

# Zipios++ Reference Manual

Generated by Doxygen 1.1.4

Sun Jul 9 00:55:44 2000



# Contents

<b>1</b>	<b>Zipios++</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Status . . . . .	1
1.3	Documentation . . . . .	1
1.4	Download . . . . .	2
1.5	Links . . . . .	2
1.6	Bugs . . . . .	3
<b>2</b>	<b>Zipios++ Hierarchical Index</b>	<b>5</b>
2.1	Zipios++ Class Hierarchy . . . . .	5
<b>3</b>	<b>Zipios++ Compound Index</b>	<b>7</b>
3.1	Zipios++ Compound List . . . . .	7
<b>4</b>	<b>Zipios++ File Index</b>	<b>9</b>
4.1	Zipios++ File List . . . . .	9
<b>5</b>	<b>Zipios++ Class Documentation</b>	<b>11</b>
5.1	zipios::BackBuffer Class Reference . . . . .	11
5.2	zipios::BasicEntry Class Reference . . . . .	13
5.3	zipios::CollectionCollection Class Reference . . . . .	21
5.4	zipios::DataDescriptor Struct Reference . . . . .	27
5.5	zipios::DirectoryCollection Class Reference . . . . .	28
5.6	zipios::EndOfCentralDirectory Class Reference . . . . .	33
5.7	zipios::FCollException Class Reference . . . . .	34
5.8	zipios::FileCollection Class Reference . . . . .	35
5.9	zipios::FileEntry Class Reference . . . . .	40
5.10	zipios::FilePath Class Reference . . . . .	47
5.11	zipios::FilterInputSteambuf Class Reference . . . . .	51
5.12	zipios::InflateInputSteambuf Class Reference . . . . .	53

5.13	zipios::InvalidStateException Class Reference . . . . .	55
5.14	zipios::IOException Class Reference . . . . .	56
5.15	zipios::FileEntry::MatchFileName Class Reference . . . . .	57
5.16	zipios::FileEntry::MatchName Class Reference . . . . .	58
5.17	zipios::OutputStringStream Class Reference . . . . .	59
5.18	zipios::SimpleSmartPointer Class Reference . . . . .	60
5.19	zipios::VirtualSeeker Class Reference . . . . .	61
5.20	zipios::ZipCDirEntry Class Reference . . . . .	62
5.21	zipios::ZipFile Class Reference . . . . .	64
5.22	zipios::ZipInputStream Class Reference . . . . .	68
5.23	zipios::ZipInputSteambuf Class Reference . . . . .	70
5.24	zipios::ZipLocalEntry Class Reference . . . . .	72
<b>6</b>	<b>Zipios++ File Documentation</b>	<b>81</b>
6.1	appendzip.cpp File Reference . . . . .	81
6.2	backbuffer.h File Reference . . . . .	83
6.3	basicentry.cpp File Reference . . . . .	84
6.4	basicentry.h File Reference . . . . .	85
6.5	collcoll.cpp File Reference . . . . .	86
6.6	collcoll.h File Reference . . . . .	87
6.7	dircoll.cpp File Reference . . . . .	88
6.8	dircoll.h File Reference . . . . .	89
6.9	example_zip.cpp File Reference . . . . .	90
6.10	fcoll.cpp File Reference . . . . .	91
6.11	fcoll.h File Reference . . . . .	92
6.12	fcoll_common.h File Reference . . . . .	93
6.13	fcollexceptions.cpp File Reference . . . . .	94
6.14	fcollexceptions.h File Reference . . . . .	95
6.15	fileentry.cpp File Reference . . . . .	96
6.16	fileentry.h File Reference . . . . .	97
6.17	filepath.cpp File Reference . . . . .	98
6.18	filepath.h File Reference . . . . .	99
6.19	filterinputstreambuf.cpp File Reference . . . . .	100
6.20	filterinputstreambuf.h File Reference . . . . .	101
6.21	inflateinputstreambuf.cpp File Reference . . . . .	102
6.22	inflateinputstreambuf.h File Reference . . . . .	103
6.23	outputstringstream.h File Reference . . . . .	104

---

6.24	simplesmartptr.h File Reference . . . . .	105
6.25	test_appzip.cpp File Reference . . . . .	106
6.26	test_collcoll.cpp File Reference . . . . .	107
6.27	test_dircoll.cpp File Reference . . . . .	108
6.28	test_zip.cpp File Reference . . . . .	109
6.29	test_zipinputstream.cpp File Reference . . . . .	110
6.30	test_zipinputstreambuf.cpp File Reference . . . . .	111
6.31	virtualeseeker.h File Reference . . . . .	112
6.32	zipfile.cpp File Reference . . . . .	113
6.33	zipfile.h File Reference . . . . .	114
6.34	ziphead.cpp File Reference . . . . .	115
6.35	ziphead.h File Reference . . . . .	116
6.36	zipinputstream.cpp File Reference . . . . .	117
6.37	zipinputstream.h File Reference . . . . .	118
6.38	zipinputstreambuf.cpp File Reference . . . . .	119
6.39	zipinputstreambuf.h File Reference . . . . .	120



# Chapter 1

## Zipios++



### 1.1 Introduction

Zipios++ is a `java.util.zip`-like C++ library for reading Zip files. Access to individual entries is provided through standard C++ iostreams. A simple read-only virtual file system that mounts regular directories and zip files is also provided.

The source code is released under the [GNU Lesser General Public License](#).

### 1.2 Status

Writing is not supported yet

Spanned archives are not supported, and support is not planned.

The library has been tested and appears to be working with

- Linux Mandrake release 7.0 (Air) / gcc 2.95.2
- Red Hat Linux release 6.2 (Zoot) / egcs-2.91.66
- SGI IRIX64 6.5 / gcc 2.95.2
- SGI IRIX64 6.5 / MIPSpro Compilers: Version 7.30

If you are aware of any other platforms that Zipios++ works on, please let me know ([thomas@miba.auc.dk](mailto:thomas@miba.auc.dk)).

### 1.3 Documentation

This web page is the front page to the library documentation which is generated from the source files using [Doxygen](#). Use the links at the top of the page to browse the API documentation. The documentation is also available in a printer-friendly format [\[pdf\]](#).

### 1.3.1 Zip file access

The two most important classes are [ZipFile](#) and [ZipInputStream](#). [ZipInputStream](#) is an istream for reading zipfiles. It can be instantiated directly, without the use of [ZipFile](#). A new [ZipInputStream](#) reads from the first entry, and the user can skip to the next entry by calling [ZipInputStream::getNextEntry\(\)](#).

[ZipFile](#) scans the central directory of a zipfile and provides an interface to access that directory. The user may search for entries with a particular filename using [ZipFile::getEntry\(\)](#), or simply get the complete list of entries with [ZipFile::entries\(\)](#). To get an istream ([ZipInputStream](#)) to a particular entry simply use [ZipFile::getInputStream\(\)](#).

[example.zip.cpp](#) demonstrates the central elements of Zipios++.

A Zip file appended to another file, e.g. a binary program, with the program [appendzip](#), can be read with [ZipFile::openEmbeddedZipFile\(\)](#).

### 1.3.2 FileCollections

A [ZipFile](#) is actually just a special kind of [FileCollection](#) that obtains its entries from a .zip Zip archive. Zipios++ also implements a [DirectoryCollection](#) that obtains its entries from a specified directory, and a [CollectionCollection](#) that obtains its entries from other collections. Using a single [CollectionCollection](#) any number of other [FileCollections](#) can be placed under its control and accessed through the same single interface that is used to access a [ZipFile](#) or a [DirectoryCollection](#). A singleton (a unique global instance) [CollectionCollection](#) can be obtained through

```
CollectionCollection::inst\(\) ;
```

To save typing [CollectionCollection](#) has been typedef'ed to [CColl](#). In the initialization part of an application [FileCollections](#) can be created, and placed under [CColl::inst\(\)](#)'s control using

```
CColl::inst\(\).addCollection\(\)
```

and later an istream can be obtained using

```
CColl::inst\(\).getInputStream\(\).
```

## 1.4 Download

Go to [Zipios++ project page](#) on SourceForge for tar balls and ChangeLog. [http://sourceforge.net/project/?group\\_id=5418](http://sourceforge.net/project/?group_id=5418)

## 1.5 Links

[zlib](#). The compression library that Zipios++ uses to perform the actual decompression.

[Java 2 Platform, Standard Edition, v 1.3 API Specification](#) . Zipios++ is heavily inspired by the [java.util.zip](#) package.

[PKWARE zip format](#) . A description of the zip file format.



## 1.6 Bugs

Submit bug reports and patches to [thomas@miba.auc.dk](mailto:thomas@miba.auc.dk)



# Chapter 2

## Zipios++ Hierarchical Index

### 2.1 Zipios++ Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

zipios::DataDescriptor . . . . .	27
zipios::EndOfCentralDirectory . . . . .	33
zipios::exception	
zipios::FCollException . . . . .	34
zipios::InvalidStateException . . . . .	55
zipios::IOException . . . . .	56
zipios::FileCollection . . . . .	35
zipios::CollectionCollection . . . . .	21
zipios::DirectoryCollection . . . . .	28
zipios::ZipFile . . . . .	64
zipios::FileEntry . . . . .	40
zipios::BasicEntry . . . . .	13
zipios::ZipLocalEntry . . . . .	72
zipios::ZipCDirEntry . . . . .	62
zipios::FilePath . . . . .	47
zipios::istream	
zipios::ZipInputStream . . . . .	68
zipios::FileEntry::MatchFileName . . . . .	57
zipios::FileEntry::MatchName . . . . .	58
zipios::ostrstream	
zipios::OutputStringStream . . . . .	59
zipios::SimpleSmartPointer . . . . .	60
zipios::streambuf	
zipios::FilterInputStreambuf . . . . .	51
zipios::InflateInputStreambuf . . . . .	53
zipios::ZipInputStreambuf . . . . .	70
zipios::vector	
zipios::BackBuffer . . . . .	11
zipios::VirtualSeeker . . . . .	61



# Chapter 3

## Zipios++ Compound Index

### 3.1 Zipios++ Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

<b>zipios::BackBuffer</b> (A <a href="#">BackBuffer</a> instance is useful for reading the last part of a file in an efficient manner, when it is not known exactly how far back (towards the front!) to go, to find the start of the desired data block) . . . . .	11
<b>zipios::BasicEntry</b> ( <a href="#">BasicEntry</a> is a <a href="#">FileEntry</a> that is suitable as a base class for basic entries, that e.g) . . . . .	13
<b>zipios::CollectionCollection</b> ( <a href="#">CollectionCollection</a> is a <a href="#">FileCollection</a> that consists of an arbitrary number of <a href="#">FileCollections</a> ) . . . . .	21
<b>zipios::DataDescriptor</b> (A struct containing fields for the entries in a zip file data descriptor, that trails the compressed data in files that were created by streaming, ie) . . . . .	27
<b>zipios::DirectoryCollection</b> ( <a href="#">DirectoryCollection</a> is a <a href="#">FileCollection</a> that obtains its entries from a directory) . . . . .	28
<b>zipios::EndOfCentralDirectory</b> (The end of the Central directory structure) . . . . .	33
<b>zipios::FCollException</b> (An <a href="#">FCollException</a> is used to signal a problem with a <a href="#">FileCollection</a> ) . . . . .	34
<b>zipios::FileCollection</b> ( <a href="#">FileCollection</a> is an abstract baseclass that represents a collection of files)	35
<b>zipios::FileEntry</b> (A <a href="#">FileEntry</a> represents an entry in a <a href="#">FileCollection</a> ) . . . . .	40
<b>zipios::FilePath</b> ( <a href="#">FilePath</a> represents a path to a file or directory name) . . . . .	47
<b>zipios::FilterInputStreambuf</b> (An input streambuf filter is a streambuf that filters the input it gets from the streambuf it is attached to) . . . . .	51
<b>zipios::InflateInputStreambuf</b> ( <a href="#">InflateInputStreambuf</a> is an input stream filter, that inflates the input from the attached input stream) . . . . .	53
<b>zipios::InvalidStateException</b> (An object member function may throw this exception, if the operation it normally performs is inappropriate or impossible to perform because of the current state of the object) . . . . .	55
<b>zipios::IOException</b> (An <a href="#">IOException</a> is used to signal an I/O error) . . . . .	56
<b>zipios::FileEntry::MatchFileName</b> (Function object to be used with the STL <code>find_if</code> algorithm to find a <a href="#">FileEntry</a> in a container, which name (as obtained with <a href="#">FileEntry::getFileName()</a> ) is identical to the name specified in the <a href="#">MatchName</a> constructor) . . . . .	57
<b>zipios::FileEntry::MatchName</b> (Function object to be used with the STL <code>find_if</code> algorithm to find a <a href="#">FileEntry</a> in a container, which name (as obtained with <a href="#">FileEntry::getName()</a> ) is identical to the name specified in the <a href="#">MatchName</a> constructor) . . . . .	58
<b>zipios::OutputStringStream</b> ( <a href="#">OutputStringStream</a> is typedefed to <code>ostreamstream</code> if <code>sstream</code> is part of the standard library (unless Zipios++ has been explicitly configured not to use it))	59

---

<b>zipios::SimpleSmartPointer</b> ( <a href="#">SimpleSmartPointer</a> is a simple reference counting smart pointer template) . . . . .	60
<b>zipios::VirtualSeeker</b> ( <a href="#">VirtualSeeker</a> is a simple class that keeps track of a set of specified 'virtual' file endings that mark a subset of a real file) . . . . .	61
<b>zipios::ZipCDirEntry</b> (Specialization of <a href="#">ZipLocalEntry</a> , that add fields for storing the extra information, that is only present in the entries in the zip central directory and not in the local entry headers) . . . . .	62
<b>zipios::ZipFile</b> ( <a href="#">ZipFile</a> is a <a href="#">FileCollection</a> , where the files are stored in a .zip file) . . . . .	64
<b>zipios::ZipInputStream</b> ( <a href="#">ZipInputStream</a> is an istream that gets it's input from a zip file) . . . . .	68
<b>zipios::ZipInputStreambuf</b> ( <a href="#">ZipInputStreambuf</a> is a zip input streambuf filter) . . . . .	70
<b>zipios::ZipLocalEntry</b> (A concrete implementation of the abstract <a href="#">FileEntry</a> base class for <a href="#">ZipFile</a> entries, specifically for representing the information present in the local headers of file entries in a zip file) . . . . .	72

# Chapter 4

## Zipios++ File Index

### 4.1 Zipios++ File List

Here is a list of all documented files with brief descriptions:

<b>appendzip.cpp</b> (Source code to a small program appendzip that appends a zip archive to another file) . . . . .	81
<b>backbuffer.h</b> (The header file for BackBuffer) . . . . .	83
<b>basicentry.cpp</b> (Implementation of BasicEntry) . . . . .	84
<b>basicentry.h</b> (Header file that defines BasicEntry) . . . . .	85
<b>collcoll.cpp</b> (Implementation of CollectionCollection) . . . . .	86
<b>collcoll.h</b> (Header file that defines CollectionCollection) . . . . .	87
<b>dircoll.cpp</b> (Implementation of DirectoryCollection) . . . . .	88
<b>dircoll.h</b> (Header file that defines DirectoryCollection) . . . . .	89
<b>example_zip.cpp</b> (Source code to a small program that demonstrates the central elements of Zipios++) . . . . .	90
<b>fcoll.cpp</b> (Implementation of FileCollection) . . . . .	91
<b>fcoll.h</b> (Header file that defines FileCollection) . . . . .	92
<b>fcoll_common.h</b> (Header file that doesn't really contain much!) . . . . .	93
<b>fcoll_exceptions.cpp</b> (Implementation of a number of Exceptions used by FileCollection and its subclasses) . . . . .	94
<b>fcoll_exceptions.h</b> (Header file that defines a number of exceptions used by FileCollection and its subclasses) . . . . .	95
<b>fileentry.cpp</b> (Implementation of FileEntry) . . . . .	96
<b>fileentry.h</b> (Header file that defines FileEntry) . . . . .	97
<b>filepath.cpp</b> (Implementation of FilePath) . . . . .	98
<b>filepath.h</b> (Header file that defines FilePath) . . . . .	99
<b>filterinputstreambuf.cpp</b> (Implementation of FilterInputSteambuf) . . . . .	100
<b>filterinputstreambuf.h</b> (Header file that defines FilterInputSteambuf) . . . . .	101
<b>inflateinputstreambuf.cpp</b> (Implementation of InflateInputSteambuf) . . . . .	102
<b>inflateinputstreambuf.h</b> (Header file that defines InflateInputSteambuf) . . . . .	103
<b>outputstringstream.h</b> (Header file that defines OutputStringStream) . . . . .	104
<b>simplesmartptr.h</b> (Header file that defines SimpleSmartPointer) . . . . .	105
<b>test_appzip.cpp</b> (Source code to a small program that demonstrates the central elements of Zipios++) . . . . .	106
<b>test_collcoll.cpp</b> (Source code to a small program that demonstrates the central elements of Zipios++) . . . . .	107

---

<b>test_dircoll.cpp</b> (Source code to a small program that demonstrates the central elements of Zipios++) . . . . .	108
<b>test_zip.cpp</b> (Source code to a small program that tests the functionality of Zipios++) . . . . .	109
<b>test_zipinputstream.cpp</b> (Source for a test program for testing ZipInputStream) . . . . .	110
<b>test_zipinputstreambuf.cpp</b> (Source for a test program for testing ZipInputSteambuf) . . . . .	111
<b>virtalseeker.h</b> (Header file that defines VirtualSeeker) . . . . .	112
<b>zipfile.cpp</b> (The implementation of ZipFile) . . . . .	113
<b>zipfile.h</b> (Header file that defines ZipFile) . . . . .	114
<b>ziphead.cpp</b> (Implementation of routines for reading the central directory and local headers of a zip archive) . . . . .	115
<b>ziphead.h</b> (Header file containing classes and functions for reading the central directory and local header fields in a zip archive) . . . . .	116
<b>zipinputstream.cpp</b> (Implementation of ZipInputStream) . . . . .	117
<b>zipinputstream.h</b> (Header file that defines ZipInputStream) . . . . .	118
<b>zipinputstreambuf.cpp</b> (Implementation of ZipInputSteambuf) . . . . .	119
<b>zipinputstreambuf.h</b> (Header file that defines ZipInputSteambuf) . . . . .	120



# Chapter 5

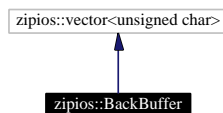
## Zipios++ Class Documentation

### 5.1 zipios::BackBuffer Class Reference

A [BackBuffer](#) instance is useful for reading the last part of a file in an efficient manner, when it is not known exactly how far back (towards the front!) to go, to find the start of the desired data block.

```
#include <backbuffer.h>
```

Inheritance diagram for zipios::BackBuffer



#### Public Methods

- [BackBuffer](#) ( istream &is, [VirtualSeeker](#) vs = [VirtualSeeker](#)(), int chunk\_size = 1024 )  
*BackBuffer constructor.*
- int [readChunk](#) ( int &read\_pointer )  
*Reads another chunk and returns the size of the chunk that has been read.*

#### 5.1.1 Detailed Description

A [BackBuffer](#) instance is useful for reading the last part of a file in an efficient manner, when it is not known exactly how far back (towards the front!) to go, to find the start of the desired data block.

[BackBuffer](#) is a `vector< unsigned char >` that fills itself with data from a file by reading chunks from the end of the file progressing towards the start. Upon construction the [BackBuffer](#) instance is associated with a file and a chunksize can be specified. To read a chunk of the file into the [BackBuffer](#) call [readChunk](#)() .

Definition at line 31 of file backbuffer.h.

## 5.1.2 Constructor & Destructor Documentation

### 5.1.2.1 `zipios::BackBuffer::BackBuffer (istream & is, VirtualSeeker vs = VirtualSeeker(), int chunk_size = 1024) [inline, explicit]`

[BackBuffer](#) constructor.

**Parameters:**

*is* The istream to read the data from. The stream must be seekable, as [BackBuffer](#) will reposition the file position to read chunks from the back of the file.

*chunk\_size* specifies the size of the chunks to read the file into the [BackBuffer](#) in.

**Exceptions:**

*FCollException* Thrown if the [VirtualSeeker](#) vs that has been specified is invalid for the istream is.

Definition at line 60 of file [backbuffer.h](#).

## 5.1.3 Member Function Documentation

### 5.1.3.1 `int zipios::BackBuffer::readChunk (int & read_pointer) [inline]`

Reads another chunk and returns the size of the chunk that has been read.

Returns 0 on I/O failure.

**Parameters:**

*read\_pointer* When a new chunk is read in the already stored bytes change position in the [BackBuffer](#). *read\_pointer* is assumed by [readChunk\(\)](#) to be a pointer into a position in the [BackBuffer](#), and is updated to point to the same position in the file as it pointed to before the new chunk was read.

Definition at line 74 of file [backbuffer.h](#).

The documentation for this class was generated from the following file:

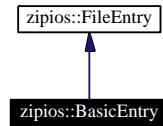
- [backbuffer.h](#)

## 5.2 zipios::BasicEntry Class Reference

[BasicEntry](#) is a [FileEntry](#) that is suitable as a base class for basic entries, that e.g.

```
#include <basicentry.h>
```

Inheritance diagram for zipios::BasicEntry



### Public Methods

- [BasicEntry](#) ( const string &filename, const string &comment, const [FilePath](#) &basepath = string() )  
*Constructor.*
- virtual string [getComment](#) () const  
*Returns the comment of the entry, if it has one.*
- virtual int [getCompressedSize](#) () const  
*Returns the compressed size of the entry.*
- virtual int [getCrc](#) () const  
*Returns the Crc for the entry, if it has one.*
- virtual vector< unsigned char > [getExtra](#) () const  
*Returns a vector of bytes of extra data that may be stored with the entry.*
- virtual StorageMethod [getMethod](#) () const  
*Returns the method used to store the entry in the [FileCollection](#).*
- virtual string [getName](#) () const  
*Returns the full filename of the entry, including a path if the entry is stored in a subfolder.*
- virtual string [getFileName](#) () const  
*Returns the filename of the entry.*
- virtual int [getSize](#) () const  
*Returns the (uncompressed) size of the entry data.*
- virtual int [getTime](#) () const  
*Returns the date and time of FIXME: what?*
- virtual bool [isValid](#) () const

Any method or operator that initializes a [FileEntry](#) may set a flag, that specifies whether the read entry is valid or not.

- virtual bool [isDirectory](#) () const  
*Returns true if the entry is a directory.*
- virtual void [setComment](#) ( const string &comment )  
*Sets the comment field for the [FileEntry](#).*
- virtual void [setCompressedSize](#) ( int size )  
*Set the compressed size field of the entry.*
- virtual void [setCrc](#) ( int crc )  
*Sets the crc field.*
- virtual void [setExtra](#) ( const vector< unsigned char > &extra )  
*Sets the extra field.*
- virtual void [setMethod](#) ( StorageMethod method )  
*Sets the storage method field for the entry.*
- virtual void [setName](#) ( const string &name )  
*Sets the name field for the entry.*
- virtual void [setSize](#) ( int size )  
*Sets the size field for the entry.*
- virtual void [setTime](#) ( int time )  
*Sets the time field for the entry.*
- virtual string [toString](#) () const  
*Returns a human-readable string representation of the entry.*
- virtual BasicEntry\* [clone](#) () const  
*Create a heap allocated clone of the object this method is called for.*
- virtual ~**BasicEntry** ()

## Protected Attributes

- string **\_filename**
- string **\_comment**
- int **\_size**
- bool **\_valid**
- [FilePath](#) **\_basepath**

## 5.2.1 Detailed Description

[BasicEntry](#) is a [FileEntry](#) that is suitable as a base class for basic entries, that e.g.

do not support any form of compression

Definition at line 17 of file basicentry.h.

## 5.2.2 Constructor & Destructor Documentation

### 5.2.2.1 zipios::BasicEntry::BasicEntry (const string & filename, const string & comment, const FilePath & basepath = string()) [explicit]

Constructor.

#### Parameters:

*filename* the filename of the entry.

*comment* a comment for the entry.

Definition at line 23 of file basicentry.cpp.

## 5.2.3 Member Function Documentation

### 5.2.3.1 BasicEntry \* zipios::BasicEntry::clone () const [virtual]

Create a heap allocated clone of the object this method is called for.

The caller is responsible for deallocating the clone when he is done with it.

#### Returns:

A heap allocated copy of the object this method is called for.

Reimplemented from [zipios::FileEntry](#).

Definition at line 131 of file basicentry.cpp.

### 5.2.3.2 string zipios::BasicEntry::getComment () const [virtual]

Returns the comment of the entry, if it has one.

Otherwise it returns an empty string.

#### Returns:

the comment associated with the entry, if there is one.

Reimplemented from [zipios::FileEntry](#).

Definition at line 41 of file basicentry.cpp.

### 5.2.3.3 `int zipios::BasicEntry::getCompressedSize () const` [virtual]

Returns the compressed size of the entry.

If the entry is not stored in a compressed format, the uncompressed size is returned.

**Returns:**

the compressed size of the entry. If the entry is stored without compression the uncompressed size is returned.

Reimplemented from [zipios::FileEntry](#).

Definition at line 45 of file basicentry.cpp.

### 5.2.3.4 `int zipios::BasicEntry::getCrc () const` [virtual]

Returns the Crc for the entry, if it has one.

FIXME: what is returned if it doesn't have one?

**Returns:**

the Crc for the entry, if it has one.

Reimplemented from [zipios::FileEntry](#).

Definition at line 49 of file basicentry.cpp.

### 5.2.3.5 `vector<unsigned char> zipios::BasicEntry::getExtra () const` [virtual]

Returns a vector of bytes of extra data that may be stored with the entry.

**Returns:**

A vector< unsigned char > of extra bytes that may potentially be associated with an entry.

Reimplemented from [zipios::FileEntry](#).

Definition at line 53 of file basicentry.cpp.

### 5.2.3.6 `string zipios::BasicEntry::getFileName () const` [virtual]

Returns the filename of the entry.

**Returns:**

Returns the filename of the entry.

Reimplemented from [zipios::FileEntry](#).

Definition at line 65 of file basicentry.cpp.

### 5.2.3.7 `StorageMethod zipios::BasicEntry::getMethod () const` [virtual]

Returns the method used to store the entry in the [FileCollection](#).

**Returns:**

the storage method used to store the entry in the collection.

**See also:**

StorageMethod.

Reimplemented from [zipios::FileEntry](#).

Definition at line 57 of file basicentry.cpp.

**5.2.3.8 string zipios::BasicEntry::getName () const [virtual]**

Returns the full filename of the entry, including a path if the entry is stored in a subfolder.

**Returns:**

the filename of the entry, including path if the entry is stored in a sub-folder.

Reimplemented from [zipios::FileEntry](#).

Definition at line 61 of file basicentry.cpp.

**5.2.3.9 int zipios::BasicEntry::getSize () const [virtual]**

Returns the (uncompressed) size of the entry data.

**Returns:**

Returns the (uncompressed) size of the entry.

Reimplemented from [zipios::FileEntry](#).

Definition at line 78 of file basicentry.cpp.

**5.2.3.10 int zipios::BasicEntry::getTime () const [virtual]**

Returns the date and time of FIXME: what?

**Returns:**

the date and time of the entry.

Reimplemented from [zipios::FileEntry](#).

Definition at line 82 of file basicentry.cpp.

**5.2.3.11 bool zipios::BasicEntry::isDirectory () const [virtual]**

Returns true if the entry is a directory.

A directory entry is an entry which name ends with a separator ('/' for Unix systems, '\' for Windows and DOS systems).

**Returns:**

true if the entry is a directory.

Reimplemented from [zipios::FileEntry](#).

Definition at line 91 of file basicentry.cpp.

### 5.2.3.12 `bool zipios::BasicEntry::isValid () const` [virtual]

Any method or operator that initializes a [FileEntry](#) may set a flag, that specifies whether the read entry is valid or not.

If it isn't this method returns false.

**Returns:**

true if the [FileEntry](#) has been parsed successfully.

Reimplemented from [zipios::FileEntry](#).

Definition at line 86 of file basicentry.cpp.

### 5.2.3.13 `void zipios::BasicEntry::setComment (const string & comment)` [virtual]

Sets the comment field for the [FileEntry](#).

**Parameters:**

*comment* string with the new comment.

Reimplemented from [zipios::FileEntry](#).

Definition at line 97 of file basicentry.cpp.

### 5.2.3.14 `void zipios::BasicEntry::setCompressedSize (int size)` [virtual]

Set the compressed size field of the entry.

**Parameters:**

*size* value to set the compressed size field of the entry to.

Reimplemented from [zipios::FileEntry](#).

Definition at line 101 of file basicentry.cpp.

### 5.2.3.15 `void zipios::BasicEntry::setCrc (int crc)` [virtual]

Sets the crc field.

**Parameters:**

*crc* value to set the crc field to.

Reimplemented from [zipios::FileEntry](#).

Definition at line 104 of file basicentry.cpp.

### 5.2.3.16 `void zipios::BasicEntry::setExtra (const vector<unsigned char>& extra)` [virtual]

Sets the extra field.

**Parameters:**

*extra* the extra field is set to this value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 107 of file basicentry.cpp.



**5.2.3.17 void zipios::BasicEntry::setMethod (StorageMethod *method*)** [virtual]

Sets the storage method field for the entry.

**Parameters:**

*method* the method field is set to the specified value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 110 of file basicentry.cpp.

**5.2.3.18 void zipios::BasicEntry::setName (const string & *name*)** [virtual]

Sets the name field for the entry.

**Parameters:**

*name* the name field is set to the specified value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 113 of file basicentry.cpp.

**5.2.3.19 void zipios::BasicEntry::setSize (int *size*)** [virtual]

Sets the size field for the entry.

**Parameters:**

*size* the size field is set to this value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 117 of file basicentry.cpp.

**5.2.3.20 void zipios::BasicEntry::setTime (int *time*)** [virtual]

Sets the time field for the entry.

**Parameters:**

*time* the time field is set to the specified value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 121 of file basicentry.cpp.

**5.2.3.21 string zipios::BasicEntry::toString () const** [virtual]

Returns a human-readable string representation of the entry.

**Returns:**

a human-readable string representation of the entry.

Reimplemented from [zipios::FileEntry](#).

Definition at line 125 of file basicentry.cpp.

The documentation for this class was generated from the following files:

- [basicentry.h](#)
- [basicentry.cpp](#)

## 5.3 zipios::CollectionCollection Class Reference

[CollectionCollection](#) is a [FileCollection](#) that consists of an arbitrary number of [FileCollections](#).

```
#include <collcoll.h>
```

Inheritance diagram for zipios::CollectionCollection



### Public Methods

- [CollectionCollection](#) ()  
*Constructor.*
- [CollectionCollection](#) ( const [CollectionCollection](#) &src )  
*Copy constructor.*
- const [CollectionCollection](#)& [operator=](#) ( const [CollectionCollection](#) &src )  
*Copy assignment operator.*
- bool [addCollection](#) ( const [FileCollection](#) &collection )  
*Adds a collection.*
- bool [addCollection](#) ( [FileCollection](#) \*collection )  
*Adds the collection pointed to by collection.*
- virtual void [close](#) ()  
*Closes the [FileCollection](#).*
- virtual vector< [ConstEntryPointer](#) > [entries](#) () const  
*Returns a vector of const pointers to the entries in the [FileCollection](#).*
- virtual [ConstEntryPointer](#) [getEntry](#) ( const string &name, [MatchPath](#) matchpath = [MATCH](#) ) const  
*Returns a [ConstEntryPointer](#) to a [FileEntry](#) object for the entry with the specified name.*
- virtual [istream](#)\* [getInputStream](#) ( const [ConstEntryPointer](#) &entry )  
*Returns a pointer to an opened istream for the specified [FileEntry](#).*
- virtual [istream](#)\* [getInputStream](#) ( const string &entry\_name, [MatchPath](#) matchpath = [MATCH](#) )  
*Returns a pointer to an opened istream for the specified entry name.*
- virtual int [size](#) () const  
*Returns the number in entries in all collections kept by the [CollectionCollection](#) object.*

- virtual `CollectionCollection* clone () const`  
*Create a heap allocated clone of the object this method is called for.*
- virtual `~CollectionCollection ()`

## Static Public Methods

- `CollectionCollection& inst ()`  
*This static method provides a singleton instance of a `CollectionCollection`.*

## Protected Methods

- void `getEntry ( const string &name, ConstEntryPoint &cep, vector< FileCollection * >::const_iterator &it, MatchPath matchpath = MATCH ) const`  
*A protected `getEntry` member function, that not only finds an entry that match the name, if such an entry exists in the collection, it also returns, which collection it was found in.*

## Protected Attributes

- `vector< FileCollection * > _collections`

### 5.3.1 Detailed Description

`CollectionCollection` is a `FileCollection` that consists of an arbitrary number of `FileCollections`.

With a `CollectionCollection` the user can use multiple `FileCollections` transparently, making it easy for a program to keep some of its files in a zip archive and others stored in ordinary files. `CollectionCollection` can be used to create a simple virtual filesystem, where all collections are mounted in `/`. If more than one collection contain a file with the same path only the one in the first added collection will be accessible.

Definition at line 26 of file `collcoll.h`.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 `zipios::CollectionCollection::CollectionCollection () [explicit]`

Constructor.

Definition at line 16 of file `collcoll.cpp`.

#### 5.3.2.2 `zipios::CollectionCollection::CollectionCollection (const CollectionCollection & src) [inline]`

Copy constructor.

Definition at line 118 of file `collcoll.h`.

### 5.3.3 Member Function Documentation

#### 5.3.3.1 `bool zipios::CollectionCollection::addCollection (FileCollection * collection)`

Adds the collection pointed to by collection.

The `CollectionCollection` will call delete on the pointer when it is destructed, so the caller should make absolutely sure to only pass in a collection created with new and be sure to leave it alone after adding it. If the collection is not added false is returned and the caller remains responsible for the collection pointed to by collection.

**Parameters:**

*collection* A pointer to the collection to add.

**Returns:**

true if the collection was added succesfully and the added collection is valid.

Definition at line 31 of file collcoll.cpp.

#### 5.3.3.2 `bool zipios::CollectionCollection::addCollection (const FileCollection & collection)`

Adds a collection.

**Parameters:**

*collection* The collection to add.

**Returns:**

true if the collection was added succesfully and the added collection is valid.

Definition at line 21 of file collcoll.cpp.

#### 5.3.3.3 `CollectionCollection * zipios::CollectionCollection::clone () const [virtual]`

Create a heap allocated clone of the object this method is called for.

The caller is responsible for deallocating the clone when he is done with it.

**Returns:**

A heap allocated copy of the object this method is called for.

Reimplemented from `zipios::FileCollection`.

Definition at line 108 of file collcoll.cpp.

#### 5.3.3.4 `void zipios::CollectionCollection::close () [virtual]`

Closes the `FileCollection`.

Reimplemented from `zipios::FileCollection`.

Definition at line 41 of file collcoll.cpp.

### 5.3.3.5 `vector<ConstEntryPoint> zipios::CollectionCollection::entries () const` [virtual]

Returns a vector of const pointers to the entries in the [FileCollection](#).

#### Returns:

a vector< ConstEntryPoint > containing the entries of the [FileCollection](#).

#### Exceptions:

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 46 of file collcoll.cpp.

### 5.3.3.6 `void zipios::CollectionCollection::getEntry (const string & name, ConstEntryPoint & cep, vector<FileCollection *>::const_iterator & it, MatchPath matchpath = MATCH) const` [protected]

A protected getEntry member function, that not only finds an entry that match the name, if such an entry exists in the collection, it also returns, which collection it was found in.

Definition at line 123 of file collcoll.cpp.

### 5.3.3.7 `ConstEntryPoint zipios::CollectionCollection::getEntry (const string & name, MatchPath matchpath = MATCH) const` [virtual]

Returns a ConstEntryPoint to a [FileEntry](#) object for the entry with the specified name.

To ignore the path part of the filename in search of a match, specify [FileCollection::IGNORE](#) as the second argument.

#### Parameters:

*name* A string containing the name of the entry to get.

*matchpath* Specially MATCH, if the path should match as well, specify IGNORE, if the path should be ignored.

#### Returns:

A ConstEntryPoint to the found entry. The returned pointer equals zero if no entry is found.

#### Exceptions:

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 58 of file collcoll.cpp.

### 5.3.3.8 `istream * zipios::CollectionCollection::getInputStream (const string & entry_name, MatchPath matchpath = MATCH)` [virtual]

Returns a pointer to an opened istream for the specified entry name.

It is the callers responsibility to delete the stream when he is done with it. Returns 0, if there is no entry with the specified name in the [FileCollection](#).

**Parameters:**

*matchpath* Specify MATCH, if the path should match as well, specify IGNORE, if the path should be ignored.

**Returns:**

an open istream for the specified entry. The istream is allocated on heap and it is the users responsibility to delete it when he is done with it.

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 80 of file collcoll.cpp.

### 5.3.3.9 istream \* zipios::CollectionCollection::getInputStream (const ConstEntryPoint & entry) [virtual]

Returns a pointer to an opened istream for the specified [FileEntry](#).

It is the callers responsibility to delete the stream when he is done with it. Returns 0, if there is no such [FileEntry](#) in the [FileCollection](#).

**Parameters:**

*entry* A ConstEntryPoint to the [FileEntry](#) to get an istream to.

**Returns:**

an open istream for the specified entry. The istream is allocated on heap and it is the users responsibility to delete it when he is done with it.

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 72 of file collcoll.cpp.

### 5.3.3.10 CollectionCollection & zipios::CollectionCollection::inst () [inline, static]

This static method provides a singleton instance of a [CollectionCollection](#).

The instance is instantiated the first time the method is called.

**Returns:**

A pointer to a singleton [CollectionCollection](#) instance.

Definition at line 111 of file collcoll.h.

### 5.3.3.11 const CollectionCollection & zipios::CollectionCollection::operator= (const CollectionCollection & src) [inline]

Copy assignment operator.

Definition at line 129 of file collcoll.h.

**5.3.3.12** `int zipios::CollectionCollection::size () const` [virtual]

Returns the number in entries in all collections kept by the [CollectionCollection](#) object.

Reimplemented from [zipios::FileCollection](#).

Definition at line 98 of file `collcoll.cpp`.

The documentation for this class was generated from the following files:

- [collcoll.h](#)
- [collcoll.cpp](#)



## 5.4 zipios::DataDescriptor Struct Reference

A struct containing fields for the entries in a zip file data descriptor, that trails the compressed data in files that were created by streaming, ie.

```
#include <ziphead.h>
```

### Public Attributes

- uint32 **crc\_32**
- uint32 **compress\_size**
- uint32 **uncompress\_size**

### 5.4.1 Detailed Description

A struct containing fields for the entries in a zip file data descriptor, that trails the compressed data in files that were created by streaming, ie.

where the zip compressor cannot seek back to the local header and store the data.

Definition at line 83 of file ziphead.h.

The documentation for this struct was generated from the following file:

- [ziphead.h](#)

## 5.5 zipios::DirectoryCollection Class Reference

[DirectoryCollection](#) is a [FileCollection](#) that obtains its entries from a directory.

```
#include <dircoll.h>
```

Inheritance diagram for zipios::DirectoryCollection



### Public Methods

- [DirectoryCollection](#) ()  
*Default Constructor.*
- [DirectoryCollection](#) ( const string &path, bool recursive = true, bool load\_now = false )  
*Constructor.*
- virtual void [close](#) ()  
*Closes the [FileCollection](#).*
- virtual vector< ConstEntryPoint > [entries](#) () const  
*Returns a vector of const pointers to the entries in the [FileCollection](#).*
- virtual ConstEntryPoint [getEntry](#) ( const string &name, MatchPath matchpath = MATCH ) const  
*Returns a ConstEntryPoint to a [FileEntry](#) object for the entry with the specified name.*
- virtual istream\* [getInputStream](#) ( const ConstEntryPoint &entry )  
*Returns a pointer to an opened istream for the specified [FileEntry](#).*
- virtual istream\* [getInputStream](#) ( const string &entry\_name, MatchPath matchpath = MATCH )  
*Returns a pointer to an opened istream for the specified entry name.*
- virtual int [size](#) () const  
*Returns the number of entries in the [FileCollection](#).*
- virtual DirectoryCollection\* [clone](#) () const  
*Create a heap allocated clone of the object this method is called for.*
- virtual [~DirectoryCollection](#) ()  
*Destructor.*

## Protected Methods

- void **loadEntries** () const
- void **Load** ( bool recursive, const [FilePath](#) &subdir = string() )

## Protected Attributes

- bool **\_entries\_loaded**
- bool **\_recursive**
- [FilePath](#) **\_filepath**

### 5.5.1 Detailed Description

[DirectoryCollection](#) is a [FileCollection](#) that obtains its entries from a directory.

Definition at line 19 of file dircoll.h.

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 zipios::DirectoryCollection::DirectoryCollection () [explicit]

Default Constructor.

Definition at line 23 of file dircoll.h.

#### 5.5.2.2 zipios::DirectoryCollection::DirectoryCollection (const string & path, bool recursive = true, bool load\_now = false) [explicit]

Constructor.

##### Parameters:

*path* A directory path name. If the name is not a valid directory the created [DirectoryCollection](#) will be invalid.

*load\_now* Load directory into memory now. Otherwise it will be done when it is first needed.

Definition at line 20 of file dircoll.cpp.

#### 5.5.2.3 zipios::DirectoryCollection::~~DirectoryCollection () [virtual]

Destructor.

Definition at line 118 of file dircoll.cpp.

### 5.5.3 Member Function Documentation

### 5.5.3.1 `DirectoryCollection * zipios::DirectoryCollection::clone () const` [virtual]

Create a heap allocated clone of the object this method is called for.

The caller is responsible for deallocating the clone when he is done with it.

#### Returns:

A heap allocated copy of the object this method is called for.

Reimplemented from [zipios::FileCollection](#).

Definition at line 114 of file `dircoll.cpp`.

### 5.5.3.2 `void zipios::DirectoryCollection::close ()` [virtual]

Closes the [FileCollection](#).

Reimplemented from [zipios::FileCollection](#).

Definition at line 33 of file `dircoll.cpp`.

### 5.5.3.3 `vector<ConstEntryPoint> zipios::DirectoryCollection::entries () const` [virtual]

Returns a vector of const pointers to the entries in the [FileCollection](#).

#### Returns:

a `vector< ConstEntryPoint >` containing the entries of the [FileCollection](#).

#### Exceptions:

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 38 of file `dircoll.cpp`.

### 5.5.3.4 `ConstEntryPoint zipios::DirectoryCollection::getEntry (const string & name, MatchPath matchpath = MATCH) const` [virtual]

Returns a `ConstEntryPoint` to a [FileEntry](#) object for the entry with the specified name.

To ignore the path part of the filename in search of a match, specify `FileCollection::IGNORE` as the second argument.

#### Parameters:

*name* A string containing the name of the entry to get.

*matchpath* Specify `MATCH`, if the path should match as well, specify `IGNORE`, if the path should be ignored.

#### Returns:

A `ConstEntryPoint` to the found entry. The returned pointer equals zero if no entry is found.

#### Exceptions:

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 49 of file dircoll.cpp.

#### 5.5.3.5 `istream * zipios::DirectoryCollection::getInputStream (const string & entry_name, MatchPath matchpath = MATCH) [virtual]`

Returns a pointer to an opened istream for the specified entry name.

It is the callers responsibility to delete the stream when he is done with it. Returns 0, if there is no entry with the specified name in the [FileCollection](#).

##### Parameters:

*matchpath* Specify MATCH, if the path should match as well, specify IGNORE, if the path should be ignored.

##### Returns:

an open istream for the specified entry. The istream is allocated on heap and it is the users responsibility to delete it when he is done with it.

##### Exceptions:

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 76 of file dircoll.cpp.

#### 5.5.3.6 `istream * zipios::DirectoryCollection::getInputStream (const ConstEntryPointer & entry) [virtual]`

Returns a pointer to an opened istream for the specified [FileEntry](#).

It is the callers responsibility to delete the stream when he is done with it. Returns 0, if there is no such [FileEntry](#) in the [FileCollection](#).

##### Parameters:

*entry* A ConstEntryPointer to the [FileEntry](#) to get an istream to.

##### Returns:

an open istream for the specified entry. The istream is allocated on heap and it is the users responsibility to delete it when he is done with it.

##### Exceptions:

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 68 of file dircoll.cpp.

#### 5.5.3.7 `int zipios::DirectoryCollection::size () const [virtual]`

Returns the number of entries in the [FileCollection](#).

##### Returns:

the number of entries in the [FileCollection](#).

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 106 of file dircoll.cpp.

The documentation for this class was generated from the following files:

- [dircoll.h](#)
- [dircoll.cpp](#)

## 5.6 zipios::EndOfCentralDirectory Class Reference

The end of the Central directory structure.

```
#include <ziphead.h>
```

### Public Methods

- uint32 **offset** () const
- uint16 **totalCount** () const
- int **eocdOffsetFromEnd** () const
- bool **read** ( vector<unsigned char> &buf, int pos )

### Friends

- class **operator**<<

### 5.6.1 Detailed Description

The end of the Central directory structure.

This structure is stored in the end of the zipfile, and contains information about the zipfile, including the position of the start of the central directory.

Definition at line 126 of file ziphead.h.

The documentation for this class was generated from the following files:

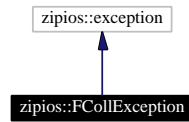
- [ziphead.h](#)
- [ziphead.cpp](#)

## 5.7 zipios::FCollException Class Reference

An [FCollException](#) is used to signal a problem with a [FileCollection](#).

```
#include <fcollexceptions.h>
```

Inheritance diagram for zipios::FCollException



### Public Methods

- **FCollException** () throw ()
- **FCollException** ( const string &msg ) throw ()
- **FCollException** ( const FCollException &src ) throw ()
- FCollException& **operator=** ( const FCollException &src ) throw ()
- virtual const char\* **what** () const throw ()
- virtual ~**FCollException** () throw ()

### 5.7.1 Detailed Description

An [FCollException](#) is used to signal a problem with a [FileCollection](#).

Definition at line 31 of file fcollexceptions.h.

The documentation for this class was generated from the following files:

- [fcollexceptions.h](#)
- [fcollexceptions.cpp](#)

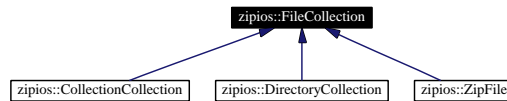


## 5.8 zipios::FileCollection Class Reference

`FileCollection` is an abstract baseclass that represents a collection of files.

```
#include <fcoll.h>
```

Inheritance diagram for `zipios::FileCollection`



### Public Types

- enum `MatchPath` { `IGNORE`, `MATCH` }

### Public Methods

- `FileCollection` ()  
*FileCollection constructor.*
- `FileCollection` ( const `FileCollection` &src )  
*Copy constructor.*
- const `FileCollection`& `operator=` ( const `FileCollection` &src )  
*Copy assignment operator.*
- virtual void `close` () = 0  
*Closes the FileCollection.*
- virtual vector< `ConstEntryPoint` > `entries` () const  
*Returns a vector of const pointers to the entries in the FileCollection.*
- virtual `ConstEntryPoint` `getEntry` ( const string &name, `MatchPath` matchpath = `MATCH` ) const  
*Returns a ConstEntryPoint to a FileEntry object for the entry with the specified name.*
- virtual `istream`\* `getInputStream` ( const `ConstEntryPoint` &entry ) = 0  
*Returns a pointer to an opened istream for the specified FileEntry.*
- virtual `istream`\* `getInputStream` ( const string &entry\_name, `MatchPath` matchpath = `MATCH` ) = 0  
*Returns a pointer to an opened istream for the specified entry name.*
- virtual string `getName` () const  
*Returns the name of the FileCollection.*

- virtual int `size ()` const  
*Returns the number of entries in the `FileCollection`.*
- bool `isValid ()` const  
*The member function returns true if the collection is valid.*
- virtual `FileCollection* clone ()` const = 0  
*Create a heap allocated clone of the object this method is called for.*
- virtual `~FileCollection ()`  
*`FileCollection` destructor.*

## Protected Attributes

- string `_filename`
- vector< `EntryPoint` > `_entries`
- bool `_valid`

### 5.8.1 Detailed Description

`FileCollection` is an abstract baseclass that represents a collection of files.

The specializations of `FileCollection` represents different origins of file collections, such as directories, simple filename lists and compressed archives.

Definition at line 20 of file `fcoll.h`.

### 5.8.2 Constructor & Destructor Documentation

#### 5.8.2.1 `zipios::FileCollection::FileCollection ()` [explicit]

`FileCollection` constructor.

Definition at line 23 of file `fcoll.h`.

#### 5.8.2.2 `zipios::FileCollection::FileCollection (const FileCollection & src)` [inline]

Copy constructor.

Definition at line 114 of file `fcoll.h`.

#### 5.8.2.3 `zipios::FileCollection::~~FileCollection ()` [virtual]

`FileCollection` destructor.

Definition at line 52 of file `fcoll.cpp`.

## 5.8.3 Member Function Documentation

### 5.8.3.1 FileCollection \* zipios::FileCollection::clone () const [pure virtual]

Create a heap allocated clone of the object this method is called for.

The caller is responsible for deallocating the clone when he is done with it.

**Returns:**

A heap allocated copy of the object this method is called for.

Reimplemented in [zipios::CollectionCollection](#), [zipios::DirectoryCollection](#), and [zipios::ZipFile](#).

### 5.8.3.2 void zipios::FileCollection::close () [pure virtual]

Closes the [FileCollection](#).

Reimplemented in [zipios::CollectionCollection](#), [zipios::DirectoryCollection](#), and [zipios::ZipFile](#).

### 5.8.3.3 vector<ConstEntryPoint> zipios::FileCollection::entries () const [virtual]

Returns a vector of const pointers to the entries in the [FileCollection](#).

**Returns:**

a vector< ConstEntryPoint > containing the entries of the [FileCollection](#).

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented in [zipios::CollectionCollection](#), and [zipios::DirectoryCollection](#).

Definition at line 17 of file fcoll.cpp.

### 5.8.3.4 ConstEntryPoint zipios::FileCollection::getEntry (const string & name, MatchPath matchpath = MATCH) const [virtual]

Returns a ConstEntryPoint to a [FileEntry](#) object for the entry with the specified name.

To ignore the path part of the filename in search of a match, specify FileCollection::IGNORE as the second argument.

**Parameters:**

*name* A string containing the name of the entry to get.

*matchpath* Specify MATCH, if the path should match as well, specify IGNORE, if the path should be ignored.

**Returns:**

A ConstEntryPoint to the found entry. The returned pointer equals zero if no entry is found.

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented in [zipios::CollectionCollection](#), and [zipios::DirectoryCollection](#).

Definition at line 23 of file fcoll.cpp.

### 5.8.3.5 `istream * zipios::FileCollection::getInputStream (const string & entry_name, MatchPath matchpath = MATCH)` [pure virtual]

Returns a pointer to an opened istream for the specified entry name.

It is the callers responsibility to delete the stream when he is done with it. Returns 0, if there is no entry with the specified name in the [FileCollection](#).

**Parameters:**

*matchpath* Specify MATCH, if the path should match as well, specify IGNORE, if the path should be ignored.

**Returns:**

an open istream for the specified entry. The istream is allocated on heap and it is the users responsibility to delete it when he is done with it.

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented in [zipios::CollectionCollection](#), [zipios::DirectoryCollection](#), and [zipios::ZipFile](#).

### 5.8.3.6 `istream * zipios::FileCollection::getInputStream (const ConstEntryPointer & entry)` [pure virtual]

Returns a pointer to an opened istream for the specified [FileEntry](#).

It is the callers responsibility to delete the stream when he is done with it. Returns 0, if there is no such [FileEntry](#) in the [FileCollection](#).

**Parameters:**

*entry* A ConstEntryPointer to the [FileEntry](#) to get an istream to.

**Returns:**

an open istream for the specified entry. The istream is allocated on heap and it is the users responsibility to delete it when he is done with it.

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented in [zipios::CollectionCollection](#), [zipios::DirectoryCollection](#), and [zipios::ZipFile](#).

### 5.8.3.7 `string zipios::FileCollection::getName () const` [virtual]

Returns the name of the [FileCollection](#).

**Returns:**

the name of the [FileCollection](#).

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Definition at line 39 of file fcoll.cpp.

**5.8.3.8 bool zipios::FileCollection::isValid () const [inline]**

The member function returns true if the collection is valid.

**Returns:**

true if the collection is valid.

Definition at line 92 of file fcoll.h.

**5.8.3.9 const FileCollection & zipios::FileCollection::operator= (const FileCollection & src) [inline]**

Copy assignment operator.

Definition at line 124 of file fcoll.h.

**5.8.3.10 int zipios::FileCollection::size () const [virtual]**

Returns the number of entries in the [FileCollection](#).

**Returns:**

the number of entries in the [FileCollection](#).

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented in [zipios::CollectionCollection](#), and [zipios::DirectoryCollection](#).

Definition at line 46 of file fcoll.cpp.

The documentation for this class was generated from the following files:

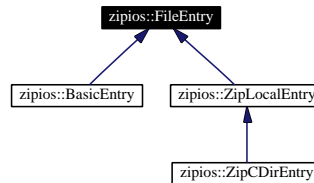
- [fcoll.h](#)
- [fcoll.cpp](#)

## 5.9 zipios::FileEntry Class Reference

A [FileEntry](#) represents an entry in a [FileCollection](#).

```
#include <fileentry.h>
```

Inheritance diagram for zipios::FileEntry



### Public Methods

- [FileEntry](#) ()  
*Constructor.*
- [FileEntry](#) ( const [FileEntry](#) &src )  
*Copy constructor.*
- const [FileEntry](#)& [operator=](#) ( const [FileEntry](#) &src )  
*Assignment operator.*
- virtual string [getComment](#) () const = 0  
*Returns the comment of the entry, if it has one.*
- virtual int [getCompressedSize](#) () const = 0  
*Returns the compressed size of the entry.*
- virtual int [getCrc](#) () const = 0  
*Returns the Crc for the entry, if it has one.*
- virtual vector< unsigned char > [getExtra](#) () const = 0  
*Returns a vector of bytes of extra data that may be stored with the entry.*
- virtual StorageMethod [getMethod](#) () const = 0  
*Returns the method used to store the entry in the [FileCollection](#).*
- virtual string [getName](#) () const = 0  
*Returns the full filename of the entry, including a path if the entry is stored in a subfolder.*
- virtual string [getFileName](#) () const = 0  
*Returns the filename of the entry.*

- virtual int `getSize () const = 0`  
*Returns the (uncompressed) size of the entry data.*
- virtual int `getTime () const = 0`  
*Returns the date and time of FIXME: what?*
- virtual bool `isValid () const = 0`  
*Any method or operator that initializes a `FileEntry` may set a flag, that specifies whether the read entry is valid or not.*
- virtual bool `isDirectory () const = 0`  
*Returns true if the entry is a directory.*
- virtual void `setComment ( const string &comment ) = 0`  
*Sets the comment field for the `FileEntry`.*
- virtual void `setCompressedSize ( int size ) = 0`  
*Set the compressed size field of the entry.*
- virtual void `setCrc ( int crc ) = 0`  
*Sets the crc field.*
- virtual void `setExtra ( const vector< unsigned char > &extra ) = 0`  
*Sets the extra field.*
- virtual void `setMethod ( StorageMethod method ) = 0`  
*Sets the storage method field for the entry.*
- virtual void `setName ( const string &name ) = 0`  
*Sets the name field for the entry.*
- virtual void `setSize ( int size ) = 0`  
*Sets the size field for the entry.*
- virtual void `setTime ( int time ) = 0`  
*Sets the time field for the entry.*
- virtual string `toString () const = 0`  
*Returns a human-readable string representation of the entry.*
- virtual `FileEntry* clone () const = 0`  
*Create a heap allocated clone of the object this method is called for.*
- virtual `~FileEntry ()`  
*`FileEntry` destructor.*

## Protected Methods

- void `ref () const`
- unsigned int `unref () const`

## Protected Attributes

- unsigned short `_ref_count`

### 5.9.1 Detailed Description

A [FileEntry](#) represents an entry in a [FileCollection](#).

The interface is a copy of the `ZipEntry` interface from the `java.util.zip` package. The name has been changed to [FileEntry](#), as [FileCollection](#) is a more general abstraction, that covers other types of file collections than just zip files.

Definition at line 46 of file `fileentry.h`.

### 5.9.2 Constructor & Destructor Documentation

#### 5.9.2.1 `zipios::FileEntry::FileEntry () [explicit]`

Constructor.

Definition at line 51 of file `fileentry.h`.

#### 5.9.2.2 `zipios::FileEntry::FileEntry (const FileEntry & src)`

Copy constructor.

New copy does `_not_` have same ref count as `src`!

Definition at line 55 of file `fileentry.h`.

#### 5.9.2.3 `zipios::FileEntry::~~FileEntry () [inline, virtual]`

[FileEntry](#) destructor.

Definition at line 166 of file `fileentry.h`.

### 5.9.3 Member Function Documentation

#### 5.9.3.1 `FileEntry * zipios::FileEntry::clone () const [pure virtual]`

Create a heap allocated clone of the object this method is called for.

The caller is responsible for deallocating the clone when he is done with it.

#### Returns:

A heap allocated copy of the object this method is called for.

Reimplemented in [zipios::BasicEntry](#), [zipios::ZipCDirEntry](#), and [zipios::ZipLocalEntry](#).



**5.9.3.2** `string zipios::FileEntry::getComment () const` [pure virtual]

Returns the comment of the entry, if it has one.

Otherwise it returns an empty string.

**Returns:**

the comment associated with the entry, if there is one.

Reimplemented in [zipios::BasicEntry](#), [zipios::ZipCDirEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.3** `int zipios::FileEntry::getCompressedSize () const` [pure virtual]

Returns the compressed size of the entry.

If the entry is not stored in a compressed format, the uncompressed size is returned.

**Returns:**

the compressed size of the entry. If the entry is stored without compression the uncompressed size is returned.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.4** `int zipios::FileEntry::getCrc () const` [pure virtual]

Returns the Crc for the entry, if it has one.

FIXME: what is returned if it doesn't have one?

**Returns:**

the Crc for the entry, if it has one.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.5** `vector<unsigned char> zipios::FileEntry::getExtra () const` [pure virtual]

Returns a vector of bytes of extra data that may be stored with the entry.

**Returns:**

A vector< unsigned char > of extra bytes that may potentially be associated with an entry.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.6** `string zipios::FileEntry::getFileName () const` [pure virtual]

Returns the filename of the entry.

**Returns:**

Returns the filename of the entry.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

### 5.9.3.7 StorageMethod zipios::FileEntry::getMethod () const [pure virtual]

Returns the method used to store the entry in the [FileCollection](#).

**Returns:**

the storage method used to store the entry in the collection.

**See also:**

[StorageMethod](#).

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

### 5.9.3.8 string zipios::FileEntry::getName () const [pure virtual]

Returns the full filename of the entry, including a path if the entry is stored in a subfolder.

**Returns:**

the filename of the entry, including path if the entry is stored in a sub-folder.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

### 5.9.3.9 int zipios::FileEntry::getSize () const [pure virtual]

Returns the (uncompressed) size of the entry data.

**Returns:**

Returns the (uncompressed) size of the entry.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

### 5.9.3.10 int zipios::FileEntry::getTime () const [pure virtual]

Returns the date and time of FIXME: what?

**Returns:**

the date and time of the entry.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

### 5.9.3.11 bool zipios::FileEntry::isDirectory () const [pure virtual]

Returns true if the entry is a directory.

A directory entry is an entry which name ends with a separator ('/' for Unix systems, '\\\' for Windows and DOS systems).

**Returns:**

true if the entry is a directory.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.12** `bool zipios::FileEntry::isValid () const` [pure virtual]

Any method or operator that initializes a [FileEntry](#) may set a flag, that specifies whether the read entry is valid or not.

If it isn't this method returns false.

**Returns:**

true if the [FileEntry](#) has been parsed successfully.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.13** `const FileEntry & zipios::FileEntry::operator= (const FileEntry & src)` [inline]

Assignment operator.

Assignment does not change our ref count.

Definition at line 59 of file fileentry.h.

**5.9.3.14** `void zipios::FileEntry::setComment (const string & comment)` [pure virtual]

Sets the comment field for the [FileEntry](#).

**Parameters:**

*comment* string with the new comment.

Reimplemented in [zipios::BasicEntry](#), [zipios::ZipCDirEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.15** `void zipios::FileEntry::setCompressedSize (int size)` [pure virtual]

Set the compressed size field of the entry.

**Parameters:**

*size* value to set the compressed size field of the entry to.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.16** `void zipios::FileEntry::setCrc (int crc)` [pure virtual]

Sets the crc field.

**Parameters:**

*crc* value to set the crc field to.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.17** `void zipios::FileEntry::setExtra (const vector<unsigned char>& extra)` [pure virtual]

Sets the extra field.

**Parameters:**

*extra* the extra field is set to this value.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.18 void zipios::FileEntry::setMethod (StorageMethod *method*) [pure virtual]**

Sets the storage method field for the entry.

**Parameters:**

*method* the method field is set to the specified value.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.19 void zipios::FileEntry::setName (const string & *name*) [pure virtual]**

Sets the name field for the entry.

**Parameters:**

*name* the name field is set to the specified value.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.20 void zipios::FileEntry::setSize (int *size*) [pure virtual]**

Sets the size field for the entry.

**Parameters:**

*size* the size field is set to this value.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.21 void zipios::FileEntry::setTime (int *time*) [pure virtual]**

Sets the time field for the entry.

**Parameters:**

*time* the time field is set to the specified value.

Reimplemented in [zipios::BasicEntry](#), and [zipios::ZipLocalEntry](#).

**5.9.3.22 string zipios::FileEntry::toString () const [pure virtual]**

Returns a human-readable string representation of the entry.

**Returns:**

a human-readable string representation of the entry.

Reimplemented in [zipios::BasicEntry](#), [zipios::ZipCDirEntry](#), and [zipios::ZipLocalEntry](#).

The documentation for this class was generated from the following file:

- [fileentry.h](#)

## 5.10 zipios::FilePath Class Reference

[FilePath](#) represents a path to a file or directory name.

```
#include <filepath.h>
```

### Public Methods

- [FilePath](#) ( const string &path = "", bool check\_exists = false )

*Constructor.*

- [FilePath& operator=](#) ( const string &rhs )
- [operator string](#) () const
- [FilePath operator+](#) ( const [FilePath](#) &name ) const

*Concatenates [FilePath](#) objects.*

- [FilePath filename](#) () const

*Returns filename of the [FilePath](#) object by pruning the path off.*

- bool [exists](#) () const
- bool [isRegular](#) () const
- bool [isDirectory](#) () const
- bool [isCharSpecial](#) () const
- bool [isBlockSpecial](#) () const
- bool [isSocket](#) () const
- bool [isFifo](#) () const

### Protected Methods

- void [pruneTrailingSeparator](#) ()

*Prunes the trailing separator of a specified path.*

- void [check](#) () const

*This function sets `_checked` to true, stats the path, to see if it exists and to determine what type of file it is.*

### Protected Attributes

- bool `_checked`
- bool `_exists`
- bool `_is_reg`
- bool `_is_dir`
- bool `_is_char`
- bool `_is_block`
- bool `_is_socket`
- bool `_is_fifo`
- string `_path`

## Static Protected Attributes

- `const char _separator`

### 5.10.1 Detailed Description

`FilePath` represents a path to a file or directory name.

`FilePath` has member functions to check if the file path is a valid file system entity, and to check what kind of file system entity it is, e.g. is it a file, a directory, a pipe etc.

Definition at line 18 of file `filepath.h`.

### 5.10.2 Constructor & Destructor Documentation

#### 5.10.2.1 `zipios::FilePath::FilePath (const string & path = "", bool check_exists = false)`

Constructor.

##### Parameters:

*path* A string representation of the path.

*check\_exists* If true is specified the constructor will check the existence and type of the path immediately, instead of deferring that task until it is needed.

Definition at line 16 of file `filepath.cpp`.

### 5.10.3 Member Function Documentation

#### 5.10.3.1 `void zipios::FilePath::check () const [protected]`

This function sets `_checked` to true, stats the path, to see if it exists and to determine what type of file it is.

All the query functions check if `_checked` is true, and if it isn't they call `check()`. This means stat'ing is deferred until it becomes necessary.

Definition at line 25 of file `filepath.cpp`.

#### 5.10.3.2 `bool zipios::FilePath::exists () const [inline]`

##### Returns:

true If the path is a valid file system entity.

Definition at line 129 of file `filepath.h`.

**5.10.3.3** `FilePath zipios::FilePath::filename () const` [inline]

Returns filename of the [FilePath](#) object by pruning the path off.

Definition at line 119 of file filepath.h.

**5.10.3.4** `bool zipios::FilePath::isBlockSpecial () const` [inline]**Returns:**

true if the path is block special (a block device file).

Definition at line 157 of file filepath.h.

**5.10.3.5** `bool zipios::FilePath::isCharSpecial () const` [inline]**Returns:**

true if the path is character special (a character device file).

Definition at line 150 of file filepath.h.

**5.10.3.6** `bool zipios::FilePath::isDirectory () const` [inline]**Returns:**

true if the path is a directory.

Definition at line 143 of file filepath.h.

**5.10.3.7** `bool zipios::FilePath::isFifo () const` [inline]**Returns:**

true if the path is a Fifo (a pipe).

Definition at line 171 of file filepath.h.

**5.10.3.8** `bool zipios::FilePath::isRegular () const` [inline]**Returns:**

true if the path is a regular file.

Definition at line 136 of file filepath.h.

**5.10.3.9** `bool zipios::FilePath::isSocket () const` [inline]**Returns:**

true if the path is a socket.

Definition at line 164 of file filepath.h.

**5.10.3.10** `FilePath zipios::FilePath::operator+ (const FilePath & name) const` [inline]

Concatenates [FilePath](#) objects.

A file separator is inserted if appropriate.

Definition at line 111 of file `filepath.h`.

**5.10.3.11** `void zipios::FilePath::pruneTrailingSeparator ()` [inline, protected]

Prunes the trailing separator of a specified path.

Definition at line 100 of file `filepath.h`.

The documentation for this class was generated from the following files:

- [filepath.h](#)
- [filepath.cpp](#)

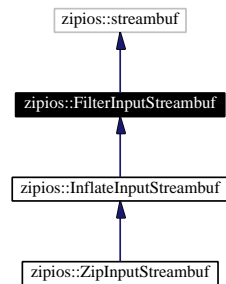


## 5.11 zipios::FilterInputStreambuf Class Reference

An input streambuf filter is a streambuf that filters the input it gets from the streambuf it is attached to.

```
#include <filterinputstreambuf.h>
```

Inheritance diagram for zipios::FilterInputStreambuf



### Public Methods

- [FilterInputStreambuf](#) ( streambuf \*inbuf, bool del\_inbuf = false )

*Constructor.*

- virtual [~FilterInputStreambuf](#) ()

*Destructor.*

### Protected Attributes

- int **\_s\_pos**
- streambuf\* **\_inbuf**
- bool **\_del\_inbuf**

#### 5.11.1 Detailed Description

An input streambuf filter is a streambuf that filters the input it gets from the streambuf it is attached to.

[FilterInputStreambuf](#) is a base class to derive input streambuf filters from.

Definition at line 15 of file filterinputstreambuf.h.

#### 5.11.2 Constructor & Destructor Documentation

### 5.11.2.1 zipios::FilterInputStreambuf::FilterInputStreambuf (streambuf \* *inbuf*, bool *del\_inbuf* = false) [explicit]

Constructor.

#### Parameters:

*inbuf* the streambuf to use for input.

*del\_inbuf* if true is specified *inbuf* will be deleted, when the [FilterInputStreambuf](#) is destructed.

Definition at line 8 of file `filterinputstreambuf.cpp`.

### 5.11.2.2 zipios::FilterInputStreambuf::~~FilterInputStreambuf () [virtual]

Destructor.

Definition at line 18 of file `filterinputstreambuf.cpp`.

The documentation for this class was generated from the following files:

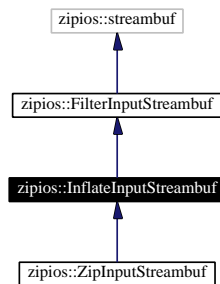
- [filterinputstreambuf.h](#)
- [filterinputstreambuf.cpp](#)

## 5.12 zipios::InflateInputStreambuf Class Reference

[InflateInputStreambuf](#) is an input stream filter, that inflates the input from the attached input stream.

```
#include <inflateinputstreambuf.h>
```

Inheritance diagram for zipios::InflateInputStreambuf



### Public Methods

- [InflateInputStreambuf](#) ( streambuf \*inbuf, int s\_pos = -1, bool del\_inbuf = false )

*InflateInputStreambuf constructor.*

- virtual ~**InflateInputStreambuf** ()
- bool [reset](#) ( int stream\_position = -1 )

*Resets the zlib stream and purges input and output buffers.*

### Protected Methods

- virtual int **underflow** ()

### Protected Attributes

- const int **\_outvecsize**
- vector< char > **\_outvec**

#### 5.12.1 Detailed Description

[InflateInputStreambuf](#) is an input stream filter, that inflates the input from the attached input stream.

Deflation/Inflation is a compression/decompression method used in gzip and zip. The zlib library is used to perform the actual inflation, this class only wraps the functionality in an input stream filter.

Definition at line 22 of file inflateinputstreambuf.h.

## 5.12.2 Constructor & Destructor Documentation

### 5.12.2.1 `zipios::InflateInputStreambuf::InflateInputStreambuf (streambuf * inbuf, int s_pos = -1, bool del_inbuf = false) [explicit]`

[InflateInputStreambuf](#) constructor.

**Parameters:**

*inbuf* the streambuf to use for input.

*s\_pos* a position to reset the inbuf to before reading. Specify -1 to read from the current position.

*del\_inbuf* if true is specified inbuf will be deleted, when the [InflateInputStreambuf](#) is destructed.

Definition at line 18 of file `inflateinputstreambuf.cpp`.

## 5.12.3 Member Function Documentation

### 5.12.3.1 `bool zipios::InflateInputStreambuf::reset (int stream_position = -1)`

Resets the zlib stream and purges input and output buffers.

repositions the input streambuf at `stream_position`.

**Parameters:**

*stream\_position* a position to reset the inbuf to before reading. Specify -1 to read from the current position.

Definition at line 116 of file `inflateinputstreambuf.cpp`.

The documentation for this class was generated from the following files:

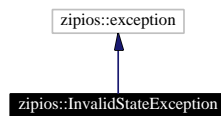
- [inflateinputstreambuf.h](#)
- [inflateinputstreambuf.cpp](#)

## 5.13 zipios::InvalidStateException Class Reference

An object member function may throw this exception, if the operation it normally performs is inappropriate or impossible to perform because of the current state of the object.

```
#include <fcollexceptions.h>
```

Inheritance diagram for zipios::InvalidStateException



### Public Methods

- **InvalidStateException** () throw ()
- **InvalidStateException** ( const string &msg ) throw ()
- **InvalidStateException** ( const InvalidStateException &src ) throw ()
- InvalidStateException& **operator=** ( const InvalidStateException &src ) throw ()
- virtual const char\* **what** () const throw ()
- virtual ~**InvalidStateException** () throw ()

#### 5.13.1 Detailed Description

An object member function may throw this exception, if the operation it normally performs is inappropriate or impossible to perform because of the current state of the object.

Definition at line 47 of file `fcollexceptions.h`.

The documentation for this class was generated from the following files:

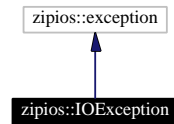
- [fcollexceptions.h](#)
- [fcollexceptions.cpp](#)

## 5.14 zipios::IOException Class Reference

An [IOException](#) is used to signal an I/O error.

```
#include <fcollexceptions.h>
```

Inheritance diagram for zipios::IOException



### Public Methods

- **IOException** () throw ()
- **IOException** ( const string &msg ) throw ()
- **IOException** ( const IOException &src ) throw ()
- **IOException& operator=** ( const IOException &src ) throw ()
- virtual const char\* **what** () const throw ()
- virtual **~IOException** () throw ()

#### 5.14.1 Detailed Description

An [IOException](#) is used to signal an I/O error.

Definition at line 16 of file `fcollexceptions.h`.

The documentation for this class was generated from the following files:

- [fcollexceptions.h](#)
- [fcollexceptions.cpp](#)

## 5.15 zipios::FileEntry::MatchFileName Class Reference

Function object to be used with the STL `find_if` algorithm to find a [FileEntry](#) in a container, which name (as obtained with [FileEntry::getFileName\(\)](#)) is identical to the name specified in the [MatchName](#) constructor.

```
#include <fileentry.h>
```

### Public Methods

- [MatchFileName](#) ( const string &name )
- `bool operator()` ( const ConstEntryPointer &entry )

### 5.15.1 Detailed Description

Function object to be used with the STL `find_if` algorithm to find a [FileEntry](#) in a container, which name (as obtained with [FileEntry::getFileName\(\)](#)) is identical to the name specified in the [MatchName](#) constructor.

Definition at line 196 of file `fileentry.h`.

The documentation for this class was generated from the following file:

- [fileentry.h](#)

## 5.16 zipios::FileEntry::MatchName Class Reference

Function object to be used with the STL `find_if` algorithm to find a [FileEntry](#) in a container, which name (as obtained with [FileEntry::getName\(\)](#)) is identical to the name specified in the [MatchName](#) constructor.

```
#include <fileentry.h>
```

### Public Methods

- [MatchName](#) ( const string &name )
- `bool operator()` ( const ConstEntryPoint &entry )

### 5.16.1 Detailed Description

Function object to be used with the STL `find_if` algorithm to find a [FileEntry](#) in a container, which name (as obtained with [FileEntry::getName\(\)](#)) is identical to the name specified in the [MatchName](#) constructor.

Definition at line 182 of file `fileentry.h`.

The documentation for this class was generated from the following file:

- [fileentry.h](#)

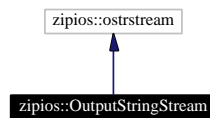


## 5.17 zipios::OutputStringStream Class Reference

[OutputStringStream](#) is typedefed to ostream if sstream is part of the standard library (unless Zipios++ has been explicitly configured not to use it).

```
#include <outputstringstream.h>
```

Inheritance diagram for zipios::OutputStringStream



### Public Methods

- `string str ()`

*Specialization of `ostrstream::str()` that takes care of null-terminating the string and unfreezing the `ostrstream`.*

### 5.17.1 Detailed Description

[OutputStringStream](#) is typedefed to ostream if sstream is part of the standard library (unless Zipios++ has been explicitly configured not to use it).

If sstream is not present [OutputStringStream](#) is a subclass of ostrstream from ostream.h. In this case [OutputStringStream](#) specializes the `str()` method, such that the caller does not have to concern himself with null-terminating the string and unfreezing the ostrstream.

Definition at line 24 of file outputstringstream.h.

### 5.17.2 Member Function Documentation

#### 5.17.2.1 `string zipios::OutputStringStream::str () [inline]`

Specialization of `ostrstream::str()` that takes care of null-terminating the string and unfreezing the `ostrstream`.

Definition at line 29 of file outputstringstream.h.

The documentation for this class was generated from the following file:

- [outputstringstream.h](#)

## 5.18 zipios::SimpleSmartPointer Class Reference

[SimpleSmartPointer](#) is a simple reference counting smart pointer template.

```
#include <simplesmartptr.h>
```

### Public Methods

- `Type* operator → () const`
- `Type& operator * () const`
- `SimpleSmartPointer ( Type *p = 0 )`
- `SimpleSmartPointer ( const SimpleSmartPointer &src )`
- `template<class T2> SimpleSmartPointer<T2> ( const SimpleSmartPointer< T2 > &src )`
- `~SimpleSmartPointer ()`
- `SimpleSmartPointer& operator= ( const SimpleSmartPointer &src )`
- `template<class T2> SimpleSmartPointer& operator=<T2> ( const SimpleSmartPointer< T2 > &src )`
- `bool operator== ( const Type *p ) const`
- `bool operator!= ( const Type *p ) const`
- `bool operator== ( const SimpleSmartPointer &sp ) const`
- `bool operator!= ( const SimpleSmartPointer &sp ) const`
- `bool operator! () const`
- `operator void * () const`
- `Type* get () const`

### 5.18.1 Detailed Description

```
template<class Type> class zipios::SimpleSmartPointer
```

[SimpleSmartPointer](#) is a simple reference counting smart pointer template.

The type pointed to must keep a reference count that is initialized to zero and accessible through the two methods `void ref() const` and `unsigned int unref() const`.

Definition at line 14 of file `simplesmartptr.h`.

The documentation for this class was generated from the following file:

- [simplesmartptr.h](#)

## 5.19 zipios::VirtualSeeker Class Reference

[VirtualSeeker](#) is a simple class that keeps track of a set of specified 'virtual' file endings that mark a subset of a real file.

```
#include <virtalseeker.h>
```

### Public Methods

- **VirtualSeeker** ( int start\_offset = 0, int end\_offset = 0)
- void **setOffsets** ( int start\_offset, int end\_offset )
- void **getOffsets** ( int &start\_offset, int &end\_offset ) const
- int **startOffset** () const
- int **endOffset** () const
- void **vseekg** ( istream &is, int offset, ios::seekdir sd ) const
- int **vtellg** ( istream &is ) const

### 5.19.1 Detailed Description

[VirtualSeeker](#) is a simple class that keeps track of a set of specified 'virtual' file endings that mark a subset of a real file.

An example of its use (and its reason for existence) is to keep track of the file endings of a Zip file embedded in another file.

Definition at line 20 of file virtalseeker.h.

The documentation for this class was generated from the following file:

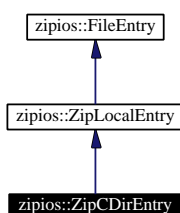
- [virtalseeker.h](#)

## 5.20 zipios::ZipCDirEntry Class Reference

Specialization of [ZipLocalEntry](#), that add fields for storing the extra information, that is only present in the entries in the zip central directory and not in the local entry headers.

```
#include <ziphead.h>
```

Inheritance diagram for zipios::ZipCDirEntry



### Public Methods

- [ZipCDirEntry](#) ()
- [ZipCDirEntry](#)& **operator=** ( const class [ZipCDirEntry](#) &src )
- virtual string [toString](#) () const  
*Returns a human-readable string representation of the entry.*
- virtual string [getComment](#) () const  
*Returns the comment of the entry, if it has one.*
- virtual void [setComment](#) ( const string &comment )  
*Sets the comment field for the [FileEntry](#).*
- virtual uint32 [getLocalHeaderOffset](#) () const
- virtual [ZipCDirEntry](#)\* [clone](#) () const  
*Create a heap allocated clone of the object this method is called for.*
- virtual ~[ZipCDirEntry](#) ()

### Friends

- class **operator**>>
- class **operator**==

#### 5.20.1 Detailed Description

Specialization of [ZipLocalEntry](#), that add fields for storing the extra information, that is only present in the entries in the zip central directory and not in the local entry headers.

Definition at line 92 of file ziphead.h.

## 5.20.2 Member Function Documentation

### 5.20.2.1 ZipCDirEntry \* zipios::ZipCDirEntry::clone () const [virtual]

Create a heap allocated clone of the object this method is called for.

The caller is responsible for deallocating the clone when he is done with it.

**Returns:**

A heap allocated copy of the object this method is called for.

Reimplemented from [zipios::ZipLocalEntry](#).

Definition at line 285 of file ziphead.cpp.

### 5.20.2.2 string zipios::ZipCDirEntry::getComment () const [virtual]

Returns the comment of the entry, if it has one.

Otherwise it returns an empty string.

**Returns:**

the comment associated with the entry, if there is one.

Reimplemented from [zipios::ZipLocalEntry](#).

Definition at line 263 of file ziphead.cpp.

### 5.20.2.3 void zipios::ZipCDirEntry::setComment (const string & comment) [virtual]

Sets the comment field for the [FileEntry](#).

**Parameters:**

*comment* string with the new comment.

Reimplemented from [zipios::ZipLocalEntry](#).

Definition at line 272 of file ziphead.cpp.

### 5.20.2.4 string zipios::ZipCDirEntry::toString () const [virtual]

Returns a human-readable string representation of the entry.

**Returns:**

a human-readable string representation of the entry.

Reimplemented from [zipios::ZipLocalEntry](#).

Definition at line 278 of file ziphead.cpp.

The documentation for this class was generated from the following files:

- [ziphead.h](#)
- [ziphead.cpp](#)

## 5.21 zipios::ZipFile Class Reference

[ZipFile](#) is a [FileCollection](#), where the files are stored in a .zip file.

```
#include <zipfile.h>
```

Inheritance diagram for zipios::ZipFile



### Public Methods

- [ZipFile](#) ()  
*Default constructor.*
- [ZipFile](#) ( const string &name, int s\_off = 0, int e\_off = 0 )  
*Constructor.*
- virtual [ZipFile\\*](#) [clone](#) () const  
*Create a heap allocated clone of the object this method is called for.*
- virtual [~ZipFile](#) ()  
*Destructor.*
- virtual void [close](#) ()  
*Closes the [FileCollection](#).*
- virtual [istream\\*](#) [getInputStream](#) ( const [ConstEntryPoint](#) &entry )  
*Returns a pointer to an opened istream for the specified [FileEntry](#).*
- virtual [istream\\*](#) [getInputStream](#) ( const string &entry\_name, [MatchPath](#) matchpath = MATCH )  
*Returns a pointer to an opened istream for the specified entry name.*

### Static Public Methods

- [ZipFile](#) [openEmbeddedZipFile](#) ( const string &name )  
*Opens a Zip archive embedded in another file, by writing the zip archive to the end of the file followed by the start offset of the zip file.*

### 5.21.1 Detailed Description

`ZipFile` is a `FileCollection`, where the files are stored in a `.zip` file.

Definition at line 20 of file `zipfile.h`.

### 5.21.2 Constructor & Destructor Documentation

#### 5.21.2.1 `zipios::ZipFile::ZipFile ()` [inline]

Default constructor.

Definition at line 37 of file `zipfile.h`.

#### 5.21.2.2 `zipios::ZipFile::ZipFile (const string & name, int s_off = 0, int e_off = 0)` [explicit]

Constructor.

Opens the zip file name. If the zip "file" is embedded in a file that contains other data, e.g. a binary program, the offset of the zip file start and end must be specified.

##### Parameters:

*name* The filename of the zip file to open.

*s\_off* Offset relative to the start of the file, that indicates the beginning of the zip file.

*e\_off* Offset relative to the end of the file, that indicates the end of the zip file. The offset is a positive number, even though the offset is towards the beginning of the file.

##### Exceptions:

*FColException* Thrown if the specified file name is not a valid zip archive.

*IOException* Thrown if an I/O problem is encountered, while the directory of the specified zip archive is being read.

Definition at line 29 of file `zipfile.cpp`.

#### 5.21.2.3 `zipios::ZipFile::~~ZipFile ()` [virtual]

Destructor.

Definition at line 45 of file `zipfile.cpp`.

### 5.21.3 Member Function Documentation

### 5.21.3.1 `ZipFile * zipios::ZipFile::clone () const` [virtual]

Create a heap allocated clone of the object this method is called for.

The caller is responsible for deallocating the clone when he is done with it.

#### Returns:

A heap allocated copy of the object this method is called for.

Reimplemented from [zipios::FileCollection](#).

Definition at line 40 of file zipfile.cpp.

### 5.21.3.2 `void zipios::ZipFile::close ()` [virtual]

Closes the [FileCollection](#).

Reimplemented from [zipios::FileCollection](#).

Definition at line 49 of file zipfile.cpp.

### 5.21.3.3 `istream * zipios::ZipFile::getInputStream (const string & entry_name, MatchPath matchpath = MATCH)` [virtual]

Returns a pointer to an opened istream for the specified entry name.

It is the callers responsibility to delete the stream when he is done with it. Returns 0, if there is no entry with the specified name in the [FileCollection](#).

#### Parameters:

*matchpath* Specify MATCH, if the path should match as well, specify IGNORE, if the path should be ignored.

#### Returns:

an open istream for the specified entry. The istream is allocated on heap and it is the users responsibility to delete it when he is done with it.

#### Exceptions:

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 60 of file zipfile.cpp.

### 5.21.3.4 `istream * zipios::ZipFile::getInputStream (const ConstEntryPoint & entry)` [virtual]

Returns a pointer to an opened istream for the specified [FileEntry](#).

It is the callers responsibility to delete the stream when he is done with it. Returns 0, if there is no such [FileEntry](#) in the [FileCollection](#).

#### Parameters:

*entry* A ConstEntryPoint to the [FileEntry](#) to get an istream to.



**Returns:**

an open istream for the specified entry. The istream is allocated on heap and it is the users responsibility to delete it when he is done with it.

**Exceptions:**

*InvalidStateException* Thrown if the collection is invalid.

Reimplemented from [zipios::FileCollection](#).

Definition at line 54 of file zipfile.cpp.

**5.21.3.5 ZipFile zipios::ZipFile::openEmbeddedZipFile (const string & name) [static]**

Opens a Zip archive embedded in another file, by writing the zip archive to the end of the file followed by the start offset of the zip file.

The offset must be written in zip-file byte-order (little endian). The program `appendzip`, which is part of the Zipios++ distribution can be used to append a Zip archive to a file, e.g. a binary program.

**Exceptions:**

*FColException* Thrown if the specified file name is not a valid zip archive.

*IOException* Thrown if an I/O problem is encountered, while the directory of the specified zip archive is being read.

Definition at line 19 of file zipfile.cpp.

The documentation for this class was generated from the following files:

- [zipfile.h](#)
- [zipfile.cpp](#)

## 5.22 zipios::ZipInputStream Class Reference

[ZipInputStream](#) is an istream that gets it's input from a zip file.

```
#include <zipinputstream.h>
```

Inheritance diagram for zipios::ZipInputStream



### Public Methods

- [ZipInputStream](#) ( istream &is, streampos pos = 0 )  
*ZipInputStream constructor.*
- **ZipInputStream** ( const string &filename, streampos pos = 0 )
- int **available** ()
- void **closeEntry** ()  
*Closes the current entry, and positions the stream read pointer at the beginning of the next entry (if there is one).*
- void **close** ()  
*Closes the istream.*
- ConstEntryPoint **getNextEntry** ()  
*Opens the next entry in the zip archive and returns a const pointer to a [FileEntry](#) object for the entry.*
- virtual **~ZipInputStream** ()  
*Destructor.*

### 5.22.1 Detailed Description

[ZipInputStream](#) is an istream that gets it's input from a zip file.

The interface approximates the interface of the Java [ZipInputStream](#).

Definition at line 20 of file zipinputstream.h.

### 5.22.2 Constructor & Destructor Documentation

### 5.22.2.1 zipios::ZipInputStream::ZipInputStream (istream & *is*, streampos *pos* = 0) [explicit]

[ZipInputStream](#) constructor.

#### Parameters:

*is* istream from which the compressed zip archive can be read.

*pos* position to reposition the istream to before reading.

Definition at line 11 of file zipinputstream.cpp.

### 5.22.2.2 zipios::ZipInputStream::~~ZipInputStream () [virtual]

Destructor.

Definition at line 51 of file zipinputstream.cpp.

## 5.22.3 Member Function Documentation

### 5.22.3.1 void zipios::ZipInputStream::close ()

Closes the istream.

Definition at line 39 of file zipinputstream.cpp.

### 5.22.3.2 void zipios::ZipInputStream::closeEntry ()

Closes the current entry, and positions the stream read pointer at the beginning of the next entry (if there is one).

Definition at line 35 of file zipinputstream.cpp.

### 5.22.3.3 ConstEntryPointer zipios::ZipInputStream::getNextEntry ()

Opens the next entry in the zip archive and returns a const pointer to a [FileEntry](#) object for the entry.

#### Returns:

a const [FileEntry](#) \* containing information about the (now) current entry.

Definition at line 46 of file zipinputstream.cpp.

The documentation for this class was generated from the following files:

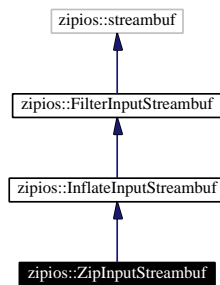
- [zipinputstream.h](#)
- [zipinputstream.cpp](#)

## 5.23 zipios::ZipInputStreambuf Class Reference

[ZipInputStreambuf](#) is a zip input streambuf filter.

```
#include <zipinputstreambuf.h>
```

Inheritance diagram for zipios::ZipInputStreambuf



### Public Methods

- [ZipInputStreambuf](#) ( streambuf \*inbuf, int s\_pos = -1, bool del\_inbuf = false )  
*ZipInputStreambuf constructor:*
- void [closeEntry](#) ()  
*Closes the current entry, and positions the stream read pointer at the beginning of the next entry (if there is one).*
- void [close](#) ()  
*Closes the streambuf.*
- ConstEntryPointer [getNextEntry](#) ()  
*Opens the next entry in the zip archive and returns a const pointer to a [FileEntry](#) object for the entry.*
- virtual [~ZipInputStreambuf](#) ()  
*Destructor.*

### Protected Methods

- virtual int [underflow](#) ()

#### 5.23.1 Detailed Description

[ZipInputStreambuf](#) is a zip input streambuf filter.

Definition at line 18 of file zipinputstreambuf.h.

## 5.23.2 Constructor & Destructor Documentation

### 5.23.2.1 zipios::ZipInputSteambuf::ZipInputSteambuf (streambuf \* *inbuf*, int *s\_pos* = -1, bool *del\_inbuf* = false) [explicit]

[ZipInputSteambuf](#) constructor.

**Parameters:**

*inbuf* the streambuf to use for input.

*s\_pos* a position to reset the inbuf to before reading. Specify -1 to read from the current position.

*del\_inbuf* if true is specified inbuf will be deleted, when the [ZipInputSteambuf](#) is destructed.

Definition at line 18 of file zipinputstreambuf.cpp.

### 5.23.2.2 zipios::ZipInputSteambuf::~~ZipInputSteambuf () [virtual]

Destructor.

Definition at line 80 of file zipinputstreambuf.cpp.

## 5.23.3 Member Function Documentation

### 5.23.3.1 void zipios::ZipInputSteambuf::close ()

Closes the streambuf.

Definition at line 42 of file zipinputstreambuf.cpp.

### 5.23.3.2 void zipios::ZipInputSteambuf::closeEntry ()

Closes the current entry, and positions the stream read pointer at the beginning of the next entry (if there is one).

Definition at line 29 of file zipinputstreambuf.cpp.

### 5.23.3.3 ConstEntryPointer zipios::ZipInputSteambuf::getNextEntry ()

Opens the next entry in the zip archive and returns a const pointer to a [FileEntry](#) object for the entry.

**Returns:**

a const [FileEntry](#) \* containing information about the (now) current entry.

Definition at line 45 of file zipinputstreambuf.cpp.

The documentation for this class was generated from the following files:

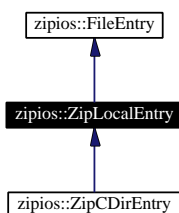
- [zipinputstreambuf.h](#)
- [zipinputstreambuf.cpp](#)

## 5.24 zipios::ZipLocalEntry Class Reference

A concrete implementation of the abstract [FileEntry](#) base class for [ZipFile](#) entries, specifically for representing the information present in the local headers of file entries in a zip file.

```
#include <ziphead.h>
```

Inheritance diagram for zipios::ZipLocalEntry



### Public Methods

- **ZipLocalEntry** ()
- ZipLocalEntry & **operator=** ( const class ZipLocalEntry &src )
- virtual string **getComment** () const  
*Returns the comment of the entry, if it has one.*
- virtual int **getCompressedSize** () const  
*Returns the compressed size of the entry.*
- virtual int **getCrc** () const  
*Returns the Crc for the entry, if it has one.*
- virtual vector< unsigned char > **getExtra** () const  
*Returns a vector of bytes of extra data that may be stored with the entry.*
- virtual StorageMethod **getMethod** () const  
*Returns the method used to store the entry in the [FileCollection](#).*
- virtual string **getName** () const  
*Returns the full filename of the entry, including a path if the entry is stored in a subfolder.*
- virtual string **getFileName** () const  
*Returns the filename of the entry.*
- virtual int **getSize** () const  
*Returns the (uncompressed) size of the entry data.*
- virtual int **getTime** () const  
*Returns the date and time of FIXME: what?*

- virtual bool `isValid () const`  
*Any method or operator that initializes a `FileEntry` may set a flag, that specifies whether the read entry is valid or not.*
- virtual bool `isDirectory () const`  
*Returns true if the entry is a directory.*
- virtual void `setComment ( const string &comment )`  
*Sets the comment field for the `FileEntry`.*
- virtual void `setCompressedSize ( int size )`  
*Set the compressed size field of the entry.*
- virtual void `setCrc ( int crc )`  
*Sets the crc field.*
- virtual void `setExtra ( const vector< unsigned char > &extra )`  
*Sets the extra field.*
- virtual void `setMethod ( StorageMethod method )`  
*Sets the storage method field for the entry.*
- virtual void `setName ( const string &name )`  
*Sets the name field for the entry.*
- virtual void `setSize ( int size )`  
*Sets the size field for the entry.*
- virtual void `setTime ( int time )`  
*Sets the time field for the entry.*
- virtual string `toString () const`  
*Returns a human-readable string representation of the entry.*
- bool `trailingDataDescriptor () const`
- virtual `ZipLocalEntry* clone () const`  
*Create a heap allocated clone of the object this method is called for.*
- virtual `~ZipLocalEntry ()`

### Protected Attributes

- uint32 `signature`
- uint16 `extract_version`
- uint16 `gp_bitfield`
- uint16 `compress_method`
- uint16 `last_mod_ftime`
- uint16 `last_mod_fdate`
- uint32 `crc_32`
- uint32 `compress_size`

- uint32 **uncompress\_size**
- uint16 **filename\_len**
- uint16 **extra\_field\_len**
- string **filename**
- vector< unsigned char > **extra\_field**
- bool **\_valid**

## Friends

- class **operator**>>
- class **operator**<<
- class **operator**==

### 5.24.1 Detailed Description

A concrete implementation of the abstract [FileEntry](#) base class for [ZipFile](#) entries, specifically for representing the information present in the local headers of file entries in a zip file.

Definition at line 24 of file ziphead.h.

### 5.24.2 Member Function Documentation

#### 5.24.2.1 `ZipLocalEntry * zipios::ZipLocalEntry::clone () const` [virtual]

Create a heap allocated clone of the object this method is called for.

The caller is responsible for deallocating the clone when he is done with it.

#### Returns:

A heap allocated copy of the object this method is called for.

Reimplemented from [zipios::FileEntry](#).

Reimplemented in [zipios::ZipCDirEntry](#).

Definition at line 254 of file ziphead.cpp.

#### 5.24.2.2 `string zipios::ZipLocalEntry::getComment () const` [virtual]

Returns the comment of the entry, if it has one.

Otherwise it returns an empty string.

#### Returns:

the comment associated with the entry, if there is one.

Reimplemented from [zipios::FileEntry](#).

Reimplemented in [zipios::ZipCDirEntry](#).

Definition at line 146 of file ziphead.cpp.



### 5.24.2.3 `int zipios::ZipLocalEntry::getCompressedSize () const` [virtual]

Returns the compressed size of the entry.

If the entry is not stored in a compressed format, the uncompressed size is returned.

**Returns:**

the compressed size of the entry. If the entry is stored without compression the uncompressed size is returned.

Reimplemented from [zipios::FileEntry](#).

Definition at line 150 of file ziphead.cpp.

### 5.24.2.4 `int zipios::ZipLocalEntry::getCrc () const` [virtual]

Returns the Crc for the entry, if it has one.

FIXME: what is returned if it doesn't have one?

**Returns:**

the Crc for the entry, if it has one.

Reimplemented from [zipios::FileEntry](#).

Definition at line 154 of file ziphead.cpp.

### 5.24.2.5 `vector<unsigned char> zipios::ZipLocalEntry::getExtra () const` [virtual]

Returns a vector of bytes of extra data that may be stored with the entry.

**Returns:**

A vector< unsigned char > of extra bytes that may potentially be associated with an entry.

Reimplemented from [zipios::FileEntry](#).

Definition at line 158 of file ziphead.cpp.

### 5.24.2.6 `string zipios::ZipLocalEntry::getFileName () const` [virtual]

Returns the filename of the entry.

**Returns:**

Returns the filename of the entry.

Reimplemented from [zipios::FileEntry](#).

Definition at line 170 of file ziphead.cpp.

### 5.24.2.7 `StorageMethod zipios::ZipLocalEntry::getMethod () const` [virtual]

Returns the method used to store the entry in the [FileCollection](#).

**Returns:**

the storage method used to store the entry in the collection.

**See also:**

[StorageMethod](#).

Reimplemented from [zipios::FileEntry](#).

Definition at line 162 of file ziphead.cpp.

**5.24.2.8 string zipios::ZipLocalEntry::getName () const [virtual]**

Returns the full filename of the entry, including a path if the entry is stored in a subfolder.

**Returns:**

the filename of the entry, including path if the entry is stored in a sub-folder.

Reimplemented from [zipios::FileEntry](#).

Definition at line 166 of file ziphead.cpp.

**5.24.2.9 int zipios::ZipLocalEntry::getSize () const [virtual]**

Returns the (uncompressed) size of the entry data.

**Returns:**

Returns the (uncompressed) size of the entry.

Reimplemented from [zipios::FileEntry](#).

Definition at line 183 of file ziphead.cpp.

**5.24.2.10 int zipios::ZipLocalEntry::getTime () const [virtual]**

Returns the date and time of FIXME: what?

**Returns:**

the date and time of the entry.

Reimplemented from [zipios::FileEntry](#).

Definition at line 187 of file ziphead.cpp.

**5.24.2.11 bool zipios::ZipLocalEntry::isDirectory () const [virtual]**

Returns true if the entry is a directory.

A directory entry is an entry which name ends with a separator ('/' for Unix systems, '\\\' for Windows and DOS systems).

**Returns:**

true if the entry is a directory.

Reimplemented from [zipios::FileEntry](#).

Definition at line 196 of file ziphead.cpp.

**5.24.2.12** `bool zipios::ZipLocalEntry::isValid () const` [virtual]

Any method or operator that initializes a [FileEntry](#) may set a flag, that specifies whether the read entry is valid or not.

If it isn't this method returns false.

**Returns:**

true if the [FileEntry](#) has been parsed successfully.

Reimplemented from [zipios::FileEntry](#).

Definition at line 192 of file ziphead.cpp.

**5.24.2.13** `void zipios::ZipLocalEntry::setComment (const string & comment)` [virtual]

Sets the comment field for the [FileEntry](#).

**Parameters:**

*comment* string with the new comment.

Reimplemented from [zipios::FileEntry](#).

Reimplemented in [zipios::ZipCDirEntry](#).

Definition at line 202 of file ziphead.cpp.

**5.24.2.14** `void zipios::ZipLocalEntry::setCompressedSize (int size)` [virtual]

Set the compressed size field of the entry.

**Parameters:**

*size* value to set the compressed size field of the entry to.

Reimplemented from [zipios::FileEntry](#).

Definition at line 206 of file ziphead.cpp.

**5.24.2.15** `void zipios::ZipLocalEntry::setCrc (int crc)` [virtual]

Sets the crc field.

**Parameters:**

*crc* value to set the crc field to.

Reimplemented from [zipios::FileEntry](#).

Definition at line 210 of file ziphead.cpp.

**5.24.2.16** `void zipios::ZipLocalEntry::setExtra (const vector<unsigned char> & extra)`  
[virtual]

Sets the extra field.

**Parameters:**

*extra* the extra field is set to this value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 214 of file ziphead.cpp.

**5.24.2.17 void zipios::ZipLocalEntry::setMethod (StorageMethod *method*) [virtual]**

Sets the storage method field for the entry.

**Parameters:**

*method* the method field is set to the specified value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 219 of file ziphead.cpp.

**5.24.2.18 void zipios::ZipLocalEntry::setName (const string & *name*) [virtual]**

Sets the name field for the entry.

**Parameters:**

*name* the name field is set to the specified value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 223 of file ziphead.cpp.

**5.24.2.19 void zipios::ZipLocalEntry::setSize (int *size*) [virtual]**

Sets the size field for the entry.

**Parameters:**

*size* the size field is set to this value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 228 of file ziphead.cpp.

**5.24.2.20 void zipios::ZipLocalEntry::setTime (int *time*) [virtual]**

Sets the time field for the entry.

**Parameters:**

*time* the time field is set to the specified value.

Reimplemented from [zipios::FileEntry](#).

Definition at line 232 of file ziphead.cpp.

**5.24.2.21** `string zipios::ZipLocalEntry::toString () const` [virtual]

Returns a human-readable string representation of the entry.

**Returns:**

a human-readable string representation of the entry.

Reimplemented from [zipios::FileEntry](#).

Reimplemented in [zipios::ZipCDirEntry](#).

Definition at line 237 of file ziphead.cpp.

The documentation for this class was generated from the following files:

- [ziphead.h](#)
- [ziphead.cpp](#)



# Chapter 6

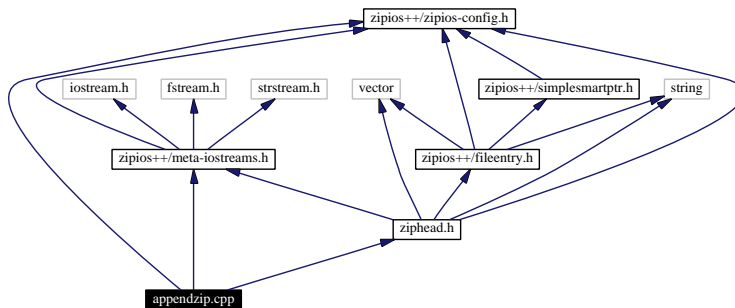
## Zipios++ File Documentation

### 6.1 appendzip.cpp File Reference

Source code to a small program appendzip that appends a zip archive to another file.

```
#include "zipios++/zipios-config.h"  
#include "zipios++/meta-iostreams.h"  
#include "ziphead.h"
```

Include dependency graph for appendzip.cpp:



#### Functions

- void **printUsage** ()
- void **exitUsage** ( int exit\_code )
- int **main** ( int argc, char \*argv[ ] )

#### Variables

- char\* **\_progname**

### 6.1.1 Detailed Description

Source code to a small program `appendzip` that appends a zip archive to another file.

Run `appendzip` without arguments to get a helpful usage message.

Definition in file [appendzip.cpp](#).

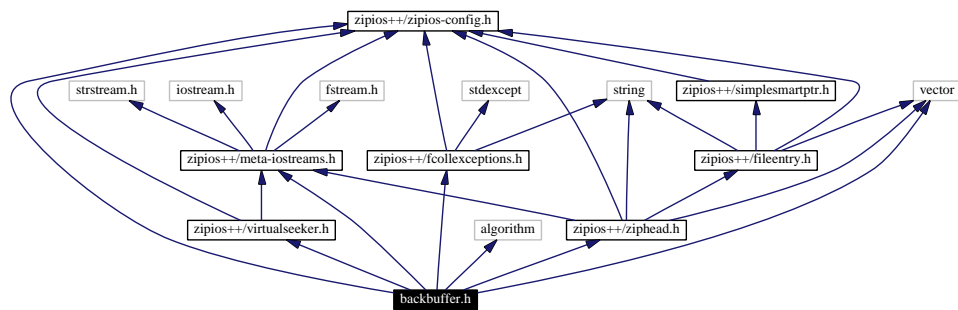


## 6.2 backbuffer.h File Reference

The header file for BackBuffer.

```
#include "zipios++/zipios-config.h"
#include <algorithm>
#include "zipios++/meta-iostreams.h"
#include <vector>
#include "zipios++/fcollexceptions.h"
#include "zipios++/ziphead.h"
#include "zipios++/virtalseeker.h"
```

Include dependency graph for backbuffer.h:



### Namespaces

- namespace **zipios**

### 6.2.1 Detailed Description

The header file for BackBuffer.

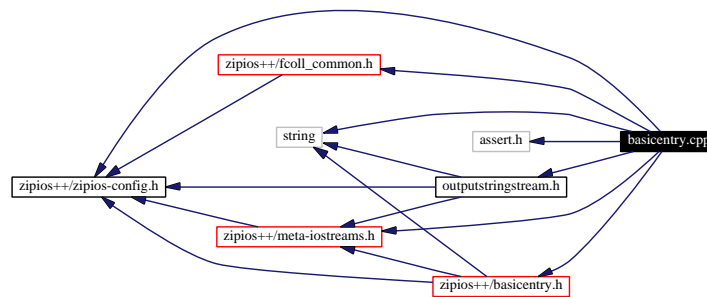
Definition in file [backbuffer.h](#).

## 6.3 basicentry.cpp File Reference

Implementation of BasicEntry.

```
#include "zipios++/zipios-config.h"
#include <assert.h>
#include "zipios++/meta-iostreams.h"
#include <string>
#include "zipios++/fcoll_common.h"
#include "zipios++/basicentry.h"
#include "outputstringstream.h"
```

Include dependency graph for basicentry.cpp:



### Namespaces

- namespace **zipios**

#### 6.3.1 Detailed Description

Implementation of BasicEntry.

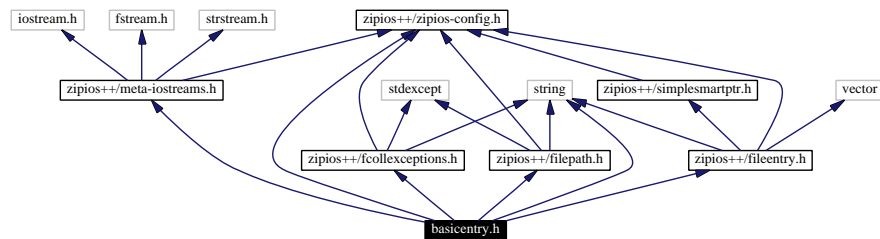
Definition in file [basicentry.cpp](#).

## 6.4 basicentry.h File Reference

Header file that defines BasicEntry.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <string>
#include "zipios++/fcollexceptions.h"
#include "zipios++/fileentry.h"
#include "zipios++/filepath.h"
```

Include dependency graph for basicentry.h:



### Namespaces

- namespace `zipios`

#### 6.4.1 Detailed Description

Header file that defines BasicEntry.

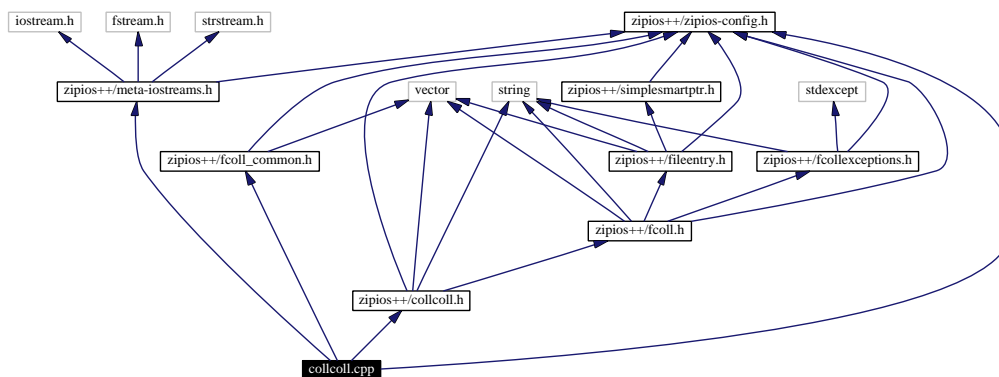
Definition in file [basicentry.h](#).

## 6.5 collcoll.cpp File Reference

Implementation of CollectionCollection.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include "zipios++/collcoll.h"
#include "zipios++/fcoll_common.h"
```

Include dependency graph for collcoll.cpp:



### Namespaces

- namespace **zipios**

### 6.5.1 Detailed Description

Implementation of CollectionCollection.

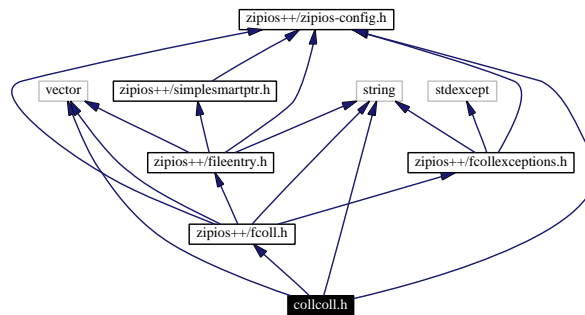
Definition in file [collcoll.cpp](#).

## 6.6 collcoll.h File Reference

Header file that defines CollectionCollection.

```
#include "zipios++/zipios-config.h"  
#include <string>  
#include <vector>  
#include "zipios++/fcoll.h"
```

Include dependency graph for collcoll.h:



### Namespaces

- namespace `zipios`

#### 6.6.1 Detailed Description

Header file that defines CollectionCollection.

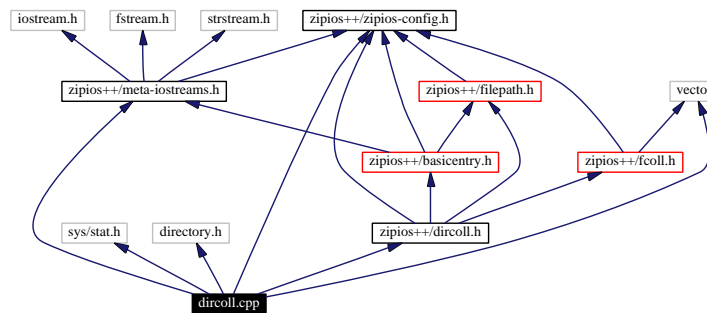
Definition in file [collcoll.h](#).

## 6.7 dircoll.cpp File Reference

Implementation of DirectoryCollection.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <vector>
#include <sys/stat.h>
#include "zipios++/dircoll.h"
#include "directory.h"
```

Include dependency graph for dircoll.cpp:



### Namespaces

- namespace `zipios`

#### 6.7.1 Detailed Description

Implementation of DirectoryCollection.

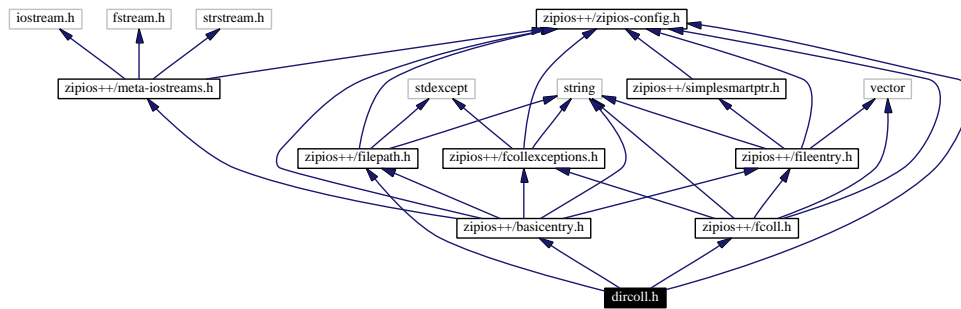
Definition in file [dircoll.cpp](#).

## 6.8 dircoll.h File Reference

Header file that defines DirectoryCollection.

```
#include "zipios++/zipios-config.h"
#include "zipios++/fcoll.h"
#include "zipios++/basicentry.h"
#include "zipios++/filepath.h"
```

Include dependency graph for dircoll.h:



### Namespaces

- namespace `zipios`

#### 6.8.1 Detailed Description

Header file that defines DirectoryCollection.

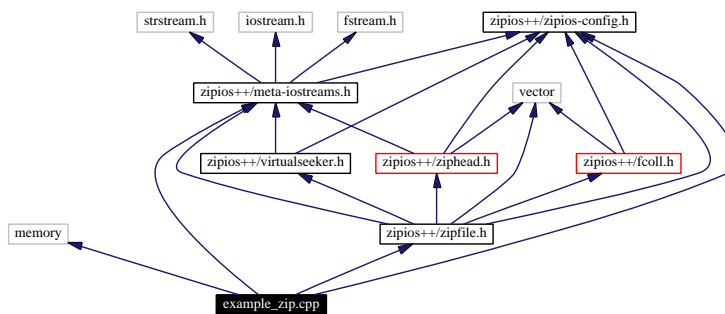
Definition in file [dircoll.h](#).

## 6.9 example\_zip.cpp File Reference

source code to a small program that demonstrates the central elements of Zipios++.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <memory>
#include "zipios++/zipfile.h"
```

Include dependency graph for example\_zip.cpp:



### Functions

- `int main ()`

#### 6.9.1 Detailed Description

source code to a small program that demonstrates the central elements of Zipios++.

Definition in file [example\\_zip.cpp](#).

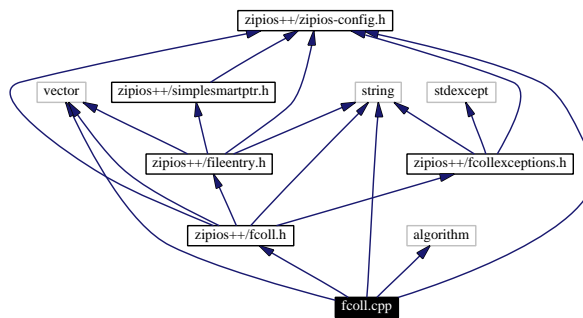


## 6.10 fcoll.cpp File Reference

Implementation of FileCollection.

```
#include "zipios++/zipios-config.h"  
#include <algorithm>  
#include <string>  
#include <vector>  
#include "zipios++/fcoll.h"
```

Include dependency graph for fcoll.cpp:



### Namespaces

- namespace `zipios`

#### 6.10.1 Detailed Description

Implementation of FileCollection.

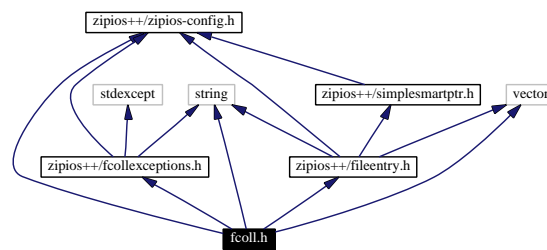
Definition in file [fcoll.cpp](#).

## 6.11 fcoll.h File Reference

Header file that defines FileCollection.

```
#include "zipios++/zipios-config.h"
#include <vector>
#include <string>
#include "zipios++/fcollexceptions.h"
#include "zipios++/fileentry.h"
```

Include dependency graph for fcoll.h:



### Namespaces

- namespace **zipios**

#### 6.11.1 Detailed Description

Header file that defines FileCollection.

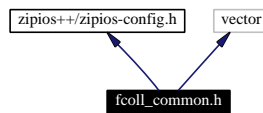
Definition in file [fcoll.h](#).

## 6.12 fcoll\_common.h File Reference

Header file that doesn't really contain much!

```
#include "zipios++/zipios-config.h"  
#include <vector>
```

Include dependency graph for fcoll\_common.h:



### Namespaces

- namespace `zipios`

#### 6.12.1 Detailed Description

Header file that doesn't really contain much!

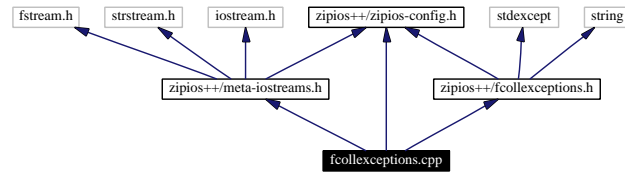
Definition in file [fcoll\\_common.h](#).

## 6.13 fcollexceptions.cpp File Reference

Implementation of a number of Exceptions used by FileCollection and its subclasses.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include "zipios++/fcollexceptions.h"
```

Include dependency graph for fcollexceptions.cpp:



### Namespaces

- namespace **zipios**

#### 6.13.1 Detailed Description

Implementation of a number of Exceptions used by FileCollection and its subclasses.

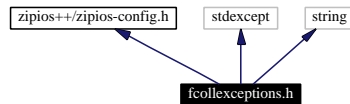
Definition in file [fcollexceptions.cpp](#).

## 6.14 fcollexceptions.h File Reference

Header file that defines a number of exceptions used by FileCollection and its subclasses.

```
#include "zipios++/zipios-config.h"  
#include <stdexcept>  
#include <string>
```

Include dependency graph for fcollexceptions.h:



### Namespaces

- namespace `zipios`

#### 6.14.1 Detailed Description

Header file that defines a number of exceptions used by FileCollection and its subclasses.

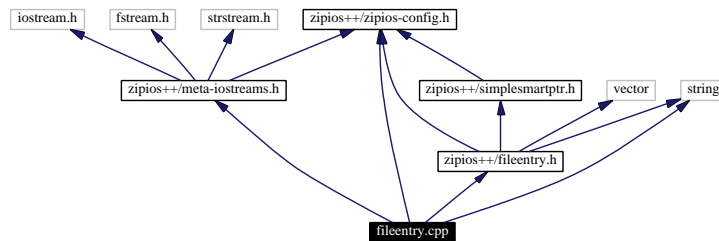
Definition in file [fcollexceptions.h](#).

## 6.15 fileentry.cpp File Reference

Implementation of FileEntry.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <string>
#include "zipios++/fileentry.h"
```

Include dependency graph for fileentry.cpp:



### Namespaces

- namespace **zipios**

#### 6.15.1 Detailed Description

Implementation of FileEntry.

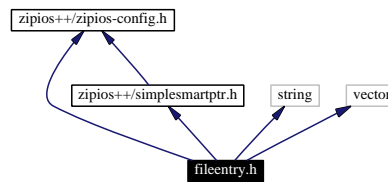
Definition in file [fileentry.cpp](#).

## 6.16 fileentry.h File Reference

Header file that defines FileEntry.

```
#include "zipios++/zipios-config.h"
#include <string>
#include <vector>
#include "zipios++/simplesmartptr.h"
```

Include dependency graph for fileentry.h:



### Namespaces

- namespace **zipios**

#### 6.16.1 Detailed Description

Header file that defines FileEntry.

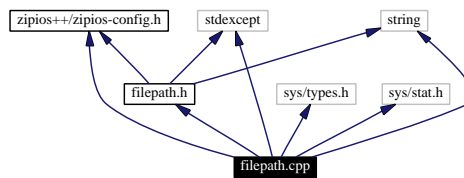
Definition in file [fileentry.h](#).

## 6.17 filepath.cpp File Reference

Implementation of FilePath.

```
#include "zipios++/zipios-config.h"
#include <stdexcept>
#include <string>
#include <sys/types.h>
#include <sys/stat.h>
#include "filepath.h"
```

Include dependency graph for filepath.cpp:



### Namespaces

- namespace `zipios`

#### 6.17.1 Detailed Description

Implementation of FilePath.

Definition in file [filepath.cpp](#).

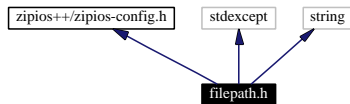


## 6.18 filepath.h File Reference

Header file that defines FilePath.

```
#include "zipios++/zipios-config.h"  
#include <stdexcept>  
#include <string>
```

Include dependency graph for filepath.h:



### Namespaces

- namespace `std`
- namespace `zipios`

### 6.18.1 Detailed Description

Header file that defines FilePath.

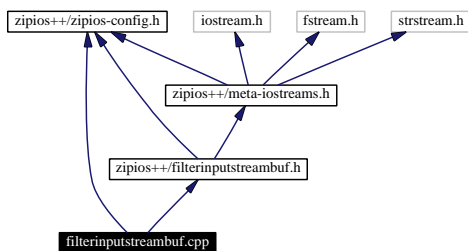
Definition in file [filepath.h](#).

## 6.19 filterinputstreambuf.cpp File Reference

Implementation of FilterInputStreambuf.

```
#include "zipios++/zipios-config.h"  
#include "zipios++/filterinputstreambuf.h"
```

Include dependency graph for filterinputstreambuf.cpp:



### Namespaces

- namespace `zipios`

#### 6.19.1 Detailed Description

Implementation of FilterInputStreambuf.

Definition in file [filterinputstreambuf.cpp](#).

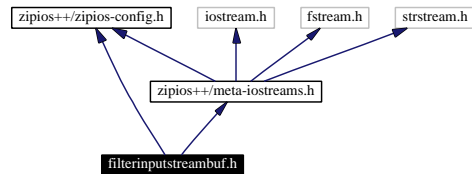
## 6.20 filterinputstreambuf.h File Reference

Header file that defines `FilterInputStreambuf`.

```
#include "zipios++/zipios-config.h"
```

```
#include "zipios++/meta-iostreams.h"
```

Include dependency graph for `filterinputstreambuf.h`:



### Namespaces

- namespace `zipios`

#### 6.20.1 Detailed Description

Header file that defines `FilterInputStreambuf`.

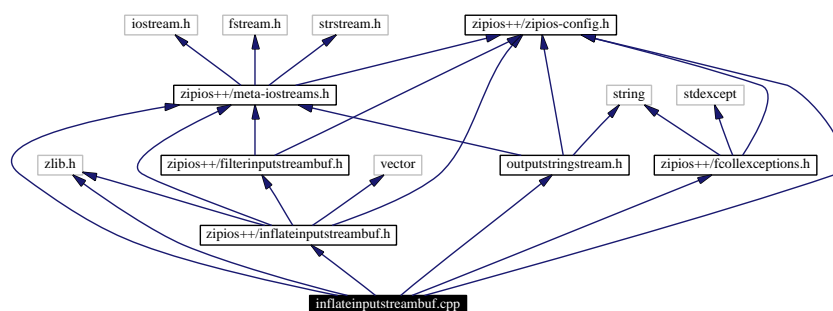
Definition in file [filterinputstreambuf.h](#).

## 6.21 inflateinputstreambuf.cpp File Reference

Implementation of InflateInputStreambuf.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <zlib.h>
#include "zipios++/fcollexceptions.h"
#include "zipios++/inflateinputstreambuf.h"
#include "outputstringstream.h"
```

Include dependency graph for inflateinputstreambuf.cpp:



### Namespaces

- namespace **zipios**

#### 6.21.1 Detailed Description

Implementation of InflateInputStreambuf.

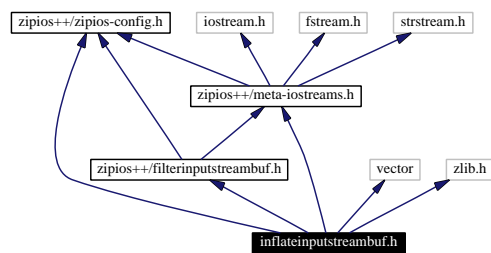
Definition in file [inflateinputstreambuf.cpp](#).

## 6.22 inflateinputstreambuf.h File Reference

Header file that defines InflateInputStreambuf.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <vector>
#include <zlib.h>
#include "zipios++/filterinputstreambuf.h"
```

Include dependency graph for inflateinputstreambuf.h:



### Namespaces

- namespace `zipios`

### 6.22.1 Detailed Description

Header file that defines InflateInputStreambuf.

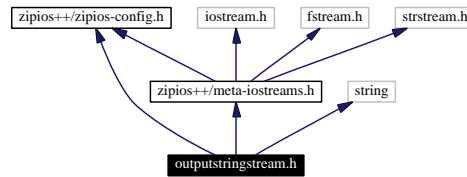
Definition in file [inflateinputstreambuf.h](#).

## 6.23 outputstringstream.h File Reference

Header file that defines OutputStringStream.

```
#include "zipios++/zipios-config.h"  
#include "zipios++/meta-iostreams.h"  
#include <string>
```

Include dependency graph for outputstringstream.h:



### Namespaces

- namespace `zipios`

#### 6.23.1 Detailed Description

Header file that defines OutputStringStream.

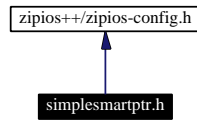
Definition in file [outputstringstream.h](#).

## 6.24 simplesmartptr.h File Reference

Header file that defines SimpleSmartPointer.

```
#include "zipios++/zipios-config.h"
```

Include dependency graph for simplesmartptr.h:



### Namespaces

- namespace `zipios`

#### 6.24.1 Detailed Description

Header file that defines SimpleSmartPointer.

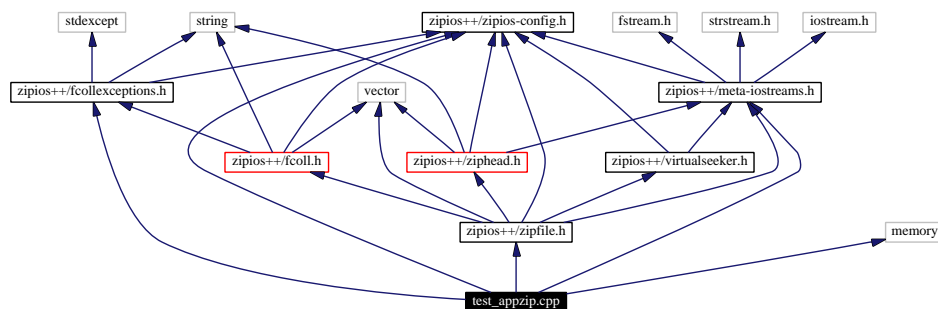
Definition in file [simplesmartptr.h](#).

## 6.25 test\_appzip.cpp File Reference

source code to a small program that demonstrates the central elements of Zipios++.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <memory>
#include "zipios++/fcollexceptions.h"
#include "zipios++/zipfile.h"
```

Include dependency graph for test\_appzip.cpp:



### Functions

- `int main ( int argc, char *argv[ ] )`

#### 6.25.1 Detailed Description

source code to a small program that demonstrates the central elements of Zipios++.

Definition in file [test\\_appzip.cpp](#).

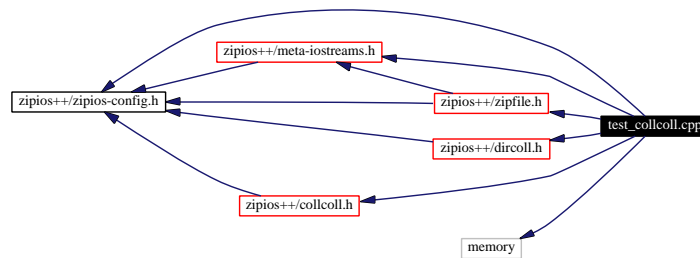


## 6.26 test\_collcoll.cpp File Reference

source code to a small program that demonstrates the central elements of Zipios++.

```
#include "zipios++/zipios-config.h"  
#include "zipios++/meta-iostreams.h"  
#include <memory>  
#include "zipios++/dircoll.h"  
#include "zipios++/zipfile.h"  
#include "zipios++/collcoll.h"
```

Include dependency graph for test\_collcoll.cpp:



### Functions

- int main ()

#### 6.26.1 Detailed Description

source code to a small program that demonstrates the central elements of Zipios++.

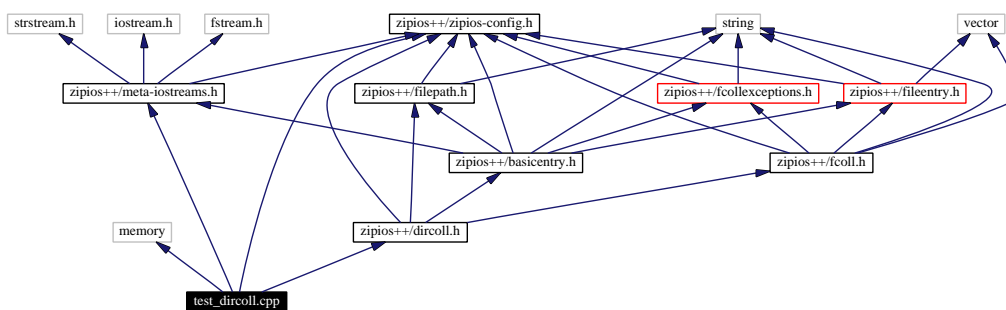
Definition in file [test\\_collcoll.cpp](#).

## 6.27 test\_dircoll.cpp File Reference

source code to a small program that demonstrates the central elements of Zipios++.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <memory>
#include "zipios++/dircoll.h"
```

Include dependency graph for test\_dircoll.cpp:



### Functions

- `int main ()`

#### 6.27.1 Detailed Description

source code to a small program that demonstrates the central elements of Zipios++.

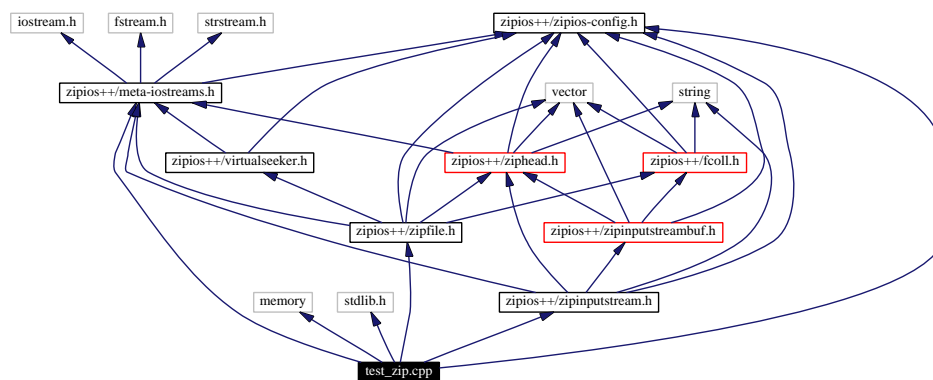
Definition in file [test\\_dircoll.cpp](#).

## 6.28 test\_zip.cpp File Reference

Source code to a small program that tests the functionality of Zipios++.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <memory>
#include <stdlib.h>
#include "zipios++/zipfile.h"
#include "zipios++/zipinputstream.h"
```

Include dependency graph for test\_zip.cpp:



### Functions

- void **entryToFile** ( const string &ent\_name, istream &is, const string &outfile, bool cerr\_report = true )
- int **main** ()

### 6.28.1 Detailed Description

Source code to a small program that tests the functionality of Zipios++.

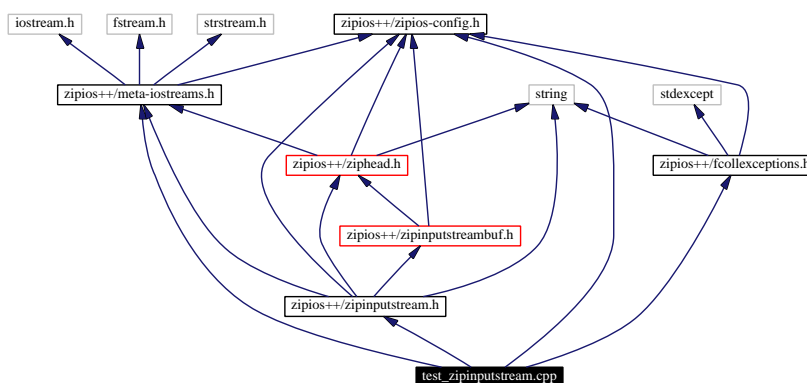
Definition in file [test\\_zip.cpp](#).

## 6.29 test\_zipinputstream.cpp File Reference

Source for a test program for testing ZipInputStream.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include "zipios++/fcollexceptions.h"
#include "zipios++/zipinputstream.h"
```

Include dependency graph for test\_zipinputstream.cpp:



### Functions

- `int main ()`

#### 6.29.1 Detailed Description

Source for a test program for testing ZipInputStream.

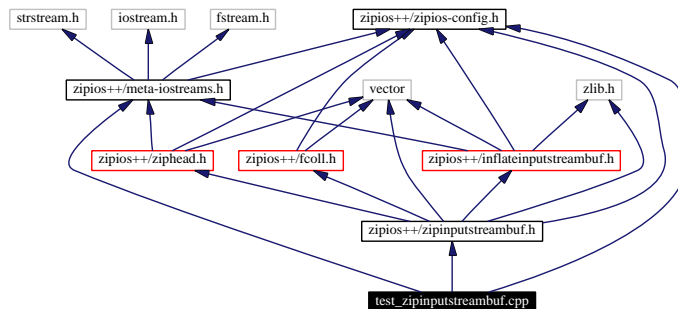
Definition in file [test\\_zipinputstream.cpp](#).

## 6.30 test\_zipinputstreambuf.cpp File Reference

Source for a test program for testing ZipInputStreambuf.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include "zipios++/zipinputstreambuf.h"
```

Include dependency graph for test\_zipinputstreambuf.cpp:



### Functions

- `int main ()`

#### 6.30.1 Detailed Description

Source for a test program for testing ZipInputStreambuf.

Definition in file [test\\_zipinputstreambuf.cpp](#).

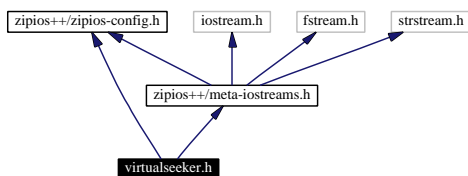
## 6.31 virtualeseeker.h File Reference

Header file that defines VirtualSeeker.

```
#include "zipios++/zipios-config.h"
```

```
#include "zipios++/meta-iostreams.h"
```

Include dependency graph for virtualeseeker.h:



### Namespaces

- namespace `zipios`

#### 6.31.1 Detailed Description

Header file that defines VirtualSeeker.

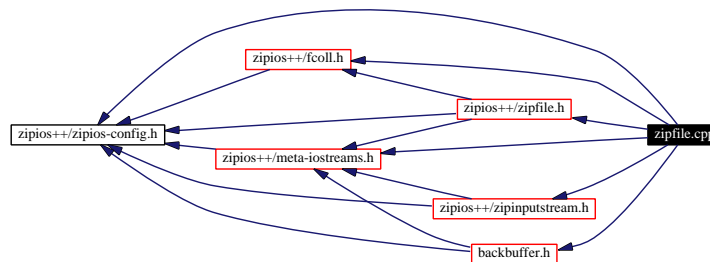
Definition in file [virtualeseeker.h](#).

## 6.32 zipfile.cpp File Reference

The implementation of ZipFile.

```
#include "zipios++/zipios-config.h"  
#include "zipios++/meta-iostreams.h"  
#include "zipios++/fcoll.h"  
#include "zipios++/zipfile.h"  
#include "zipios++/zipinputstream.h"  
#include "backbuffer.h"
```

Include dependency graph for zipfile.cpp:



### Namespaces

- namespace `zipios`

### 6.32.1 Detailed Description

The implementation of ZipFile.

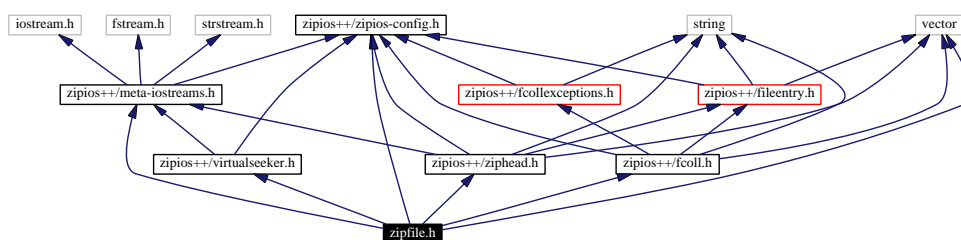
Definition in file [zipfile.cpp](#).

## 6.33 zipfile.h File Reference

Header file that defines ZipFile.

```
#include "zipios++/zipios-config.h"
#include <vector>
#include "zipios++/meta-iostreams.h"
#include "zipios++/fcoll.h"
#include "zipios++/ziphead.h"
#include "zipios++/virtalseeker.h"
```

Include dependency graph for zipfile.h:



### Namespaces

- namespace `zipios`

#### 6.33.1 Detailed Description

Header file that defines ZipFile.

Definition in file [zipfile.h](#).

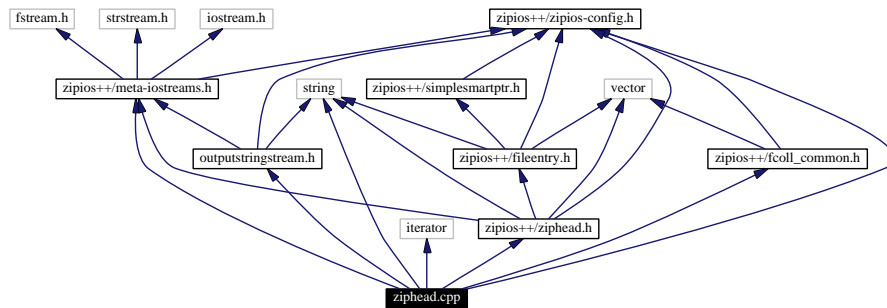


## 6.34 ziphead.cpp File Reference

Implementation of routines for reading the central directory and local headers of a zip archive.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <iterator>
#include <string>
#include "zipios++/fcoll_common.h"
#include "zipios++/ziphead.h"
#include "outputstringstream.h"
```

Include dependency graph for ziphead.cpp:



### Namespaces

- namespace **zipios**

#### 6.34.1 Detailed Description

Implementation of routines for reading the central directory and local headers of a zip archive.

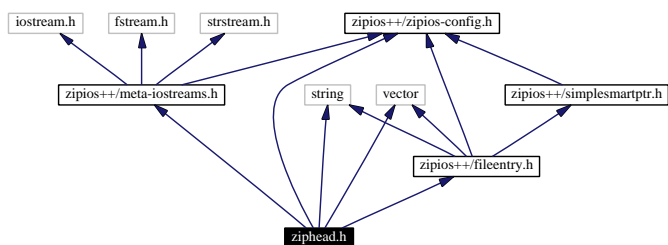
Definition in file [ziphead.cpp](#).

## 6.35 ziphead.h File Reference

Header file containing classes and functions for reading the central directory and local header fields in a zip archive.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <string>
#include <vector>
#include "zipios++/fileentry.h"
```

Include dependency graph for ziphead.h:



### Namespaces

- namespace **zipios**

#### 6.35.1 Detailed Description

Header file containing classes and functions for reading the central directory and local header fields in a zip archive.

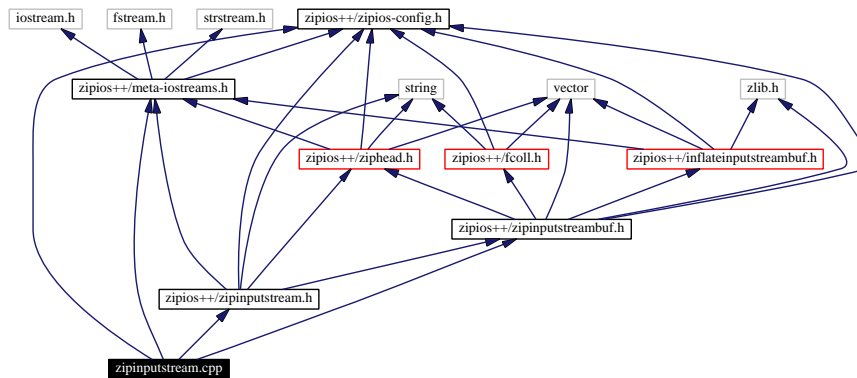
Definition in file [ziphead.h](#).

## 6.36 zipinputstream.cpp File Reference

Implementation of ZipInputStream.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include "zipios++/zipinputstreambuf.h"
#include "zipios++/zipinputstream.h"
```

Include dependency graph for zipinputstream.cpp:



### Namespaces

- namespace `zipios`

### 6.36.1 Detailed Description

Implementation of ZipInputStream.

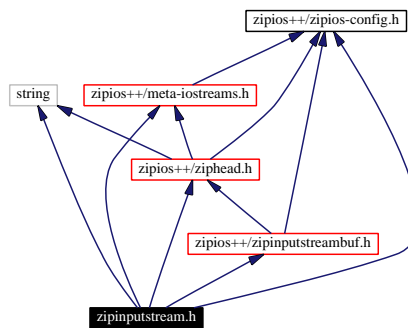
Definition in file [zipinputstream.cpp](#).

## 6.37 zipinputstream.h File Reference

Header file that defines ZipInputStream.

```
#include "zipios++/zipios-config.h"
#include "zipios++/meta-iostreams.h"
#include <string>
#include "zipios++/ziphead.h"
#include "zipios++/zipinputstreambuf.h"
```

Include dependency graph for zipinputstream.h:



### Namespaces

- namespace `zipios`

#### 6.37.1 Detailed Description

Header file that defines ZipInputStream.

Definition in file [zipinputstream.h](#).

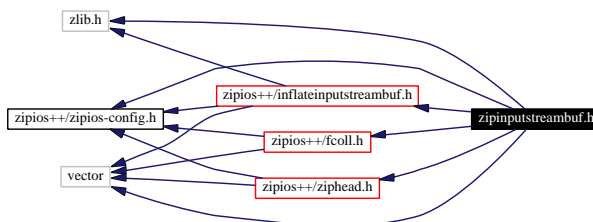


## 6.39 zipinputstreambuf.h File Reference

Header file that defines ZipInputStreambuf.

```
#include "zipios++/zipios-config.h"
#include <vector>
#include <zlib.h>
#include "zipios++/fcoll.h"
#include "zipios++/inflateinputstreambuf.h"
#include "zipios++/ziphead.h"
```

Include dependency graph for zipinputstreambuf.h:



### Namespaces

- namespace **zipios**

#### 6.39.1 Detailed Description

Header file that defines ZipInputStreambuf.

Definition in file [zipinputstreambuf.h](#).

# Index

- ~BasicEntry
  - zipios::BasicEntry, 14
- ~CollectionCollection
  - zipios::CollectionCollection, 22
- ~DirectoryCollection
  - zipios::DirectoryCollection, 29
- ~FCollException
  - zipios::FCollException, 34
- ~FileCollection
  - zipios::FileCollection, 36
- ~FileEntry
  - zipios::FileEntry, 42
- ~FilterInputStreambuf
  - zipios::FilterInputStreambuf, 52
- ~IOException
  - zipios::IOException, 56
- ~InflateInputStreambuf
  - zipios::InflateInputStreambuf, 53
- ~InvalidStateException
  - zipios::InvalidStateException, 55
- ~SimpleSmartPointer
  - zipios::SimpleSmartPointer, 60
- ~ZipCDirEntry
  - zipios::ZipCDirEntry, 62
- ~ZipFile
  - zipios::ZipFile, 65
- ~ZipInputStream
  - zipios::ZipInputStream, 69
- ~ZipInputStreambuf
  - zipios::ZipInputStreambuf, 71
- ~ZipLocalEntry
  - zipios::ZipLocalEntry, 73
- .\_basepath
  - zipios::BasicEntry, 14
- .\_checked
  - zipios::FilePath, 47
- .\_collections
  - zipios::CollectionCollection, 22
- .\_comment
  - zipios::BasicEntry, 14
- .\_del\_inbuf
  - zipios::FilterInputStreambuf, 51
- .\_entries
  - zipios::FileCollection, 36
- .\_entries\_loaded
  - zipios::DirectoryCollection, 29
- .\_exists
  - zipios::FilePath, 47
- .\_filename
  - zipios::BasicEntry, 14
  - zipios::FileCollection, 36
- .\_filepath
  - zipios::DirectoryCollection, 29
- .\_inbuf
  - zipios::FilterInputStreambuf, 51
- .\_is\_block
  - zipios::FilePath, 47
- .\_is\_char
  - zipios::FilePath, 47
- .\_is\_dir
  - zipios::FilePath, 47
- .\_is\_fifo
  - zipios::FilePath, 47
- .\_is\_reg
  - zipios::FilePath, 47
- .\_is\_socket
  - zipios::FilePath, 47
- .\_outvec
  - zipios::InflateInputStreambuf, 53
- .\_outvecsize
  - zipios::InflateInputStreambuf, 53
- .\_path
  - zipios::FilePath, 47
- .\_progname
  - appendzip.cpp, 81
- .\_recursive
  - zipios::DirectoryCollection, 29
- .\_ref\_count
  - zipios::FileEntry, 42
- .\_s\_pos
  - zipios::FilterInputStreambuf, 51
- .\_separator
  - zipios::FilePath, 48
- .\_size
  - zipios::BasicEntry, 14
- .\_valid
  - zipios::BasicEntry, 14
  - zipios::FileCollection, 36
  - zipios::ZipLocalEntry, 74

- addCollection
  - zipios::CollectionCollection, 23
- appendzip.cpp, 81
  - \_progrname, 81
  - exitUsage, 81
  - main, 81
  - printUsage, 81
- available
  - zipios::ZipInputStream, 68
- BackBuffer
  - zipios::BackBuffer, 12
- backbuffer.h, 83
- BasicEntry
  - zipios::BasicEntry, 15
- basicentry.cpp, 84
- basicentry.h, 85
- check
  - zipios::FilePath, 48
- clone
  - zipios::BasicEntry, 15
  - zipios::CollectionCollection, 23
  - zipios::DirectoryCollection, 29
  - zipios::FileCollection, 37
  - zipios::FileEntry, 42
  - zipios::ZipCDirEntry, 63
  - zipios::ZipFile, 65
  - zipios::ZipLocalEntry, 74
- close
  - zipios::CollectionCollection, 23
  - zipios::DirectoryCollection, 30
  - zipios::FileCollection, 37
  - zipios::ZipFile, 66
  - zipios::ZipInputStream, 69
  - zipios::ZipInputSteambuf, 71
- closeEntry
  - zipios::ZipInputStream, 69
  - zipios::ZipInputSteambuf, 71
- collcoll.cpp, 86
- collcoll.h, 87
- CollectionCollection
  - zipios::CollectionCollection, 22
- compress\_method
  - zipios::ZipLocalEntry, 73
- compress\_size
  - zipios::DataDescriptor, 27
  - zipios::ZipLocalEntry, 73
- crc\_32
  - zipios::DataDescriptor, 27
  - zipios::ZipLocalEntry, 73
- dircoll.cpp, 88
- dircoll.h, 89
- DirectoryCollection
  - zipios::DirectoryCollection, 29
- endOffset
  - zipios::VirtualSeeker, 61
- entries
  - zipios::CollectionCollection, 23
  - zipios::DirectoryCollection, 30
  - zipios::FileCollection, 37
- entryToFile
  - test.zip.cpp, 109
- eocdOffSetFromEnd
  - zipios::EndOfCentralDirectory, 33
- example.zip.cpp, 90
  - main, 90
- exists
  - zipios::FilePath, 48
- exitUsage
  - appendzip.cpp, 81
- extra\_field
  - zipios::ZipLocalEntry, 74
- extra\_field\_len
  - zipios::ZipLocalEntry, 74
- extract\_version
  - zipios::ZipLocalEntry, 73
- fcoll.cpp, 91
- fcoll.h, 92
- fcoll\_common.h, 93
- FCollException
  - zipios::FCollException, 34
- fcollexceptions.cpp, 94
- fcollexceptions.h, 95
- FileCollection
  - zipios::FileCollection, 36
- FileEntry
  - zipios::FileEntry, 42
- fileentry.cpp, 96
- fileentry.h, 97
- filename
  - zipios::FilePath, 48
  - zipios::ZipLocalEntry, 74
- filename\_len
  - zipios::ZipLocalEntry, 74
- FilePath
  - zipios::FilePath, 48
- filepath.cpp, 98
- filepath.h, 99
- FilterInputSteambuf
  - zipios::FilterInputSteambuf, 51
- filterinputstreambuf.cpp, 100
- filterinputstreambuf.h, 101
- get



- zipios::SimpleSmartPointer, 60
- getComment
  - zipios::BasicEntry, 15
  - zipios::FileEntry, 42
  - zipios::ZipCDirEntry, 63
  - zipios::ZipLocalEntry, 74
- getCompressedSize
  - zipios::BasicEntry, 15
  - zipios::FileEntry, 43
  - zipios::ZipLocalEntry, 74
- getCrc
  - zipios::BasicEntry, 16
  - zipios::FileEntry, 43
  - zipios::ZipLocalEntry, 75
- getEntry
  - zipios::CollectionCollection, 24
  - zipios::DirectoryCollection, 30
  - zipios::FileCollection, 37
- getExtra
  - zipios::BasicEntry, 16
  - zipios::FileEntry, 43
  - zipios::ZipLocalEntry, 75
- getFileName
  - zipios::BasicEntry, 16
  - zipios::FileEntry, 43
  - zipios::ZipLocalEntry, 75
- getInputStream
  - zipios::CollectionCollection, 24, 25
  - zipios::DirectoryCollection, 31
  - zipios::FileCollection, 38
  - zipios::ZipFile, 66
- getLocalHeaderOffset
  - zipios::ZipCDirEntry, 62
- getMethod
  - zipios::BasicEntry, 16
  - zipios::FileEntry, 43
  - zipios::ZipLocalEntry, 75
- getName
  - zipios::BasicEntry, 17
  - zipios::FileCollection, 38
  - zipios::FileEntry, 44
  - zipios::ZipLocalEntry, 76
- getNextEntry
  - zipios::ZipInputStream, 69
  - zipios::ZipInputStreambuf, 71
- getOffsets
  - zipios::VirtualSeeker, 61
- getSize
  - zipios::BasicEntry, 17
  - zipios::FileEntry, 44
  - zipios::ZipLocalEntry, 76
- getTime
  - zipios::BasicEntry, 17
  - zipios::FileEntry, 44
- zipios::ZipLocalEntry, 76
- gp\_bitfield
  - zipios::ZipLocalEntry, 73
- InflateInputStreambuf
  - zipios::InflateInputStreambuf, 54
- inflateinputstreambuf.cpp, 102
- inflateinputstreambuf.h, 103
- inst
  - zipios::CollectionCollection, 25
- InvalidStateException
  - zipios::InvalidStateException, 55
- IOException
  - zipios::IOException, 56
- isBlockSpecial
  - zipios::FilePath, 49
- isCharSpecial
  - zipios::FilePath, 49
- isDirectory
  - zipios::BasicEntry, 17
  - zipios::FileEntry, 44
  - zipios::FilePath, 49
  - zipios::ZipLocalEntry, 76
- isFifo
  - zipios::FilePath, 49
- isRegular
  - zipios::FilePath, 49
- isSocket
  - zipios::FilePath, 49
- isValid
  - zipios::BasicEntry, 17
  - zipios::FileCollection, 39
  - zipios::FileEntry, 44
  - zipios::ZipLocalEntry, 76
- last\_mod\_fdate
  - zipios::ZipLocalEntry, 73
- last\_mod\_mtime
  - zipios::ZipLocalEntry, 73
- Load
  - zipios::DirectoryCollection, 29
- loadEntries
  - zipios::DirectoryCollection, 29
- main
  - appendzip.cpp, 81
  - example\_zip.cpp, 90
  - test\_appzip.cpp, 106
  - test\_collcoll.cpp, 107
  - test\_dircoll.cpp, 108
  - test\_zip.cpp, 109
  - test\_zipinputstream.cpp, 110
  - test\_zipinputstreambuf.cpp, 111
- MatchFileName

- zipios::FileEntry::MatchFileName, 57
- MatchName
  - zipios::FileEntry::MatchName, 58
- offset
  - zipios::EndOfCentralDirectory, 33
- openEmbeddedZipFile
  - zipios::ZipFile, 67
- operator \*
  - zipios::SimpleSmartPointer, 60
- operator string
  - zipios::FilePath, 47
- operator void \*
  - zipios::SimpleSmartPointer, 60
- operator()
  - zipios::FileEntry::MatchFileName, 57
  - zipios::FileEntry::MatchName, 58
- operator+
  - zipios::FilePath, 49
- operator →
  - zipios::SimpleSmartPointer, 60
- operator=
  - zipios::CollectionCollection, 25
  - zipios::FCollException, 34
  - zipios::FileCollection, 39
  - zipios::FileEntry, 45
  - zipios::FilePath, 47
  - zipios::InvalidStateException, 55
  - zipios::IOException, 56
  - zipios::SimpleSmartPointer, 60
  - zipios::ZipCDirEntry, 62
  - zipios::ZipLocalEntry, 72
- operator==
  - zipios::SimpleSmartPointer, 60
- outputstringstream.h, 104
- printUsage
  - appendzip.cpp, 81
- pruneTrailingSeparator
  - zipios::FilePath, 50
- read
  - zipios::EndOfCentralDirectory, 33
- readChunk
  - zipios::BackBuffer, 12
- ref
  - zipios::FileEntry, 41
- reset
  - zipios::InflateInputStreambuf, 54
- setComment
  - zipios::BasicEntry, 18
  - zipios::FileEntry, 45
  - zipios::ZipCDirEntry, 63
  - zipios::ZipLocalEntry, 77
- setCompressedSize
  - zipios::BasicEntry, 18
  - zipios::FileEntry, 45
  - zipios::ZipLocalEntry, 77
- setCrc
  - zipios::BasicEntry, 18
  - zipios::FileEntry, 45
  - zipios::ZipLocalEntry, 77
- setExtra
  - zipios::BasicEntry, 18
  - zipios::FileEntry, 45
  - zipios::ZipLocalEntry, 77
- setMethod
  - zipios::BasicEntry, 18
  - zipios::FileEntry, 46
  - zipios::ZipLocalEntry, 78
- setName
  - zipios::BasicEntry, 19
  - zipios::FileEntry, 46
  - zipios::ZipLocalEntry, 78
- setOffsets
  - zipios::VirtualSeeker, 61
- setSize
  - zipios::BasicEntry, 19
  - zipios::FileEntry, 46
  - zipios::ZipLocalEntry, 78
- setTime
  - zipios::BasicEntry, 19
  - zipios::FileEntry, 46
  - zipios::ZipLocalEntry, 78
- signature
  - zipios::ZipLocalEntry, 73
- SimpleSmartPointer
  - zipios::SimpleSmartPointer, 60
- simplesmartptr.h, 105
- size
  - zipios::CollectionCollection, 25
  - zipios::DirectoryCollection, 31
  - zipios::FileCollection, 39
- startOffset
  - zipios::VirtualSeeker, 61
- str
  - zipios::OutputStringStream, 59
- test\_appzip.cpp, 106
  - main, 106
- test\_collcoll.cpp, 107
  - main, 107
- test\_dircoll.cpp, 108
  - main, 108
- test\_zip.cpp, 109
  - entryToFile, 109
  - main, 109

- test\_zipinputstream.cpp, 110
  - main, 110
- test\_zipinputstreambuf.cpp, 111
  - main, 111
- toString
  - zipios::BasicEntry, 19
  - zipios::FileEntry, 46
  - zipios::ZipCDirEntry, 63
  - zipios::ZipLocalEntry, 78
- totalCount
  - zipios::EndOfCentralDirectory, 33
- trailingDataDescriptor
  - zipios::ZipLocalEntry, 73
- uncompress\_size
  - zipios::DataDescriptor, 27
  - zipios::ZipLocalEntry, 74
- underflow
  - zipios::InflateInputSteambuf, 53
  - zipios::ZipInputSteambuf, 70
- unref
  - zipios::FileEntry, 41
- VirtualSeeker
  - zipios::VirtualSeeker, 61
- virtalseeker.h, 112
- vseekg
  - zipios::VirtualSeeker, 61
- vtellg
  - zipios::VirtualSeeker, 61
- what
  - zipios::FCollException, 34
  - zipios::InvalidStateException, 55
  - zipios::IOException, 56
- ZipCDirEntry
  - zipios::ZipCDirEntry, 62
- ZipFile
  - zipios::ZipFile, 65
- zipfile.cpp, 113
- zipfile.h, 114
- ziphead.cpp, 115
- ziphead.h, 116
- ZipInputStream
  - zipios::ZipInputStream, 68
- zipinputstream.cpp, 117
- zipinputstream.h, 118
- ZipInputSteambuf
  - zipios::ZipInputSteambuf, 71
- zipinputstreambuf.cpp, 119
- zipinputstreambuf.h, 120
- zipios::BackBuffer, 11
  - BackBuffer, 12
  - readChunk, 12
- zipios::BasicEntry, 13
  - ~BasicEntry, 14
  - \_basepath, 14
  - \_comment, 14
  - \_filename, 14
  - \_size, 14
  - \_valid, 14
  - BasicEntry, 15
  - clone, 15
  - getComment, 15
  - getCompressedSize, 15
  - getCrc, 16
  - getExtra, 16
  - getFileName, 16
  - getMethod, 16
  - getName, 17
  - getSize, 17
  - getTime, 17
  - isDirectory, 17
  - isValid, 17
  - setComment, 18
  - setCompressedSize, 18
  - setCrc, 18
  - setExtra, 18
  - setMethod, 18
  - setName, 19
  - setSize, 19
  - setTime, 19
  - toString, 19
- zipios::CollectionCollection, 21
  - ~CollectionCollection, 22
  - \_collections, 22
  - addCollection, 23
  - clone, 23
  - close, 23
  - CollectionCollection, 22
  - entries, 23
  - getEntry, 24
  - getInputStream, 24, 25
  - inst, 25
  - operator=, 25
  - size, 25
- zipios::DataDescriptor, 27
  - compress\_size, 27
  - crc\_32, 27
  - uncompress\_size, 27
- zipios::DirectoryCollection, 28
  - ~DirectoryCollection, 29
  - \_entries\_loaded, 29
  - \_filepath, 29
  - \_recursive, 29
  - clone, 29
  - close, 30

- DirectoryCollection, 29
- entries, 30
- getEntry, 30
- getInputStream, 31
- Load, 29
- loadEntries, 29
- size, 31
- zipios::EndOfCentralDirectory, 33
  - eocdOffSetFromEnd, 33
  - offset, 33
  - read, 33
  - totalCount, 33
- zipios::FCollException, 34
  - ~FCollException, 34
  - FCollException, 34
  - operator=, 34
  - what, 34
- zipios::FileCollection, 35
  - ~FileCollection, 36
  - \_entries, 36
  - \_filename, 36
  - \_valid, 36
  - clone, 37
  - close, 37
  - entries, 37
  - FileCollection, 36
  - getEntry, 37
  - getInputStream, 38
  - getName, 38
  - isValid, 39
  - operator=, 39
  - size, 39
- zipios::FileEntry, 40
  - ~FileEntry, 42
  - \_ref\_count, 42
  - clone, 42
  - FileEntry, 42
  - getComment, 42
  - getCompressedSize, 43
  - getCrc, 43
  - getExtra, 43
  - getFileName, 43
  - getMethod, 43
  - getName, 44
  - getSize, 44
  - getTime, 44
  - isDirectory, 44
  - isValid, 44
  - operator=, 45
  - ref, 41
  - setComment, 45
  - setCompressedSize, 45
  - setCrc, 45
  - setExtra, 45
  - setMethod, 46
  - setName, 46
  - setSize, 46
  - setTime, 46
  - toString, 46
  - unref, 41
- zipios::FileEntry::MatchFileName, 57
  - MatchFileName, 57
  - operator(), 57
- zipios::FileEntry::MatchName, 58
  - MatchName, 58
  - operator(), 58
- zipios::FilePath, 47
  - \_checked, 47
  - \_exists, 47
  - \_is\_block, 47
  - \_is\_char, 47
  - \_is\_dir, 47
  - \_is\_fifo, 47
  - \_is\_reg, 47
  - \_is\_socket, 47
  - \_path, 47
  - \_separator, 48
  - check, 48
  - exists, 48
  - filename, 48
  - FilePath, 48
  - isBlockSpecial, 49
  - isCharSpecial, 49
  - isDirectory, 49
  - isFifo, 49
  - isRegular, 49
  - isSocket, 49
  - operator string, 47
  - operator+, 49
  - operator=, 47
  - pruneTrailingSeparator, 50
- zipios::FilterInputStreambuf, 51
  - ~FilterInputStreambuf, 52
  - \_del\_inbuf, 51
  - \_inbuf, 51
  - \_s\_pos, 51
  - FilterInputStreambuf, 51
- zipios::InflateInputStreambuf, 53
  - ~InflateInputStreambuf, 53
  - \_outvec, 53
  - \_outvecsize, 53
  - InflateInputStreambuf, 54
  - reset, 54
  - underflow, 53
- zipios::InvalidStateException, 55
  - ~InvalidStateException, 55
  - InvalidStateException, 55
  - operator=, 55

- what, 55
- zipios::IOException, 56
  - ~IOException, 56
  - IOException, 56
  - operator=, 56
  - what, 56
- zipios::OutputStringStream, 59
  - str, 59
- zipios::SimpleSmartPointer, 60
  - ~SimpleSmartPointer, 60
  - get, 60
  - operator \*, 60
  - operator void \*, 60
  - operator → , 60
  - operator=, 60
  - operator==, 60
  - SimpleSmartPointer, 60
- zipios::VirtualSeeker, 61
  - endOffset, 61
  - getOffsets, 61
  - setOffsets, 61
  - startOffset, 61
  - VirtualSeeker, 61
  - vseekg, 61
  - vtellg, 61
- zipios::ZipCDirEntry, 62
  - ~ZipCDirEntry, 62
  - clone, 63
  - getComment, 63
  - getLocalHeaderOffset, 62
  - operator=, 62
  - setComment, 63
  - toString, 63
  - ZipCDirEntry, 62
- zipios::ZipFile, 64
  - ~ZipFile, 65
  - clone, 65
  - close, 66
  - getInputStream, 66
  - openEmbeddedZipFile, 67
  - ZipFile, 65
- zipios::ZipInputStream, 68
  - ~ZipInputStream, 69
  - available, 68
  - close, 69
  - closeEntry, 69
  - getNextEntry, 69
  - ZipInputStream, 68
- zipios::ZipInputSteambuf, 70
  - ~ZipInputSteambuf, 71
  - close, 71
  - closeEntry, 71
  - getNextEntry, 71
  - underflow, 70
  - ZipInputSteambuf, 71
- zipios::ZipLocalEntry, 72
  - ~ZipLocalEntry, 73
  - \_valid, 74
  - clone, 74
  - compress\_method, 73
  - compress\_size, 73
  - crc\_32, 73
  - extra\_field, 74
  - extra\_field\_len, 74
  - extract\_version, 73
  - filename, 74
  - filename\_len, 74
  - getComment, 74
  - getCompressedSize, 74
  - getCrc, 75
  - getExtra, 75
  - getFileName, 75
  - getMethod, 75
  - getName, 76
  - getSize, 76
  - getTime, 76
  - gp\_bitfield, 73
  - isDirectory, 76
  - isValid, 76
  - last\_mod\_fdate, 73
  - last\_mod\_mtime, 73
  - operator=, 72
  - setComment, 77
  - setCompressedSize, 77
  - setCrc, 77
  - setExtra, 77
  - setMethod, 78
  - setName, 78
  - setSize, 78
  - setTime, 78
  - signature, 73
  - toString, 78
  - trailingDataDescriptor, 73
  - uncompress\_size, 74
  - ZipLocalEntry, 72
- ZipLocalEntry
  - zipios::ZipLocalEntry, 72