    {
        Flavor = SelfExtractorFlavor.WinFormsApplicati
        DefaultExtractDirectory = "%USERPROFILE%\\Extr
        PostExtractCommandLine = ExeToRunAfterExtract,
        SfxExeWindowTitle = "My Custom Window Title",
        RemoveUnpackedFilesAfterExecute = true
    };
    zip.SaveSelfExtractor("archive.exe", options);
}
```

VB.NET

```
Dim DirectoryPath As String = "c:\Documents\Project7
Using zip As New ZipFile()
    zip.AddDirectory(DirectoryPath, System.IO.Path.G
```

```
zip.Comment = "This will be embedded into a self
Dim options As New SelfExtractorOptions()
options.Flavor = SelfExtractorFlavor.WinFormsApp
options.DefaultExtractDirectory = "%USERPROFILE%
options.PostExtractCommandLine = ExeToRunAfterEx
options.SfxExeWindowTitle = "My Custom Window Ti
options.RemoveUnpackedFilesAfterExecute = True
zip.SaveSelfExtractor("archive.exe", options)
End Using
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

|

Ionic Zip Library v1.9.1.6 SelectEntries Method

Reference ► Ionic.Zip ► ZipFile ► SelectEntries()

C#

■ Members

Icon	Member	Description
≅	SelectEntries(String)	Retrieve entries from the zipfile by specified criteria.
∄	SelectEntries(String, String)	Retrieve entries from the zipfile by specified criteria.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **SelectEntries Method** (selection Criteria)

Reference ► Ionic.Zip ► ZipFile ► SelectEntries(String)

Retrieve entries from the zipfile by specified criteria.

C#

Declaration Syntax

Parameters

selectionCriteria (String)

the string that specifies which entries to select

■ Return Value

a collection of ZipEntry objects that conform to the inclusion spec

Remarks

This method allows callers to retrieve the collection of entries from the zipfile that fit the specified criteria. The criteria are described in a string format, and can include patterns for the filename; constraints on the size of the entry; constraints on the last modified, created, or last accessed time for the file described by the entry; or the attributes of the entry.

For details on the syntax for the selectionCriteria parameter, see <u>AddSelectedFiles(String)</u>.

This method is intended for use with a ZipFile that has been read from storage. When creating a new ZipFile, this method will work only after the ZipArchive has been Saved to the disk (the ZipFile class subsequently and implicitly reads the Zip archive from storage.) Calling SelectEntries on a ZipFile that has not yet been saved will deliver undefined results.

Examples

This example selects all the PhotoShop files from within an archive, and extracts them to the current working directory.

```
c#
using (ZipFile zip1 = ZipFile.Read(ZipFileName))
{
  var PhotoShopFiles = zip1.SelectEntries("*.psd")
  foreach (ZipEntry psd in PhotoShopFiles)
  {
    psd.Extract();
  }
}
```

```
Using zip1 As ZipFile = ZipFile.Read(ZipFileName)
    Dim PhotoShopFiles as ICollection(Of ZipEntry)
    PhotoShopFiles = zip1.SelectEntries("*.psd")
    Dim psd As ZipEntry
    For Each psd In PhotoShopFiles
        psd.Extract
    Next
End Using
```

Exceptions

Exception	Condition

Exception	Thrown if selectionCriteria has an invalid
	syntax.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SelectEntries Method (selectionCriteria, directoryPathInArchive)

Reference ► Ionic.Zip ► ZipFile ► SelectEntries(String, String)

Retrieve entries from the zipfile by specified criteria.

C#

Declaration Syntax

Parameters

selectionCriteria (String)

the string that specifies which entries to select

directoryPathInArchive (String)

the directory in the archive from which to select entries. If null, then all directories in the archive are used.

Return Value

a collection of ZipEntry objects that conform to the inclusion spec

Remarks

This method allows callers to retrieve the collection of entries from the zipfile that fit the specified criteria. The criteria are described in a string format, and can include patterns for the filename; constraints on the size of the entry; constraints on the last modified, created, or last accessed time for the file described by the entry; or the attributes of the entry.

For details on the syntax for the selectionCriteria parameter, see AddSelectedFiles(String).

This method is intended for use with a ZipFile that has been read from storage. When creating a new ZipFile, this method will work only after the ZipArchive has been Saved to the disk (the ZipFile class subsequently and implicitly reads the Zip archive from storage.) Calling SelectEntries on a ZipFile that has not yet been saved will deliver undefined results.

Examples

```
using (ZipFile zip1 = ZipFile.Read(ZipFileName))
{
   var UpdatedPhotoShopFiles = zip1.SelectEntries("
   foreach (ZipEntry e in UpdatedPhotoShopFiles)
   {
      // prompt for extract here
      if (WantExtract(e.FileName))
            e.Extract();
   }
}
```

```
VB.NET

Using zip1 As ZipFile = ZipFile.Read(ZipFileName)
    Dim UpdatedPhotoShopFiles As ICollection(Of ZipE
    Dim e As ZipEntry
    For Each e In UpdatedPhotoShopFiles
    ' prompt for extract here
    If Me.WantExtract(e.FileName) Then
```

e.Extract End If Next End Using

Exceptions

Exception	Condition
Exception	Thrown if selectionCriteria has an invalid syntax.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

•1

```
| Ionic Zip Library v1.9.1.6 SetCompression Property | Reference ► Ionic.Zip ► ZipFile ► SetCompression | C# | ▼
```

A callback that allows the application to specify the compression level to use for entries subsequently added to the zip archive.

Declaration Syntax

```
C# Visual Basic Visual C++
public SetCompressionCallback SetCompression { get;

Public Property SetCompression As SetCompressionCall Get Set

public:
property SetCompressionCallback^ SetCompression {
    SetCompressionCallback^ get ();
    void set (SetCompressionCallback^ value);
}
```

Remarks

With this callback, the DotNetZip library allows the application to determine whether compression will be used, at the time of the Save. This may be useful if the application wants to favor speed over size, and wants to defer the decision until the time of Save.

Typically applications set the <u>CompressionLevel</u> property on the ZipFile or on each ZipEntry to determine the level of compression used. This is done at the time the entry is added to the ZipFile. Setting the property to Ionic.Zlib.CompressionLevel.None means no compression will be used.

This callback allows the application to defer the decision on the

CompressionLevel to use, until the time of the call to ZipFile.Save(). The callback is invoked once per ZipEntry, a the time the data for the entry is being written out as part of a Save() operation. The application can use whatever criteria it likes in determining the level to return. For example, an application may wish that no .mp3 files should be compressed, because they are already compressed and the extra compression is not worth the CPU time incurred, and so can return None for all .mp3 entries.

The library determines whether compression will be attempted for an entry this way: If the entry is a zero length file, or a directory, no compression is used. Otherwise, if this callback is set, it is invoked and the CompressionLevel is set to the return value. If this callback han not been set, then the previously set value for CompressionLevel used.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

4

Ionic Zip Library v1.9.1.6 SortEntries Before Saving Property

Reference ► Ionic.Zip ► ZipFile ► SortEntriesBeforeSaving

Whether to sort the ZipEntries before saving the file.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++
public bool SortEntriesBeforeSaving { get; set; }

Public Property SortEntriesBeforeSaving As Boolean Get Set

public:
property bool SortEntriesBeforeSaving {
    bool get ();
    void set (bool value);
}
```

Remarks

The default is false. If you have a large number of zip entries, the sort alone can consume significant time.

Examples

```
Using (var zip = new ZipFile())
{
    zip.AddFiles(filesToAdd);
    zip.SortEntriesBeforeSaving = true;
    zip.Save(name);
}
```

VB.NET

Cop

```
Using zip As New ZipFile
    zip.AddFiles(filesToAdd)
    zip.SortEntriesBeforeSaving = True
    zip.Save(name)
End Using
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Status Message Text Writer Property

Reference ► Ionic.Zip ► ZipFile ► StatusMessageTextWriter

Gets or sets the TextWriter to which status messages are delivere for the instance.

■ Declaration Syntax

■ Remarks

If the TextWriter is set to a non-null value, then verbose output is sent to the TextWriter during Add, Read, Save and Extract operations. Typically, console applications might use Console.Out and graphical or headless applications might use a System.IO.StringWriter. The output of this is suitable for viewing by humans.

Examples

In this example, a console application instantiates a ZipFile, then se the StatusMessageTextWriter to Console.Out. At that point, all verbose status messages for that ZipFile are sent to the console. C# □ Cop

```
using (ZipFile zip= ZipFile.Read(FilePath))
{
  zip.StatusMessageTextWriter= System.Console.Out;
  // messages are sent to the console during extract
  zip.ExtractAll();
}
```

VB.NET

```
Using zip As ZipFile = ZipFile.Read(FilePath)
  zip.StatusMessageTextWriter= System.Console.Out
  'Status Messages will be sent to the console durin
  zip.ExtractAll()
End Using
```

In this example, a Windows Forms application instantiates a ZipFile then sets the StatusMessageTextWriter to a StringWriter. At that point, all verbose status messages for that ZipFile are sent to the StringWriter.

```
C# □ Cop
```

```
var sw = new System.IO.StringWriter();
using (ZipFile zip= ZipFile.Read(FilePath))
{
   zip.StatusMessageTextWriter= sw;
   zip.ExtractAll();
}
Console.WriteLine("{0}", sw.ToString());
```

VB.NET

```
Dim sw as New System.IO.StringWriter
Using zip As ZipFile = ZipFile.Read(FilePath)
  zip.StatusMessageTextWriter= sw
  zip.ExtractAll()
End Using
```

'Status Messages are now available in sw

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

F

4

```
Ionic Zip Library v1.9.1.6
Strategy Property

Reference ► Ionic.Zip ► ZipFile ► Strategy
C#
```

The compression strategy to use for all entries.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public CompressionStrategy Strategy { get; set; }

Public Property Strategy As CompressionStrategy
    Get
    Set

public:
property CompressionStrategy Strategy {
    CompressionStrategy get ();
    void set (CompressionStrategy value);
}
```

■ Remarks

Set the Strategy used by the ZLIB-compatible compressor, when compressing entries using the DEFLATE method. Different compression strategies work better on different sorts of data. The strategy parameter can affect the compression ratio and the speed of compression but not the correctness of the compression. For more information see lonic.zlib.compressionStrategy.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Gets or sets the name for the folder to store the temporary file this librar writes when saving a zip archive.

Declaration Syntax

Remarks

This library will create a temporary file when saving a Zip archive to a fil This file is written when calling one of the Save() methods that does not save to a stream, or one of the SaveSelfExtractor() methods.

By default, the library will create the temporary file in the directory specified for the file itself, via the <u>Name</u> property or via the <u>Save(String)</u> method.

Setting this property allows applications to override this default behavior so that the library will create the temporary file in the specified folder. For example, to have the library create the temporary file in the current working directory, regardless where the <code>ZipFile</code> is saved, specfy ".".

To revert to the default behavior, set this property to null (Nothing VB).

When setting the property to a non-null value, the folder specified must exist; if it does not an exception is thrown. The application should have write and delete permissions on the folder. The permissions are not explicitly checked ahead of time; if the application does not have the appropriate rights, an exception will be thrown at the time Save() is called.

There is no temporary file created when reading a zip archive. When saving to a Stream, there is no temporary file created. For example, if the application is an ASP.NET application and calls <code>Save()</code> specifying the <code>Response.OutputStream</code> as the output stream, there is no temporary file created.

Exceptions

Exception	Condition
FileNotFoundException	Thrown when setting the property if the directory does not exist.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ToString Method

Reference ► Ionic.Zip ► ZipFile ► ToString()

C# -

Provides a string representation of the instance.

■ Declaration Syntax

C# Visual Basic Visual C++

public override string ToString()

Public Overrides Function ToString As String

public:

virtual String^ ToString() override

Return Value

a string representation of the instance.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Update Directory Method

Reference ► Ionic.Zip ► ZipFile ► UpdateDirectory()

C#

■ Members

Icon	Member	Description
∄	<u>UpdateDirectory(String)</u>	Add or update a directory in a zip archive.
€₩	<u>UpdateDirectory(String,</u> <u>String)</u>	Add or update a directory in the zip archive at the specified root directory in the archive.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Update Directory Method (directory Name)

Reference ► Ionic.Zip ► ZipFile ► UpdateDirectory(String)

Add or update a directory in a zip archive.

■ Declaration Syntax

C#

Parameters

directoryName (String)

The path to the directory to be added to the zip archive, or updated in the zip archive.

■ Return Value

The ZipEntry corresponding to the Directory that was added or updated.

Remarks

If the specified directory does not exist in the archive, then this method equivalent to calling AddDirectory(). If the specified directory already exists in the archive, then this method updates any existing

entries, and adds any new entries. Any entries that are in the zip archive but not in the specified directory, are left alone. In other words, the contents of the zip file will be a union of the previous contents and the new files.

■ See Also

<u>UpdateFile(String)</u> <u>AddDirectory(String)</u> <u>UpdateItem(String)</u>

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Update Directory Method (directory Name, directory Path In Archive)

Reference ► Ionic.Zip ► ZipFile ► UpdateDirectory(String, String)

Add or update a directory in the zip archive at the specified root director in the archive.

■ Declaration Syntax

Parameters

directoryName (String)

The path to the directory to be added to the zip archive, or updated in the zip archive.

directoryPathInArchive (String)

Specifies a directory path to use to override any path in the directoryName. This path may, or may not, correspond to a re directory in the current filesystem. If the files within the zip are later extracted, this is the path used for the extracted file. Passing null

(Nothing in VB) will use the path on the directoryName, if any. Passing the empty string ("") will insert the item at the root path within the archive.

■ Return Value

The ZipEntry corresponding to the Directory that was added or updated.

Remarks

If the specified directory does not exist in the archive, then this method equivalent to calling AddDirectory(). If the specified directory already exists in the archive, then this method updates any existing entries, and adds any new entries. Any entries that are in the zip archive but not in the specified directory, are left alone. In other words, the contents of the zip file will be a union of the previous contents and the new files.

See Also

<u>UpdateFile(String, String)</u>
<u>AddDirectory(String, String)</u>
<u>UpdateItem(String, String)</u>

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **UpdateEntry Method**Reference ► Ionic.Zip ► ZipFile ► UpdateEntry()

C#

■ Members

Icon	Member	Description
∄	<u>UpdateEntry(String,</u> <u>String)</u>	Updates the given entry in the ZipFile, using the given string as content for the ZipEntry.
∄	UpdateEntry(String, String, Encoding)	Updates the given entry in the ZipFile, using the given string as content for the ZipEntry.
∄ `	UpdateEntry(String, WriteDelegate)	Updates the given entry in the ZipFile, using the given delegate as the source for content for the ZipEntry.
∃	UpdateEntry(String, OpenDelegate, CloseDelegate)	Updates the given entry in the ZipFile, using the given delegate to open and close the stream that provides the content for the ZipEntry.
€	UpdateEntry(String, Stream)	Updates the given entry in the ZipFile, using the given stream a input, and the given filename and given directory Path.

UpdateEntry(String, Byte[])
Updates the given entry in the ZipFile, using the given byte arra as content for the entry.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 UpdateEntry Method (entryName,

opener, closer)

Reference ► <u>lonic.Zip</u> ► <u>ZipFile</u> ► UpdateEntry(String, OpenDelegate, CloseDelegate)

Updates the given entry in the ZipFile, using the given delegates to open and close the stream that provides the content for the ZipEntry

Declaration Syntax

Parameters

entryName (String)

The name, including any path, to use within the archive for the entr

opener (OpenDelegate)

the delegate that will be invoked to open the stream

closer (CloseDelegate)

the delegate that will be invoked to close the stream

Return Value

The ZipEntry added or updated.

■ Remarks

Calling this method is equivalent to removing the ZipEntry for the given file name and directory path, if it exists, and then calling AddEntry(String, OpenDelegate, CloseDelegate). See the documentatic for that method for further explanation.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Update Entry Method (entry Name,

writer)

Reference ► Ionic.Zip ► ZipFile ► UpdateEntry(String,

WriteDelegate)

C# •

Updates the given entry in the ZipFile, using the given delegate as the source for content for the ZipEntry.

Declaration Syntax

```
Public Function UpdateEntry ( _ entryName As String, _ writer As WriteDelegate _ ) As ZipEntry
```

Parameters

entryName (String)

The name, including any path, to use within the archive for the entr *writer* (WriteDelegate)

the delegate which will write the entry content.

■ Return Value

The ZipEntry added.

■ Remarks

Calling this method is equivalent to removing the ZipEntry for the given file name and directory path, if it exists, and then calling AddEntry(String, WriteDelegate). See the documentation for that metho for further explanation.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateEntry Method (entryName, byteContent)

Reference ► Ionic.Zip ► ZipFile ► UpdateEntry(String, Byte[])

Updates the given entry in the ZipFile, using the given byte array as content for the entry.

Declaration Syntax

Parameters

entryName (String)

The name, including any path, to use within the archive for the entr

byteContent (Byte[])

The content to use for the ZipEntry.

■ Return Value

The ZipEntry added.

■ Remarks

Calling this method is equivalent to removing the ZipEntry for the given filename and directory path, if it exists, and then calling AddEntry(String, Byte]). See the documentation for that method for further explanation.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateEntry Method (entryName, stream)

Reference ► Ionic.Zip ► ZipFile ► UpdateEntry(String, Stream)

Updates the given entry in the ZipFile, using the given stream as input, and the given filename and given directory Path.

Declaration Syntax

Parameters

entryName (String)

The name, including any path, to use within the archive for the entr

stream (Stream)

The input stream from which to read file data.

■ Return Value

The ZipEntry added.

■ Remarks

Calling the method is equivalent to calling RemoveEntry() if an ent by the same name already exists, and then calling AddEntry() with the given fileName and stream.

The stream must be open and readable during the call to **ZipFile.Save**. You can dispense the stream on a just-in-time basis using the **InputStream** property. Check the documentation of that property for more information.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

See Also

AddEntry(String, Stream)
InputStream

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateEntry Method (entryName, content)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► **UpdateEntry(String, String)**

Updates the given entry in the ZipFile, using the given string as content for the ZipEntry.

Declaration Syntax

Parameters

entryName (String)

The name, including any path, to use within the archive for the entr *content* (String)

The content of the file, should it be extracted from the zip.

■ Return Value

The ZipEntry added.

■ Remarks

Calling this method is equivalent to removing the ZipEntry for the given file name and directory path, if it exists, and then calling AddEntry(String, String). See the documentation for that method for further explanation. The string content is encoded using the default encoding for the machine, or on Silverlight, using UTF-8. This encoding distinct from the encoding used for the filename itself. See AlternateEncoding.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateEntry Method (entryName, content, encoding)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► **UpdateEntry(String, String, Encoding)**

Updates the given entry in the ZipFile, using the given string as content for the ZipEntry.

Declaration Syntax

Parameters

entryName (String)

The name, including any path, to use within the archive for the entr *content* (String)

The content of the file, should it be extracted from the zip.

encoding (Encoding)

The text encoding to use when encoding the string. Be aware: This is distinct from the text encoding used to encode the filename. See <u>AlternateEncoding</u>.

Return Value

The ZipEntry added.

Remarks

Calling this method is equivalent to removing the ZipEntry for the given file name and directory path, if it exists, and then calling AddEntry(String, String, Encoding). See the documentation for that method for further explanation.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateFile Method

Reference ► Ionic.Zip ► ZipFile ► UpdateFile()

C#

■ Members

Icon	Member	Description
≅	<u>UpdateFile(String)</u>	Adds or Updates a File in a Zip file archive.
∄ 🗳	<u>UpdateFile(String, String)</u>	Adds or Updates a File in a Zip file archive.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateFile Method (fileName)

Reference ► Ionic.Zip ► ZipFile ► UpdateFile(String)

Adds or Updates a File in a Zip file archive.

Declaration Syntax

Parameters

fileName (String)

The name of the file to add or update. It should refer to a file in the filesystem. The name of the file may be a relative path or a fully-qualified path.

■ Return Value

The ZipEntry corresponding to the File that was added or updated.

Remarks

This method adds a file to a zip archive, or, if the file already exists in th zip archive, this method Updates the content of that given filename in th zip archive. The UpdateFile method might more accurately be calle "AddOrUpdateFile".

Upon success, there is no way for the application to learn whether the fi was added versus updated.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

Examples

This example shows how to Update an existing entry in a zipfile. The fire call to UpdateFile adds the file to the newly-created zip archive. The second call to UpdateFile updates the content for that file in the zip archive.

C# □ Cop

```
using (ZipFile zip1 = new ZipFile())
{
    // UpdateFile might more accurately be called "Add zip1.UpdateFile("MyDocuments\\Readme.txt");
    zip1.UpdateFile("CustomerList.csv");
    zip1.Comment = "This zip archive has been created. zip1.Save("Content.zip");
}

using (ZipFile zip2 = ZipFile.Read("Content.zip"))
{
    zip2.UpdateFile("Updates\\Readme.txt");
    zip2.Comment = "This zip archive has been updated: zip2.Save();
}
```

VB.NET

```
Using zip1 As New ZipFile
   ' UpdateFile might more accurately be called "Ad zip1.UpdateFile("MyDocuments\Readme.txt")
   zip1.UpdateFile("CustomerList.csv")
   zip1.Comment = "This zip archive has been create
```

```
zip1.Save("Content.zip")
End Using

Using zip2 As ZipFile = ZipFile.Read("Content.zip")
    zip2.UpdateFile("Updates\Readme.txt")
    zip2.Comment = "This zip archive has been update zip2.Save
End Using
```

■ See Also

AddFile(String)
UpdateDirectory(String)
UpdateItem(String)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

√

Ionic Zip Library v1.9.1.6 UpdateFile Method (fileName, directoryPathInArchive)

Reference ► Ionic.Zip ► ZipFile ► UpdateFile(String, String)

Adds or Updates a File in a Zip file archive.

C# __

■ Declaration Syntax

Parameters

fileName (String)

The name of the file to add or update. It should refer to a file in the filesystem. The name of the file may be a relative path or a fully-qualified path.

directoryPathInArchive (String)

Specifies a directory path to use to override any path in the fileName. This path may, or may not, correspond to a real directory in the current filesystem. If the files within the zip are later extracted, this is the path used for the extracted file. Passing null

(Nothing in VB) will use the path on the fileName, if any. Passing the empty string ("") will insert the item at the root path within the archive.

Return Value

The ZipEntry corresponding to the File that was added or updated.

Remarks

This method adds a file to a zip archive, or, if the file already exists in th zip archive, this method Updates the content of that given filename in th zip archive.

This version of the method allows the caller to explicitly specify the directory path to be used in the archive. The entry to be added or updated is found by using the specified directory path, combined with the basename of the specified filename.

Upon success, there is no way for the application to learn if the file was added versus updated.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

See Also

AddFile(String, String)

UpdateDirectory(String, String)

UpdateItem(String, String)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateFiles Method

Reference ► Ionic.Zip ► ZipFile ► UpdateFiles()

C#

■ Members

Icon	Member	Description
≅	<u>UpdateFiles(IEnumerable<string>)</string></u>	Adds or updates a set of files in the ZipFile.
≡ ₩	<u>UpdateFiles(IEnumerable<string>, String)</string></u>	Adds or updates a set of files to the ZipFile, using the specified directory path in the archive.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateFiles Method (fileNames)

Reference ► Ionic.Zip ► ZipFile ►

UpdateFiles(IEnumerable<String>)

Adds or updates a set of files in the ZipFile.

C#

Declaration Syntax

Parameters

fileNames (<u>IEnumerable</u><<u>String</u>>)

The collection of names of the files to update. Each string should refer to a file in the filesystem. The name of the file may be a relative path or a fully-qualified path.

■ Remarks

Any files that already exist in the archive are updated. Any files that don yet exist in the archive are added.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the

time of this call will be applied to each ZipEntry added.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateFiles Method (fileNames, directoryPathInArchive)

Reference ► Ionic.Zip ► ZipFile ►

UpdateFiles(IEnumerable<String>, String)

Adds or updates a set of files to the **ZipFile**, using the specified directory path in the archive.

Declaration Syntax

IEnumerable<String^>^ fileNames,
String^ directoryPathInArchive

■ Parameters

fileNames (<u>IEnumerable</u><<u>String</u>>)

The names of the files to add or update. Each string should refer to file in the filesystem. The name of the file may be a relative path or fully-qualified path.

directoryPathInArchive (String)

Specifies a directory path to use to override any path in the file name. This path may, or may not, correspond to a real directory in

the current filesystem. If the files within the zip are later extracted, this is the path used for the extracted file. Passing null (Nothing in VB) will use the path on each of the fileNames, i any. Passing the empty string ("") will insert the item at the root path within the archive.

■ Remarks

Any files that already exist in the archive are updated. Any files that don yet exist in the archive are added.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to each ZipEntry added.

See Also

AddSelectedFiles(String, String)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateItem Method

Reference ► Ionic.Zip ► ZipFile ► UpdateItem()

C#

■ Members

Icon	Member	Description
ĕ	<u>UpdateItem(String)</u>	Add or update a file or directory in the zip archive.
=	<u>UpdateItem(String, String)</u>	Add or update a file or directory.

Send comments on this topic to ${\underline{\sf dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateItem Method (itemName)

Reference ► Ionic.Zip ► ZipFile ► UpdateItem(String) C#

Add or update a file or directory in the zip archive.

■ Declaration Syntax

Parameters

itemName (String)

the path to the file or directory to be added or updated.

Remarks

This is useful when the application is not sure or does not care if the iter to be added is a file or directory, and does not know or does not care if the item already exists in the ZipFile. Calling this method is equivalent to calling RemoveEntry() if an entry by the same name already exists, followed calling by AddItem().

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the

time of this call will be applied to the ZipEntry added.

■ See Also

AddItem(String)
UpdateFile(String)
UpdateDirectory(String)

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 UpdateItem Method (itemName, directoryPathInArchive)

Reference ► Ionic.Zip ► ZipFile ► UpdateItem(String, String)

Add or update a file or directory.

C#

Declaration Syntax

Parameters

itemName (String)

The path for the File or Directory to be added or updated.

directoryPathInArchive (String)

Specifies a directory path to use to override any path in the itemName. This path may, or may not, correspond to a real directory in the current filesystem. If the files within the zip are later extracted, this is the path used for the extracted file. Passing null (Nothing in VB) will use the path on the itemName, if any. Passing the empty string ("") will insert the item at the root path

within the archive.

Remarks

This method is useful when the application is not sure or does not care the item to be added is a file or directory, and does not know or does no care if the item already exists in the ZipFile. Calling this method is equivalent to calling RemoveEntry(), if an entry by that name exists and then calling AddItem().

This version of the method allows the caller to explicitly specify the directory path to be used for the item being added to the archive. The entry or entries that are added or updated will use the specified <code>DirectoryPathInArchive</code>. Extracting the entry from the archive will result in a file stored in that directory path.

For ZipFile properties including <u>Encryption</u>, <u>Password</u>, <u>SetCompression</u>, <u>ProvisionalAlternateEncoding</u>, <u>ExtractExistingFile</u>, <u>ZipErrorAction</u>, and <u>CompressionLevel</u>, their respective values at the time of this call will be applied to the ZipEntry added.

See Also

AddItem(String, String)

UpdateFile(String, String)

UpdateDirectory(String, String)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UpdateSelectedFiles Method (selectionCriteria, directoryOnDisk, directoryPathInArchive, recurseDirectories)

Reference ► <u>Ionic.Zip</u> ► <u>ZipFile</u> ► UpdateSelectedFiles(String, String, String, Boolean)

Updates the ZipFile with a selection of files from the disk that conform to the specified criteria.

Declaration Syntax

```
C# Visual Basic Visual C++

public void UpdateSelectedFiles(
    string selectionCriteria,
    string directoryOnDisk,
    string directoryPathInArchive,
    bool recurseDirectories
)
```

Parameters

selectionCriteria (String)

The criteria for selection of files to add to the ZipFile.

directoryOnDisk (String)

The path to the directory in the filesystem from which to select files

directoryPathInArchive (String)

Specifies a directory path to use to in place of the directoryOnDisk. This path may, or may not, correspond to a real directory in the current filesystem. If the files within the zip are later extracted, this is the path used for the extracted file. Passing null (nothing in VB) will use the path on the file name, if any; in othe words it would use directoryOnDisk, plus any subdirectory. Passing the empty string ("") will insert the item at the root path within the archive.

recurseDirectories (Boolean)

If true, the method also scans subdirectories for files matching the criteria.

■ Remarks

This method selects files from the specified disk directory that match the specified selection criteria, and Updates the ZipFile with those files, using the specified directory path in the archive. If recurseDirectories is true, files are also selected from subdirectories, and the directory structure in the filesystem is reproduce in the zip archive, rooted at the directory specified by directoryOnDisk. For details on the syntax for the selectionCriter parameter, see AddSelectedFiles(String).

See Also

AddSelectedFiles(String, String, String, Boolean)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UseUnicodeAsNecessary Property

Reference ► Ionic.Zip ► ZipFile ► UseUnicodeAsNecessary

C# . . .

Indicates whether to encode entry filenames and entry comments using Unicode (UTF-8).

Declaration Syntax

```
C# Visual Basic Visual C++

[ObsoleteAttribute("Beginning with v1.9.1.6 of DotNe public bool UseUnicodeAsNecessary { get; set; }

<ObsoleteAttribute("Beginning with v1.9.1.6 of DotNe Public Property UseUnicodeAsNecessary As Boolean Get Set
```

```
[ObsoleteAttribute(L"Beginning with v1.9.1.6 of DotN
public:
property bool UseUnicodeAsNecessary {
        bool get ();
        void set (bool value);
}
```

■ Remarks

The PKWare zip specification provides for encoding file names and file comments in either the IBM437 code page, or in UTF-8. This flag select the encoding according to that specification. By default, this flag is false and filenames and comments are encoded into the zip file in the IBM43 codepage. Setting this flag to true will specify that filenames and comments that cannot be encoded with IBM437 will be encoded with UTF-8.

Zip files created with strict adherence to the PKWare specification with respect to UTF-8 encoding can contain entries with filenames containing any combination of Unicode characters, including the full range of

characters from Chinese, Latin, Hebrew, Greek, Cyrillic, and many othe alphabets. However, because at this time, the UTF-8 portion of the PKWare specification is not broadly supported by other zip libraries and utilities, such zip files may not be readable by your favorite zip tool or archiver. In other words, interoperability will decrease if you set this flag to true.

In particular, Zip files created with strict adherence to the PKWare specification with respect to UTF-8 encoding will not work well with Explorer in Windows XP or Windows Vista, because Windows compressed folders, as far as I know, do not support UTF-8 in zip files. Vista can read the zip files, but shows the filenames incorrectly. Unpacking from Windows Vista Explorer will result in filenames that hav rubbish characters in place of the high-order UTF-8 bytes.

Also, zip files that use UTF-8 encoding will not work well with Java applications that use the java.util.zip classes, as of v5.0 of the Java runtime. The Java runtime does not correctly implement the PKWare specification in this regard.

As a result, we have the unfortunate situation that "correct" behavior by the DotNetZip library with regard to Unicode encoding of filenames during creation will result in zip files that are readable by strictly compliant and current tools (for example the most recent release of the commercial WinZip tool); but these zip files will not be readable by various other too or libraries, including Windows Explorer.

The DotNetZip library can read and write zip files with UTF8-encoded entries, according to the PKware spec. If you use DotNetZip for both creating and reading the zip file, and you use UTF-8, there will be no los of information in the filenames. For example, using a self-extractor created by this library will allow you to unpack files correctly with no loss of information in the filenames.

If you do not set this flag, it will remain false. If this flag is false, your ZipFile will encode all filenames and comments using the IBM437 codepage. This can cause "loss of information" on some filenames, but the resulting zipfile will be more interoperable with other utilities. As an example of the loss of information, diacritics can be lost. The o-tilde

character will be down-coded to plain o. The c with a cedilla (Unicode 0xE7) used in Portugese will be downcoded to a c. Likewise, the Ostroke character (Unicode 248), used in Danish and Norwegian, will be down-coded to plain o. Chinese characters cannot be represented in codepage IBM437; when using the default encoding, Chinese character in filenames will be represented as ?. These are all examples of "information loss".

The loss of information associated to the use of the IBM437 encoding is inconvenient, and can also lead to runtime errors. For example, using IBM437, any sequence of 4 Chinese characters will be encoded as ??? If your application creates a ZipFile, then adds two files, each with names of four Chinese characters each, this will result in a duplicate filename exception. In the case where you add a single file with a name containing four Chinese characters, calling Extract() on the entry that he question marks in the filename will result in an exception, because the question mark is not legal for use within filenames on Windows. These are just a few examples of the problems associated to loss of information

This flag is independent of the encoding of the content within the entries in the zip file. Think of the zip file as a container - it supports an encodin Within the container are other "containers" - the file entries themselves. The encoding within those entries is independent of the encoding of the zip archive container for those entries.

Rather than specify the encoding in a binary fashion using this flag, an application can specify an arbitrary encoding via the ProvisionalAlternateEncoding property. Setting the encoding explicitly when creating zip archives will result in non-compliant zip files that, curiously, are fairly interoperable. The challenge is, the PKWare specification does not provide for a way to specify that an entry in a zip archive uses a code page that is neither IBM437 nor UTF-8. Therefore i you set the encoding explicitly when creating a zip archive, you must tal care upon reading the zip archive to use the same code page. If you ge it wrong, the behavior is undefined and may result in incorrect filenames exceptions, stomach upset, hair loss, and acne.

■ See Also

ProvisionalAlternateEncoding

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 UseZip64WhenSaving Property

Reference ► Ionic.Zip ► ZipFile ► UseZip64WhenSaving

Specify whether to use ZIP64 extensions when saving a zip archive.

Declaration Syntax

Remarks

When creating a zip file, the default value for the property is <u>Never</u>. <u>AsNecessary</u> is safest, in the sense that you will not get an Exception if pre-ZIP64 limit is exceeded.

You may set the property at any time before calling Save().

When reading a zip file via the Zipfile.Read() method, DotNetZi will properly read ZIP64-endowed zip archives, regardless of the value this property. DotNetZip will always read ZIP64 archives. This property governs only whether DotNetZip will write them. Therefore, when updating archives, be careful about setting this property after reading ar archive that may use ZIP64 extensions.

An interesting question is, if you have set this property to AsNecessary, and then successfully saved, does the resulting

archive use ZIP64 extensions or not? To learn this, check the OutputUsedZip64 property, after calling Save().

Have you thought about donating?

■ See Also

RequiresZip64

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
ZipError Event

Reference ► Ionic.Zip ► ZipFile ► ZipError
C#
```

An event that is raised when an error occurs during open or read of files while saving a zip archive.

Declaration Syntax

```
C# Visual Basic Visual C++

public event <u>EventHandler<ZipErrorEventArgs> ZipErrorEventArgs> ZipErrorEventArgs> ZipErrorEventArgs> ZipErrorEventArgs> ZipErrorEventArgs  

public:
    event <u>EventHandler<ZipErrorEventArgs</u>> ZipError {
        void add (<u>EventHandler<ZipErrorEventArgs</u>> void remove (<u>EventHandler</u><ZipErrorEventArgs> }
}</u>
```

■ Remarks

Errors can occur as a file is being saved to the zip archive. For example the File.Open may fail, or a File.Read may fail, because of lock conflicts or other reasons. If you add a handler to this event, you can handle suc errors in your own code. If you don't add a handler, the library will throw an exception if it encounters an I/O error during a call to Save().

Setting a handler implicitly sets <u>ZipErrorAction</u> to ZipErrorAction. InvokeErrorEvent.

The handler you add applies to all <u>ZipEntry</u> items that are subsequently added to the <u>ZipFile</u> instance. If you set this property after you have added items to the <u>ZipFile</u>, but before you have called <u>Save()</u>, errors that occur while saving those items will not cause the error handle to be invoked.

If you want to handle any errors that occur with any entry in the zip file using the same error handler, then add your error handler once, before adding any entries to the zip archive.

In the error handler method, you need to set the ZipErrorAction property on the ZipErrorEventArgs. CurrentEntry. This communicates back to DotNetZip what you would like to do with this particular error. Within an error handler, if you set the ZipEntry.ZipErrorAction property on the ZipEntry to ZipErrorAction.InvokeErrorEvent or if you don't set it at all, the library will throw the exception. (It is the same as if you had set the ZipEntry.ZipErrorAction property on the ZipEntry to ZipErrorAction.Throw.) If you set the ZipErrorEventArgs.Cancel to true, the entire Save() will be canceled.

In the case that you use <code>ZipErrorAction.Skip</code>, implying that yo want to skip the entry for which there's been an error, <code>DotNetZip</code> tries to seek backwards in the output stream, and truncate all bytes written on behalf of that particular entry. This works only if the output stream is seekable. It will not work, for example, when using <code>ASPNET</code>'s Response.OutputStream.

Examples

This example shows how to use an event handler to handle errors durin save of the zip file.

```
public static void MyZipError(object sender, ZipErro
{
    Console.WriteLine("Error saving {0}...", e.FileN
    Console.WriteLine(" Exception: {0}", e.excepti
    ZipEntry entry = e.CurrentEntry;
    string response = null;
    // Ask the user whether he wants to skip this er
    do
    {
```

```
Console.Write("Retry, Skip, Throw, or Cancel
        response = Console.ReadLine();
        Console.WriteLine();
    } while (response != null &&
             response[0]!='S' && response[0]!='s' &&
             response[0]!='R' && response[0]!='r'
                                                   &&
             response[0]!='T' && response[0]!='t' &&
             response[0]!='C' && response[0]!='c');
    e.Cancel = (response[0]=='C' || response[0]=='c'
    if (response[0]=='S' || response[0]=='s')
        entry.ZipErrorAction = ZipErrorAction.Skip;
    else if (response[0]=='R' || response[0]=='r')
        entry.ZipErrorAction = ZipErrorAction.Retry;
    else if (response[0]=='T' || response[0]=='t')
        entry.ZipErrorAction = ZipErrorAction.Throw;
}
public void SaveTheFile()
{
  string directoryToZip = "fodder";
  string directoryInArchive = "files";
  string zipFileToCreate = "Archive.zip";
 using (var zip = new ZipFile())
  {
    // set the event handler before adding any entri
    zip.ZipError += MyZipError;
    zip.AddDirectory(directoryToZip, directoryInArch
    zip.Save(zipFileToCreate);
 }
}
```

VB.NET

Private Sub MyZipError(ByVal sender As Object, ByVal
' At this point, the application could prompt th
' But in this case, this application will simply

```
Console.WriteLine("Zip Error, entry {0}", e.Cur
                         Exception: {0}", e.excepti
    Console.WriteLine("
      set the desired ZipErrorAction on the CurrentE
    e.CurrentEntry.ZipErrorAction = Zip.ZipErrorActi
End Sub
Public Sub SaveTheFile()
    Dim directoryToZip As String = "fodder"
    Dim directoryInArchive As String = "files"
    Dim zipFileToCreate as String = "Archive.zip"
    Using zipArchive As ZipFile = New ZipFile
          set the event handler before adding any en
        AddHandler zipArchive.ZipError, AddressOf My
        zipArchive.AddDirectory(directoryToZip, dire
        zipArchive.Save(zipFileToCreate)
    End Using
End Sub
```

See Also

ZipErrorAction

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

4

The action the library should take when an error is encountered while opening or reading files as they are saved into a zip archive.

Declaration Syntax

Remarks

Errors can occur as a file is being saved to the zip archive. For example the File.Open may fail, or a File.Read may fail, because of lock conflicts or other reasons.

The first problem might occur after having called AddDirectory() on a directory that contains a Clipper .dbf file; the file is locked by Clipper and cannot be opened for read by another process. An example of the second problem might occur when trying to zip a .pst file that is in use b Microsoft Outlook. Outlook locks a range on the file, which allows other processes to open the file, but not read it in its entirety.

This property tells DotNetZip what you would like to do in the case of these errors. The primary options are: ZipErrorAction. Throw to throw an exception (this is the default behavior if you don't set this

property); ZipErrorAction.Skip to Skip the file for which there was an error and continue saving; ZipErrorAction.Retry to Retry the entry that caused the problem; or ZipErrorAction.InvokeErrorEvent to invoke an event handler.

This property is implicitly set to

ZipErrorAction. InvokeErrorEvent if you add a handler to the ZipError event. If you set this property to something other than ZipErrorAction. InvokeErrorEvent, then the ZipError event is implicitly cleared. What it means is you can set one or the other (or neither), depending on what you want, but you never need to set both.

As with some other properties on the ZipFile class, like <u>Password</u>, <u>Encryption</u>, and <u>CompressionLevel</u>, setting this property on a ZipFile instance will cause the specified ZipErrorAction to be used on al <u>ZipEntry</u> items that are subsequently added to the ZipFile instance. you set this property after you have added items to the ZipFile, but before you have called Save(), those items will not use the specified error handling action.

If you want to handle any errors that occur with any entry in the zip file in the same way, then set this property once, before adding any entries to the zip archive.

If you set this property to ZipErrorAction. Skip and you'd like to learn which files may have been skipped after a Save(), you can set the <u>StatusMessageTextWriter</u> on the ZipFile before calling Save(). A message will be emitted into that writer for each skipped file, if any.

Examples

This example shows how to tell DotNetZip to skip any files for which an error is generated during the Save().

VB.NET

Public Sub SaveZipFile()

```
Dim SourceFolder As String = "fodder"
Dim DestFile As String = "eHandler.zip"
Dim sw as New StringWriter
Using zipArchive As ZipFile = New ZipFile
    ' Tell DotNetZip to skip any files for which
    zipArchive.ZipErrorAction = ZipErrorAction.S
    zipArchive.StatusMessageTextWriter = sw
    zipArchive.AddDirectory(SourceFolder)
    zipArchive.Save(DestFile)
End Using
    ' examine sw here to see any messages
End Sub
```

See Also

ZipErrorAction ZipError

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 ZipInputStream Class

Reference ► Ionic.Zip ► ZipInputStream

C#

Provides a stream metaphor for reading zip files.

■ Declaration Syntax

C# Visual Basic Visual C++

public class ZipInputStream : Stream

Public Class ZipInputStream _ Inherits Stream

public ref class ZipInputStream : public Stream

■ Members

All M	embers	Constructors	Me	thods	Properties	
✓ Public✓ Protected			✓ Instance✓ Static		Declare Inherite	
lcon	Member			Description		
± ∅	ZipInput	Stream(Stream)			ipInputStr t around an exis	-
Ξ Φ	ZipInputStream(String)		Create a ZipInputStream, given the name of an existing zip file.			
∉	ZipInput(Boolean)	Stream(Stream,			LipInputStr pecifying whethe	

		keep the underlying stream open.
≅©	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous read operation.
∃	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	(Inherited from <u>Stream</u> .) HostProtectionAttribute. Begins an asynchronous write operation. (Inherited from <u>Stream</u> .)
	CanRead	Always returns true. (Overrides <u>Stream</u> . <u>CanRead</u> .)
	CanSeek	Returns the value of CanSeek fo the underlying (wrapped) stream. (Overrides <u>Stream.CanSeek.</u>)
	CanTimeout	Gets a value that determines whether the current stream can time out. (Inherited from <u>Stream</u> .)
	<u>CanWrite</u>	Always returns false. (Overrides <u>Stream</u> . <u>CanWrite</u> .)
∃	<u>Close()</u>	Closes the current stream and releases any resources (such as sockets and file handles) associated with the current stream (Inherited from Stream .)
	CodecBufferSize	Size of the work buffer to use for the ZLIB codec during

		decompression.
₫ 🗳	CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object (Inherited from MarshalByRefObject.)
ē óo	CreateWaitHandle()	Obsolete. Allocates a <u>WaitHandle</u> object. (Inherited from <u>Stream</u> .)
ĕ	Dispose(Boolean)	Dispose the stream. (Overrides Stream.Dispose(Boolean).)
∄	Dispose()	Releases all resources used by the <u>Stream</u> . (Inherited from <u>Stream</u> .)
≡	EndRead(IAsyncResult)	Waits for the pending asynchronous read to complete. (Inherited from <u>Stream</u> .)
≡ ♠	EndWrite(IAsyncResult)	Ends an asynchronous write operation. (Inherited from <u>Stream</u> .)
≟	Equals(Object)	Object is equal to the current Object.
		(Inherited from Object.)

9 P	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
₫♦	Flush()	This is a no-op. (Overrides <u>Stream.Flush()</u> .)
≡©	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
≡ ♦	GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
≡	GetNextEntry()	Read the next entry from the zip file.
≡©	GetType()	Gets the <u>Type</u> of the current instance. (Inherited from <u>Object</u> .)
≅₩	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
	Length	Returns the length of the underlying stream.

		(Overrides <u>Stream</u> . <u>Length</u> .)
ē	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object. (Inherited from
		MarshalByRefObject.)
Ģ [™]	MemberwiseClone()	Creates a shallow copy of the current Object.
		(Inherited from Object.)
	Password	Sets the password to be used on the ZipInputStream instance
	Position	Gets or sets the position of the underlying stream. (Overrides <u>Stream.Position.</u>)
	<u>ProvisionalAlternateEncoding</u>	The text encoding to use when reading entries into the zip archive for those entries whose filenames or comments cannot be encoded with the default (IBM437) encoding
ã ù	Read(Byte[], Int32, Int32)	Read the data from the stream into the buffer. (Overrides <u>Stream.Read(Byte[], Int32, Int32).</u>)
≡ ∳	ReadByte()	Reads a byte from the stream and advances the position within the stream by one byte, or returns -1 is at the end of the stream.

		(Inherited from <u>Stream</u> .)
	ReadTimeout	Gets or sets a value, in miliseconds, that determines how long the stream will attempt to reabefore timing out. (Inherited from Stream .)
∄	Seek(Int64, SeekOrigin)	This method seeks in the underlying stream. (Overrides <u>Stream.Seek(Int64, SeekOrigin)</u> .)
∃	SetLength(Int64)	This method always throws a NotSupportedException. (Overrides Stream.SetLength(Int64).)
=0	ToString()	Provides a string representation of the instance. (Overrides Object.ToString().)
∄	Write(Byte[], Int32, Int32)	This method always throws a NotSupportedException. (Overrides <u>Stream.Write(Byte[], Int32, Int32</u>).)
≓	WriteByte(Byte)	Writes a byte to the current positio in the stream and advances the position within the stream by one byte. (Inherited from Stream .)
	WriteTimeout	Gets or sets a value, in miliseconds, that determines how long the stream will attempt to writ

before timing out.
(Inherited from <u>Stream</u> .)

Remarks

This class provides an alternative programming model for reading zip files to the one enabled by the <u>ZipFile</u> class. Use this when reading zip files, as an alternative to the <u>ZipFile</u> class, when you would like to use a Stream class to read the file.

Some application designs require a readable stream for input. This stream can be used to read a zip file, and extract entries.

Both the ZipInputStream class and the ZipFile class can be used to read and extract zip files. Both of them support many of the common zip features, including Unicode, different compression levels, and ZIP64. The programming models differ. For example, when extracting entries via calls to the GetNextEntry() and Read() methods on the ZipInputStream class, the caller is responsible fo creating the file, writing the bytes into the file, setting the attributes on the file, and setting the created, last modified, and last accessed timestamp on the file. All of these things are done automatically by a call to ZipEntry.Extract(). For this reason, the ZipInputStream is generall recommended for when your application wants to extract the data, without storing that data into a file.

Aside from the obvious differences in programming model, there are some differences in capability between the ZipFile class and the ZipInputStream class.

- ZipFile can be used to create or update zip files, or read and extract zip files. ZipInputStream can be used only to read an extract zip files. If you want to use a stream to create zip files, chec out the ZipOutputStream.
- ZipInputStream cannot read segmented or spanned zip files.

- ZipInputStream will not read Zip file comments.
- When reading larger files, ZipInputStream will always underperform ZipFile. This is because the ZipInputStreadoes a full scan on the zip file, while the ZipFile class reads the central directory of the zip file.

■ Inheritance Hierarchy

Object

MarshalByRefObject

Stream

ZipInputStream

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Input Stream Constructor

Reference ► Ionic.Zip ► ZipInputStream ► ZipInputStream()

C#

■ Members

Icon	Member	Description
∃ ₩	ZipInputStream(Stream)	Create a ZipInputStream, wrapping it around an existing stream.
≡	ZipInputStream(String)	Create a ZipInputStream, give the name of an existing zip file.
₫ 🏟	ZipInputStream(Stream, Boolean)	Create a ZipInputStream, explicitly specifying whether to keep the underlying stream open.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZipInputStream Constructor

(stream)

Reference ► Ionic.Zip ► ZipInputStream ► ZipInputStream(Stream)

Create a ZipInputStream, wrapping it around an existing stream.

Declaration Syntax

Parameters

stream (Stream)

The stream to read. It must be readable. This stream will be closed at the time the ZipInputStream is closed.

Remarks

While the <u>ZipFile</u> class is generally easier to use, this class provides an alternative to those applications that want to read from a zipfile directly, using a <u>Stream</u>.

Both the ZipInputStream class and the ZipFile class can be used to read and extract zip files. Both of them support many of the

common zip features, including Unicode, different compression levels, and ZIP64. The programming models differ. For example, when extracting entries via calls to the <code>GetNextEntry()</code> and <code>Read()</code> methods on the <code>ZipInputStream</code> class, the caller is responsible fo creating the file, writing the bytes into the file, setting the attributes on the file, and setting the created, last modified, and last accessed timestamp on the file. All of these things are done automatically by a call to <code>ZipEntry.Extract()</code>. For this reason, the <code>ZipInputStream</code> is generall recommended for when your application wants to extract the data, without storing that data into a file.

Aside from the obvious differences in programming model, there are some differences in capability between the ZipFile class and the ZipInputStream class.

- ZipFile can be used to create or update zip files, or read and extract zip files. ZipInputStream can be used only to read an extract zip files. If you want to use a stream to create zip files, chec out the ZipOutputStream.
- ZipInputStream cannot read segmented or spanned zip files.
- ZipInputStream will not read Zip file comments.
- When reading larger files, ZipInputStream will always underperform ZipFile. This is because the ZipInputStrea does a full scan on the zip file, while the ZipFile class reads the central directory of the zip file.

Examples

This example shows how to read a zip file, and extract entries, using the ZipInputStream class.

```
C#
private void Unzip()
{
   byte[] buffer= new byte[2048];
   int n;
```

```
using (var raw = File.Open(inputFileName, FileMo
    {
        using (var input= new ZipInputStream(raw))
        {
            ZipEntry e;
            while (( e = input.GetNextEntry()) != nu
            {
                if (e.IsDirectory) continue;
                string outputPath = Path.Combine(ext
                using (var output = File.Open(output
                {
                    while ((n= input.Read(buffer, 0,
                     {
                         output.Write(buffer, 0, n);
                     }
                }
            }
        }
    }
}
```

VB.NET

```
Private Sub UnZip()

Dim inputFileName As String = "MyArchive.zip"

Dim extractDir As String = "extract"

Dim buffer As Byte() = New Byte(2048) {}

Using raw As FileStream = File.Open(inputFileNam Using input As ZipInputStream = New ZipInput Dim e As ZipEntry

Do While (Not e = input.GetNextEntry Is

If Not e.IsDirectory Then

Using output As FileStream = Fil

Dim n As Integer

Do While (n = input.Read(buf output.Write(buffer, 0, Loop End Using)
```

End If

Loop

End Using

End Using

End Sub

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 Zip Input Stream Constructor

(stream, leaveOpen)

Reference ► Ionic.Zip ► ZipInputStream ► ZipInputStream(Stream, **Boolean)**

C#

Create a ZipInputStream, explicitly specifying whether to keep the underlying stream open.

Declaration Syntax

```
C#
                   Visual Basic
                                       Visual C++
public ZipInputStream(
         Stream stream,
         bool leaveOpen
```

```
Public Sub New (
       stream As Stream, _
       leaveOpen As Boolean
)
```

```
public:
ZipInputStream(
        Stream^ stream,
        bool leaveOpen
```

Parameters

stream (Stream)

The stream to read from. It must be readable.

leaveOpen (Boolean)

true if the application would like the stream to remain open after the ZipInputStream has been closed.

■ Remarks

See the documentation for the ZipInputStream(Stream) constructor for a discussion of the class, and an example of how to use the class.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZipInputStream Constructor

(fileName)

Reference ► Ionic.Zip ► ZipInputStream ► ZipInputStream(String)

Create a ZipInputStream, given the name of an existing zip file.

Declaration Syntax

Parameters

fileName (String)

The name of the filesystem file to read.

■ Remarks

This constructor opens a FileStream for the given zipfile, and wraps a ZipInputStream around that. See the documentation for the ZipInputStream(Stream) constructor for full details.

While the <u>ZipFile</u> class is generally easier to use, this class provides an alternative to those applications that want to read from a zipfile directly, using a <u>Stream</u>.

Examples

This example shows how to read a zip file, and extract entries, using the ZipInputStream class.

C# □ Cop

```
private void Unzip()
{
    byte[] buffer= new byte[2048];
    int n;
    using (var input= new ZipInputStream(inputFileNa
    {
        ZipEntry e;
        while (( e = input.GetNextEntry()) != null)
        {
            if (e.IsDirectory) continue;
            string outputPath = Path.Combine(extract
            using (var output = File.Open(outputPath
            {
                while ((n= input.Read(buffer, 0, buf
                {
                    output.Write(buffer, 0, n);
                }
            }
        }
   }
}
```

VB.NET

```
Dim n As Integer
Do While (n = input.Read(buffer,
output.Write(buffer, 0, n)
Loop
End Using
End If
Loop
End Using
End Sub
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.Zip ► ZipInputStream ► CanRead C#

Always returns true.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:

virtual property bool CanRead {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Returns the value of CanSeek for the underlying (wrapped) stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:
    virtual property bool CanSeek {
        bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanWrite Property

Reference ► Ionic.Zip ► ZipInputStream ► CanWrite C#

Always returns false.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
virtual property bool CanWrite {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Codec Buffer Size Property

Reference ► Ionic.Zip ► ZipInputStream ► CodecBufferSize

Size of the work buffer to use for the ZLIB codec during decompression

Declaration Syntax

Remarks

Setting this affects the performance and memory efficiency of compression and decompression. For larger files, setting this to a larger size may improve performance, but the exact numbers vary depending on available memory, and a bunch of other variables. I don't have good firm recommendations on how to set it. You'll have to test it yourself. Or just leave it alone and accept the default.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Dispose Method (disposing)

Reference ► Ionic.Zip ► ZipInputStream ► Dispose(Boolean)

Dispose the stream.

Declaration Syntax

Parameters

disposing (Boolean)

true if the Dispose method was invoked by user code.

■ Remarks

This method disposes the ZipInputStream. It may also close the underlying stream, depending on which constructor was used.

Typically the application will call Dispose() implicitly, via a using statement in C#, or a Using statement in VB.

Application code won't call this code directly. This method may be invoked in two distinct scenarios. If disposing == true, the method has been called directly or indirectly by a user's code, for example via the

public Dispose() method. In this case, both managed and unmanaged resources can be referenced and disposed. If disposing == false, the method has been called by the runtime from inside the object finalizer and this method should not reference other objects; in that case only unmanaged resources must be referenced or disposed.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Method

Reference ► Ionic.Zip ► ZipInputStream ► Flush()

C#

This is a no-op.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

public:

virtual void Flush() override

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **GetNextEntry Method**

Reference ► Ionic.Zip ► ZipInputStream ► GetNextEntry()

Read the next entry from the zip file.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public ZipEntry GetNextEntry()

Public Function GetNextEntry As ZipEntry

public:
ZipEntry^ GetNextEntry()
```

C#

Return Value

The ZipEntry read. Returns null (or Nothing in VB) if there are no more entries in the zip file.

■ Remarks

Call this method just before calling Read(Byte[], Int32, to positior the pointer in the zip file to the next entry that can be read. Subsequent calls to Read(), will decrypt and decompress the data in the zip file, until Read() returns 0.

Each time you call **GetNextEntry()**, the pointer in the wrapped stream is moved to the next entry in the zip file. If you call **Seek(Int64**, **SeekOrigin)**, and thus re-position the pointer within the file, you will nee to call **GetNextEntry()** again, to insure that the file pointer is positioned at the beginning of a zip entry.

This method returns the ZipEntry. Using a stream approach, you wi read the raw bytes for an entry in a zip file via calls to Read(). Alternatively, you can extract an entry into a file, or a stream, by calling

Extract(), or one of its siblings.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Length Property

Reference ► Ionic.Zip ► ZipInputStream ► Length
C#
```

Returns the length of the underlying stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:

virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Sets the password to be used on the ZipInputStream instance.

Declaration Syntax

```
C# Visual Basic Visual C++

public string Password { set; }

Public WriteOnly Property Password As String Set

public:
property String^ Password {
    void set (String^ value);
}
```

Remarks

When reading a zip archive, this password is used to read and decrypt the entries that are encrypted within the zip file. When entries within a z file use different passwords, set the appropriate password for the entry before the first call to Read () for each entry.

When reading an entry that is not encrypted, the value of this property is ignored.

Examples

This example uses the ZipInputStream to read and extract entries from zip file, using a potentially different password for each entry.

```
byte[] buffer= new byte[2048];
int n;
using (var raw = File.Open(_inputFileName, FileMode.
```

```
{
    using (var input= new ZipInputStream(raw))
    {
        ZipEntry e;
        while (( e = input.GetNextEntry()) != null)
            input.Password = PasswordForEntry(e.File
            if (e.IsDirectory) continue;
            string outputPath = Path.Combine(_extrac
            using (var output = File.Open(outputPath
            {
                while ((n= input.Read(buffer, 0, buffe
                {
                     output.Write(buffer, 0, n);
                }
            }
        }
    }
}
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

F

Gets or sets the position of the underlying stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

■ Remarks

Setting the position is equivalent to calling Seek(value, SeekOrigin.Begin).

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Provisional Alternate Encoding

Property

Reference ► Ionic.Zip ► ZipInputStream ►

Provisional Alternate Encoding

The text encoding to use when reading entries into the zip archive, for those entries whose filenames or comments cannot be encoded with the default (IBM437) encoding.

Declaration Syntax

■ Remarks

In <u>its zip specification</u>, PKWare describes two options for encoding filenames and comments: using IBM437 or UTF-8. But, some archiving tools or libraries do not follow the specification, and instead encode characters using the system default code page. For example, WinRAR when run on a machine in Shanghai may encode filenames with the Big 5 Chinese (950) code page. This behavior is contrary to the Zip specification, but it occurs anyway.

When using DotNetZip to read zip archives that use something other the UTF-8 or IBM437, set this property to specify the code page to use whe

reading encoded filenames and comments for each ZipEntry in the zip file.

This property is "provisional". When the entry in the zip archive is not explicitly marked as using UTF-8, then IBM437 is used to decode filenames and comments. If a loss of data would result from using IBM436 - specifically when encoding and decoding is not reflexive - the codepage specified here is used. It is possible, therefore, to have a give entry with a Comment encoded in IBM437 and a FileName encoded with the specified "provisional" codepage.

When a zip file uses an arbitrary, non-UTF8 code page for encoding, there is no standard way for the reader application - whether DotNetZip WinZip, WinRar, or something else - to know which codepage has been used for the entries. Readers of zip files are not able to inspect the zip f and determine the codepage that was used for the entries contained within it. It is left to the application or user to determine the necessary codepage when reading zip files encoded this way. If you use an incorrect codepage when reading a zipfile, you will get entries with filenames that are incorrect, and the incorrect filenames may even contain characters that are not legal for use within filenames in Window Extracting entries with illegal characters in the filenames will lead to exceptions. It's too bad, but this is just the way things are with code pages in zip files. Caveat Emptor.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

Reference ► Ionic.Zip ► ZipInputStream ► Read(Byte[], Int32, Int32)

C#

Read the data from the stream into the buffer.

Declaration Syntax

```
C# Visual Basic Visual C++

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

```
public:
virtual int Read(
          array<unsigned char>^ buffer,
          int offset,
          int count
) override
```

Parameters

buffer (Byte∏)

The buffer to hold the data read from the stream.

offset (Int32)

the offset within the buffer to copy the first byte read.

count (Int32)

the number of bytes to read.

Return Value

the number of bytes read, after decryption and decompression.

■ Remarks

The data for the zipentry will be decrypted and uncompressed, as necessary, before being copied into the buffer.

You must set the <u>Password</u> property before calling Read() the first tin for an encrypted entry. To determine if an entry is encrypted and require a password, check the <u>ZipEntry.Encryption</u> property.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.Zip ► ZipInputStream ► Seek(Int64, SeekOrigin)

C#

This method seeks in the underlying stream.

Declaration Syntax

```
C# Visual Basic Visual C++

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

```
Public Overrides Function Seek ( _ offset As Long, _ origin As SeekOrigin _ ) As Long
```

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

Parameters

offset (Int64)

the offset point to seek to

origin (SeekOrigin)

the reference point from which to seek

■ Return Value

The new position

■ Remarks

Call this method if you want to seek around within the zip file for randon

access.

Applications can intermix calls to Seek() with calls to GetNextEntry(). After a call to Seek(), GetNextEntry() will get the next ZipEntry that falls after the current position in the input stream. You' on your own for finding out just where to seek in the stream, to get to the various entries.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.Zip ► ZipInputStream ► SetLength(Int64)

This method always throws a NotSupportedException.

1.

■ Declaration Syntax

Parameters

value (Int64) ignored

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ToString Method

Reference ► Ionic.Zip ► ZipInputStream ► ToString() C#

Provides a string representation of the instance.

■ Declaration Syntax

C# Visual Basic Visual C++

public override string ToString()

Public Overrides Function ToString As String

public:

virtual String^ ToString() override

Return Value

a string representation of the instance.

■ Remarks

This can be useful for debugging purposes.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)
```

Reference ► Ionic.Zip ► ZipInputStream ► Write(Byte[], Int32, Int32)

This method always throws a NotSupportedException.

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

```
buffer (Byte[])
  ignored

offset (Int32)
  ignored

count (Int32)
  ignored
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Option Enumeration

Reference ► <u>lonic.Zip</u> ► **ZipOption**

C#

An enum representing the values on a three-way toggle switch for various options in the library. This might be used to specify whether to employ a particular text encoding, or to use ZIP64 extensions, or some other option.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enum ZipOp	otion	

Public Enumeration ZipOption

public enum class ZipOption

Members

Member	Description
Default	The default behavior. This is the same as "Never". (For COM clients, this is a 0 (zero).)
Never	Never use the associated option. (For COM clients, this is a 0 (zero).)
AsNecessary	Use the associated behavior "as necessary." (For COM clients, this is a 1.)
Always	Use the associated behavior Always, whether necessary or not. (For COM clients, this is a 2.)

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Output Stream Class

Reference ► Ionic.Zip ► ZipOutputStream

C#

Provides a stream metaphor for generating zip files.

■ Declaration Syntax

C# Visual Basic Visual C++

public class ZipOutputStream : Stream

Public Class ZipOutputStream _ Inherits <u>Stream</u>

public ref class ZipOutputStream : public Stream

Members

All M	embers	Constructors	Met	hods	Properties	
PublicProtected			nstance static		Declare Inherite	
lcon	Member			Descripti	on	
∃	ZipOutpu	utStream(Stream)			ZipOutputStrean an existing strea	=
± □	ZipOutpu	utStream(String)			ZipOutputStrean a filesystem file.	1 that
=0	ZipOutpu Boolean)	utStream(Stream,)		Create a 2	ZipOutputStrean	1.
	Alternate	Encoding		A Text En	coding to use wh	nen

		encoding the filenames and comments for all the ZipEntry items, during a ZipFile.Save() operation.
	<u>AlternateEncodingUsage</u>	A flag that tells if and when this instance should apply AlternateEncoding to encode the filenames and comments associated to of ZipEntry objects contained within this instance.
≡ ♠	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous read operation. (Inherited from <u>Stream</u> .)
₫ 心	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous write operation. (Inherited from <u>Stream</u> .)
	CanRead	Always returns false. (Overrides <u>Stream.CanRead</u> .)
	CanSeek	Always returns false. (Overrides <u>Stream.CanSeek</u> .)
	CanTimeout	Gets a value that determines whether the current stream can time out. (Inherited from <u>Stream</u> .)
	CanWrite	Always returns true.

		(Overrides <u>Stream.CanWrite</u> .)
Ξ₩	Close()	Closes the current stream and releases any resources (such as sockets and file handles) associated with the current stream. (Inherited from <u>Stream</u> .)
	CodecBufferSize	Size of the work buffer to use for the ZLIB codec during compression.
	Comment	A comment attached to the zip archive.
	CompressionLevel	Sets the compression level to be used for entries subsequently added to the zip archive.
	CompressionMethod	The compression method used or each entry added to the ZipOutputStream.
∃	ContainsEntry(String)	Returns true if an entry by the given name has already been written to the ZipOutputStream.
≡	CreateObjRef(Type)	Creates an object that contains al the relevant information required to generate a proxy used to

		communicate with a remote objec
		(Inherited from MarshalByRefObject.)
ġ ©	CreateWaitHandle()	Obsolete. Allocates a <u>WaitHandle</u> object. (Inherited from <u>Stream</u> .)
₽ S	DefaultEncoding	The default text encoding used in zip archives. It is numeric 437, also known as IBM437.
ē	Dispose(Boolean)	Dispose the stream (Overrides Stream.Dispose(Boolean).)
≡©	Dispose()	Releases all resources used by the <u>Stream</u> . (Inherited from <u>Stream</u> .)
	EnableZip64	Specify whether to use ZIP64 extensions when saving a zip archive.
	Encryption	The Encryption to use for entries added to the ZipOutputStream.
≡	EndRead(IAsyncResult)	Waits for the pending asynchronous read to complete.
= (English (IA B II)	(Inherited from <u>Stream</u> .)
	EndWrite(IAsyncResult)	Ends an asynchronous write

		operation.
		(Inherited from <u>Stream</u> .)
=	Equals(Object)	Object is equal to the current Object.
		(Inherited from Object.)
<u> </u>	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.
		(Inherited from Object.)
≡	Flush()	This is a no-op.
		(Overrides <u>Stream.Flush()</u> .)
≟	GetHashCode()	Serves as a hash function for a particular type.
		(Inherited from Object.)
≘	GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
≟	GetType()	Gets the <u>Type</u> of the current instance.
		(Inherited from Object.)
	<u>IgnoreCase</u>	Whether the ZipOutputStream should use case-insensitive comparisons when checking for uniqueness of zip entries.

∃	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
	Length	Always returns a NotSupportedException. (Overrides <u>Stream.Length.</u>)
⊕	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject.)
Ģ ♥	MemberwiseClone()	Creates a shallow copy of the current <u>Object</u> . (Inherited from <u>Object</u> .)
	OutputUsedZip64	Indicates whether ZIP64 extensions were used when saving the zip archive.
	<u>ParallelDeflateMaxBufferPairs</u>	The maximum number of buffer pairs to use when performing parallel compression.
	<u>ParallelDeflateThreshold</u>	The size threshold for an entry, above which a parallel deflate is used.

	Password	Sets the password to be used on the ZipOutputStream instance.
	Position	Setting this property always returns a NotSupportedException Getting it returns the value of the Position on the underlying stream (Overrides <u>Stream.Position.</u>)
	ProvisionalAlternateEncoding	Obsolete. The text encoding to use when emitting entries into the zip archive, for those entries whose filenames or comments cannot be encoded with the default (IBM437 encoding.
≡ ♠	PutNextEntry(String)	Specify the name of the next entry that will be written to the zip file.
₫∳	Read(Byte[], Int32, Int32)	This method always throws a NotSupportedException. (Overrides Stream.Read(Byte] , Int32 , Int32).)
≡ ©	ReadByte()	Reads a byte from the stream and advances the position within the stream by one byte, or returns -1 at the end of the stream. (Inherited from Stream .)
	ReadTimeout	Gets or sets a value, in

		miliseconds, that determines how long the stream will attempt to read before timing out. (Inherited from <u>Stream</u> .)
=	Seek(Int64, SeekOrigin)	This method always throws a NotSupportedException.
		(Overrides <u>Stream.Seek(Int64, SeekOrigin)</u> .)
≡	SetLength(Int64)	This method always throws a NotSupportedException.
		(Overrides Stream .SetLength(Int64).)
	Strategy	The compression strategy to use for all entries.
	Timestamp	The type of timestamp attached to the ZipEntry.
= Q	ToString()	Provides a string representation c the instance. (Overrides Object.ToString().)
	UseUnicodeAsNecessary	Obsolete. Indicates whether to encode entry filenames and entry comments using Unicode (UTF-8).
=0	Write(Byte[], Int32, Int32)	Write the data from the buffer to the stream.

		(Overrides <u>Stream.Write(Byte[], Int32, Int32</u>).)
≟	WriteByte(Byte)	Writes a byte to the current position in the stream and advances the position within the stream by one byte. (Inherited from <u>Stream</u> .)
	WriteTimeout	Gets or sets a value, in miliseconds, that determines how long the stream will attempt to write before timing out. (Inherited from Stream .)

■ Remarks

This class writes zip files, as defined in the <u>specification for zip files</u> <u>described by PKWare</u>. The compression for this implementation is provided by a managed-code version of Zlib, included with DotNetZip in the classes in the Ionic.Zlib namespace.

This class provides an alternative programming model to the one enable by the <u>ZipFile</u> class. Use this when creating zip files, as an alternative to the <u>ZipFile</u> class, when you would like to use a **Stream** type to write the zip file.

Both the ZipOutputStream class and the ZipFile class can be used to create zip files. Both of them support many of the common zip features, including Unicode, different compression levels, and ZIP64. They provide very similar performance when creating zip files.

The ZipFile class is generally easier to use than ZipOutputStream and should be considered a higher-level interface. For example, when creating a zip file via calls to the PutNextEntry() and Write() methods on the ZipOutputStream class, the caller is responsible for opening the

file, reading the bytes from the file, writing those bytes into the ZipOutputStream, setting the attributes on the ZipEntry, and setting the created, last modified, and last accessed timestamps on the zip entry. All of these things are done automatically by a call to ZipFile.AddFile(). For this reason, the ZipOutputStream is genera recommended for use only when your application emits arbitrary data, not necessarily data from a filesystem file, directly into a zip file, and do so using a Stream metaphor.

Aside from the differences in programming model, there are other differences in capability between the two classes.

- ZipFile can be used to read and extract zip files, in addition to creating zip files. ZipOutputStream cannot read zip files. If you want to use a stream to read zip files, check out the ZipInputStream class.
- ZipOutputStream does not support the creation of segmenter or spanned zip files.
- ZipOutputStream cannot produce a self-extracting archive.

Be aware that the ZipOutputStream class implements the IDisposable interface. In order for ZipOutputStream to produce a valid zip file, you use use it within a using clause (Using in VB), or call the Dispose() method explicitly. See the examples for how to emploa using clause.

Also, a note regarding compression performance: On the desktop .NET Framework, DotNetZip can use a multi-threaded compression implementation that provides significant speed increases on large files, over 300k or so, at the cost of increased memory use at runtime. (The output of the compression is almost exactly the same size). But, the multi-threaded approach incurs a performance hit on smaller files. There's no way for the ZipOutputStream to know whether parallel compression will be beneficial, because the ZipOutputStream does not know how much data you will write through the stream. You may wish to set the ParallelDeflateThreshold property to zero, if you are compressin

large files through ZipOutputStream. This will cause parallel compression to be used, always.

■ Inheritance Hierarchy

Object

MarshalByRefObject

Stream

ZipOutputStream

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Output Stream Constructor

Reference ► Ionic.Zip ► ZipOutputStream ► ZipOutputStream()

C#

■ Members

Icon	Member	Description
∄	ZipOutputStream(Stream)	Create a ZipOutputStream, wrapping an existing stream.
∄	ZipOutputStream(String)	Create a ZipOutputStream that write to a filesystem file.
≡	ZipOutputStream(Stream, Boolean)	Create a ZipOutputStream.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Output Stream Constructor

(stream)

Reference ► Ionic.Zip ► ZipOutputStream ►

ZipOutputStream(Stream)

Create a ZipOutputStream, wrapping an existing stream.

■ Declaration Syntax

Parameters

stream (<u>Stream</u>)

The stream to wrap. It must be writable. This stream will be closed the time the ZipOutputStream is closed.

■ Remarks

The <u>ZipFile</u> class is generally easier to use when creating zip files. The ZipOutputStream offers a different metaphor for creating a zip file, base on the <u>Stream</u> class.

Examples

This example shows how to create a zip file, using the ZipOutputStream

C# □ Cop

```
private void Zipup()
{
    if (filesToZip.Count == 0)
    {
        System.Console.WriteLine("Nothing to do.");
        return;
    }
    using (var raw = File.Open(_outputFileName, File
        using (var output= new ZipOutputStream(raw))
        {
            output.Password = "VerySecret!";
            output.Encryption = EncryptionAlgorithm.
            foreach (string inputFileName in filesTo
            {
                System.Console.WriteLine("file: {0}"
                output.PutNextEntry(inputFileName);
                using (var input = File.Open(inputFi
                {
                    byte[] buffer= new byte[2048];
                     int n;
                    while ((n= input.Read(buffer, 0, b
                     {
                         output.Write(buffer, 0, n);
                    }
                }
            }
       }
    }
}
```

VB.NET

Cop

```
Private Sub Zipup()
    Dim outputFileName As String = "XmlData.zip"
    Dim filesToZip As String() = Directory.GetFiles(
    If (filesToZip.Length = 0) Then
        Console.WriteLine("Nothing to do.")
    Else
        Using raw As FileStream = File.Open(outputFi
            Using output As ZipOutputStream = New Zi
                output.Password = "VerySecret!"
                output.Encryption = EncryptionAlgori
                Dim inputFileName As String
                For Each inputFileName In filesToZip
                    Console.WriteLine("file: {0}", i
                    output.PutNextEntry(inputFileNam
                    Using input As FileStream = File
                        Dim n As Integer
                        Dim buffer As Byte() = New B
                        Do While (n = input.Read(buf
                            output.Write(buffer, 0,
                        Loop
                    End Using
                Next
            End Using
        End Using
    End If
End Sub
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Output Stream Constructor

(stream, leaveOpen)

Reference ► Ionic.Zip ► ZipOutputStream ► ZipOutputStream(Stream, Boolean)

Create a ZipOutputStream.



Declaration Syntax

Parameters

stream (Stream)

The stream to wrap. It must be writable.

leaveOpen (Boolean)

true if the application would like the stream to remain open after the ZipOutputStream has been closed.

■ Remarks

See the documentation for the **ZipOutputStream(Stream)** constructor fo

an example.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Output Stream Constructor

(fileName)

Reference ► <u>lonic.Zip</u> ► <u>ZipOutputStream</u> ► **ZipOutputStream**(String)

Create a ZipOutputStream that writes to a filesystem file.

■ Declaration Syntax

Parameters

fileName (String)

The name of the zip file to create.

Remarks

The <u>ZipFile</u> class is generally easier to use when creating zip files. The ZipOutputStream offers a different metaphor for creating a zip file, base on the <u>Stream</u> class.

Examples

This example shows how to create a zip file, using the ZipOutputStream class.

C# □ Cop

```
private void Zipup()
    if (filesToZip.Count == 0)
    {
        System.Console.WriteLine("Nothing to do.");
        return;
    }
    using (var output= new ZipOutputStream(outputFil
    {
        output.Password = "VerySecret!";
        output.Encryption = EncryptionAlgorithm.WinZ
        foreach (string inputFileName in filesToZip)
        {
            System.Console.WriteLine("file: {0}", in
            output.PutNextEntry(inputFileName);
            using (var input = File.Open(inputFileNa
                                           FileShare.R
            {
                byte[] buffer= new byte[2048];
                int n;
                while ((n= input.Read(buffer, 0, buffe
                {
                    output.Write(buffer, 0, n);
                }
            }
        }
    }
}
```

VB.NET

```
Private Sub Zipup()
    Dim outputFileName As String = "XmlData.zip"
    Dim filesToZip As String() = Directory.GetFiles(
```

```
If (filesToZip.Length = 0) Then
        Console.WriteLine("Nothing to do.")
    Else
        Using output As ZipOutputStream = New ZipOut
            output.Password = "VerySecret!"
            output.Encryption = EncryptionAlgorithm.
            Dim inputFileName As String
            For Each inputFileName In filesToZip
                Console.WriteLine("file: {0}", input
                output.PutNextEntry(inputFileName)
                Using input As FileStream = File.Ope
                    Dim n As Integer
                    Dim buffer As Byte() = New Byte(
                    Do While (n = input.Read(buffer,
                        output.Write(buffer, 0, n)
                    Loop
                End Using
            Next
        End Using
    End If
End Sub
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Alternate Encoding Property

Reference ► Ionic.Zip ► ZipOutputStream ► AlternateEncoding

A Text Encoding to use when encoding the filenames and comments for all the ZipEntry items, during a ZipFile.Save() operation.

■ Declaration Syntax

■ Remarks

Whether the encoding specified here is used during the save depends calternateEncodingUsage.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Alternate Encoding Usage Property

Reference ► Ionic.Zip ► ZipOutputStream ►

AlternateEncodingUsage

A flag that tells if and when this instance should apply AlternateEncodin to encode the filenames and comments associated to of ZipEntry object contained within this instance.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.Zip ► ZipOutputStream ► CanRead C#

Always returns false.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:
virtual property bool CanRead {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Can Seek Property

Reference ► Ionic.Zip ► ZipOutputStream ► CanSeek C#

Always returns false.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:

virtual property bool CanSeek {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanWrite Property

Reference ► Ionic.Zip ► ZipOutputStream ► CanWrite C#

Always returns true.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
virtual property bool CanWrite {
    bool get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Codec Buffer Size Property

Reference ► Ionic.Zip ► ZipOutputStream ► CodecBufferSize

Size of the work buffer to use for the ZLIB codec during compression.

■ Declaration Syntax

Remarks

Setting this may affect performance. For larger files, setting this to a larger size may improve performance, but I'm not sure. Sorry, I don't currently have good recommendations on how to set it. You can test it if you like.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 Comment Property | Reference ► Ionic.Zip ► ZipOutputStream ► Comment C#

A comment attached to the zip archive.

■ Declaration Syntax

■ Remarks

The application sets this property to specify a comment to be embedded into the generated zip archive.

According to <u>PKWARE's zip specification</u>, the comment is not encrypted even if there is a password set on the zip file.

The specification does not describe how to indicate the encoding used a comment string. Many "compliant" zip tools and libraries use IBM437 as the code page for comments; DotNetZip, too, follows that practice. C the other hand, there are situations where you want a Comment to be encoded with something else, for example using code page 950 "Big-5 Chinese". To fill that need, DotNetZip will encode the comment following the same procedure it follows for encoding filenames: (a) if AlternateEncodingUsage is Never, it uses the default encoding (IBM437). (b) if Always, it always uses the

alternate encoding (<u>AlternateEncoding</u>). (c) if <u>AlternateEncodingUsage</u> | AsNecessary, it uses the alternate encoding only if the default encoding is not sufficient for encoding the comment - in other words if decoding the result does not produce the original string. This decision is taken at the time of the call to ZipFile.Save().

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Compression Level Property

Reference ► Ionic.Zip ► ZipOutputStream ► CompressionLevel

Sets the compression level to be used for entries subsequently added to the zip archive.

■ Declaration Syntax

■ Remarks

Varying the compression level used on entries can affect the size-vsspeed tradeoff when compression and decompressing data streams or files.

As with some other properties on the ZipOutputStream class, like Password, and Encryption, setting this property on a ZipOutputStream instance will cause the specified CompressionLevel to be used on all ZipEntry items that are subsequently added to the ZipOutputStream instance.

If you do not set this property, the default compression level is used, which normally gives a good balance of compression efficiency and compression speed. In some tests, using BestCompression can

double the time it takes to compress, while delivering just a small increase in compression efficiency. This behavior will vary with the type of data you compress. If you are in doubt, just leave this setting alone, and accept the default.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compression Method Property

Reference ► Ionic.Zip ► ZipOutputStream ► CompressionMethod

The compression method used on each entry added to the ZipOutputStream.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Contains Entry Method (name)

Reference ► Ionic.Zip ► ZipOutputStream ► ContainsEntry(String)

Returns true if an entry by the given name has already been written to the ZipOutputStream.

Declaration Syntax

```
C# Visual Basic Visual C++

public bool ContainsEntry(
    string name
)

Public Function ContainsEntry ( _
    name As String _
) As Boolean
```

Parameters

name (String)

The name of the entry to scan for.

■ Return Value

true if an entry by the given name has already been written.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 DefaultEncoding Property

Reference ► Ionic.Zip ► ZipOutputStream ► DefaultEncoding

The default text encoding used in zip archives. It is numeric 437, also known as IBM437.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public static Encoding DefaultEncoding { get; }

Public Shared ReadOnly Property DefaultEncoding As E Get

public:
static property Encoding^ DefaultEncoding {
    Encoding^ get ();
}
```

■ See Also

ProvisionalAlternateEncoding

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

Ionic Zip Library v1.9.1.6 Dispose Method (disposing)

Reference ► Ionic.Zip ► ZipOutputStream ► Dispose(Boolean)

C#

Dispose the stream

■ Declaration Syntax

Parameters

disposing (Boolean) set this to true, always.

■ Remarks

This method writes the Zip Central directory, then closes the stream. Th application must call Dispose() (or Close) in order to produce a valid zip file.

Typically the application will call Dispose() implicitly, via a using statement in C#, or a Using statement in VB.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Enable Zip 64 Property

Reference ► Ionic.Zip ► ZipOutputStream ► EnableZip64

Specify whether to use ZIP64 extensions when saving a zip archive.

Declaration Syntax

Remarks

The default value for the property is <u>Never</u>. <u>AsNecessary</u> is safest, in th sense that you will not get an Exception if a pre-ZIP64 limit is exceeded

You must set this property before calling Write().

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Encryption Property

Reference ► Ionic.Zip ► ZipOutputStream ► Encryption

The Encryption to use for entries added to the ZipOutputStream.

Declaration Syntax

Remarks

The specified Encryption is applied to the entries subsequently written to the ZipOutputStream instance.

If you set this to something other than EncryptionAlgorithm.None, you walso need to set the <u>Password</u> to a non-null, non-empty value in order to actually get encryption on the entry.

See Also

ZipOutputStream.Password ZipEntry.Encryption

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Method

Reference ► Ionic.Zip ► ZipOutputStream ► Flush() C#

This is a no-op.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

public:
virtual void Flush() override

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 gnoreCase Property

Reference ► Ionic.Zip ► ZipOutputStream ► IgnoreCase

C#

Whether the ZipOutputStream should use case-insensitive comparisons when checking for uniqueness of zip entries.

Declaration Syntax

■ Remarks

Though the zip specification doesn't prohibit zipfiles with duplicate entries, Sane zip files have no duplicates, and the DotNetZip library cannot create zip files with duplicate entries. If an application attempts t call PutNextEntry(String) with a name that duplicates one already used within the archive, the library will throw an Exception.

This property allows the application to specify whether the ZipOutputStream instance considers ordinal case when checking for uniqueness of zip entries.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

```
Ionic Zip Library v1.9.1.6
Length Property

Reference ► Ionic.Zip ► ZipOutputStream ► Length
C#
```

Always returns a NotSupportedException.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:

virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Output Used Zip 64 Property

Reference ► Ionic.Zip ► ZipOutputStream ► OutputUsedZip64

Indicates whether ZIP64 extensions were used when saving the zip archive.

Declaration Syntax

```
C# Visual Basic Visual C++
public bool OutputUsedZip64 { get; }

Public ReadOnly Property OutputUsedZip64 As Boolean Get

public:
property bool OutputUsedZip64 {
    bool get ();
}
```

■ Remarks

The value is defined only after the ZipOutputStream has been closed.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel Deflate Max Buffer Pairs

Property

Reference ► Ionic.Zip ► ZipOutputStream ►

ParallelDeflateMaxBufferPairs

The maximum number of buffer pairs to use when performing parallel compression.

Declaration Syntax

■ Remarks

This property sets an upper limit on the number of memory buffer pairs create when performing parallel compression. The implementation of th parallel compression stream allocates multiple buffers to facilitate parall compression. As each buffer fills up, the stream uses <a href="https://doi.org/10.2016/nc.201

A higher number of buffer pairs enables a higher degree of parallelism, which tends to increase the speed of compression on multi-cpu computers. On the other hand, a higher number of buffer pairs also

implies a larger memory consumption, more active worker threads, and higher cpu utilization for any compression. This property enables the application to limit its memory consumption and CPU utilization behavio depending on requirements.

For each compression "task" that occurs in parallel, there are 2 buffers allocated: one for input and one for output. This property sets a limit for the number of pairs. The total amount of storage space allocated for buffering will then be (N*S*2), where N is the number of buffer pairs, S i the size of each buffer (CodecBufferSize). By default, DotNetZip allocates 4 buffer pairs per CPU core, so if your machine has 4 cores, and you retain the default buffer size of 128k, then the ParallelDeflateOutputStream will use 4 * 4 * 2 * 128kb of buffer memory in total, or 4mb, in blocks of 128kb. If you then set this property to 8, the the number will be 8 * 2 * 128kb of buffer memory, or 2mb.

CPU utilization will also go up with additional buffers, because a larger number of buffer pairs allows a larger number of background threads to compress in parallel. If you find that parallel compression is consuming too much memory or CPU, you can adjust this value downward.

The default value is 16. Different values may deliver better or worse results, depending on your priorities and the dynamic performance characteristics of your storage and compute resources.

This property is not the number of buffer pairs to use; it is an upper limit An illustration: Suppose you have an application that uses the default value of this property (which is 16), and it runs on a machine with 2 CPL cores. In that case, DotNetZip will allocate 4 buffer pairs per CPU core, for a total of 8 pairs. The upper limit specified by this property has no effect.

The application can set this value at any time, but it is effective only if so before calling ZipOutputStream.Write() for the first time.

■ See Also

ParallelDeflateThreshold

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

•

Ionic Zip Library v1.9.1.6 Parallel Deflate Threshold Property

Reference ► Ionic.Zip ► ZipOutputStream ►

ParallelDeflateThreshold

The size threshold for an entry, above which a parallel deflate is used.

■ Declaration Syntax

■ Remarks

DotNetZip will use multiple threads to compress any ZipEntry, when the CompressionMethod is Deflate, and if the entry is larger than the given size. Zero means "always use parallel deflate", while -1 means "never use parallel deflate".

If the entry size cannot be known before compression, as with any entry added via a ZipOutputStream, then Parallel deflate will never be performed, unless the value of this property is zero.

A parallel deflate operations will speed up the compression of large files on computers with multiple CPUs or multiple CPU cores. For files above 1mb, on a dual core or dual-cpu (2p) machine, the time required to compress the file can be 70% of the single-threaded deflate. For very large files on 4p machines the compression can be done in 30% of the

normal time. The downside is that parallel deflate consumes extra memory during the deflate, and the deflation is slightly less effective.

Parallel deflate tends to not be as effective as single-threaded deflate because the original data stream is split into multiple independent buffers, each of which is compressed in parallel. But because they are treated independently, there is no opportunity to share compression dictionaries, and additional framing bytes must be added to the output stream. For that reason, a deflated stream may be slightly larger when compressed using parallel deflate, as compared to a traditional single-threaded deflate. For files of about 512k, the increase over the normal deflate is as much as 5% of the total compressed size. For larger files, the difference can be as small as 0.1%.

Multi-threaded compression does not give as much an advantage when using Encryption. This is primarily because encryption tends to slow down the entire pipeline. Also, multi-threaded compression gives less of an advantage when using lower compression levels, for example BestSpeed. You may have to perform some tests to determine the best approach for your situation.

The default value for this property is -1, which means parallel compression will not be performed unless you set it to zero.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Sets the password to be used on the ZipOutputStream instance.

Declaration Syntax

```
C# Visual Basic Visual C++

public string Password { set; }

Public WriteOnly Property Password As String Set

public:
property String^ Password {
    void set (String^ value);
}
```

Remarks

When writing a zip archive, this password is applied to the entries, not to the zip archive itself. It applies to any ZipEntry subsequently written to the ZipOutputStream.

Using a password does not encrypt or protect the "directory" of the archive - the list of entries contained in the archive. If you set the Password property, the password actually applies to individual entries that are added to the archive, subsequent to the setting of this property. The list of filenames in the archive that is eventually created will appear in clear text, but the contents of the individual files are encrypted. This is how Zip encryption works.

If you set this property, and then add a set of entries to the archive via calls to PutNextEntry, then each entry is encrypted with that password. You may also want to change the password between adding different entries. If you set the password, add an entry, then set the

password to null (Nothing in VB), and add another entry, the first entry is encrypted and the second is not.

When setting the Password, you may also want to explicitly set the Encryption property, to specify how to encrypt the entries added to the ZipFile. If you set the Password to a non-null value and do not set Encryption, then PKZip 2.0 ("Weak") encryption is used. This encryption is relatively weak but is very interoperable. If you set the password to a null value (Nothing in VB), Encryption is reset to None.

Special case: if you wrap a ZipOutputStream around a non-seekable stream, and use encryption, and emit an entry of zero bytes, the Close() or PutNextEntry() following the entry will throw an exception.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Setting this property always returns a NotSupportedException. Getting i returns the value of the Position on the underlying stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Provisional Alternate Encoding

Property

Reference ► Ionic.Zip ► ZipOutputStream ►

Provisional Alternate Encoding

The text encoding to use when emitting entries into the zip archive, for those entries whose filenames or comments cannot be encoded with the default (IBM437) encoding.

Declaration Syntax

```
C# Visual Basic Visual C++

[ObsoleteAttribute("use AlternateEncoding and Altern public Encoding ProvisionalAlternateEncoding { get;

<ObsoleteAttribute("use AlternateEncoding and Altern Public Property ProvisionalAlternateEncoding As EncoGet Set
```

Remarks

In <u>its zip specification</u>, PKWare describes two options for encoding filenames and comments: using IBM437 or UTF-8. But, some archiving tools or libraries do not follow the specification, and instead encode characters using the system default code page. For example, WinRAR when run on a machine in Shanghai may encode filenames with the Big 5 Chinese (950) code page. This behavior is contrary to the Zip specification, but it occurs anyway.

When using DotNetZip to write zip archives that will be read by one of these other archivers, set this property to specify the code page to use when encoding the FileName and Comment for each ZipEntry in the zip file, for values that cannot be encoded with the default codepage for zip files, IBM437. This is why this property is "provisional". In all cases, IBM437 is used where possible, in other words, where no loss of data would result. It is possible, therefore, to have a given entry with a Comment encoded in IBM437 and a FileName encoded with the specified "provisional" codepage.

Be aware that a zip file created after you've explicitly set the ProvisionalAlternateEncoding property to a value other than IBM437 may not be compliant to the PKWare specification, and may not be readable by compliant archivers. On the other hand, many (most archivers are non-compliant and can read zip files created in arbitrary code pages. The trick is to use or specify the proper codepage when reading the zip.

When creating a zip archive using this library, it is possible to change th value of ProvisionalAlternateEncoding between each entry you add, and between adding entries and the call to Close(). Don't c this. It will likely result in a zipfile that is not readable. For best interoperability, either leave ProvisionalAlternateEncoding alone, or specify it only once, before adding any entries to the ZipOutputStream instance. There is one exception to this recommendation, described later.

When using an arbitrary, non-UTF8 code page for encoding, there is no standard way for the creator application - whether DotNetZip, WinZip, WinRar, or something else - to formally specify in the zip file which codepage has been used for the entries. As a result, readers of zip files are not able to inspect the zip file and determine the codepage that was used for the entries contained within it. It is left to the application or user to determine the necessary codepage when reading zip files encoded this way. If you use an incorrect codepage when reading a zipfile, you w get entries with filenames that are incorrect, and the incorrect filenames may even contain characters that are not legal for use within filenames Windows. Extracting entries with illegal characters in the filenames will

lead to exceptions. It's too bad, but this is just the way things are with code pages in zip files. Caveat Emptor.

One possible approach for specifying the code page for a given zip file i to describe the code page in a human-readable form in the Zip commer For example, the comment may read "Entries in this archive are encode in the Big5 code page". For maximum interoperability, the zip comment this case should be encoded in the default, IBM437 code page. In this case, the zip comment is encoded using a different page than the filenames. To do this, Specify

ProvisionalAlternateEncoding to your desired regionspecific code page, once before adding any entries, and then set the <u>Comment</u> property and reset ProvisionalAlternateEncodin to IBM437 before calling Close().

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 PutNextEntry Method (entryName)

Reference ► Ionic.Zip ► ZipOutputStream ► PutNextEntry(String)

Specify the name of the next entry that will be written to the zip file.

Declaration Syntax

Parameters

entryName (String)

The name of the entry to be added, including any path to be used within the zip file.

■ Return Value

The ZipEntry created.

■ Remarks

Call this method just before calling Write(Byte[], Int32, Int32), to specify the name of the entry that the next set of bytes written to the ZipOutputStream belongs to. All subsequent calls to Write, until the next call to PutNextEntry, will be inserted into the named entry in the zip file.

If the *entryName* used in PutNextEntry() ends in a slash, then the entry added is marked as a directory. Because directory entries do not contain data, a call to Write(), before an intervening additional call to PutNextEntry(), will throw an exception.

If you don't call Write() between two calls to PutNextEntry(), the first entry is inserted into the zip file as a file of zero size. This may k what you want.

Because PutNextEntry() closes out the prior entry, if any, this method may throw if there is a problem with the prior entry.

This method returns the ZipEntry. You can modify public properties on the ZipEntry, such as Encryption, Password, and so on, until the first call to ZipOutputStream.Write(), or until the next call to PutNextEntry(). If you modify the ZipEntryafter having called Write(), you may get a runtime exception, or you may silently get ar invalid zip archive.

Examples

This example shows how to create a zip file, using the ZipOutputStream class.

```
private void Zipup()
{
    using (FileStream fs raw = File.Open(_outputFile {
        using (var output= new ZipOutputStream(fs)) {
            output.Password = "VerySecret!";
            output.Encryption = EncryptionAlgorithm.
            output.PutNextEntry("entry1.txt");
            byte[] buffer= System.Text.Encoding.ASCI output.Write(buffer,0,buffer.Length);
            output.PutNextEntry("entry2.txt"); // t
```

```
output.PutNextEntry("entry3.txt");
buffer= System.Text.Encoding.ASCII.GetBy
output.Write(buffer,0,buffer.Length);
}
}
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

Reference ► Ionic.Zip ► ZipOutputStream ► Read(Byte[], Int32, Int32)

This method always throws a NotSupportedException.

C# -

Declaration Syntax

Parameters

```
buffer (Byte[])
ignored

offset (Int32)
ignored

count (Int32)
ignored
```

■ Return Value

nothing

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.Zip ► ZipOutputStream ► Seek(Int64, SeekOrigin)

This method always throws a NotSupportedException.

Declaration Syntax

```
C# Visual Basic Visual C++

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

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

Parameters

```
offset (Int64)
ignored

origin (SeekOrigin)
ignored
```

■ Return Value

nothing

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.Zip ► ZipOutputStream ► SetLength(Int64)

This method always throws a NotSupportedException.

■ Declaration Syntax

Parameters

value (Int64) ignored

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

The compression strategy to use for all entries.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public CompressionStrategy Strategy { get; set; }

Public Property Strategy As CompressionStrategy
    Get
    Set

public:
property CompressionStrategy Strategy {
    CompressionStrategy get ();
    void set (CompressionStrategy value);
}
```

■ Remarks

Set the Strategy used by the ZLIB-compatible compressor, when compressing data for the entries in the zip archive. Different compressic strategies work better on different sorts of data. The strategy parameter can affect the compression ratio and the speed of compression but not the correctness of the compression. For more information see CompressionStrategy.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Timestamp Property

Reference ► Ionic.Zip ► ZipOutputStream ► Timestamp

The type of timestamp attached to the ZipEntry.

C#

Declaration Syntax

```
C# Visual Basic Visual C++
public ZipEntryTimestamp Timestamp { get; set; }

Public Property Timestamp As ZipEntryTimestamp
    Get
    Set

public:
property ZipEntryTimestamp Timestamp {
    ZipEntryTimestamp get ();
    void set (ZipEntryTimestamp value);
}
```

■ Remarks

Set this in order to specify the kind of timestamp that should be emitted into the zip file for each entry.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ToString Method

Reference ► Ionic.Zip ► ZipOutputStream ► ToString() C#

Provides a string representation of the instance.

■ Declaration Syntax

C# Visual Basic Visual C++

public override string ToString()

Public Overrides Function ToString As String

public:

virtual String^ ToString() override

Return Value

a string representation of the instance.

■ Remarks

This can be useful for debugging purposes.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 UseUnicodeAsNecessary Property

Reference ► Ionic.Zip ► ZipOutputStream ►

UseUnicodeAsNecessary

Indicates whether to encode entry filenames and entry comments using Unicode (UTF-8).

Declaration Syntax

```
C# Visual Basic Visual C++

[ObsoleteAttribute("Beginning with v1.9.1.6 of DotNe public bool UseUnicodeAsNecessary { get; set; }

<ObsoleteAttribute("Beginning with v1.9.1.6 of DotNe Public Property UseUnicodeAsNecessary As Boolean Get Set

[ObsoleteAttribute(L"Beginning with v1.9.1.6 of DotNe public: property bool UseUnicodeAsNecessary { bool get (); void set (bool value);
```

Remarks

}

The PKWare zip specification provides for encoding file names and file comments in either the IBM437 code page, or in UTF-8. This flag select the encoding according to that specification. By default, this flag is false and filenames and comments are encoded into the zip file in the IBM43 codepage. Setting this flag to true will specify that filenames and comments that cannot be encoded with IBM437 will be encoded with UTF-8.

Zip files created with strict adherence to the PKWare specification with respect to UTF-8 encoding can contain entries with filenames containing

any combination of Unicode characters, including the full range of characters from Chinese, Latin, Hebrew, Greek, Cyrillic, and many othe alphabets. However, because at this time, the UTF-8 portion of the PKWare specification is not broadly supported by other zip libraries and utilities, such zip files may not be readable by your favorite zip tool or archiver. In other words, interoperability will decrease if you set this flag to true.

In particular, Zip files created with strict adherence to the PKWare specification with respect to UTF-8 encoding will not work well with Explorer in Windows XP or Windows Vista, because Windows compressed folders, as far as I know, do not support UTF-8 in zip files. Vista can read the zip files, but shows the filenames incorrectly. Unpacking from Windows Vista Explorer will result in filenames that hav rubbish characters in place of the high-order UTF-8 bytes.

Also, zip files that use UTF-8 encoding will not work well with Java applications that use the java.util.zip classes, as of v5.0 of the Java runtime. The Java runtime does not correctly implement the PKWare specification in this regard.

As a result, we have the unfortunate situation that "correct" behavior by the DotNetZip library with regard to Unicode encoding of filenames during zip creation will result in zip files that are readable by strictly compliant and current tools (for example the most recent release of the commercial WinZip tool); but these zip files will not be readable by various other too or libraries, including Windows Explorer.

The DotNetZip library can read and write zip files with UTF8-encoded entries, according to the PKware spec. If you use DotNetZip for both creating and reading the zip file, and you use UTF-8, there will be no los of information in the filenames. For example, using a self-extractor created by this library will allow you to unpack files correctly with no loss of information in the filenames.

If you do not set this flag, it will remain false. If this flag is false, the ZipOutputStream will encode all filenames and comments using the IBM437 codepage. This can cause "loss of information" on some filenames, but the resulting zipfile will be more interoperable with other

utilities. As an example of the loss of information, diacritics can be lost. The o-tilde character will be down-coded to plain o. The c with a cedilla (Unicode 0xE7) used in Portugese will be downcoded to a c. Likewise, the O-stroke character (Unicode 248), used in Danish and Norwegian, will be down-coded to plain o. Chinese characters cannot be represente in codepage IBM437; when using the default encoding, Chinese characters in filenames will be represented as ?. These are all example of "information loss".

The loss of information associated to the use of the IBM437 encoding is inconvenient, and can also lead to runtime errors. For example, using IBM437, any sequence of 4 Chinese characters will be encoded as ??? If your application creates a ZipOutputStream, does not set the encoding, then adds two files, each with names of four Chinese characters each, this will result in a duplicate filename exception. In the case where you add a single file with a name containing four Chinese characters, the zipfile will save properly, but extracting that file later, with any zip tool, will result in an error, because the question mark is not legator use within filenames on Windows. These are just a few examples of the problems associated to loss of information.

This flag is independent of the encoding of the content within the entries in the zip file. Think of the zip file as a container - it supports an encodin Within the container are other "containers" - the file entries themselves. The encoding within those entries is independent of the encoding of the zip archive container for those entries.

Rather than specify the encoding in a binary fashion using this flag, an application can specify an arbitrary encoding via the ProvisionalAlternateEncoding property. Setting the encoding explicitly when creating zip archives will result in non-compliant zip files that, curiously, are fairly interoperable. The challenge is, the PKWare specification does not provide for a way to specify that an entry in a zip archive uses a code page that is neither IBM437 nor UTF-8. Therefore i you set the encoding explicitly when creating a zip archive, you must tal care upon reading the zip archive to use the same code page. If you ge it wrong, the behavior is undefined and may result in incorrect filenames exceptions, stomach upset, hair loss, and acne.

See Also

ProvisionalAlternateEncoding

Send comments on this topic to ${\underline{\sf dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

Reference ► Ionic.Zip ► ZipOutputStream ► Write(Byte[], Int32, Int32)

Write the data from the buffer to the stream.

C#

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

```
public:
virtual void Write(
          array<<u>unsigned char</u>>^ buffer,
          <u>int offset</u>,
          <u>int count</u>
) override
```

Parameters

buffer (Byte[])

The buffer holding data to write to the stream.

offset (Int32)

the offset within that data array to find the first byte to write.

count (Int32)

the number of bytes to write.

■ Remarks

As the application writes data into this stream, the data may be compressed and encrypted before being written out to the underlying stream, depending on the settings of the CompressionLevel and the Encryption properties.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zip Progress Event Args Class

Reference ► Ionic.Zip ► ZipProgressEventArgs

C#

Provides information about the progress of a save, read, or extract operation. This is a base class; you will probably use one of the classes derived from this one.

■ Declaration Syntax

C# Visual Basic Visual C++
public class ZipProgressEventArgs : EventArgs

Public Class ZipProgressEventArgs _ Inherits <u>EventArgs</u>

public ref class ZipProgressEventArgs : public Event

Members

All M	All Members Methods			Properties	
✓ Public✓ Protected			✓ Instance✓ Static	Declare Inherite	
Icon	Icon Member		Description		
	<u>ArchiveName</u>		Returns the archive name associated to this event.		
	BytesTransferred			mber of bytes read or or this entry.	written
	Cancel			vent handler, set this t e or extract operation ss.	

	CurrentEntry	The name of the last entry saved or extracted.
	EntriesTotal	The total number of entries to be saved or extracted.
≡	Equals(Object)	Determines whether the specified Object is equal to the current Object . (Inherited from Object .)
	EventType	The type of event being reported.
Ģ	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
≡	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
∃	GetType()	Gets the <u>Type</u> of the current instance (Inherited from <u>Object</u> .)
Ģ [©]	MemberwiseClone()	Creates a shallow copy of the curren Object . (Inherited from Object .)
=0	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> .

	(Inherited from Object.)
<u>TotalBytesToTransfer</u>	Total number of bytes that will be rea or written for this entry. This number will be -1 if the value cannot be determined.

■ Inheritance Hierarchy

Object

EventArgs

ZipProgressEventArgs

- AddProgressEventArgs
- <u>ExtractProgressEventArgs</u>
- <u>ReadProgressEventArgs</u>
- <u>SaveProgressEventArgs</u>
- ZipErrorEventArgs

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 ArchiveName Property

Reference ► Ionic.Zip ► ZipProgressEventArgs ► ArchiveName

Returns the archive name associated to this event.

C#

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Bytes Transferred Property

Reference ► Ionic.Zip ► ZipProgressEventArgs ► BytesTransferred

The number of bytes read or written so far for this entry.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Cancel Property

Reference ► Ionic.Zip ► ZipProgressEventArgs ► Cancel

C#

In an event handler, set this to cancel the save or extract operation that in progress.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CurrentEntry Property

Reference ► Ionic.Zip ► ZipProgressEventArgs ► CurrentEntry

C#

The name of the last entry saved or extracted.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Entries Total Property

Reference ► Ionic.Zip ► ZipProgressEventArgs ► EntriesTotal

The total number of entries to be saved or extracted.

C#

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 EventType Property

Reference ► Ionic.Zip ► ZipProgressEventArgs ► EventType

The type of event being reported.

■ Declaration Syntax

C#

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Total Bytes To Transfer Property

Reference ► Ionic.Zip ► ZipProgressEventArgs ►

TotalBytesToTransfer

Total number of bytes that will be read or written for this entry. This number will be -1 if the value cannot be determined.

Declaration Syntax

```
C# Visual Basic Visual C++
public long TotalBytesToTransfer { get; set; }

Public Property TotalBytesToTransfer As Long
    Get
    Set

public:
property long long TotalBytesToTransfer {
    long long get ();
    void set (long long value);
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | Zip Progress Event Type Enumeration | Reference ► Ionic.Zip ► Zip Progress Event Type | C#

In an EventArgs type, indicates which sort of progress event is being reported.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enur	m ZipProgressEventTy	rpe
Public Enur	meration ZipProgress	EventType
public enur	m class ZipProgressE	EventType

Members

Member	Description
Adding_Started	Indicates that a Add() oper
Adding_AfterAddEntry	Indicates that an individual been added.
Adding_Completed	Indicates that a Add() oper
Reading_Started	Indicates that a Read() ope
Reading_BeforeReadEntry	Indicates that an individual about to be read.
Reading_AfterReadEntry	Indicates that an individual just been read.
Reading_Completed	Indicates that a Read() ope
Reading_ArchiveBytesRead	The given event reports the far during a Read() operation
Saving_Started	Indicates that a Save() ope
Saving_BeforeWriteEntry	Indicates that an individual

	about to be written.
Saving_AfterWriteEntry	Indicates that an individual just been saved.
Saving_Completed	Indicates that a Save() ope
Saving_AfterSaveTempArchive	Indicates that the zip architemporary location during
Saving_BeforeRenameTempArchive	Indicates that the temporal renamed to the final archiv operation.
Saving_AfterRenameTempArchive	Indicates that the temporal renamed to the final archiv operation.
Saving_AfterCompileSelfExtractor	Indicates that the self-extra compiled during a Save() of
Saving_EntryBytesRead	The given event is reportin bytes that have run throug during a Save() operation.
Extracting_BeforeExtractEntry	Indicates that an entry is a
Extracting_AfterExtractEntry	Indicates that an entry has
Extracting_ExtractEntryWouldOverwrite	Indicates that extraction of an existing filesystem file. ExtractExistingFileAction.I in the call to ZipEntry. receive this event.
Extracting_EntryBytesWritten	The given event is reportin written so far for the currer operation.
Extracting_BeforeExtractAll	Indicates that an ExtractAl begin.
Extracting_AfterExtractAll	Indicates that an ExtractAl
Error_Saving	Indicates that an error has file. This generally means

because it has been removed by another process. It can cannot be Read, because

■ Remarks

There are events for reading, events for saving, and events for extractin This enumeration allows a single EventArgs type to be sued to describe one of multiple subevents. For example, a SaveProgress event is invoked before, after, and during the saving of a single entry. The value an enum with this type, specifies which event is being triggered. The same applies to Extraction, Reading and Adding events.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4

▶

lonic Zip Library v1.9.1.6 lonic.Zlib Namespace Reference ► lonic.Zlib C#

Ionic.Zlib namespace includes classes for doing ZLIB, DEFLATE, and GZIP compression and decompression, according to RFC 1950, RFC 1951, and RFC 1952. It includes stream classes that can directly replac the DeflateStream and GZipStream classes that are included in the Bas Class Library of the .NET Framework.

■ Declaration Syntax

C#	Visual Basic	Visual C++
namespace Ionic.	Zlib	
Namespace Ionic.	Zlib	
namespace Ionic.	Zlib	

Types

All Types		Classes		Enumerations
Icon	Туре		Descript	ion
2	CompressionLe	vel	when usi ZlibStrea	pression level to be useding a DeflateStream or mith sionMode.Compress.
₂ 2	CompressionMo	ode	of transco	to specify the direction oding - whether to s or decompress.
	CompressionSt	rategy		s options for how the sion algorithm is

		executed. Different strategies work better on different sorts of data. The strategy parameter car affect the compression ratio and the speed of compression but not the correctness of the compression.
9 \$	<u>DeflateStream</u>	A class for compressing and decompressing streams using the Deflate algorithm.
(g)31	FlushType	Describes how to flush the currer deflate operation.
9 3	<u>GZipStream</u>	A class for compressing and decompressing GZIP streams.
4 \$	ParallelDeflateOutputStream	A class for compressing streams using the Deflate algorithm with multiple threads.
* \$	ZlibCodec	Encoder and Decoder for ZLIB and DEFLATE (IETF RFC1950 and RFC1951).
P	<u>ZlibConstants</u>	A bunch of constants used in the Zlib interface.
*		

	ZlibException	A general purpose exception clas for exceptions in the Zlib library.
43	ZlibStream	Represents a Zlib stream for compression or decompression.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CompressionLevel Enumeration

Reference ► <u>lonic.Zlib</u> ► CompressionLevel

C# -

The compression level to be used when using a DeflateStream or ZlibStream with CompressionMode.Compress.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enum Comp	ressionLevel	
Public Enumeration	on CompressionLeve	1
public enum class	s CompressionLevel	

Members

Member	Description
None	None means that the data will be simply stored, with no change at all. If you are producing ZIPs for use on Mac OSX, be aware that archives produced with CompressionLevel.None cannot be opened with the default zip reader. Use a different CompressionLevel.
Level0	Same as None.
BestSpeed	The fastest but least effective compression.
Level1	A synonym for BestSpeed.
Level2	A little slower, but better, than level 1.
Level3	A little slower, but better, than level 2.
Level4	A little slower, but better, than level 3.
Level5	A little slower than level 4, but with better

	compression.
Default	The default compression level, with a good balance of speed and compression efficiency.
Level6	A synonym for Default.
Level7	Pretty good compression!
Level8	Better compression than Level7!
BestCompression	The "best" compression, where best means greatest reduction in size of the input data stream. This is also the slowes compression.
Level9	A synonym for BestCompression.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compression Mode Enumeration

Reference ► Ionic.Zlib ► CompressionMode

C#

An enum to specify the direction of transcoding - whether to compress c decompress.

■ Declaration Syntax

C#	Visual Basic	Visual C++	
public enum CompressionMode			
Public Enumeration CompressionMode			
public enum class	s CompressionMode		

Members

Member	Description
Compress	Used to specify that the stream should compress the data.
Decompress	Used to specify that the stream should decompress the data.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compression Strategy Enumeration

Reference ► <u>lonic.Zlib</u> ► CompressionStrategy

C#

Describes options for how the compression algorithm is executed. Different strategies work better on different sorts of data. The strategy parameter can affect the compression ratio and the speed of compression but not the correctness of the compression.

■ Declaration Syntax

C#	Visual Basic	Visual C++
public enum Comp	ressionStrategy	
Public Enumeration	on CompressionStra	tegy

public enum class CompressionStrategy

■ Members

Member	Description
Default	The default strategy is probably the best for normal data.
Filtered	The Filtered strategy is intended to be used most effectively with data produced by a filter or predictor. By this definition, filtered data consists mostly of small values with a somewhat random distribution. In this case, the compression algorithm is tuned to compress them better. The effect of Filtered is to force more Huffman coding and less strin matching; it is a half-step between Default and HuffmanOnly.
HuffmanOnly	Using HuffmanOnly will force the compressor to do Huffman encoding only

with no string matching.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 DeflateStream Class

Reference ► Ionic.Zlib ► DeflateStream

C#

A class for compressing and decompressing streams using the Deflate algorithm.

■ Declaration Syntax

C# Visual Basic Visual C++
public class DeflateStream : Stream

Public Class DeflateStream _ Inherits <u>Stream</u>

public ref class DeflateStream : public Stream

■ Members

All M	embers	Constructors	M	ethods	Properties	
✓ Pul	blic otected			Instance Static		Declare Inherite
lcon	Member			Description	n	
≡		tream(Stream, ssionMode)			eflateStream usi ompressionMod	•
ĕ₩	Compres	tream(Stream, ssionMode, ssionLevel)		specified C	eflateStream usi ompressionMod d CompressionI	e and
≟ ∳		tream(Stream, ssionMode,		Create a De the specifie	eflateStrea d	am usinç

		CompressionMode, and explicitly specify whether the strear should be left open after Deflation (Inflation.
∃ 🗳	DeflateStream(Stream, CompressionMode, CompressionLevel, Boolean)	Create a DeflateStream using the specified CompressionMode and the specified CompressionLevel, and explicitly specify whether the stream should be left open after Deflation or Inflation.
€ 0	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous read operation. (Inherited from <u>Stream</u> .)
Ξ ῷ	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous write operation. (Inherited from <u>Stream</u> .)
	<u>BufferSize</u>	The size of the working buffer for the compression codec.
	CanRead	Indicates whether the stream can b read. (Overrides <u>Stream.CanRead.</u>)
	CanSeek	Indicates whether the stream supports Seek operations.

		(Overrides <u>Stream.CanSeek</u> .)
	CanTimeout	Gets a value that determines whether the current stream can tim out.
		(Inherited from <u>Stream</u> .)
	CanWrite	Indicates whether the stream can b written. (Overrides Stream CanWrite)
-0		(Overrides <u>Stream.CanWrite</u> .)
- V	Close()	Closes the current stream and releases any resources (such as sockets and file handles) associate with the current stream. (Inherited from Stream .)
= \S	CompressBuffer(Byte[])	,
-45	Compressibilier(bytett)	Compress a byte array into a new byte array using DEFLATE.
=♦\$	CompressString(String)	Compress a string into a byte array using DEFLATE (RFC 1951).
Ξ ◊	CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from
		MarshalByRefObject.)
Ģ [©]	CreateWaitHandle()	Obsolete. Allocates a <u>WaitHandle</u> object.
		(Inherited from <u>Stream</u> .)
9	Dispose(Boolean)	Dispose the stream.

		(Overrides Stream.Dispose(Boolean).)
≡	Dispose()	Releases all resources used by the Stream.
•		(Inherited from <u>Stream</u> .)
=0	EndRead(IAsyncResult)	Waits for the pending asynchronouread to complete. (Inherited from <u>Stream</u> .)
-A		,
€ Q	EndWrite(IAsyncResult)	Ends an asynchronous write operation.
		(Inherited from <u>Stream</u> .)
=0	Equals(Object)	Determines whether the specified Object is equal to the current Object .
		(Inherited from Object.)
ē	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.
		(Inherited from Object.)
≡	Flush()	Flush the stream.
		(Overrides Stream.Flush().)
	FlushMode	This property sets the flush behavior on the stream.
≡	GetHashCode()	Serves as a hash function for a

		portion dor type
		particular type.
		(Inherited from Object.)
€	GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
=₩	GetType()	Gets the Type of the current instance. (Inherited from Object.)
€Û	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
	Length	Reading this property always throw a NotImplementedException. (Overrides Stream.Length.)
ē	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object (Inherited from MarshalByRefObject.)
ē	MemberwiseClone()	Creates a shallow copy of the current <u>Object</u> . (Inherited from <u>Object</u> .)
	Position	The position of the stream pointer. (Overrides <u>Stream.Position.</u>)
≡	Read(Byte[], Int32, Int32)	Read data from the stream.
	· · · · · · · · · · · · · · · · · · ·	. 15 da data ii 5111 ti 15 da datiii

		(Overrides <u>Stream.Read(Byte[], Int32, Int32</u>).)
∃©	ReadByte()	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. (Inherited from Stream .)
	ReadTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to read before timing out. (Inherited from Stream.)
₫∳	Seek(Int64, SeekOrigin)	Calling this method always throws a NotImplementedException. (Overrides Stream.Seek(Int64, SeekOrigin).)
□	SetLength(Int64)	Calling this method always throws a NotImplementedException. (Overrides Stream.SetLength(Int64).)
	Strategy	The ZLIB strategy to be used during compression.
₫	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
	<u>Totalln</u>	Returns the total number of bytes input so far.

	TotalOut	Returns the total number of bytes output so far.
= \$	UncompressBuffer(Byte[])	Uncompress a DEFLATE'd byte array into a byte array.
= \$	UncompressString(Byte[])	Uncompress a DEFLATE'd byte array into a single string.
≡	Write(Byte[], Int32, Int32)	Write data to the stream. (Overrides <u>Stream.Write(Byte[], Int32, Int32)</u> .)
≓	WriteByte(Byte)	Writes a byte to the current positior in the stream and advances the position within the stream by one byte. (Inherited from <u>Stream</u> .)
in the second	WriteTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to write before timing out.
		(Inherited from <u>Stream</u> .)

■ Remarks

The DeflateStream is a <u>Decorator</u> on a <u>Stream</u>. It adds DEFLATE compression or decompression to any stream.

Using this stream, applications can compress or decompress data via

stream Read and Write operations. Either compression or decompression can occur through either reading or writing. The compression format used is DEFLATE, which is documented in IETERFC 1951, "DEFLATE Compressed Data Format Specification version 1.3.".

This class is similar to <u>ZlibStream</u>, except that <u>ZlibStream</u> adds the <u>RFC 1950 - ZLIB</u> framing bytes to a compressed stream when compressing, or expects the RFC1950 framing bytes when decompressing. The <u>DeflateStream</u> does not.

■ Inheritance Hierarchy

<u>Object</u>

MarshalByRefObject

Stream

DeflateStream

See Also

ZlibStream GZipStream

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 DeflateStream Constructor

Reference ► Ionic.Zlib ► DeflateStream ► DeflateStream()

C#

■ Members

Icon	Member	Description
∃	DeflateStream(Stream, CompressionMode)	Create a DeflateStream using the specified CompressionMode.
∄	DeflateStream(Stream, CompressionMode, CompressionLevel)	Create a DeflateStream using the specified CompressionMode and the specified CompressionLevel.
≘ ₩	DeflateStream(Stream, CompressionMode, Boolean)	Create a DeflateStream using the specified CompressionMode and explicitly specify whether the stream should be left open after Deflation or Inflation.
⊒	DeflateStream(Stream, CompressionMode, CompressionLevel, Boolean)	Create a DeflateStream using the specified CompressionMode and the specified CompressionLevel, and explicitly specify whether the stream should be left open after Deflation or Inflation.

Send comments on this topic to ${\underline{\sf dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 DeflateStream Constructor (stream, mode)

Reference ► <u>lonic.Zlib</u> ► <u>DeflateStream</u> ► **DeflateStream**(Stream, CompressionMode)

Create a DeflateStream using the specified CompressionMode.

■ Declaration Syntax

Parameters

stream (Stream)

The stream which will be read or written.

mode (CompressionMode)

Indicates whether the DeflateStream will compress or decompress.

Remarks

When mode is CompressionMode. Compress, the DeflateStream will use the default compression level. The "captive" stream will be close

when the DeflateStream is closed.

Examples

This example uses a DeflateStream to compress data from a file, and writes the compressed data to another file.

VB.NET

```
Using input As Stream = File.OpenRead(fileToCompress
    Using raw As FileStream = File.Create(fileToComp
    Using compressor As Stream = New DeflateStre
        Dim buffer As Byte() = New Byte(4096) {}
        Dim n As Integer = -1
        Do While (n <> 0)
        If (n > 0) Then
            compressor.Write(buffer, 0, n)
        End If
        n = input.Read(buffer, 0, buffer.Len
        Loop
        End Using
        End Using
```

End Using

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

√

Ionic Zip Library v1.9.1.6 DeflateStream Constructor (stream, mode, level)

Reference ► Ionic.Zlib ► DeflateStream ► DeflateStream(Stream, CompressionMode, CompressionLevel)

Create a DeflateStream using the specified CompressionMode and the specified CompressionLevel.

Declaration Syntax

```
C# Visual Basic Visual C++

public DeflateStream(
    Stream stream,
    CompressionMode mode,
    CompressionLevel level
)
```

```
Public Sub New ( __ stream As Stream, _ mode As CompressionMode, _ level As CompressionLevel _ )
```

Parameters

stream (Stream)

The stream to be read or written while deflating or inflating.

mode (CompressionMode)

Indicates whether the DeflateStream will compress or

decompress.

level (CompressionLevel)

A tuning knob to trade speed for effectiveness.

Remarks

When mode is CompressionMode. Decompress, the level parameter is ignored. The "captive" stream will be closed when the DeflateStream is closed.

Examples

This example uses a DeflateStream to compress data from a file, and writes the compressed data to another file.

```
C#
                                                 Cop
using (System.IO.Stream input = System.IO.File.OpenR
{
    using (var raw = System.IO.File.Create(fileToCom
    {
        using (Stream compressor = new DeflateStream
         {
             byte[] buffer = new byte[WORKING_BUFFER_
             int n= -1;
             while (n != 0)
             {
                 if (n > 0)
                     compressor.Write(buffer, 0, n);
                 n= input.Read(buffer, 0, buffer.Leng
             }
        }
    }
}
```

VB.NET Cop

Using input As Stream = File.OpenRead(fileToCompress

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 DeflateStream Constructor (stream, mode, level, leaveOpen)

Reference ► <u>lonic.Zlib</u> ► <u>DeflateStream</u> ► **DeflateStream**(Stream, CompressionMode, CompressionLevel, Boolean)

Create a DeflateStream using the specified CompressionMod and the specified CompressionLevel, and explicitly specify wheth the stream should be left open after Deflation or Inflation.

Declaration Syntax

```
Visual Basic
                                     Visual C++
C#
public DeflateStream(
        Stream stream,
        <u>CompressionMode</u> mode,
        CompressionLevel level,
        bool leaveOpen
Public Sub New ( _
        stream As Stream,
        mode As CompressionMode,
        level As CompressionLevel,
        leaveOpen As Boolean _
public:
DeflateStream(
        Stream^ stream,
        CompressionMode mode,
        CompressionLevel level,
        bool leaveOpen
```

Parameters

stream (Stream)

The stream which will be read or written.

mode (CompressionMode)

Indicates whether the DeflateStream will compress or decompress.

level (CompressionLevel)

A tuning knob to trade speed for effectiveness.

leaveOpen (Boolean)

true if the application would like the stream to remain open after inflation/deflation.

Remarks

When mode is CompressionMode. Decompress, the level parameter is ignored.

This constructor allows the application to request that the captive strear remain open after the deflation or inflation occurs. By default, after Close() is called on the stream, the captive stream is also closed. In some cases this is not desired, for example if the stream is a MemoryStream that will be re-read after compression. Specify true for the leaveOpen parameter to leave the stream open.

Examples

This example shows how to use a DeflateStream to compress date from a file, and store the compressed data into another file.

```
using (var output = System.IO.File.Create(fileToComp
{
   using (System.IO.Stream input = System.IO.File.0
   {
      using (Stream compressor = new DeflateStream
      {
        byte[] buffer = new byte[WORKING_BUFFER_int n= -1;
        while (n != 0)
```

VB.NET

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 DeflateStream Constructor (stream, mode, leaveOpen)

Reference ► <u>Ionic.Zlib</u> ► <u>DeflateStream</u> ► **DeflateStream**(Stream, CompressionMode, Boolean)

Create a DeflateStream using the specified

CompressionMode, and explicitly specify whether the stream shoul be left open after Deflation or Inflation.

Declaration Syntax

```
mode As CompressionMode, _
    leaveOpen As Boolean _
)
public:
```

Parameters

stream (Stream)

The stream which will be read or written. This is called the "captive' stream in other places in this documentation.

mode (CompressionMode)

Indicates whether the **DeflateStream** will compress or decompress.

leaveOpen (Boolean)

true if the application would like the stream to remain open after inflation/deflation.

■ Remarks

This constructor allows the application to request that the captive strear remain open after the deflation or inflation occurs. By default, after Close() is called on the stream, the captive stream is also closed. In some cases this is not desired, for example if the stream is a memory stream that will be re-read after compression. Specify true for the *leaveOpen* parameter to leave the stream open.

The DeflateStream will use the default compression level.

See the other overloads of this constructor for example code.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

The size of the working buffer for the compression codec.

Declaration Syntax

■ Remarks

The working buffer is used for all stream operations. The default size is 1024 bytes. The minimum size is 128 bytes. You may get better performance with a larger buffer. Then again, you might not. You would have to test it.

Set this before the first call to Read() or Write() on the stream. If you try to set it afterwards, it will throw.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.Zlib ► DeflateStream ► CanRead

C#

Indicates whether the stream can be read.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:

virtual property bool CanRead {
    bool get () override;
}
```

Remarks

The return value depends on whether the captive stream supports reading.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Can Seek Property

Reference ► Ionic.Zlib ► DeflateStream ► CanSeek

C#

Indicates whether the stream supports Seek operations.

Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:
virtual property bool CanSeek {
    bool get () override;
}
```

■ Remarks

Always returns false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanWrite Property

Reference ► Ionic.Zlib ► DeflateStream ► CanWrite

C#

Indicates whether the stream can be written.

Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
virtual property bool CanWrite {
    bool get () override;
}
```

Remarks

The return value depends on whether the captive stream supports writing.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compress Buffer Method (b)

Reference ► Ionic.Zlib ► DeflateStream ► CompressBuffer(Byte[])

Compress a byte array into a new byte array using DEFLATE.

Declaration Syntax

Parameters

b (**Byte**[])

A buffer to compress.

■ Return Value

The data in compressed form

■ Remarks

Uncompress it with UncompressBuffer(Byte]).

See Also

DeflateStream.CompressString(string)
DeflateStream.UncompressBuffer(byte[])
GZipStream.CompressBuffer(byte[])

ZlibStream.CompressBuffer(byte[])

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compress String Method (s)

Reference ► Ionic.Zlib ► DeflateStream ► CompressString(String)

Compress a string into a byte array using DEFLATE (RFC 1951).

Declaration Syntax

```
Public Shared Function CompressString ( _ s As String _ )

As Byte()
```

Parameters

s (String)

A string to compress. The string will first be encoded using UTF8, then compressed.

■ Return Value

The string in compressed form

■ Remarks

Uncompress it with UncompressString(Byte]).

- See Also

DeflateStream.UncompressString(byte[])
DeflateStream.CompressBuffer(byte[])

GZipStream.CompressString(string) ZlibStream.CompressString(string)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Dispose Method (disposing)

Reference ► Ionic.Zlib ► DeflateStream ► Dispose(Boolean)

Dispose the stream.

■ Declaration Syntax

Parameters

disposing (Boolean)

true if the Dispose method was invoked by user code.

■ Remarks

This may or may not result in a Close() call on the captive stream. See the constructors that have a leaveOpen parameter for more information.

Application code won't call this code directly. This method may be invoked in two distinct scenarios. If disposing == true, the method has been called directly or indirectly by a user's code, for example via the public Dispose() method. In this case, both managed and unmanaged resources can be referenced and disposed. If disposing == false, the

method has been called by the runtime from inside the object finalizer and this method should not reference other objects; in that case only unmanaged resources must be referenced or disposed.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Method

Reference ► Ionic.Zlib ► DeflateStream ► Flush()

C#

Flush the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

public:

virtual void Flush() override

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Mode Property

Reference ► Ionic.Zlib ► DeflateStream ► FlushMode C#

This property sets the flush behavior on the stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public virtual FlushType FlushMode { get; set; }

Public Overridable Property FlushMode As FlushType
    Get
    Set

public:
virtual property FlushType FlushMode {
    FlushType get ();
    void set (FlushType value);
}
```

■ Remarks

See the ZLIB documentation for the meaning of the flush behavior.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Length Property

Reference
► Ionic.Zlib
► DeflateStream
► Length

C#
```

Reading this property always throws a **NotImplementedException**.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:
virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

The position of the stream pointer.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

■ Remarks

Setting this property always throws a <u>NotImplementedException</u>. Reading will return the total bytes written out, if used in writing, or the total bytes read in, if used in reading. The count may refer to compresse bytes or uncompressed bytes, depending on how you've used the stream.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

Reference ► Ionic.Zlib ► DeflateStream ► Read(Byte[], Int32, Int32)

C#

Read data from the stream.

Declaration Syntax

```
C# Visual Basic Visual C++

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

Parameters

buffer (Byte∏)

The buffer into which the read data should be placed.

offset (Int32)

the offset within that data array to put the first byte read.

count (Int32)

the number of bytes to read.

Return Value

the number of bytes actually read

■ Remarks

If you wish to use the DeflateStream to compress data while reading, you can create a DeflateStream with CompressionMode.Compress, providing an uncompressed data stream. Then call Read() on that DeflateStream, and the data read will be compressed as you read. If you wish to use the DeflateStream to decompress data while reading, you can create DeflateStream with CompressionMode.Decompress, providing a readable compressed data stream. Then call Read() on that DeflateStream, and the data read will be decompressed as you read.

A DeflateStream can be used for Read() or Write(), but not both.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.Zlib ► DeflateStream ► Seek(Int64, SeekOrigin)

Calling this method always throws a NotImplementedException.

Declaration Syntax

```
C# Visual Basic Visual C++

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

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

Parameters

offset (Int64)

this is irrelevant, since it will always throw!

origin (SeekOrigin)

this is irrelevant, since it will always throw!

■ Return Value

irrelevant!

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.Zlib ► DeflateStream ► SetLength(Int64)

Calling this method always throws a NotImplementedException.

■ Declaration Syntax

Parameters

value (Int64)

this is irrelevant, since it will always throw!

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Strategy Property Reference ► Ionic.Zlib ► DeflateStream ► Strategy C#

The ZLIB strategy to be used during compression.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public CompressionStrategy Strategy { get; set; }

Public Property Strategy As CompressionStrategy
    Get
    Set

public:
property CompressionStrategy Strategy {
    CompressionStrategy get ();
    void set (CompressionStrategy value);
}
```

■ Remarks

By tweaking this parameter, you may be able to optimize the compression for data with particular characteristics.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

```
| Ionic Zip Library v1.9.1.6 Totalin Property
| Reference ► Ionic.Zlib ► DeflateStream ► Totalin | C#
```

Returns the total number of bytes input so far.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public virtual long TotalIn { get; }

Public Overridable ReadOnly Property TotalIn As Long
    Get

public:
virtual property long long TotalIn {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 TotalOut Property Reference ► Ionic.Zlib ► DeflateStream ► TotalOut C#

Returns the total number of bytes output so far.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public virtual long TotalOut { get; }

Public Overridable ReadOnly Property TotalOut As Lon
    Get

public:
virtual property long long TotalOut {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Uncompress Buffer Method

(compressed)

Reference ► Ionic.Zlib ► DeflateStream ►

UncompressBuffer(Byte[])

Uncompress a DEFLATE'd byte array into a byte array.

C# 🔽

Declaration Syntax

Parameters

compressed (Byte[])

A buffer containing data that has been compressed with DEFLATE.

Return Value

The data in uncompressed form

See Also

DeflateStream.CompressBuffer(byte[])
DeflateStream.UncompressString(byte[])
GZipStream.UncompressBuffer(byte[])
ZlibStream.UncompressBuffer(byte[])

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Uncompress String Method

(compressed)

Reference ► Ionic.Zlib ► DeflateStream ►

UncompressString(Byte[])

Uncompress a DEFLATE'd byte array into a single string.

■ Declaration Syntax

array<unsigned char>^ compressed

Parameters

)

compressed (Byte[])

A buffer containing DEFLATE-compressed data.

■ Return Value

The uncompressed string

See Also

DeflateStream.CompressString(String)
DeflateStream.UncompressBuffer(byte[])
GZipStream.UncompressString(byte[])
ZlibStream.UncompressString(byte[])

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

Reference ► Ionic.Zlib ► DeflateStream ► Write(Byte[], Int32, Int32)

C#

Write data to the stream.

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

buffer (Byte∏)

The buffer holding data to write to the stream.

offset (Int32)

the offset within that data array to find the first byte to write.

count (Int32)

the number of bytes to write.

■ Remarks

If you wish to use the DeflateStream to compress data while writing, you can create a DeflateStream with CompressionMode.Compress, and a writable output stream. Then call Write() on that DeflateStream, providing uncompressed data as input. The data sent to the output stream will be the compressed form of the data written. If you wish to use the DeflateStream to decompress data while writing, you can create a DeflateStream with CompressionMode.Decompress, and writable output stream. Then call Write() on that stream, providing previously compressed data. The data sent to the output stream will be the decompressed form of the data written.

A DeflateStream can be used for Read() or Write(), but not both.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | FlushType Enumeration | Reference | Ionic.Zlib | FlushType | FlushType |

C#

Describes how to flush the current deflate operation.

■ Declaration Syntax

C#	Visual Basic	Visual C++					
public enum FlushType							
Public Enumeration FlushType							
public enum class	FlushType						

■ Members

Member	Description
None	No flush at all.
Partial	Closes the current block, but doesn't flush it to the output. Used internally only in hypothetical scenarios. This was supposed to be removed by Zlib, but it is still in use in some edge cases.
Sync	Use this during compression to specify that all pending output should be flushed to the output buffer and the output should be aligned on a byte boundary. You might use this in a streaming communication scenario, so that the decompressor can get all input data available so far. When using this with a ZlibCodec, AvailableBytesIn will be zero afte the call if enough output space has been provided before the call. Flushing will degrade compression and so it should be used only when necessary.

Full	Use this during compression to specify that all output should be flushed, as with FlushType.Sync, but also, the compression state should be reset so tha decompression can restart from this point if previous compressed data has been damaged or if random access is desired. Using FlushType.Full too often car significantly degrade the compression.
Finish	Signals the end of the compression/decompression stream.

■ Remarks

The different FlushType values are useful when using a Deflate in a streaming application.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **GZipStream Class**

Reference ► Ionic.Zlib ► GZipStream

C#

A class for compressing and decompressing GZIP streams.

■ Declaration Syntax

C# Visual Basic Visual C++

public class GZipStream : Stream

Public Class GZipStream _ Inherits Stream

public ref class GZipStream : public Stream

Members

All M	embers	Constructors	Met	hods	Properties	Fields
			Instance Static		DeclaredInherited	
Icon Member			Description			
∃		ream(Stream, essionMode)			a GZipStrea d Compressi	_
Ξ ψ	Compre	ream(Stream, essionMode, essionLevel)		Create a GZipStream using the specified CompressionMode and the specified CompressionLevel.		onMode.
GZipStream(Stream, CompressionMode, Boolean)		Create a GZipStream using the specified CompressionMode, and explicitly specify whether the				

		stream should be left open after Deflation or Inflation.
€	GZipStream(Stream, CompressionMode, CompressionLevel, Boolean)	Create a GZipStream using the specified CompressionMode and the specified CompressionLevel, and explicitly specify whether the strear should be left open after Deflation conflation.
≡	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous read operation.
		(Inherited from <u>Stream</u> .)
≡₩	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous write operation.
		(Inherited from <u>Stream</u> .)
	<u>BufferSize</u>	The size of the working buffer for the compression codec.
	CanRead	Indicates whether the stream can b read.
		(Overrides <u>Stream</u> . <u>CanRead</u> .)
	CanSeek	Indicates whether the stream supports Seek operations.
		(Overrides <u>Stream</u> . <u>CanSeek</u> .)
	CanTimeout	Gets a value that determines
	BufferSize CanRead CanSeek	(Inherited from Stream.) The size of the working buffer for the compression codec. Indicates whether the stream can read. (Overrides Stream.CanRead.) Indicates whether the stream supports Seek operations. (Overrides Stream.CanSeek.)

		whether the current stream can timout.
		(Inherited from <u>Stream</u> .)
	CanWrite	Indicates whether the stream can buritten. (Overrides <u>Stream.CanWrite.</u>)
∄	Close()	Closes the current stream and releases any resources (such as sockets and file handles) associate with the current stream. (Inherited from Stream .)
	Comment	The comment on the GZIP stream
=\$ \$	CompressBuffer(Byte[])	Compress a byte array into a new byte array using GZip.
• \$	CompressString(String)	Compress a string into a byte arrausing GZip.
	Crc32	The CRC on the GZIP stream.
≓	CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object (Inherited from MarshalByRefObject.)
		,

	<u>CreateWaitHandle()</u>	Obsolete. Allocates a <u>WaitHandle</u> object. (Inherited from <u>Stream</u> .)
ē.	Dispose(Boolean)	Dispose the stream.
	<u>=.opoco(=co.cu)</u>	(Overrides Stream.Dispose(Boolean).)
∃	Dispose()	Releases all resources used by the Stream . (Inherited from Stream .)
∃	EndRead(IAsyncResult)	Waits for the pending asynchronouread to complete. (Inherited from <u>Stream</u> .)
∄	EndWrite(IAsyncResult)	Ends an asynchronous write operation. (Inherited from <u>Stream</u> .)
≅	Equals(Object)	Determines whether the specified Object is equal to the current Object . (Inherited from Object .)
	<u>FileName</u>	The FileName for the GZIP stream.
Ģ ♥	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
≓	Flush()	Flush the stream.

		(Overrides <u>Stream.Flush()</u> .)
	FlushMode	This property sets the flush behavior on the stream.
≅©	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
∃ ©	GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
≡	GetType()	Gets the <u>Type</u> of the current instance. (Inherited from <u>Object</u> .)
∃ 🗳	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)
•	LastModified	The last modified time for the GZIP stream.
	Length	Reading this property always throw a NotImplementedException. (Overrides Stream.Length.)
Ģ ₩	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object

		(Inherited from MarshalByRefObject.)
Ģ 🍑	MemberwiseClone()	Creates a shallow copy of the current Object.
		(Inherited from Object.)
	Position	The position of the stream pointer. (Overrides <u>Stream.Position.</u>)
∉ ©	Read(Byte[], Int32, Int32)	Read and decompress data from the source stream.
		(Overrides <u>Stream.Read(Byte[], Int32, Int32)</u> .)
≓	ReadByte()	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. (Inherited from Stream .)
ET .	ReadTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to read before timing out. (Inherited from <u>Stream</u> .)
Ξ₩	Seek(Int64, SeekOrigin)	Calling this method always throws a NotImplementedException. (Overrides Stream.Seek(Int64, SeekOrigin).)
≡	SetLength(Int64)	Calling this method always throws a NotImplementedException. (Overrides
		Stream.SetLength(Int64).)

=₩	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
	Totalin	Returns the total number of bytes input so far.
	TotalOut	Returns the total number of bytes output so far.
=\$	UncompressBuffer(Byte[])	Uncompress a GZip'ed byte array into a byte array.
=♦S	UncompressString(Byte[])	Uncompress a GZip'ed byte array into a single string.
≡	Write(Byte[], Int32, Int32)	Write data to the stream. (Overrides <u>Stream.Write(Byte[], Int32, Int32)</u> .)
∃	WriteByte(Byte)	Writes a byte to the current positior in the stream and advances the position within the stream by one byte. (Inherited from <u>Stream</u> .)
		,
	WriteTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to write before timing out.
		(Inherited from <u>Stream</u> .)

■ Remarks

The GZipStream is a <u>Decorator</u> on a <u>Stream</u>. It adds GZIP compression or decompression to any stream.

Like the System. IO. Compression. GZipStream in the .NET Base Class Library, the Ionic.Zlib.GZipStream can compress while writing, or decompress while reading, but not vice versa. The compression method used is GZIP, which is documented in IETF RFC
1952, "GZIP file format specification version 4.3".

A GZipStream can be used to decompress data (through Read()) to compress data (through Write()), but not both.

If you wish to use the GZipStream to compress data, you must wrap around a write-able stream. As you call Write() on the GZipStream, the data will be compressed into the GZIP format. If yo want to decompress data, you must wrap the GZipStream around a readable stream that contains an IETF RFC 1952-compliant stream. Th data will be decompressed as you call Read() on the GZipStream

Though the GZIP format allows data from multiple files to be concatenated together, this stream handles only a single segment of GZIP format, typically representing a single file.

This class is similar to <u>ZlibStream</u> and <u>DeflateStream</u>. **ZlibStream** handles RFC1950-compliant streams. <u>DeflateStream</u> handles RFC1951 compliant streams. This class handles RFC1952-compliant streams.

■ Inheritance Hierarchy

Object

MarshalByRefObject

Stream

GZipStream

See Also

<u>DeflateStream</u> <u>ZlibStream</u>

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | GZipStream Constructor | Reference | Ionic.Zlib | GZipStream | GZipStream() | C#

■ Members

Icon	Member	Description
∄ 🍑	GZipStream(Stream, CompressionMode)	Create a GZipStream using the specified CompressionMode.
≡ ₩	GZipStream(Stream, CompressionMode, CompressionLevel)	Create a GZipStream using the specified CompressionMode and the specified CompressionLevel.
₫ ♠	GZipStream(Stream, CompressionMode, Boolean)	Create a GZipStream using the specified CompressionMode, and explicitly specify whether the stream should be left open after Deflation or Inflation.
∃ 🔷	GZipStream(Stream, CompressionMode, CompressionLevel, Boolean)	Create a GZipStream using the specified CompressionMode and the specified CompressionLevel, and explicitly specify whether the stream should be left open after Deflation or Inflation.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 GZipStream Constructor (stream,

mode)

Reference ► Ionic.Zlib ► GZipStream ► GZipStream(Stream, CompressionMode)

Create a GZipStream using the specified CompressionMode.

■ Declaration Syntax

Parameters

stream (<u>Stream</u>)

The stream which will be read or written.

mode (CompressionMode)

Indicates whether the GZipStream will compress or decompress.

■ Remarks

When mode is CompressionMode.Compress, the GZipStrea

will use the default compression level.

As noted in the class documentation, the CompressionMode (Compress or Decompress) also establishes the "direction" of the stream A GZipStream with CompressionMode. Compress works only through Write(). A GZipStream with CompressionMode. Decompress works only through Read().

Examples

This example shows how to use a GZipStream to compress data.

```
using (System.IO.Stream input = System.IO.File.OpenR
{
   using (var raw = System.IO.File.Create(outputFil) {
      using (Stream compressor = new GZipStream(ra) {
        byte[] buffer = new byte[WORKING_BUFFER_int n;
      while ((n= input.Read(buffer, 0, buffer.) {
            compressor.Write(buffer, 0, n);
        }
    }
}
```

VB.NET □ Cop

```
compressor.Write(buffer, 0, n)
    End If
    n = input.Read(buffer, 0, buffer.Length)
    Loop
    End Using
    End Using
End Using
End Using
```

Examples

This example shows how to use a GZipStream to uncompress a file.

C#

```
private void GunZipFile(string filename)
{
    if (!filename.EndsWith(".gz))
        throw new ArgumentException("filename");
    var DecompressedFile = filename.Substring(0, file
    byte[] working = new byte[WORKING_BUFFER_SIZE];
    int n= 1;
    using (System.IO.Stream input = System.IO.File.0
    {
        using (Stream decompressor= new Ionic.Zlib.G
        {
            using (var output = System.IO.File.Creat
            {
                while (n !=0)
                {
                    n= decompressor.Read(working, 0,
                    if (n > 0)
                    {
                        output.Write(working, ⊙, n);
                }
           }
   }
}
```

VB.NET

```
Private Sub GunZipFile(ByVal filename as String)
    If Not (filename.EndsWith(".gz)) Then
        Throw New ArgumentException("filename")
    End If
    Dim DecompressedFile as String = filename.Substr
    Dim working(WORKING_BUFFER_SIZE) as Byte
    Dim n As Integer = 1
    Using input As Stream = File.OpenRead(filename)
        Using decompressor As Stream = new Ionic.Zli
            Using output As Stream = File.Create(Unc
                Do
                    n= decompressor.Read(working, 0,
                    If n > 0 Then
                        output.Write(working, 0, n)
                    End IF
                Loop While (n > 0)
            End Using
        End Using
    End Using
End Sub
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 GZipStream Constructor (stream,

mode, level)

Reference ► Ionic.Zlib ► GZipStream ► GZipStream(Stream, CompressionMode, CompressionLevel)

Create a GZipStream using the specified CompressionMode an the specified CompressionLevel.

Declaration Syntax

```
C# Visual Basic Visual C++

public GZipStream(
    Stream stream,
    CompressionMode mode,
    CompressionLevel level
)
```

Parameters

stream (Stream)

The stream to be read or written while deflating or inflating.

mode (CompressionMode)

Indicates whether the GZipStream will compress or decompress

level (CompressionLevel)

A tuning knob to trade speed for effectiveness.

■ Remarks

The CompressionMode (Compress or Decompress) also establishe the "direction" of the stream. A GZipStream with CompressionMode.Compress works only through Write(). A GZipStream with CompressionMode.Decompress works on through Read().

Examples

This example shows how to use a GZipStream to compress a file int a .gz file.

```
C#
                                                  Cop
using (System.IO.Stream input = System.IO.File.OpenR
{
    using (var raw = System.IO.File.Create(fileToCom
    {
         using (Stream compressor = new GZipStream(ra
                                                     Co
                                                     Co
         {
             byte[] buffer = new byte[WORKING_BUFFER_
             int n;
             while ((n= input.Read(buffer, 0, buffer.
             {
                 compressor.Write(buffer, 0, n);
             }
         }
    }
}
```

VB.NET □ Cop

Using input As Stream = File.OpenRead(fileToCompress

```
Using raw As FileStream = File.Create(fileToComp
    Using compressor As Stream = New GZipStream(
        Dim buffer As Byte() = New Byte(4096) {}
        Dim n As Integer = -1
        Do While (n <> 0)
        If (n > 0) Then
            compressor.Write(buffer, 0, n)
        End If
        n = input.Read(buffer, 0, buffer.Len
        Loop
        End Using
        End Using
        End Using
End Using
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 GZipStream Constructor (stream, mode, level, leaveOpen)

Reference ► Ionic.Zlib ► GZipStream ► GZipStream(Stream, CompressionMode, CompressionLevel, Boolean)

Create a GZipStream using the specified CompressionMode and the specified CompressionLevel, and explicitly specify whether the stream should be left open after Deflation or Inflation.

Declaration Syntax

```
Visual Basic
                                     Visual C++
C#
public GZipStream(
        Stream stream,
        CompressionMode mode,
        CompressionLevel level,
        bool leaveOpen
Public Sub New ( _
        stream As Stream,
        mode As CompressionMode,
        level As CompressionLevel,
        leaveOpen As Boolean _
public:
GZipStream(
        Stream^ stream,
        CompressionMode mode,
        CompressionLevel level,
        bool leaveOpen
```

Parameters

stream (Stream)

The stream which will be read or written.

mode (CompressionMode)

Indicates whether the GZipStream will compress or decompress.

level (CompressionLevel)

A tuning knob to trade speed for effectiveness.

leaveOpen (Boolean)

true if the application would like the stream to remain open after inflation/deflation.

Remarks

This constructor allows the application to request that the captive strear remain open after the deflation or inflation occurs. By default, after Close() is called on the stream, the captive stream is also closed. In some cases this is not desired, for example if the stream is a memory stream that will be re-read after compressed data has been written to it. Specify true for the *leaveOpen* parameter to leave the stream open.

As noted in the class documentation, the CompressionMode (Compress or Decompress) also establishes the "direction" of the stream A GZipStream with CompressionMode. Compress works only through Write(). A GZipStream with CompressionMode. Decompress works only through Read().

Examples

This example shows how to use a GZipStream to compress data.

```
Using (System.IO.Stream input = System.IO.File.OpenR
{
    using (var raw = System.IO.File.Create(outputFil) {
        using (Stream compressor = new GZipStream(ra) {
            byte[] buffer = new byte[WORKING_BUFFER_]
```

VB.NET

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

· | _______

Ionic Zip Library v1.9.1.6 GZipStream Constructor (stream, mode, leaveOpen)

Reference ► Ionic.Zlib ► GZipStream ► GZipStream(Stream, CompressionMode, Boolean)

Create a GZipStream using the specified CompressionMode, are explicitly specify whether the stream should be left open after Deflation Inflation.

Declaration Syntax

Parameters

stream (Stream)

The stream which will be read or written. This is called the "captive' stream in other places in this documentation.

mode (CompressionMode)

Indicates whether the GZipStream will compress or decompress.

leaveOpen (Boolean)

true if the application would like the base stream to remain open after inflation/deflation.

■ Remarks

This constructor allows the application to request that the captive strear remain open after the deflation or inflation occurs. By default, after Close() is called on the stream, the captive stream is also closed. In some cases this is not desired, for example if the stream is a memory stream that will be re-read after compressed data has been written to it. Specify true for the *leaveOpen* parameter to leave the stream open.

The <u>CompressionMode</u> (Compress or Decompress) also establishes the "direction" of the stream. A GZipStream with CompressionMode.Compress works only through Write(). A GZipStream with CompressionMode.Decompress works on through Read().

The GZipStream will use the default compression level. If you want t specify the compression level, see <u>GZipStream(Stream, CompressionMode, CompressionLevel, Boolean)</u>.

See the other overloads of this constructor for example code.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BufferSize Property Reference ► Ionic.Zlib ► GZipStream ► BufferSize C#

The size of the working buffer for the compression codec.

■ Declaration Syntax

Remarks

The working buffer is used for all stream operations. The default size is 1024 bytes. The minimum size is 128 bytes. You may get better performance with a larger buffer. Then again, you might not. You would have to test it.

Set this before the first call to Read() or Write() on the stream. If you try to set it afterwards, it will throw.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.Zlib ► GZipStream ► CanRead

C# _

Indicates whether the stream can be read.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:
    virtual property bool CanRead {
        bool get () override;
}
```

Remarks

The return value depends on whether the captive stream supports reading.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Can Seek Property

Reference ► Ionic.Zlib ► GZipStream ► CanSeek

C#

Indicates whether the stream supports Seek operations.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:
virtual property bool CanSeek {
    bool get () override;
}
```

■ Remarks

Always returns false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 CanWrite Property | Reference | Ionic.Zlib | GZipStream | CanWrite | C#

Indicates whether the stream can be written.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
    virtual property bool CanWrite {
        bool get () override;
}
```

Remarks

The return value depends on whether the captive stream supports writing.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 Comment Property | Reference | Ionic.Zlib | GZipStream | C# C#

The comment on the GZIP stream.

Declaration Syntax

```
C# Visual Basic Visual C++

public string Comment { get; set; }

Public Property Comment As String
    Get
    Set

public:
property String^ Comment {
    String^ get ();
    void set (String^ value);
}
```

■ Remarks

The GZIP format allows for each file to optionally have an associated comment stored with the file. The comment is encoded with the ISO-8859-1 code page. To include a comment in a GZIP stream you create, set this property before calling Write() for the first time on the GZipStream.

When using GZipStream to decompress, you can retrieve this property after the first call to Read(). If no comment has been set in the GZIP bytestream, the Comment property will return null (Nothing i VB).

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compress Buffer Method (b)

Reference ► Ionic.Zlib ► GZipStream ► CompressBuffer(Byte[])

Compress a byte array into a new byte array using GZip.

Declaration Syntax

Parameters

b (**Byte**[])

A buffer to compress.

■ Return Value

The data in compressed form

■ Remarks

Uncompress it with UncompressBuffer(Byte]).

■ See Also

CompressString(String)
UncompressBuffer(Byte[])

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compress String Method (s)

Reference ► Ionic.Zlib ► GZipStream ► CompressString(String)

C#

Compress a string into a byte array using GZip.

Declaration Syntax

Parameters

s (String)

A string to compress. The string will first be encoded using UTF8, then compressed.

Return Value

The string in compressed form

■ Remarks

Uncompress it with UncompressString(Byte]).

- See Also

UncompressString(Byte[])
CompressBuffer(Byte[])

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

```
Ionic Zip Library v1.9.1.6
Crc32 Property

Reference ► Ionic.Zlib ► GZipStream ► Crc32
C#
```

The CRC on the GZIP stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public int Crc32 { get; }

Public ReadOnly Property Crc32 As Integer
    Get

public:
property int Crc32 {
    int get ();
}
```

Remarks

This is used for internal error checking. You probably don't need to look at this property.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Dispose Method (disposing)

Reference ► Ionic.Zlib ► GZipStream ► Dispose(Boolean)

Dispose the stream.

■ Declaration Syntax

C#

Parameters

disposing (Boolean)

indicates whether the Dispose method was invoked by user code.

■ Remarks

This may or may not result in a Close() call on the captive stream. See the constructors that have a leaveOpen parameter for more information.

This method may be invoked in two distinct scenarios. If disposing == true, the method has been called directly or indirectly by a user's code, for example via the public Dispose() method. In this case, both manage and unmanaged resources can be referenced and disposed. If disposin == false, the method has been called by the runtime from inside the

object finalizer and this method should not reference other objects; in the case only unmanaged resources must be referenced or disposed.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 FileName Property Reference ► Ionic.Zlib ► GZipStream ► FileName C#

The FileName for the GZIP stream.

■ Declaration Syntax

■ Remarks

The GZIP format optionally allows each file to have an associated filename. When compressing data (through Write()), set this FileName before calling Write() the first time on the GZipStream The actual filename is encoded into the GZIP bytestream with the ISO-8859-1 code page, according to RFC 1952. It is the application's responsibility to insure that the FileName can be encoded and decoded correctly with this code page.

When decompressing (through Read()), you can retrieve this value at time after the first Read(). In the case where there was no filename encoded into the GZIP bytestream, the property will return null (Nothing in VB).

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Flush the stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

public:
virtual void Flush() override
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | FlushMode Property | Reference | Ionic.Zlib | GZipStream | FlushMode | C# | C# | C#

This property sets the flush behavior on the stream.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 LastModified Field

Reference ► Ionic.Zlib ► GZipStream ► LastModified C#

The last modified time for the GZIP stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public Nullable < DateTime > LastModified

Public LastModified As Nullable(Of DateTime)

public:

Nullable < DateTime > LastModified

■ Remarks

GZIP allows the storage of a last modified time with each GZIP entry. When compressing data, you can set this before the first call to Write(). When decompressing, you can retrieve this value any time after the first call to Read().

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Length Property

Reference
► Ionic.Zlib
► GZipStream
► Length

C#
```

Reading this property always throws a **NotImplementedException**.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:
virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Position Property Reference ► Ionic.Zlib ► GZipStream ► Position C#

The position of the stream pointer.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

Remarks

Setting this property always throws a <u>NotImplementedException</u>. Reading will return the total bytes written out, if used in writing, or the total bytes read in, if used in reading. The count may refer to compresse bytes or uncompressed bytes, depending on how you've used the stream.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assault: Lapis 7in (Madula: Lapis 7in) Varsian: 1, 0, 1, 0, (1, 0, 1)

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

C#

Reference ► Ionic.Zlib ► GZipStream ► Read(Byte[], Int32, Int32)

Read and decompress data from the source stream.

■ Declaration Syntax

```
C# Visual Basic Visual C++

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

Parameters

buffer (Byte∏)

The buffer into which the decompressed data should be placed.

offset (Int32)

the offset within that data array to put the first byte read.

count (Int32)

the number of bytes to read.

■ Return Value

the number of bytes actually read

■ Remarks

With a GZipStream, decompression is done through reading.

Examples

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

 $|\bullet|$

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.Zlib ► GZipStream ► Seek(Int64, SeekOrigin)

Calling this method always throws a NotImplementedException.

Declaration Syntax

```
C# Visual Basic Visual C++

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

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

Parameters

offset (Int64)

irrelevant; it will always throw!

origin (SeekOrigin)

irrelevant; it will always throw!

■ Return Value

irrelevant!

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.Zlib ► GZipStream ► SetLength(Int64)

C#

Calling this method always throws a **NotImplementedException**.

■ Declaration Syntax

Parameters

value (Int64)

irrelevant; this method will always throw!

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Totalin Property

Reference ► Ionic.Zlib ► GZipStream ► Totalin
C#
```

Returns the total number of bytes input so far.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public virtual long TotalIn { get; }

Public Overridable ReadOnly Property TotalIn As Long Get

public:
virtual property long long TotalIn {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 TotalOut Property Reference ► Ionic.Zlib ► GZipStream ► TotalOut C#

Returns the total number of bytes output so far.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public virtual long TotalOut { get; }

Public Overridable ReadOnly Property TotalOut As Lon Get

public:
virtual property long long TotalOut {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Uncompress Buffer Method

(compressed)

Reference ► Ionic.Zlib ► GZipStream ► UncompressBuffer(Byte[])

Uncompress a GZip'ed byte array into a byte array.

C#

Declaration Syntax

Parameters

compressed (Byte[])

A buffer containing data that has been compressed with GZip.

■ Return Value

The data in uncompressed form

See Also

CompressBuffer(Byte[])
UncompressString(Byte[])

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Uncompress String Method

(compressed)

Reference ► Ionic.Zlib ► GZipStream ► UncompressString(Byte[])

C#

Uncompress a GZip'ed byte array into a single string.

■ Declaration Syntax

Parameters

compressed (Byte[])

A buffer containing GZIP-compressed data.

■ Return Value

The uncompressed string

See Also

CompressString(String)
UncompressBuffer(Byte[])

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

C#

Reference ► Ionic.Zlib ► GZipStream ► Write(Byte[], Int32, Int32)

Write data to the stream.

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

buffer (Byte∏)

The buffer holding data to write to the stream.

offset (Int32)

the offset within that data array to find the first byte to write.

count (Int32)

the number of bytes to write.

■ Remarks

If you wish to use the GZipStream to compress data while writing, you can create a GZipStream with CompressionMode.Compress and a writable output stream. Then call Write() on that GZipStream, providing uncompressed data as input. The data sent to the output stream will be the compressed form of the data written.

A GZipStream can be used for Read() or Write(), but not both Writing implies compression. Reading implies decompression.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel Deflate Output Stream Class

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream C#

A class for compressing streams using the Deflate algorithm with multip threads.

■ Declaration Syntax

C# Visual Basic Visual C++
public class ParallelDeflateOutputStream : Stream

Public Class ParallelDeflateOutputStream _ Inherits <u>Stream</u>

public ref class ParallelDeflateOutputStream : publi

■ Members

All M	embers	Constructors	Methods	Properties	
PublicProtected		✓ Instance✓ Static		Declare Inherite	
lcon	Member			Description	
≡ ♠	Parallel	<u>DeflateOutputStrea</u>	am(Stream)	Create a ParallelDeflateOut	putStrea
₫ 🍑		DeflateOutputStrea ssionLevel)	am(Stream,	Create a ParallelDeflateOut using the specified CompressionLeve	k
∉	Parallel Boolean	DeflateOutputStrea)	am(Stream,	Create a ParallelDeflateOut	putStrea

		and specify whether to lea the captive stream open when the ParallelDeflateOutputStrea is closed.
₫ 🍑	ParallelDeflateOutputStream(Stream, CompressionLevel, Boolean)	Create a ParallelDeflateOutputStrea and specify whether to lea the captive stream open when the ParallelDeflateOutputStrea is closed.
∄	ParallelDeflateOutputStream(Stream, CompressionLevel, CompressionStrategy, Boolean)	Create a ParallelDeflateOutputStreausing the specified CompressionLevel and CompressionStrategy, and specifying whether to leav the captive stream open when the ParallelDeflateOutputStrea is closed.
∄	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous read operation. (Inherited from <u>Stream</u> .)
€ ©	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous write operation.

		(Inherited from <u>Stream</u> .)
	BufferSize	The size of the buffers use by the compressor threads
	BytesProcessed	The total number of uncompressed bytes processed by the ParallelDeflateOutputStream
	CanRead	Indicates whether the stre supports Read operations (Overrides <u>Stream.CanRead.</u>)
	CanSeek	Indicates whether the stresupports Seek operations. (Overrides Stream.CanSeek.)
	CanTimeout	Gets a value that determine whether the current stream can time out. (Inherited from Stream .)
	CanWrite	Indicates whether the stre supports Write operations (Overrides Stream.CanWrite.)
=	Close()	Close the stream. (Overrides <u>Stream.Close(</u>)
	Crc32	The CRC32 for the data the was written out, prior to

		compression.
=	CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remorbject. (Inherited from MarshalByRefObject.)
ē	CreateWaitHandle()	Obsolete. Allocates a <u>WaitHandle</u> object. (Inherited from <u>Stream</u> .)
∃	<u>Dispose()</u>	Dispose the object
Ģ [©]	Dispose(Boolean)	The Dispose method (Overrides Stream.Dispose(Boolean)
€	EndRead(IAsyncResult)	Waits for the pending asynchronous read to complete. (Inherited from Stream.)
=	EndWrite(IAsyncResult)	Ends an asynchronous wr operation. (Inherited from <u>Stream</u> .)
=0	Equals(Object)	Determines whether the specified <u>Object</u> is equal to the current <u>Object</u> .

		(Inherited from Object.)
₹	Finalize()	Allows an Object to attempt of free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
=	Flush()	Flush the stream.
		(Overrides <u>Stream.Flush(</u>)
≡	GetHashCode()	Serves as a hash function a particular type.
		(Inherited from Object.)
≓	GetLifetimeService()	Retrieves the current lifeting service object that controls the lifetime policy for this instance. (Inherited from
		MarshalByRefObject.)
≅©	GetType()	Gets the <u>Type</u> of the curre instance. (Inherited from <u>Object</u> .)
- 🗛		
∄₩	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetim policy for this instance. (Inherited from MarshalByRefObject.)
	Length	Reading this property always throws a NotSupportedException.

		(Overrides Stream.Length
	<u>MaxBufferPairs</u>	The maximum number of buffer pairs to use.
Ģ	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object (Inherited from MarshalByRefObject.)
Ģ [™]	MemberwiseClone()	Creates a shallow copy of the current <u>Object</u> . (Inherited from <u>Object</u> .)
	Position	Returns the current position of the output stream. (Overrides <u>Stream.Positio</u>
€₩	Read(Byte[], Int32, Int32)	This method always throw NotSupportedException. (Overrides Stream.Read(Byte] , Int32 Int32).)
=	ReadByte()	Reads a byte from the stream and advances the position within the stream one byte, or returns -1 if a the end of the stream. (Inherited from Stream .)
	ReadTimeout	Gets or sets a value, in miliseconds, that determin how long the stream will attempt to read before tim

		out.
		(Inherited from <u>Stream</u> .)
≅	Reset(Stream)	Resets the stream for use with another stream.
∃	Seek(Int64, SeekOrigin)	This method always throw NotSupportedException. (Overrides Stream.Seek(Int64, SeekOrigin).)
≓	SetLength(Int64)	This method always throw NotSupportedException. (Overrides Stream.SetLength(Int64).)
	Strategy	The ZLIB strategy to be us during compression.
∄	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
∃	Write(Byte[], Int32, Int32)	Write data to the stream. (Overrides Stream.Write(Byte[], Int32 Int32).)
≓	WriteByte(Byte)	Writes a byte to the currer position in the stream and advances the position with the stream by one byte.

	(Inherited from <u>Stream</u> .)
WriteTimeout	Gets or sets a value, in miliseconds, that determine how long the stream will attempt to write before time out. (Inherited from Stream.)

Remarks

This class performs DEFLATE compression through writing. For more information on the Deflate algorithm, see IETF RFC 1951, "DEFLATE Compressed Data Format Specification version 1.3."

This class is similar to <u>DeflateStream</u>, except that this class is for compression only, and this implementation uses an approach that employs multiple worker threads to perform the DEFLATE. On a multicupu or multi-core computer, the performance of this class can be significantly higher than the single-threaded DeflateStream, particularly for larger streams. How large? Anything over 10mb is a good candidate for parallel compression.

The tradeoff is that this class uses more memory and more CPU than the vanilla DeflateStream, and also is less efficient as a compressor. For large files the size of the compressed data stream can be less than 1% larger than the size of a compressed data stream from the vanialla DeflateStream. For smaller files the difference can be larger. The difference will also be larger if you set the BufferSize to be lower than the default value. Your mileage may vary. Finally, for small files, the ParallelDeflateOutputStream can be much slower than the vanilla DeflateStream, because of the overhead associated to using the thread pool.

■ Inheritance Hierarchy

Object

<u> MarshalByRefObject</u>

Stream

ParallelDeflateOutputStream

See Also

DeflateStream

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

•

1

${\sf Ionic\ Zip\ Library\ v1.9.1.6} \\ {\color{red}\textbf{ParallelDeflateOutputStream}}$

C#

Constructor

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► ParallelDeflateOutputStream()

■ Members

ParallelDeflateOutputStream(Stream) ParallelDeflateOutputStream(Stream, CompressionLevel) ParallelDeflateOutputStream(Stream, ParallelDeflateOutputsusing the specified CompressionLevel. ParallelDeflateOutputStream(Stream, Boolean) ParallelDeflateOutputStream(Stream, ParallelDeflateOutputsusing the specified CompressionLevel.	oon M	lambar	Description
ParallelDeflateOutputStream(Stream, Create a ParallelDeflateOutputStream(Stream, CompressionLevel) ParallelDeflateOutputStream(Stream, ParallelDeflateOutputStream(Stream, CompressionLevel) ParallelDeflateOutputStream(Stream, Boolean) ParallelDeflateOutputStream(Stream, ParallelDeflateOutputStream) ParallelDeflateOutputStream(Stream, ParallelDeflateOutputStream)		nember	Description
ParallelDeflateOutputStream(Stream, Create a ParallelDeflateOutputStream CompressionLevel. ParallelDeflateOutputStream Create a CompressionLevel. ParallelDeflateOutputStream Create a ParallelDeflateOutputStream and specify whether to the compression create a ParallelDeflateOutputStream Create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream and specify whether to the compression create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream (Stream and specify whether to the compression create a ParallelDeflateOutputStream and specify whether to the compression create a ParallelDeflateOutputStream and specify whether to the compression create a ParallelDeflateOutputStream and specify whether to the compression create a ParallelDeflateOutputStream and specify whether a ParallelDeflateOutputStream and specify whe	• Pa	ParallelDeflateOutputStream(Stream)	Create a ParallelDeflateOutputStrea
Boolean) Boolean Bo	<u> </u>	·	ParallelDeflateOutputStreausing the specified
when the	<u>Pc</u>	·	ParallelDeflateOutputStream and specify whether to lead the captive stream open when the ParallelDeflateOutputStream open when the ParallelDeflateOutputStre
and specify whether to the captive stream op- when the	<u>Pc</u>	·	ParallelDeflateOutputStream of specify whether to lead the captive stream open when the ParallelDeflateOutputStream of the captive stream open when the parallelDeflateOutputStream of the captive stream open when the parallelDeflateOutputStream open when the parallelDeflateOutputStream open open open open open open open open
≡♦	•		

<u>ParallelDeflateOutputStream(Stream,</u> Create a <u>CompressionLevel,</u> ParallelD <u>CompressionStrategy, Boolean)</u> using the

ParallelDeflateOutputStreausing the specified
CompressionLevel and
CompressionStrategy, and
specifying whether to leav
the captive stream open
when the
ParallelDeflateOutputStrea
is closed.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel Deflate Output Stream

Constructor (stream)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

ParallelDeflateOutputStream(Stream)

Create a ParallelDeflateOutputStream.

C#

■ Declaration Syntax

Parameters

stream (<u>Stream</u>)

The stream to which compressed data will be written.

■ Remarks

This stream compresses data written into it via the DEFLATE algorithm (see RFC 1951), and writes out the compressed byte stream.

The instance will use the default compression level, the default buffer sizes and the default number of threads and buffers per thread.

This class is similar to **DeflateStream**, except that this implementation

uses an approach that employs multiple worker threads to perform the DEFLATE. On a multi-cpu or multi-core computer, the performance of the class can be significantly higher than the single-threaded DeflateStream particularly for larger streams. How large? Anything over 10mb is a goo candidate for parallel compression.

Examples

This example shows how to use a ParallelDeflateOutputStream to compress data. It reads a file, compresses it, and writes the compresse data to a second, output file.

C# □ Cop

VB.NET

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 Parallel Deflate Output Stream

Constructor (stream, level)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

ParallelDeflateOutputStream(Stream, CompressionLevel)

Create a ParallelDeflateOutputStream using the specified CompressionLevel.

Declaration Syntax

Parameters

stream (Stream)

The stream to which compressed data will be written.

level (CompressionLevel)

A tuning knob to trade speed for effectiveness.

■ Remarks

See the <u>ParallelDeflateOutputStream(Stream)</u> constructor for example

code.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel Deflate Output Stream Constructor (stream, level, strategy, leave Open)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

ParallelDeflateOutputStream(Stream, CompressionLevel, CompressionStrategy, Boolean)

Create a ParallelDeflateOutputStream using the specified

CompressionLevel and CompressionStrategy, and specifying whether to leave the captive stream open when the ParallelDeflateOutputStream is closed.

Declaration Syntax

```
C# Visual Basic Visual C++

public ParallelDeflateOutputStream(
    Stream stream,
    CompressionLevel level,
    CompressionStrategy strategy,
    bool leaveOpen
)
```

Parameters

stream (Stream)

The stream to which compressed data will be written.

level (CompressionLevel)

A tuning knob to trade speed for effectiveness.

strategy (CompressionStrategy)

By tweaking this parameter, you may be able to optimize the compression for data with particular characteristics.

leaveOpen (Boolean)

true if the application would like the stream to remain open after inflation/deflation.

Remarks

See the <u>ParallelDeflateOutputStream(Stream)</u> constructor for example code.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel Deflate Output Stream

Constructor (stream, level, leaveOpen)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

ParallelDeflateOutputStream(Stream, CompressionLevel, Boolean)

Create a ParallelDeflateOutputStream and specify whether to leave the captive stream open when the ParallelDeflateOutputStream is closed.

Declaration Syntax

```
Visual C++
C#
                   Visual Basic
public ParallelDeflateOutputStream(
        Stream stream,
        CompressionLevel level,
        bool leaveOpen
)
```

```
Public Sub New ( _
        stream As Stream, _
        level As CompressionLevel, _
        leaveOpen As <u>Boolean</u> _
```

```
public:
ParallelDeflateOutputStream(
        Stream^ stream,
        CompressionLevel level,
        bool leaveOpen
```

Parameters

stream (Stream)

The stream to which compressed data will be written.

level (CompressionLevel)

A tuning knob to trade speed for effectiveness.

leaveOpen (Boolean)

true if the application would like the stream to remain open after inflation/deflation.

■ Remarks

See the <u>ParallelDeflateOutputStream(Stream)</u> constructor for example code.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Parallel Deflate Output Stream

Constructor (stream, leaveOpen)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

ParallelDeflateOutputStream(Stream, Boolean)

C#

Create a ParallelDeflateOutputStream and specify whether to leave the captive stream open when the ParallelDeflateOutputStream is closed.

Declaration Syntax

Parameters

stream (Stream)

The stream to which compressed data will be written.

leaveOpen (Boolean)

true if the application would like the stream to remain open after inflation/deflation.

■ Remarks

See the <u>ParallelDeflateOutputStream(Stream)</u> constructor for example code.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BufferSize Property

 $\underline{\mathsf{Reference}} \hspace{0.1cm}\blacktriangleright\hspace{0.1cm} \underline{\mathsf{Ionic.Zlib}} \hspace{0.1cm}\blacktriangleright\hspace{0.1cm} \underline{\mathsf{ParallelDeflateOutputStream}} \hspace{0.1cm}\blacktriangleright\hspace{0.1cm} \underline{\mathsf{BufferSize}}$

The size of the buffers used by the compressor threads.

Declaration Syntax

Remarks

The default buffer size is 128k. The application can set this value at any time, but it is effective only before the first Write().

Larger buffer sizes implies larger memory consumption but allows more efficient compression. Using smaller buffer sizes consumes less memor but may result in less effective compression. For example, using the default buffer size of 128k, the compression delivered is within 1% of the compression delivered by the single-threaded DeflateStream. On the other hand, using a BufferSize of 8k can result in a compressed data stream that is 5% larger than that delivered by the single-threaded DeflateStream. Excessively small buffer sizes can also cause the speed of the ParallelDeflateOutputStream to drop, because of larger thread scheduling overhead dealing with many many small buffers.

The total amount of storage space allocated for buffering will be (N*S*2)

where N is the number of buffer pairs, and S is the size of each buffer (this property). There are 2 buffers used by the compressor, one for inplicant one for output. By default, DotNetZip allocates 4 buffer pairs per CPU core, so if your machine has 4 cores, then the number of buffer pairs used will be 16. If you accept the default value of this property, 128k, then the ParallelDeflateOutputStream will use 16 * 2 * 128kb of buffer memory in total, or 4mb, in blocks of 128kb. If you set this proper to 64kb, then the number will be 16 * 2 * 64kb of buffer memory, or 2mb

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Bytes Processed Property

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

BytesProcessed

The total number of uncompressed bytes processed by the ParallelDeflateOutputStream.

Declaration Syntax

```
C# Visual Basic Visual C++
public long BytesProcessed { get; }

Public ReadOnly Property BytesProcessed As Long
    Get

public:
property long long BytesProcessed {
    long long get ();
}
```

Remarks

This value is meaningful only after a call to Close().

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► CanRead

Indicates whether the stream supports Read operations.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea
    Get

public:
virtual property bool CanRead {
    bool get () override;
}
```

■ Remarks

Always returns false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Can Seek Property

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► CanSeek

Indicates whether the stream supports Seek operations.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:
virtual property bool CanSeek {
    bool get () override;
}
```

■ Remarks

Always returns false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanWrite Property

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► CanWrite

Indicates whether the stream supports Write operations.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
    virtual property bool CanWrite {
        bool get () override;
}
```

■ Remarks

Returns true if the provided stream is writable.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Close Method

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► Close()

Close the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Close()

Public Overrides Sub Close

public:
virtual void Close() override

C#

■ Remarks

You must call Close on the stream to guarantee that all of the data written in has been compressed, and the compressed data has been written ou

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Crc32 Property

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► Crc32

The CRC32 for the data that was written out, prior to compression.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public int Crc32 { get; }

Public ReadOnly Property Crc32 As Integer
    Get

public:
property int Crc32 {
    int get ();
}
```

■ Remarks

This value is meaningful only after a call to Close().

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Dispose Method

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► Dispose()

C#

■ Members

Icon	Member	Description
	Dispose()	Dispose the object
Ģ [™]	Dispose(Boolean)	The Dispose method (Overrides <u>Stream.Dispose(Boolean)</u> .

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Dispose Method

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► Dispose()

C#

Dispose the object

■ Declaration Syntax

```
C# Visual Basic Visual C++

public void Dispose()

Public Sub Dispose

public:
void Dispose()
```

■ Remarks

Because ParallelDeflateOutputStream is IDisposable, the application must call this method when finished using the instance.

This method is generally called implicitly upon exit from a using scope in C# (Using in VB).

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Dispose Method (disposing)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

Dispose(Boolean)

The Dispose method



Declaration Syntax

Parameters

disposing (Boolean)

indicates whether the Dispose method was invoked by user code.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Method

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► Flush()

Flush the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public override void Flush()

Public Overrides Sub Flush

public:
virtual void Flush() override

C#

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Length Property

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► Length

Reading this property always throws a NotSupportedException.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:
virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 MaxBufferPairs Property

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

MaxBufferPairs

The maximum number of buffer pairs to use.

C#

■ Declaration Syntax

■ Remarks

This property sets an upper limit on the number of memory buffer pairs create. The implementation of this stream allocates multiple buffers to facilitate parallel compression. As each buffer fills up, this stream uses <a href="https://doi.org/10.2016/nc.

A higher number of buffer pairs enables a higher degree of parallelism, which tends to increase the speed of compression on multi-cpu computers. On the other hand, a higher number of buffer pairs also implies a larger memory consumption, more active worker threads, and higher cpu utilization for any compression. This property enables the application to limit its memory consumption and CPU utilization behavio depending on requirements.

For each compression "task" that occurs in parallel, there are 2 buffers allocated: one for input and one for output. This property sets a limit for the number of pairs. The total amount of storage space allocated for buffering will then be (N*S*2), where N is the number of buffer pairs, S i the size of each buffer (BufferSize). By default, DotNetZip allocates 4 buffer pairs per CPU core, so if your machine has 4 cores, and you reta the default buffer size of 128k, then the ParallelDeflateOutputStream wi use 4 * 4 * 2 * 128kb of buffer memory in total, or 4mb, in blocks of 128kb. If you then set this property to 8, then the number will be 8 * 2 * 128kb of buffer memory, or 2mb.

CPU utilization will also go up with additional buffers, because a larger number of buffer pairs allows a larger number of background threads to compress in parallel. If you find that parallel compression is consuming too much memory or CPU, you can adjust this value downward.

The default value is 16. Different values may deliver better or worse results, depending on your priorities and the dynamic performance characteristics of your storage and compute resources.

This property is not the number of buffer pairs to use; it is an upper limit An illustration: Suppose you have an application that uses the default value of this property (which is 16), and it runs on a machine with 2 CPL cores. In that case, DotNetZip will allocate 4 buffer pairs per CPU core, for a total of 8 pairs. The upper limit specified by this property has no effect.

The application can set this value at any time, but it is effective only before the first call to Write(), which is when the buffers are allocated.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Position Property

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► Position

C#

Returns the current position of the output stream.

Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

■ Remarks

Because the output gets written by a background thread, the value may change asynchronously. Setting this property always throws a NotSupportedException.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

Read(Byte[], Int32, Int32)

This method always throws a NotSupportedException.

C# •

Declaration Syntax

```
C# Visual Basic Visual C++

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

Parameters

buffer (Byte[])

The buffer into which data would be read, IF THIS METHOD ACTUALLY DID ANYTHING.

offset (Int32)

The offset within that data array at which to insert the data that is read, IF THIS METHOD ACTUALLY DID ANYTHING.

count (Int32)

The number of bytes to write, IF THIS METHOD ACTUALLY DID ANYTHING.

■ Return Value

nothing.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Reset Method (stream)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

Reset(Stream)

Resets the stream for use with another stream.

C# -

■ Declaration Syntax

- Parameters

stream (Stream)

The new output stream for this era.

■ Remarks

Because the ParallelDeflateOutputStream is expensive to create, it has been designed so that it can be recycled and re-used. You have to call Close() on the stream first, then you can call Reset() on it, to use it agai on another stream.

Examples

C#



Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

4|

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► Seek(Int64, SeekOrigin)

This method always throws a NotSupportedException.

C# -

Declaration Syntax

```
C# Visual Basic Visual C++

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

Parameters

offset (Int64)

The offset to seek to.... IF THIS METHOD ACTUALLY DID ANYTHING.

origin (SeekOrigin)

The reference specifying how to apply the offset.... IF THIS METHOD ACTUALLY DID ANYTHING.

■ Return Value

nothing. It always throws.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

SetLength(Int64)

This method always throws a NotSupportedException.

C# -

Declaration Syntax

Parameters

value (Int64)

The new value for the stream length.... IF THIS METHOD ACTUALLY DID ANYTHING.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **Strategy Property**

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ► Strategy

C#

The ZLIB strategy to be used during compression.

■ Declaration Syntax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

_

▶

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

Reference ► Ionic.Zlib ► ParallelDeflateOutputStream ►

Write(Byte[], Int32, Int32)

Write data to the stream.

C#

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

```
public:
virtual void Write(
          array<<u>unsigned char</u>>^ buffer,
          <u>int</u> offset,
          <u>int</u> count
) override
```

Parameters

buffer (Byte∏)

The buffer holding data to write to the stream.

offset (Int32)

the offset within that data array to find the first byte to write.

count (Int32)

the number of bytes to write.

■ Remarks

To use the ParallelDeflateOutputStream to compress data, create a ParallelDeflateOutputStream with CompressionMode.Compress, passir a writable output stream. Then call Write() on that ParallelDeflateOutputStream, providing uncompressed data as input. The data sent to the output stream will be the compressed form of the data written.

To decompress data, use the **DeflateStream** class.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZlibCodec Class

Reference ► Ionic.Zlib ► ZlibCodec

C#

Encoder and Decoder for ZLIB and DEFLATE (IETF RFC1950 and RFC1951).

■ Declaration Syntax

C#	Visual Basic	Visual C++	
public sealed class ZlibCodec			
Public NotInheritable Class ZlibCodec			

public ref class ZlibCodec sealed

■ Members

All M	embers	Constructors	Methods	Properties	Fields
		✓ Instance✓ Static		DeclaredInherited	
Icon	Membe	er		Description	
€ \	ZlibCod	<u>lec()</u>		Create a Zlib	Codec.
≡ ₩	ZlibCodec(CompressionMode)		Create a ZlibCodec that either compresses or decompresses.		
	Adler32			The Adler32 of the data trans through the co You probably to look at this	sferred odec so far. don't need

	AvailableBytesIn	The number of bytes available in the InputBuffer, starting at NextIn.
•	AvailableBytesOut	The number of bytes available in the OutputBuffer, starting at NextOut.
•	CompressLevel	The compression level to use in this codec. Useful only in compression mode.
₫ ◊	Deflate(FlushType)	Deflate one batch of data
₫∳	EndDeflate()	End a deflation session.
₫	EndInflate()	Ends an inflation session
₫ ◊	Equals(Object)	Determines whether the specified <u>Object</u> is equal to the current <u>Object</u> . (Inherited from <u>Object</u> .)
Ģ Û	Finalize()	Allows an Object to attempt to free resources and perform other cleanu

		operations before the Object is reclaimed by garbage collection. (Inherited from Object.)
=	GetHashCode()	Serves as a hash function for a particular type. (Inherited from Object.)
≡	GetType()	Gets the Type of the current instance. (Inherited from Object.)
∄	Inflate(FlushType)	Inflate the data in the InputBuffer, placing the result in the OutputBuffer
≡	InitializeDeflate()	Initialize the ZlibCodec fo deflation operation.
∃ ©	InitializeDeflate(CompressionLevel)	Initialize the ZlibCodec fo deflation operation, using the specified CompressionLevel.
Ξ₩	InitializeDeflate(CompressionLevel, Boolean)	Initialize the ZlibCodec fo deflation operation, using the specified CompressionLevel, and the explicit flag governing whether to emit an RFC1950 header byte pair.

≓	InitializeDeflate(CompressionLevel, Int32)	Initialize the ZlibCodec fo deflation operation, using the specified CompressionLevel, and the specified number of window bits.
∄	InitializeDeflate(CompressionLevel, Int32, Boolean)	Initialize the ZlibCodec fo deflation operation, using the specified CompressionLevel, the specified number of window bits, and the explicit flag governing whether to emit an RFC1950 header byte pair.
=	InitializeInflate()	Initialize the inflation state
∄ 🗘	InitializeInflate(Boolean)	Initialize the inflation state with an explicit flag to govern the handling of RFC1950 header bytes.
∄ 🗘	InitializeInflate(Int32)	Initialize the ZlibCodec fo inflation, with the specifie number of window bits.
=0	InitializeInflate(Int32, Boolean)	Initialize the inflation state

		with an explicit flag to govern the handling of RFC1950 header bytes.
•	<u>InputBuffer</u>	The buffer from which data is taken.
Ģ [¯] ©	MemberwiseClone()	Creates a shallow copy o the current <u>Object</u> . (Inherited from <u>Object</u> .)
•	Message	used for diagnostics, when something goes wrong!
•	NextIn	An index into the InputBuffer array, indicating where to start reading.
•	NextOut	An index into the OutputBuffer array, indicating where to start writing.
•	OutputBuffer	Buffer to store output data.
≡	ResetDeflate()	Reset a codec for anothe deflation session.

∃	SetDeflateParams(CompressionLevel, CompressionStrategy)	Set the CompressionStrategy and CompressionLevel for a deflation session.
₫∳	SetDictionary(Byte[])	Set the dictionary to be used for either Inflation of Deflation.
•	Strategy	The compression strategy to use.
≡	SyncInflate()	I don't know what this does!
≡ ◊	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> . (Inherited from <u>Object</u> .)
٠	TotalBytesIn	Total number of bytes read so far, through all calls to Inflate()/Deflate().
•	TotalBytesOut	Total number of bytes written to the output so fa through all calls to Inflate()/Deflate().

•	WindowBits	The number of Window Bits to use.

■ Remarks

This class compresses and decompresses data according to the Deflate algorithm and optionally, the ZLIB format, as documented in RFC 1950 ZLIB and RFC 1951 - DEFLATE.

■ Inheritance Hierarchy

Object

ZlibCodec

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZlibCodec Constructor

Reference ► Ionic.Zlib ► ZlibCodec ► ZlibCodec()

C# •

■ Members

Icon	Member	Description
≟	ZlibCodec()	Create a ZlibCodec.
∄ 🗳	ZlibCodec(CompressionMode)	Create a ZlibCodec that either compresses or decompresses.

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZlibCodec Constructor

Reference ► Ionic.Zlib ► ZlibCodec ► ZlibCodec()

C#

Create a ZlibCodec.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public ZlibCodec()

Public Sub New

public:
ZlibCodec()
```

■ Remarks

If you use this default constructor, you will later have to explicitly call InitializeInflate() or InitializeDeflate() before using the ZlibCodec to compress or decompress.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZlibCodec Constructor (mode)

Reference ► Ionic.Zlib ► ZlibCodec ►

ZlibCodec(CompressionMode)

Create a ZlibCodec that either compresses or decompresses.

■ Declaration Syntax

Parameters

mode (CompressionMode)

Indicates whether the codec should compress (deflate) or decompress (inflate).

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Adler32 Property

Reference ► Ionic.Zlib
► ZlibCodec ► Adler32

C#
```

The Adler32 checksum on the data transferred through the codec so far You probably don't need to look at this.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public int Adler32 { get; }

Public ReadOnly Property Adler32 As Integer
    Get

public:
property int Adler32 {
    int get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Available Bytes In Field

Reference ► Ionic.Zlib ► ZlibCodec ► AvailableBytesIn

The number of bytes available in the InputBuffer, starting at NextIn.

Declaration Syntax

C# Visual Basic Visual C++

public int AvailableBytesIn

Public AvailableBytesIn As Integer

public:

int AvailableBytesIn

■ Remarks

Generally you should set this to InputBuffer.Length before the first Inflate() or Deflate() call. The class will update this number as calls to Inflate/Deflate are made.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Available Bytes Out Field

Reference ► Ionic.Zlib ► ZlibCodec ► AvailableBytesOut

C# To NovtOut

The number of bytes available in the OutputBuffer, starting at NextOut.

Declaration Syntax

C# Visual Basic Visual C++

public int AvailableBytesOut

Public AvailableBytesOut As Integer

public:

int AvailableBytesOut

■ Remarks

Generally you should set this to OutputBuffer.Length before the first Inflate() or Deflate() call. The class will update this number as calls to Inflate/Deflate are made.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CompressLevel Field

Reference ► Ionic.Zlib ► ZlibCodec ► CompressLevel C#

The compression level to use in this codec. Useful only in compression mode.

■ Declaration Syntax

C# Visual Basic Visual C++

public CompressionLevel CompressLevel

Public CompressLevel As CompressionLevel

public:

CompressionLevel CompressLevel

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Deflate Method (flush)

Reference ► Ionic.Zlib ► ZlibCodec ► Deflate(FlushType)

Deflate one batch of data.

Declaration Syntax

C#

Parameters

flush (FlushType)

whether to flush all data as you deflate. Generally you will want to use Z_NO_FLUSH here, in a series of calls to Deflate(), and then call EndDeflate() to flush everything.

■ Return Value

Z_OK if all goes well.

■ Remarks

You must have set InputBuffer and OutputBuffer before calling this method.

Examples

C# □ Cop

```
private void DeflateBuffer(CompressionLevel level)
   int bufferSize = 1024;
   byte[] buffer = new byte[bufferSize];
   ZlibCodec compressor = new ZlibCodec();
   Console.WriteLine("Size of Buffer to Deflate: {0
   MemoryStream ms = new MemoryStream();
   int rc = compressor.InitializeDeflate(level);
   compressor.InputBuffer = UncompressedBytes;
   compressor.NextIn = 0;
   compressor.AvailableBytesIn = UncompressedBytes.
   compressor.OutputBuffer = buffer;
   // pass 1: deflate
   do
   {
       compressor.NextOut = 0;
       compressor.AvailableBytesOut = buffer.Length
       rc = compressor.Deflate(FlushType.None);
       if (rc != ZlibConstants.Z_OK && rc != ZlibCo
           throw new Exception("deflating: " + comp
       ms.Write(compressor.OutputBuffer, 0, buffer.
   while (compressor.AvailableBytesIn > 0 || compre
   // pass 2: finish and flush
   do
   {
       compressor.NextOut = 0;
       compressor.AvailableBytesOut = buffer.Length
```

```
rc = compressor.Deflate(FlushType.Finish);
        if (rc != ZlibConstants.Z_STREAM_END && rc !
            throw new Exception("deflating: " + comp
        if (buffer.Length - compressor.AvailableByte
            ms.Write(buffer, 0, buffer.Length - comp
   while (compressor.AvailableBytesIn > 0 || compre
    compressor.EndDeflate();
    ms.Seek(0, SeekOrigin.Begin);
    CompressedBytes = new byte[compressor.TotalBytes
    ms.Read(CompressedBytes, 0, CompressedBytes.Leng
}
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 End Deflate Method

Reference ► Ionic.Zlib ► ZlibCodec ► EndDeflate()

C#

End a deflation session.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public int EndDeflate()

Public Function EndDeflate As Integer
```

```
public:
int EndDeflate()
```

Return Value

Z OK if all goes well.

■ Remarks

Call this after making a series of one or more calls to Deflate(). All buffe are flushed.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 EndInflate Method

Reference ► Ionic.Zlib ► ZlibCodec ► EndInflate()

C#

Ends an inflation session.

Declaration Syntax

```
C# Visual Basic Visual C++
public int EndInflate()

Public Function EndInflate As Integer

public:
int EndInflate()
```

Return Value

Z OK if everything goes well.

■ Remarks

Call this after successively calling Inflate(). This will cause all buffers to be flushed. After calling this you cannot call Inflate() without a intervenir call to one of the InitializeInflate() overloads.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Inflate Method (flush)

Reference ► Ionic.Zlib ► ZlibCodec ► Inflate(FlushType)

Inflate the data in the InputBuffer, placing the result in the OutputBuffer.

■ Declaration Syntax

)

flush (FlushType)

Parameters

The flush to use when inflating.

FlushType *flush*

■ Return Value

Z_OK if everything goes well.

■ Remarks

You must have set InputBuffer and OutputBuffer, NextIn and NextOut, and AvailableBytesIn and AvailableBytesOut before calling this method.

Examples

C#

```
private void InflateBuffer()
   int bufferSize = 1024;
   byte[] buffer = new byte[bufferSize];
   ZlibCodec decompressor = new ZlibCodec();
   Console.WriteLine("Size of Buffer to Inflate: {0
   MemoryStream ms = new MemoryStream(DecompressedB)
   int rc = decompressor.InitializeInflate();
   decompressor.InputBuffer = CompressedBytes;
   decompressor.NextIn = 0;
   decompressor.AvailableBytesIn = CompressedBytes.
   decompressor.OutputBuffer = buffer;
   // pass 1: inflate
   do
   {
       decompressor.NextOut = 0;
       decompressor.AvailableBytesOut = buffer.Leng
       rc = decompressor.Inflate(FlushType.None);
       if (rc != ZlibConstants.Z_OK && rc != ZlibCo
           throw new Exception("inflating: " + deco
       ms.Write(decompressor.OutputBuffer, 0, buffe
   while (decompressor.AvailableBytesIn > 0 || deco
   // pass 2: finish and flush
   do
   {
       decompressor.NextOut = 0;
       decompressor.AvailableBytesOut = buffer.Leng
       rc = decompressor.Inflate(FlushType.Finish);
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Initialize Deflate Method

Reference ► Ionic.Zlib ► ZlibCodec ► InitializeDeflate()

C#

■ Members

Icon	Member	Description
=	InitializeDeflate()	Initialize the ZlibCodec for deflation operation.
∃∅	InitializeDeflate(CompressionLevel)	Initialize the ZlibCodec for deflation operation, using the specified CompressionLevel
€₩	InitializeDeflate(CompressionLevel, Boolean)	Initialize the ZlibCodec for deflation operation, using the specified CompressionLevel and the explicit flag governing whether to emit a RFC1950 header byte pair.
∃	InitializeDeflate(CompressionLevel, Int32)	Initialize the ZlibCodec for deflation operation, using the specified CompressionLevel and the specified number of window bits.
≘₽	InitializeDeflate(CompressionLevel, Int32, Boolean)	Initialize the ZlibCodec for deflation operation, using the specified CompressionLevel the specified number of window bits, and the explicit flag governing whether to emit an RFC1950 header

|--|

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Initialize Deflate Method

Reference ► Ionic.Zlib ► ZlibCodec ► InitializeDeflate()

Initialize the ZlibCodec for deflation operation.

C# -

Declaration Syntax

```
C# Visual Basic Visual C++
public int InitializeDeflate()

Public Function InitializeDeflate As Integer

public:
int InitializeDeflate()
```

Return Value

Z_OK if all goes well. You generally don't need to check the return code

■ Remarks

The codec will use the MAX window bits and the default level of compression.

Examples

C#

```
int bufferSize = 40000;
byte[] CompressedBytes = new byte[bufferSize];
byte[] DecompressedBytes = new byte[bufferSize];

ZlibCodec compressor = new ZlibCodec();

compressor.InitializeDeflate(CompressionLevel.Defaul)

compressor.InputBuffer = System.Text.ASCIIEncoding.Accompressor.NextIn = 0;
compressor.AvailableBytesIn = compressor.InputBuffer
```

```
compressor.OutputBuffer = CompressedBytes;
compressor.NextOut = 0;
compressor.AvailableBytesOut = CompressedBytes.Lengt
while (compressor.TotalBytesIn != TextToCompress.Len
{
   compressor.Deflate(FlushType.None);
}
while (true)
{
   int rc= compressor.Deflate(FlushType.Finish);
   if (rc == ZlibConstants.Z_STREAM_END) break;
}
compressor.EndDeflate();
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

▶

Ionic Zip Library v1.9.1.6 Initialize Deflate Method (level)

Reference ► Ionic.Zlib ► ZlibCodec ►

InitializeDeflate(CompressionLevel)

Initialize the ZlibCodec for deflation operation, using the specified CompressionLevel.

■ Declaration Syntax

Parameters

level (CompressionLevel)

The compression level for the codec.

■ Return Value

Z_OK if all goes well.

■ Remarks

The codec will use the maximum window bits (15) and the specified CompressionLevel. It will emit a ZLIB stream as it compresses.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Initialize Deflate Method (level,

wantRfc1950Header)

Reference ► Ionic.Zlib ► ZlibCodec ►

InitializeDeflate(CompressionLevel, Boolean)

Initialize the ZlibCodec for deflation operation, using the specified CompressionLevel, and the explicit flag governing whether to emit an RFC1950 header byte pair.

Declaration Syntax

Parameters

level (CompressionLevel)

The compression level for the codec.

wantRfc1950Header (Boolean)

whether to emit an initial RFC1950 byte pair in the compressed stream.

- Return Value

Z_OK if all goes well.

■ Remarks

The codec will use the maximum window bits (15) and the specified CompressionLevel. If you want to generate a zlib stream, you should specify true for wantRfc1950Header. In this case, the library will emit a ZLIB header, as defined in RFC 1950, in the compressed stream.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Initialize Deflate Method (level, bits)

Reference ► Ionic.Zlib ► ZlibCodec ►

InitializeDeflate(CompressionLevel, Int32)

Initialize the ZlibCodec for deflation operation, using the specified CompressionLevel, and the specified number of window bits.

Declaration Syntax

Parameters

level (CompressionLevel)

The compression level for the codec.

bits (Int32)

the number of window bits to use. If you don't know what this mean don't use this method.

■ Return Value

Z OK if all goes well.

■ Remarks

The codec will use the specified number of window bits and the specifie CompressionLevel.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Initialize Deflate Method (level, bits, wantRfc1950Header)

Reference ► Ionic.Zlib ► ZlibCodec ►

InitializeDeflate(CompressionLevel, Int32, Boolean)

Initialize the ZlibCodec for deflation operation, using the specified CompressionLevel, the specified number of window bits, and the explicitlag governing whether to emit an RFC1950 header byte pair.

Declaration Syntax

Parameters

level (CompressionLevel)

The compression level for the codec.

bits (Int32)

the number of window bits to use. If you don't know what this mean don't use this method.

wantRfc1950Header (Boolean)

whether to emit an initial RFC1950 byte pair in the compressed stream.

■ Return Value

Z_OK if all goes well.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Initialize Inflate Method

Reference ► Ionic.Zlib ► ZlibCodec ► InitializeInflate() C#

■ Members

Icon	Member	Description
≅©	InitializeInflate()	Initialize the inflation state.
≡ •	InitializeInflate(Boolean)	Initialize the inflation state with an explicit flag to govern the handling of RFC1950 header bytes.
Ξ₩	InitializeInflate(Int32)	Initialize the ZlibCodec for inflation, with the specified number of window bits.
∃	InitializeInflate(Int32, Boolean)	Initialize the inflation state with an explicit flag to govern the handling of RFC1950 header bytes.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Initialize Inflate Method

Reference ► Ionic.Zlib ► ZlibCodec ► InitializeInflate() C#

Initialize the inflation state.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public int InitializeInflate()

Public Function InitializeInflate As Integer

public:
int InitializeInflate()
```

Return Value

Z OK if everything goes well.

■ Remarks

It is not necessary to call this before using the ZlibCodec to inflate data; is implicitly called when you call the constructor.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Initialize Inflate Method (expectRfc1950Header)

Reference ► Ionic.Zlib ► ZlibCodec ► InitializeInflate(Boolean)

Initialize the inflation state with an explicit flag to govern the handling of RFC1950 header bytes.

■ Declaration Syntax

Parameters

expectRfc1950Header (Boolean)

whether to expect an RFC1950 header byte pair when reading the stream of data to be inflated.

■ Return Value

Z_OK if everything goes well.

■ Remarks

By default, the ZLIB header defined in RFC 1950 is expected. If you wa to read a zlib stream you should specify true for expectRfc1950Header. you have a deflate stream, you will want to specify false. It is only

necessary to invoke this initializer explicitly if you want to specify false.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Initialize Inflate Method (windowBits)

Reference ► Ionic.Zlib ► ZlibCodec ► InitializeInflate(Int32)

Initialize the ZlibCodec for inflation, with the specified number of window bits.

Declaration Syntax

```
C# Visual Basic Visual C++

public int InitializeInflate(
    int windowBits
)

Public Function InitializeInflate ( __
    windowBits As Integer __
) As Integer
```

Parameters

windowBits (Int32)

The number of window bits to use. If you need to ask what that is, then you shouldn't be calling this initializer.

Return Value

Z_OK if all goes well.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Initialize Inflate Method (windowBits, expectRfc1950Header)

Reference ► <u>Ionic.Zlib</u> ► <u>ZlibCodec</u> ► <u>InitializeInflate(Int32, Boolean)</u>

Initialize the inflation state with an explicit flag to govern the handling of RFC1950 header bytes.

Declaration Syntax

```
C# Visual Basic Visual C++

public int InitializeInflate(
    int windowBits,
    bool expectRfc1950Header
)
```

```
Public Function InitializeInflate ( _ windowBits As Integer, _ expectRfc1950Header As Boolean _ ) As Integer
```

Parameters

windowBits (Int32)

The number of window bits to use. If you need to ask what that is, then you shouldn't be calling this initializer.

expectRfc1950Header (Boolean)

whether to expect an RFC1950 header byte pair when reading the stream of data to be inflated.

Return Value

Z OK if everything goes well.

■ Remarks

If you want to read a zlib stream you should specify true for expectRfc1950Header. In this case, the library will expect to find a ZLIB header, as defined in RFC 1950, in the compressed stream. If you will be reading a DEFLATE or GZIP stream, which does not have such a heade you will want to specify false.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

The buffer from which data is taken.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public byte[] InputBuffer
Public InputBuffer As Byte()

public:
array<unsigned char>^ InputBuffer
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Message Field

Reference ► Ionic.Zlib ► ZlibCodec ► Message

C# -

used for diagnostics, when something goes wrong!

■ Declaration Syntax

C# Visual Basic Visual C++

public <u>string</u> Message

Public Message As <u>String</u>

public:

String^ Message

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

An index into the InputBuffer array, indicating where to start reading.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public int NextIn

Public NextIn As Integer

public:
int NextIn
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

An index into the OutputBuffer array, indicating where to start writing.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public int NextOut

Public NextOut As Integer

public:
int NextOut
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 OutputBuffer Field

Reference ► Ionic.Zlib ► ZlibCodec ► OutputBuffer

C#

Buffer to store output data.

■ Declaration Syntax

C# Visual Basic Visual C++

public byte[] OutputBuffer

Public OutputBuffer As Byte()

public:

array<<u>unsigned char</u>>^ OutputBuffer

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ResetDeflate Method

Reference ► Ionic.Zlib ► ZlibCodec ► ResetDeflate()

C#

Reset a codec for another deflation session.

■ Declaration Syntax

C# Visual Basic Visual C++

public void ResetDeflate()

Public Sub ResetDeflate

public:

public: void ResetDeflate()

Return Value

Z_OK if all goes well.

■ Remarks

Call this to reset the deflation state. For example if a thread is deflating non-consecutive blocks, you can call Reset() after the Deflate(Sync) of the first block and before the next Deflate(None) of the second block.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetDeflateParams Method (level,

strategy)

Reference ► Ionic.Zlib ► ZlibCodec ►

SetDeflateParams(CompressionLevel, CompressionStrategy)

Set the CompressionStrategy and CompressionLevel for a deflation session.

Declaration Syntax

Parameters

level (CompressionLevel)

the level of compression to use.

strategy (CompressionStrategy)

the strategy to use for compression.

■ Return Value

Z_OK if all goes well.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Set Dictionary Method (dictionary)

Reference ► Ionic.Zlib ► ZlibCodec ► SetDictionary(Byte[])

C#

Set the dictionary to be used for either Inflation or Deflation.

■ Declaration Syntax

```
dictionary As Byte() _
) As Integer
```

Parameters

dictionary (Byte[])

The dictionary bytes to use.

■ Return Value

Z_OK if all goes well.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 | Strategy Field | | Reference | Ionic.Zlib | ZlibCodec | Strategy | C# | → |

The compression strategy to use.

■ Declaration Syntax

C# Visual Basic Visual C++

public CompressionStrategy Strategy

Public Strategy As CompressionStrategy

public:

CompressionStrategy Strategy

■ Remarks

This is only effective in compression. The theory offered by ZLIB is that different strategies could potentially produce significant differences in compression behavior for different data sets. Unfortunately I don't have any good recommendations for how to set it differently. When I tested changing the strategy I got minimally different compression performance It's best to leave this property alone if you don't have a good feel for it. Or, you may want to produce a test harness that runs through the different strategy options and evaluates them on different file types. If yo do that, let me know your results.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SyncInflate Method

Reference ► Ionic.Zlib ► ZlibCodec ► SyncInflate()

C#

I don't know what this does!

■ Declaration Syntax

```
C# Visual Basic Visual C++
public int SyncInflate()
```

Public Function SyncInflate As Integer

```
public:
int SyncInflate()
```

■ Return Value

Z OK if everything goes well.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

| Ionic Zip Library v1.9.1.6 Total Bytes In Field | Reference ► Ionic.Zlib ► ZlibCodec ► Total Bytes In C#

Total number of bytes read so far, through all calls to Inflate()/Deflate().

■ Declaration Syntax

```
C# Visual Basic Visual C++

public long TotalBytesIn

Public TotalBytesIn As Long

public:
long long TotalBytesIn
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Total BytesOut Field

Reference ► Ionic.Zlib ► ZlibCodec ► TotalBytesOut C#

Total number of bytes written to the output so far, through all calls to Inflate()/Deflate().

■ Declaration Syntax

C# Visual Basic Visual C++

public long TotalBytesOut

Public TotalBytesOut As Long

public:
long long TotalBytesOut

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

The number of Window Bits to use.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public int WindowBits

Public WindowBits As Integer

public:
int WindowBits
```

■ Remarks

This gauges the size of the sliding window, and hence the compression effectiveness as well as memory consumption. It's best to just leave this setting alone if you don't know what it is. The maximum value is 15 bits, which implies a 32k window.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZlibConstants Class

Reference ► Ionic.Zlib ► ZlibConstants

C#

A bunch of constants used in the Zlib interface.

■ Declaration Syntax

C# Visual Basic Visual C++
public static class ZlibConstants

Public NotInheritable Class ZlibConstants

public ref class ZlibConstants abstract sealed

■ Members

All M	embers	Fields			
PublicProtected			✓ Instance✓ Static	DeclaredInherited	
Icon	Member		Description		
♦ \$	WindowBitsDefau	<u>ult</u>	The default if for the Defla		
♦ \$	WindowBitsMax		The maximu for the Defla		
4 S	WorkingBufferSiz	<u>eDefault</u>	The size of the ZlibCode bytes.	•	
•\$	WorkingBufferSiz	<u>ceMin</u>	The minimur buffer used i		•

		Currently it is 128 bytes.
9 S	Z_BUF_ERROR	There was an error with the working buffer.
9 S	Z_DATA_ERROR	There was an error with the data - not enough data, bad data, etc.
•3	Z_NEED_DICT	The operation ended in need of a dictionary.
•3	Z_OK	indicates everything is A-OK
S	Z_STREAM_END	Indicates that the last operation reached the end of the stream.
•\$	Z_STREAM_ERROR	There was an error with the stream not enough data, not open and readable, etc.

■ Inheritance Hierarchy

Object

ZlibConstants

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Window Bits Default Field

Reference ► Ionic.Zlib ► ZlibConstants ► WindowBitsDefault

C#

The default number of window bits for the Deflate algorithm.

■ Declaration Syntax

C# Visual Basic Visual C++

public const <u>int</u> WindowBitsDefault

Public Const WindowBitsDefault As Integer

public:

literal <u>int</u> WindowBitsDefault

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 WindowBitsMax Field

Reference ► Ionic.Zlib ► ZlibConstants ► WindowBitsMax

The maximum number of window bits for the Deflate algorithm.

■ Declaration Syntax

C# Visual Basic Visual C++

public const int WindowBitsMax

Public Const WindowBitsMax As <u>Integer</u>

public:

literal <u>int</u> WindowBitsMax

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Working Buffer Size Default Field

Reference ► Ionic.Zlib ► ZlibConstants ► WorkingBufferSizeDefault

The size of the working buffer used in the ZlibCodec class. Defaults to 8192 bytes.

■ Declaration Syntax

C# Visual Basic Visual C++
public const <u>int</u> WorkingBufferSizeDefault

Public Const WorkingBufferSizeDefault As Integer

public: literal int WorkingBufferSizeDefault

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Working Buffer Size Min Field

Reference ► Ionic.Zlib ► ZlibConstants ► WorkingBufferSizeMin

The minimum size of the working buffer used in the ZlibCodec class. Currently it is 128 bytes.

■ Declaration Syntax

C# Visual Basic Visual C++

public const int WorkingBufferSizeMin

Public Const WorkingBufferSizeMin As Integer

public: literal int WorkingBufferSizeMin

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Z_BUF_ERROR Field

Reference ► Ionic.Zlib ► ZlibConstants ► Z_BUF_ERROR

There was an error with the working buffer.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++

public const int Z_BUF_ERROR

Public Const Z_BUF_ERROR As Integer

public:
literal int Z_BUF_ERROR
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Z_DATA_ERROR Field

Reference ► Ionic.Zlib ► ZlibConstants ► Z_DATA_ERROR

There was an error with the data - not enough data, bad data, etc.

■ Declaration Syntax

C# Visual Basic Visual C++

public const int Z_DATA_ERROR

Public Const Z_DATA_ERROR As Integer

public:

literal <u>int</u> Z_DATA_ERROR

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Z_NEED_DICT Field

Reference ► Ionic.Zlib ► ZlibConstants ► Z_NEED_DICT

The operation ended in need of a dictionary.

C#

■ Declaration Syntax

```
C# Visual Basic Visual C++

public const int Z_NEED_DICT

Public Const Z_NEED_DICT As Integer

public:
literal int Z_NEED_DICT
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Z_OK Field

Reference ► Ionic.Zlib ► ZlibConstants ► Z_OK

C#

indicates everything is A-OK

■ Declaration Syntax

C# Visual Basic Visual C++

public const int Z_OK

Public Const Z_OK As Integer

public:

literal <u>int</u> Z_OK

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Z STREAM END Field

Reference ► Ionic.Zlib ► ZlibConstants ► Z_STREAM_END

C#

Indicates that the last operation reached the end of the stream.

■ Declaration Syntax

C# Visual Basic Visual C++

public const int Z_STREAM_END

Public Const Z_STREAM_END As Integer

public: literal int Z_STREAM_END

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Z_STREAM_ERROR Field

Reference ► Ionic.Zlib ► ZlibConstants ► Z_STREAM_ERROR

There was an error with the stream - not enough data, not open and readable, etc.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public const int Z_STREAM_ERROR

Public Const Z_STREAM_ERROR As Integer

public:
literal int Z_STREAM_ERROR
```

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zlib Exception Class

Reference ► Ionic.Zlib ► ZlibException

C# -

A general purpose exception class for exceptions in the Zlib library.

■ Declaration Syntax

C# Visual Basic Visual C++

public class ZlibException : Exception

Public Class ZlibException _ Inherits Exception

public ref class ZlibException : public <a>Exception

■ Members

All M	embers	Constructors	Metho	ods	Properties	
✓ Public✓ Protected		✓ Instance✓ Static		Declare Inherite		
Icon Member				Description		
≓	ZlibException()		The ZlibException class captures exception information generated by the Zlib library.			
± ⊕	ZlibExce	eption(String)			or collects a mes d to the exception	•
	<u>Data</u>			pairs th	collection of key at provide additi fined information eption.	onal

		(Inherited from Exception.)
≟∳	Equals(Object)	Determines whether the specified <u>Object</u> is equal to the current <u>Object</u> .
		(Inherited from Object.)
ૄ	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.
		(Inherited from Object.)
∃ ₩	GetBaseException()	When overridden in a derived class, returns the Exception that is the root cause of one or more subsequent exceptions. (Inherited from Exception.)
≡	CotHashCodo()	Serves as a hash function for a
	GetHashCode()	particular type. (Inherited from Object.)
Ξ Φ	GetObjectData(SerializationInfo, StreamingContext)	When overridden in a derived class, sets the <u>SerializationInfo</u> with information about the exception. (Inherited from <u>Exception</u> .)
€ 🔷	GetType()	Gets the runtime type of the current instance.
		(Inherited from Exception.)
	<u>HelpLink</u>	Gets or sets a link to the help file associated with this exception.

		(Inherited from Exception.)
	HResult	Gets or sets HRESULT, a code numerical value that is assigned to a specific exception. (Inherited from Exception.)
	InnerException	Gets the Exception instance the caused the current exception. (Inherited from Exception.)
Ģ [©]	MemberwiseClone()	Creates a shallow copy of the current <u>Object</u> . (Inherited from <u>Object</u> .)
	Message	Gets a message that describes the current exception. (Inherited from Exception.)
	Source	Gets or sets the name of the application or the object that causes the error. (Inherited from Exception.)
	<u>StackTrace</u>	Gets a string representation of the frames on the call stack at the time the current exception was thrown. (Inherited from Exception.)
	<u>TargetSite</u>	Gets the method that throws the current exception. (Inherited from Exception.)
=•	ToString()	Creates and returns a string

representation of the current exception.

(Inherited from **Exception**.)

■ Inheritance Hierarchy

Object

Exception

ZlibException

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 **ZlibException Constructor**

Reference ► Ionic.Zlib ► ZlibException ► ZlibException()

C# -

■ Members

Icon	Member	Description
€₩	ZlibException()	The ZlibException class captures exception information generated by the Zlib library.
≅	ZlibException(String)	This ctor collects a message attache to the exception.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zlib Exception Constructor

Reference ► Ionic.Zlib ► ZlibException ► ZlibException()

The ZlibException class captures exception information generated by the Zlib library.

■ Declaration Syntax

```
Visual C++
C#
                   Visual Basic
public ZlibException()
Public Sub New
public:
ZlibException()
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Zlib Exception Constructor (s)

Reference ► Ionic.Zlib ► ZlibException ► ZlibException(String)

This ctor collects a message attached to the exception.

■ Declaration Syntax

Parameters

s (String)

the message for the exception.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZlibStream Class

Reference ► Ionic.Zlib ► ZlibStream

C#

Represents a Zlib stream for compression or decompression.

■ Declaration Syntax

C# Visual Basic Visual C++

public class ZlibStream : Stream

Public Class ZlibStream _ Inherits Stream

public ref class ZlibStream : public <u>Stream</u>

Members

All M	embers	Constructors	M	ethods	Properties	
✓ Public✓ Protected			✓ Instance✓ Static		Declare Inherite	
Icon Member			Description			
≡		am(Stream, ssionMode)			libStream us ompression!	•
≅	Compres	am(Stream, ssionMode, ssionLevel)		specified C and the spe	libStream us ompression! ecified sionLevel.	•
≡		am(Stream, ssionMode,		Create a Z	libStream us	sing the

	Boolean)	specified CompressionMode, and explicitly specify whether the captive stream should be left open after Deflation or Inflation.
≅	ZlibStream(Stream, CompressionMode, CompressionLevel, Boolean)	Create a ZlibStream using the specified CompressionMode and the specified CompressionLevel, and explicitly specify whether the strear should be left open after Deflation of Inflation.
₫ ∳	BeginRead(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous read operation. (Inherited from <u>Stream</u> .)
∃	BeginWrite(Byte[], Int32, Int32, AsyncCallback, Object)	HostProtectionAttribute. Begins an asynchronous write operation. (Inherited from <u>Stream</u> .)
	BufferSize	The size of the working buffer for the compression codec.
	CanRead	Indicates whether the stream can b read. (Overrides <u>Stream.CanRead.</u>)
	CanSeek	Indicates whether the stream supports Seek operations.

	(Overrides <u>Stream.CanSeek.</u>)
<u>CanTimeout</u>	Gets a value that determines whether the current stream can tim out.
	(Inherited from <u>Stream</u> .)
<u>CanWrite</u>	Indicates whether the stream can b written.
	(Overrides <u>Stream</u> . <u>CanWrite</u> .)
Close()	Closes the current stream and releases any resources (such as sockets and file handles) associate with the current stream. (Inherited from Stream .)
CompressBuffer(Byte[])	Compress a byte array into a new byte array using ZLIB.
CompressString(String)	Compress a string into a byte array using ZLIB.
<u>CreateObjRef(Type)</u>	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.
	(Inherited from MarshalByRefObject.)
CreateWaitHandle()	•
<u>CreateWaitHandle()</u>	MarshalByRefObject.)
CreateWaitHandle()	MarshalByRefObject.) Obsolete.
	CanWrite Close() CompressBuffer(Byte[]) CompressString(String)

		(Overrides <u>Stream.Dispose(Boolean)</u> .)
≓	Dispose()	Releases all resources used by the Stream.
		(Inherited from <u>Stream</u> .)
≟	EndRead(IAsyncResult)	Waits for the pending asynchronouread to complete.
		(Inherited from <u>Stream</u> .)
≟	EndWrite(IAsyncResult)	Ends an asynchronous write operation.
		(Inherited from <u>Stream</u> .)
≟	Equals(Object)	Determines whether the specified Object is equal to the current Object .
		(Inherited from Object.)
ġ ⊕	Finalize()	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection.
		(Inherited from Object.)
=	Flush()	Flush the stream.
		(Overrides Stream.Flush().)
	FlushMode	This property sets the flush behavion the stream. Sorry, though, not sure exactly how to describe all the various settings.

	GetHashCode()	Serves as a hash function for a particular type.
		(Inherited from Object.)
≡	GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance.
		(Inherited from MarshalByRefObject.)
≡	GetType()	Gets the <u>Type</u> of the current instance.
		(Inherited from Object.)
≡	InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance.
		(Inherited from MarshalByRefObject.)
	Length	Reading this property always throw a NotSupportedException.
		(Overrides <u>Stream.Length.</u>)
Ģ Û	MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object
		(Inherited from MarshalByRefObject.)
ē Û	MemberwiseClone()	Creates a shallow copy of the current Object.
		(Inherited from Object.)
	Position	The position of the stream pointer.
		(Overrides <u>Stream.Position</u> .)

≡	Read(Byte[], Int32, Int32)	Read data from the stream.
		(Overrides <u>Stream.Read(Byte[], Int32, Int32</u>).)
≓	ReadByte()	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. (Inherited from Stream .)
	ReadTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to read before timing out. (Inherited from Stream .)
=	Seek(Int64, SeekOrigin)	Calling this method always throws
	Sook(into i, SookSiigiii)	NotSupportedException. (Overrides Stream.Seek(Int64, SeekOrigin).)
≡	SetLength(Int64)	Calling this method always throws a NotSupportedException.
		(Overrides Stream .SetLength(Int64).)
=0	ToString()	Returns a <u>String</u> that represents the current <u>Object</u> .
		(Inherited from Object.)
	Totalln	Returns the total number of bytes input so far.
	TotalOut	Returns the total number of bytes output so far.

=\$S	UncompressBuffer(Byte[])	Uncompress a ZLIB-compressed byte array into a byte array.
= \$	UncompressString(Byte[])	Uncompress a ZLIB-compressed byte array into a single string.
≡	Write(Byte[], Int32, Int32)	Write data to the stream. (Overrides <u>Stream.Write(Byte[], Int32, Int32)</u> .)
Ξ ψ	WriteByte(Byte)	Writes a byte to the current positior in the stream and advances the position within the stream by one byte. (Inherited from <u>Stream</u> .)
	WriteTimeout	Gets or sets a value, in miliseconds that determines how long the stream will attempt to write before timing out. (Inherited from Stream .)

■ Remarks

The ZlibStream is a <u>Decorator</u> on a <u>Stream</u>. It adds ZLIB compression c decompression to any stream.

Using this stream, applications can compress or decompress data via stream Read() and Write() operations. Either compression or decompression can occur through either reading or writing. The compression format used is ZLIB, which is documented in IETF RFC
1950, "ZLIB Compressed Data Format Specification version 3.3". This

implementation of ZLIB always uses DEFLATE as the compression method. (see IETF RFC 1951, "DEFLATE Compressed Data Format Specification version 1.3.")

The ZLIB format allows for varying compression methods, window sizes and dictionaries. This implementation always uses the DEFLATE compression method, a preset dictionary, and 15 window bits by default

This class is similar to <u>DeflateStream</u>, except that it adds the RFC1950 header and trailer bytes to a compressed stream when compressing, or expects the RFC1950 header and trailer bytes when decompressing. It also similar to the <u>GZipStream</u>.

■ Inheritance Hierarchy

Object

<u>MarshalByRefObject</u>

- Stream

ZlibStream

See Also

<u>DeflateStream</u> <u>GZipStream</u>

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZlibStream Constructor

Reference ► Ionic.Zlib ► ZlibStream ► ZlibStream()

C#

■ Members

Icon	Member	Description
∄	ZlibStream(Stream, CompressionMode)	Create a ZlibStream using the specified CompressionMode.
≘ ₩	ZlibStream(Stream, CompressionMode, CompressionLevel)	Create a ZlibStream using the specified CompressionMode and the specified CompressionLevel
≅ ₩	ZlibStream(Stream, CompressionMode, Boolean)	Create a ZlibStream using the specified CompressionMode, and explicitly specify whether the captive stream should be left open after Deflation or Inflation.
⊕	ZlibStream(Stream, CompressionMode, CompressionLevel, Boolean)	Create a ZlibStream using the specified CompressionMode and the specified CompressionLevel and explicitly specify whether the stream should be left open after Deflation or Inflation.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 ZlibStream Constructor (stream,

mode)

Reference ► <u>lonic.Zlib</u> ► <u>ZlibStream</u> ► **ZlibStream**(Stream, CompressionMode)

Create a ZlibStream using the specified CompressionMode.

Declaration Syntax

Parameters

stream (Stream)

The stream which will be read or written.

mode (CompressionMode)

Indicates whether the ZlibStream will compress or decompress.

■ Remarks

When mode is CompressionMode.Compress, the ZlibStrea

will use the default compression level. The "captive" stream will be close when the ZlibStream is closed.

Examples

This example uses a ZlibStream to compress a file, and writes the compressed data to another file.

VB.NET □ Cop

```
Using input As Stream = File.OpenRead(fileToCompress
    Using raw As FileStream = File.Create(fileToComp
Using compressor As Stream = New ZlibStream(raw,
    Dim buffer As Byte() = New Byte(4096) {}
    Dim n As Integer = -1
    Do While (n <> 0)
        If (n > 0) Then
            compressor.Write(buffer, 0, n)
        End If
        n = input.Read(buffer, 0, buffer.Length)
        Loop
End Using
```

End Using End Using

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

1

Ionic Zip Library v1.9.1.6 ZlibStream Constructor (stream,

mode, level)

Reference ► Ionic.Zlib ► ZlibStream ► ZlibStream(Stream,

CompressionMode, CompressionLevel)

Create a ZlibStream using the specified CompressionMode an the specified CompressionLevel.

Declaration Syntax

```
C#
                   Visual Basic
                                      Visual C++
public ZlibStream(
        Stream stream,
         CompressionMode mode,
        CompressionLevel level
```

```
Public Sub New ( _
        stream As Stream, _
        mode As CompressionMode, _
        level As CompressionLevel _
```

```
public:
ZlibStream(
        Stream^ stream,
        CompressionMode mode,
        CompressionLevel level
```

Parameters

stream (Stream)

The stream to be read or written while deflating or inflating.

mode (CompressionMode)

Indicates whether the ZlibStream will compress or decompress.

level (CompressionLevel)

A tuning knob to trade speed for effectiveness.

■ Remarks

When mode is CompressionMode. Decompress, the level parameter is ignored. The "captive" stream will be closed when the ZlibStream is closed.

Examples

This example uses a **ZlibStream** to compress data from a file, and writes the compressed data to another file.

```
C#
                                                 Cop
using (System.IO.Stream input = System.IO.File.OpenR
{
    using (var raw = System.IO.File.Create(fileToCom
    {
         using (Stream compressor = new ZlibStream(ra
                                                     Co
                                                     Co
         {
             byte[] buffer = new byte[WORKING_BUFFER_
             int n:
             while ((n= input.Read(buffer, 0, buffer.
             {
                 compressor.Write(buffer, 0, n);
             }
        }
    }
}
```

VB.NET
□ Cop

```
Dim n As Integer = -1
Do While (n <> 0)
If (n > 0) Then
compressor.Write(buffer, 0, n)
End If
n = input.Read(buffer, 0, buffer.Len
Loop
End Using
End Using
End Using
End Using
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

•

Ionic Zip Library v1.9.1.6 ZlibStream Constructor (stream, mode, level, leaveOpen)

Reference ► <u>Ionic.Zlib</u> ► <u>ZlibStream</u> ► **ZlibStream**(Stream, CompressionMode, CompressionLevel, Boolean)

Create a ZlibStream using the specified CompressionMode an the specified CompressionLevel, and explicitly specify whether th stream should be left open after Deflation or Inflation.

Declaration Syntax

```
Visual Basic
                                    Visual C++
C#
public ZlibStream(
        Stream stream,
        CompressionMode mode,
        CompressionLevel level,
        bool leaveOpen
Public Sub New ( _
        stream As Stream,
        mode As CompressionMode,
        level As CompressionLevel,
        leaveOpen As Boolean _
public:
ZlibStream(
        Stream^ stream,
        CompressionMode mode,
        CompressionLevel level,
        bool leaveOpen
```

Parameters

stream (Stream)

The stream which will be read or written.

mode (CompressionMode)

Indicates whether the ZlibStream will compress or decompress.

level (CompressionLevel)

A tuning knob to trade speed for effectiveness. This parameter is effective only when mode is CompressionMode.Compress.

leaveOpen (Boolean)

true if the application would like the stream to remain open after inflation/deflation.

■ Remarks

This constructor allows the application to request that the captive strear remain open after the deflation or inflation occurs. By default, after Close() is called on the stream, the captive stream is also closed. In some cases this is not desired, for example if the stream is a MemoryStream that will be re-read after compression. Specify true for the leaveOpen parameter to leave the stream open.

When mode is CompressionMode. Decompress, the level parameter is ignored.

Examples

This example shows how to use a ZlibStream to compress the data fror a file, and store the result into another file. The filestream remains open to allow additional data to be written to it.

```
using (var output = System.IO.File.Create(fileToComp
{
   using (System.IO.Stream input = System.IO.File.0
   {
      using (Stream compressor = new ZlibStream(ou
      {
            byte[] buffer = new byte[WORKING_BUFFER_
```

VB.NET

```
Using output As FileStream = File.Create(fileToComprousing input As Stream = File.OpenRead(fileToComprousing compressor As Stream = New ZlibStream(Dim buffer As Byte() = New Byte(4096) {}
Dim n As Integer = -1
Do While (n <> 0)
If (n > 0) Then
compressor.Write(buffer, 0, n)
End If
n = input.Read(buffer, 0, buffer.Len Loop
End Using
I can write additional data to the output stream
End Using
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 ZlibStream Constructor (stream, mode, leaveOpen)

Reference ► <u>Ionic.Zlib</u> ► <u>ZlibStream</u> ► **ZlibStream**(Stream,

CompressionMode, Boolean)

Create a ZlibStream using the specified CompressionMode, ar explicitly specify whether the captive stream should be left open after Deflation or Inflation.

Declaration Syntax

```
C# Visual Basic Visual C++

public ZlibStream(
    Stream stream,
    CompressionMode mode,
    bool leaveOpen
)

Public Sub New ( _
    stream As Stream, _
    mode As CompressionMode, _
    leaveOpen As Boolean _
)
```

Parameters

stream (Stream)

The stream which will be read or written. This is called the "captive' stream in other places in this documentation.

mode (CompressionMode)

Indicates whether the ZlibStream will compress or decompress.

leaveOpen (Boolean)

true if the application would like the stream to remain open after inflation/deflation.

■ Remarks

When mode is CompressionMode.Compress, the ZlibStrea will use the default compression level.

This constructor allows the application to request that the captive strear remain open after the deflation or inflation occurs. By default, after Close() is called on the stream, the captive stream is also closed. In some cases this is not desired, for example if the stream is a MemoryStream that will be re-read after compression. Specify true for the leaveOpen parameter to leave the stream open.

See the other overloads of this constructor for example code.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 BufferSize Property Reference ► Ionic.Zlib ► ZlibStream ► BufferSize C#

The size of the working buffer for the compression codec.

Declaration Syntax

Remarks

The working buffer is used for all stream operations. The default size is 1024 bytes. The minimum size is 128 bytes. You may get better performance with a larger buffer. Then again, you might not. You would have to test it.

Set this before the first call to Read() or Write() on the stream. If you try to set it afterwards, it will throw.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 CanRead Property

Reference ► Ionic.Zlib ► ZlibStream ► CanRead

C# _

Indicates whether the stream can be read.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override bool CanRead { get; }

Public Overrides ReadOnly Property CanRead As Boolea Get

public:

virtual property bool CanRead {
    bool get () override;
}
```

Remarks

The return value depends on whether the captive stream supports reading.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Can Seek Property

Reference ► Ionic.Zlib ► ZlibStream ► CanSeek

C#

Indicates whether the stream supports Seek operations.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanSeek { get; }

Public Overrides ReadOnly Property CanSeek As Boolea Get

public:
virtual property bool CanSeek {
    bool get () override;
}
```

■ Remarks

Always returns false.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 CanWrite Property

Reference ► Ionic.Zlib ► ZlibStream ► CanWrite

C#

Indicates whether the stream can be written.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public override bool CanWrite { get; }

Public Overrides ReadOnly Property CanWrite As Boole Get

public:
    virtual property bool CanWrite {
        bool get () override;
}
```

Remarks

The return value depends on whether the captive stream supports writing.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compress Buffer Method (b)

Reference ► Ionic.Zlib ► ZlibStream ► CompressBuffer(Byte[])

Compress a byte array into a new byte array using ZLIB.

Declaration Syntax

Parameters

b (**Byte**[])

A buffer to compress.

■ Return Value

The data in compressed form

■ Remarks

Uncompress it with UncompressBuffer(Byte]).

■ See Also

CompressString(String)
UncompressBuffer(Byte[])

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Compress String Method (s)

Reference ► Ionic.Zlib ► ZlibStream ► CompressString(String)

C#

Compress a string into a byte array using ZLIB.

Declaration Syntax

Parameters

s (String)

A string to compress. The string will first be encoded using UTF8, then compressed.

■ Return Value

The string in compressed form

■ Remarks

Uncompress it with UncompressString(Byte]).

See Also

UncompressString(Byte[])
CompressBuffer(Byte[])

CompressString(String)

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Dispose Method (disposing)

Reference ► Ionic.Zlib ► ZlibStream ► Dispose(Boolean)

Dispose the stream.

Declaration Syntax

C#

Parameters

disposing (Boolean)

indicates whether the Dispose method was invoked by user code.

■ Remarks

This may or may not result in a Close() call on the captive stream. See the constructors that have a leaveOpen parameter for more information.

This method may be invoked in two distinct scenarios. If disposing == true, the method has been called directly or indirectly by a user's code, for example via the public Dispose() method. In this case, both manage and unmanaged resources can be referenced and disposed. If disposin == false, the method has been called by the runtime from inside the

object finalizer and this method should not reference other objects; in the case only unmanaged resources must be referenced or disposed.

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Flush Method

Reference ► Ionic.Zlib ► ZlibStream ► Flush()

C#

Flush the stream.

■ Declaration Syntax

C# Visual Basic Visual C++
public override void Flush()

Public Overrides Sub Flush

public: virtual void Flush() override

Send comments on this topic to $\underline{\text{dpchiesa@hotmail.com}}$

Copyright 2009-2011, Dino Chiesa

This property sets the flush behavior on the stream. Sorry, though, not sure exactly how to describe all the various settings.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public virtual FlushType FlushMode { get; set; }

Public Overridable Property FlushMode As FlushType
    Get
    Set

public:
virtual property FlushType FlushMode {
    FlushType get ();
    void set (FlushType value);
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Length Property

Reference ► Ionic.Zlib ► ZlibStream ► Length
C#
```

Reading this property always throws a NotSupportedException.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Length { get; }

Public Overrides ReadOnly Property Length As Long Get

public:
virtual property long long Length {
    long long get () override;
}
```

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Position Property Reference ► Ionic.Zlib ► ZlibStream ► Position C#

The position of the stream pointer.

■ Declaration Syntax

```
C# Visual Basic Visual C++

public override long Position { get; set; }

Public Overrides Property Position As Long
    Get
    Set

public:

virtual property long long Position {
    long long get () override;
    void set (long long value) override;
}
```

Remarks

Setting this property always throws a <u>NotSupportedException</u>. Reading will return the total bytes written out, if used in writing, or the total bytes read in, if used in reading. The count may refer to compressed bytes or uncompressed bytes, depending on how you've used the stream.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa
Assembly: Ionic.Zip (Module: Ionic.Zip) Version: 1.9.1.8 (1.9.1.8)

Ionic Zip Library v1.9.1.6 Read Method (buffer, offset, count)

C#

Reference ► Ionic.Zlib ► ZlibStream ► Read(Byte[], Int32, Int32)

Read data from the stream.

Declaration Syntax

```
C# Visual Basic Visual C++

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

Parameters

buffer (Byte∏)

The buffer into which the read data should be placed.

offset (Int32)

the offset within that data array to put the first byte read.

count (Int32)

the number of bytes to read.

Return Value

the number of bytes read

■ Remarks

If you wish to use the ZlibStream to compress data while reading, you can create a ZlibStream with CompressionMode.Compress, providing an uncompressed data stream. Then call Read() on that ZlibStream, and the data read will be compressed. If you wish to use the ZlibStream to decompredata while reading, you can create a ZlibStream with CompressionMode.Decompress, providing a readable compressed data stream. Then call Read() on that ZlibStream, and the data will be decompressed as it is read.

A ZlibStream can be used for Read() or Write(), but not both

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Seek Method (offset, origin)

Reference ► Ionic.Zlib ► ZlibStream ► Seek(Int64, SeekOrigin)

C#

Calling this method always throws a NotSupportedException.

Declaration Syntax

```
C# Visual Basic Visual C++

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

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

Parameters

offset (Int64)

The offset to seek to.... IF THIS METHOD ACTUALLY DID ANYTHING.

origin (SeekOrigin)

The reference specifying how to apply the offset.... IF THIS METHOD ACTUALLY DID ANYTHING.

■ Return Value

nothing. This method always throws.

Send comments on this topic to dpchiesa@hotmail.com
Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 SetLength Method (value)

Reference ► Ionic.Zlib ► ZlibStream ► SetLength(Int64)

C#

Calling this method always throws a NotSupportedException.

■ Declaration Syntax

Parameters

value (<u>Int64</u>)

The new value for the stream length.... IF THIS METHOD ACTUALLY DID ANYTHING.

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

```
Ionic Zip Library v1.9.1.6
Totalin Property

Reference ► Ionic.Zlib ► ZlibStream ► Totalin
C#
```

Returns the total number of bytes input so far.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public virtual long TotalIn { get; }

Public Overridable ReadOnly Property TotalIn As Long
    Get

public:
virtual property long long TotalIn {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 TotalOut Property Reference ► Ionic.Zlib ► ZlibStream ► TotalOut C#

Returns the total number of bytes output so far.

■ Declaration Syntax

```
C# Visual Basic Visual C++
public virtual long TotalOut { get; }

Public Overridable ReadOnly Property TotalOut As Lon Get

public:
virtual property long long TotalOut {
    long long get ();
}
```

Send comments on this topic to dpchiesa@hotmail.com

Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Uncompress Buffer Method

(compressed)

Reference ► Ionic.Zlib ► ZlibStream ► UncompressBuffer(Byte[])

Uncompress a ZLIB-compressed byte array into a byte array.

Declaration Syntax

```
Visual Basic
                                        Visual C++
C#
public static byte[] UncompressBuffer(
         byte[] compressed
Public Shared Function UncompressBuffer ( _
         compressed As Byte() _
) As <a href="Byte">Byte</a>()
public:
static array<<u>unsigned char</u>>^ UncompressBuffer(
         array<unsigned char>^ compressed
```

Parameters

compressed (Byte∏)

A buffer containing ZLIB-compressed data.

Return Value

The data in uncompressed form

See Also

CompressBuffer(Byte[]) UncompressString(Byte[])

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Uncompress String Method

(compressed)

Reference ► Ionic.Zlib ► ZlibStream ► UncompressString(Byte[])

Uncompress a ZLIB-compressed byte array into a single string.

Declaration Syntax

Parameters

compressed (Byte[])

A buffer containing ZLIB-compressed data.

■ Return Value

The uncompressed string

See Also

CompressString(String)
UncompressBuffer(Byte[])

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa

Ionic Zip Library v1.9.1.6 Write Method (buffer, offset, count)

C#

Reference ► Ionic.Zlib ► ZlibStream ► Write(Byte[], Int32, Int32)

Write data to the stream.

Declaration Syntax

```
Public Overrides Sub Write ( _ buffer As Byte(), _ offset As Integer, _ count As Integer _ )
```

Parameters

buffer (Byte[])

The buffer holding data to write to the stream.

offset (Int32)

the offset within that data array to find the first byte to write.

count (Int32)

the number of bytes to write.

Remarks

If you wish to use the ZlibStream to compress data while writing, you can create a ZlibStream with CompressionMode. Compress and a writable output stream. Then call Write() on that

ZlibStream, providing uncompressed data as input. The data sent t the output stream will be the compressed form of the data written. If you wish to use the ZlibStream to decompress data while writing, you can create a ZlibStream with

CompressionMode.Decompress, and a writable output stream. Then call Write() on that stream, providing previously compressed data. The data sent to the output stream will be the decompressed form of the data written.

A ZlibStream can be used for Read() or Write(), but not both

Send comments on this topic to dpchiesa@hotmail.com Copyright 2009-2011, Dino Chiesa