Zipios++ Reference Manual

Generated by Doxygen 1.1.4

Sun Jul 9 00:55:44 2000

Contents

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).

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 CColll::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

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

Chapter 2

Zipios++ Hierarchical Index

2.1 Zipios++ Class Hierarchy

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

Chapter 3

Zipios++ Compound Index

3.1 Zipios++ Compound List

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

Chapter 4

Zipios++ File Index

4.1 Zipios++ File List

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

Chapter 5

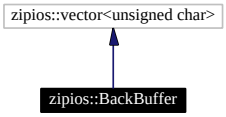
Zipios++ Class Documentation

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:

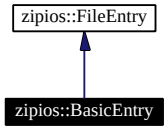
- backbuffer.h

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 succesfully.

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, ConstEntryPointer &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<ConstEntryPointer> zipios::CollectionCollection::entries () const [virtual]

Returns a vector of const pointers to the entries in the FileCollection.

Returns:
a vector< ConstEntryPointer > 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*, ConstEntryPointer & *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 ConstEntryPointer zipios::CollectionCollection::getEntry (const string & *name*, MatchPath *matchpath* = MATCH) const [virtual]

Returns a ConstEntryPointer 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 Speficy MATCH, if the path should match as well, specify IGNORE, if the path should be ignored.

Returns:
A ConstEntryPointer 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 Speficy 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 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 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< 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 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<ConstEntryPointer> zipios::DirectoryCollection::entries () const [virtual]

Returns a vector of const pointers to the entries in the FileCollection.

Returns:
a vector< ConstEntryPointer > 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 ConstEntryPointer zipios::DirectoryCollection::getEntry (const string & *name*, MatchPath *matchpath* = MATCH) const [virtual]

Returns a ConstEntryPointer 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 Speficy MATCH, if the path should match as well, specify IGNORE, if the path should be ignored.

Returns:
A ConstEntryPointer 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 Speficy 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:

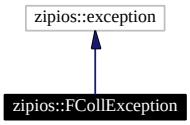
- ziphead.h
- ziphead.cpp

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:

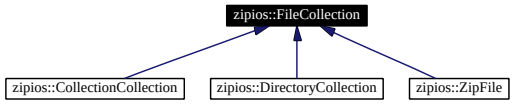
- fcollexceptions.h
- fcollexceptions.cpp

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< 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) = 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< EntryPointer > **_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<ConstEntryPointer> zipios::FileCollection::entries () const [virtual]

Returns a vector of const pointers to the entries in the FileCollection.

Returns:
a vector< ConstEntryPointer > 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 ConstEntryPointer zipios::FileCollection::getEntry (const string & *name*, MatchPath *matchpath* = MATCH) const [virtual]

Returns a ConstEntryPointer 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 Speficy MATCH, if the path should match as well, specify IGNORE, if the path should be ignored.

Returns:
A ConstEntryPointer 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 Speficy 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:

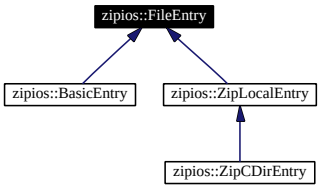
- fcoll.h
- fcoll.cpp

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 succesfully.

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 immidiately, 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:

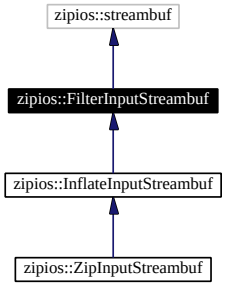
- filepath.h
- filepath.cpp

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:

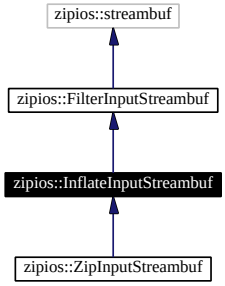
- filterinputstreambuf.h
- filterinputstreambuf.cpp

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:

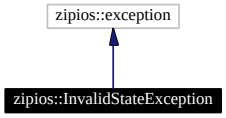
- inflateinputstreambuf.h
- inflateinputstreambuf.cpp

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:

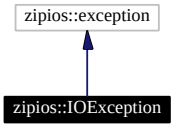
- fcollexceptions.h
- fcollexceptions.cpp

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 ConstEntryPointer &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:

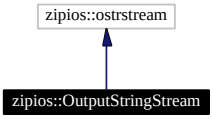
- fileentry.h

5.17 zipios::OutputStringStream Class Reference

OutputStringStream is typedefed to ostringstream 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 ostringstream 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 strstream.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 <virtualseeker.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 virtualseeker.h.

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

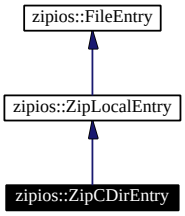
- virtualseeker.h

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 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.

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 Speficy 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 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 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:

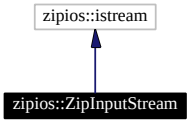
- zipfile.h
- zipfile.cpp

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.

- ConstEntryPointer 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:

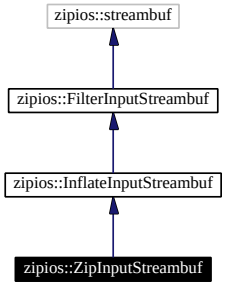
- zipinputstream.h
- zipinputstream.cpp

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::ZipInputStreambuf::ZipInputStreambuf (streambuf * *inbuf*, int *s_pos* = -1, bool *del_inbuf* = false) [explicit]

ZipInputStreambuf 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 ZipInputStreambuf is destructed.

Definition at line 18 of file zipinputstreambuf.cpp.

5.23.2.2 zipios::ZipInputStreambuf::~ZipInputStreambuf () [virtual]

Destructor.

Definition at line 80 of file zipinputstreambuf.cpp.

5.23.3 Member Function Documentation

5.23.3.1 void zipios::ZipInputStreambuf::close ()

Closes the streambuf.

Definition at line 42 of file zipinputstreambuf.cpp.

5.23.3.2 void zipios::ZipInputStreambuf::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::ZipInputStreambuf::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:

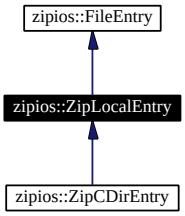
- zipinputstreambuf.h
- zipinputstreambuf.cpp

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 succesfully.

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

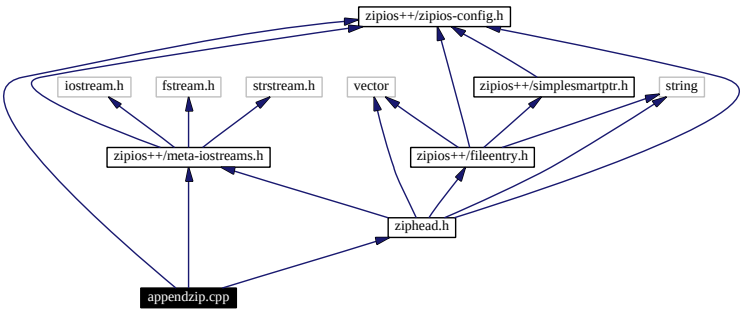
Zipios++ File Documentation

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.

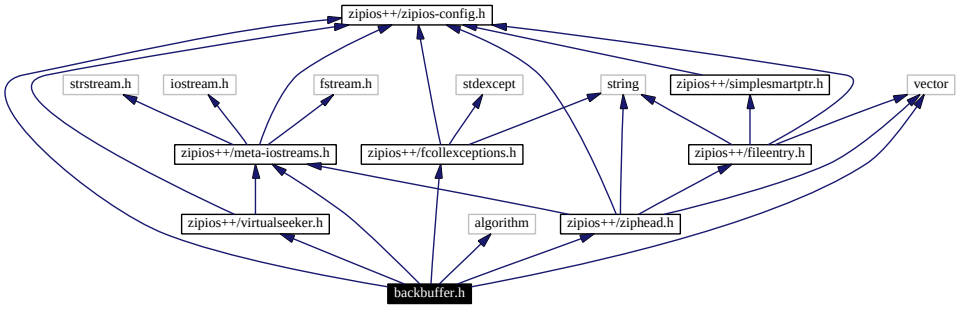
Definition in file appendzip.cpp.

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++/virtualseeker.h"
```

Include dependency graph for backbuffer.h:



Namespaces

- namespace **zipios**

6.2.1 Detailed Description

The header file for BackBuffer.

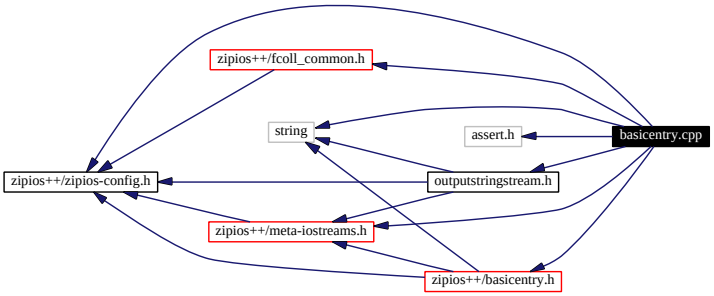
Definition in file backbuffer.h.

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.

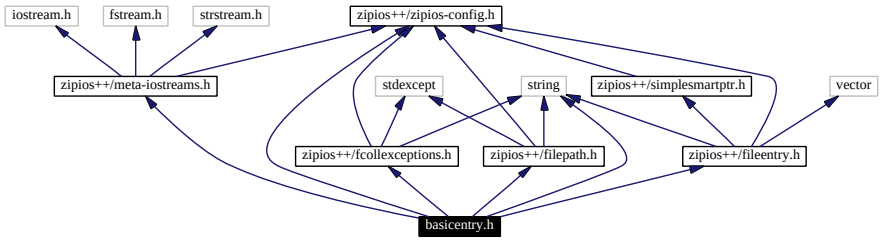
Definition in file basicentry.cpp.

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.

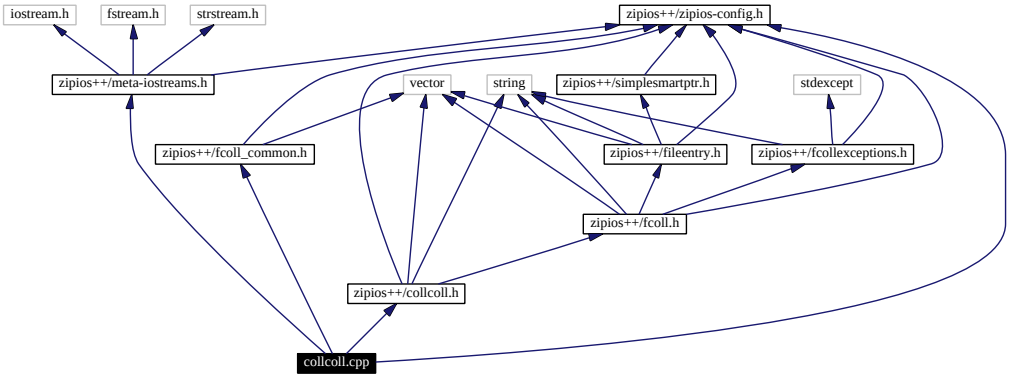
Definition in file basicentry.h.

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.

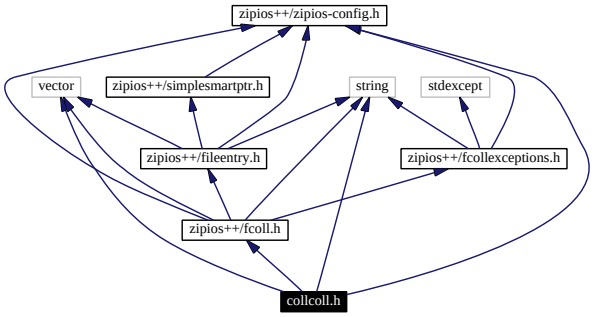
Definition in file collcoll.cpp.

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.

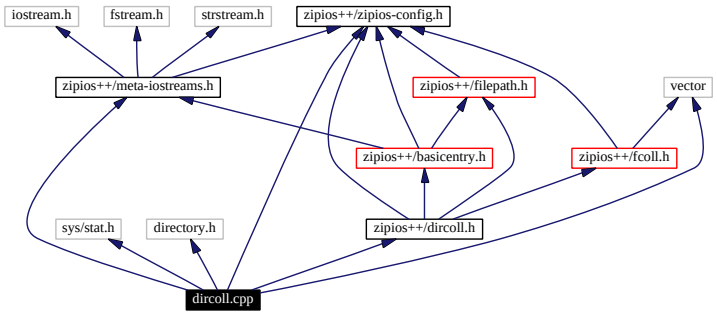
Definition in file collcoll.h.

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.

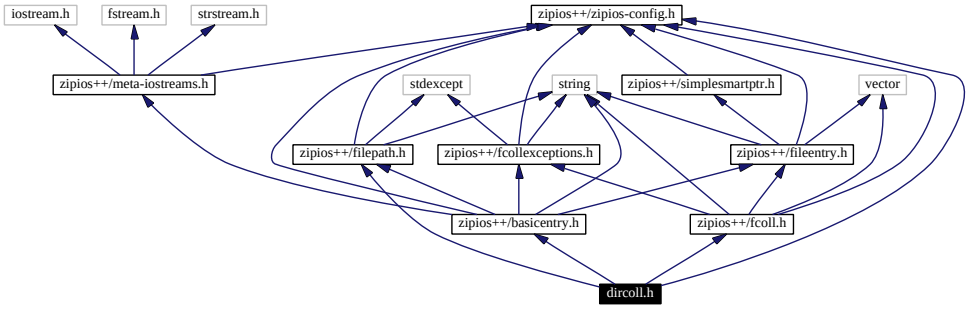
Definition in file dircoll.cpp.

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.

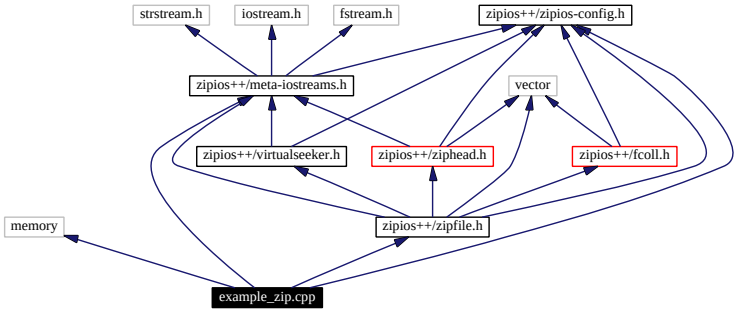
Definition in file dircoll.h.

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++.

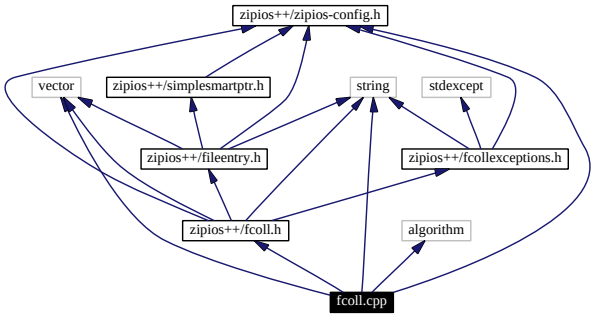
Definition in file example_zip.cpp.

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.

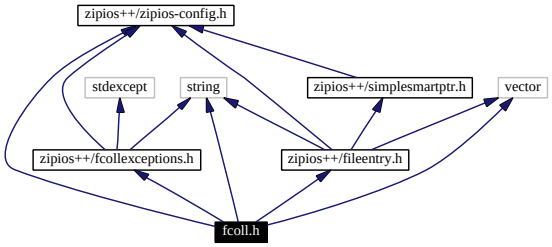
Definition in file fcoll.cpp.

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.

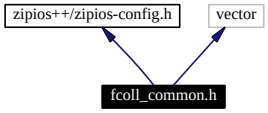
Definition in file fcoll.h.

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!

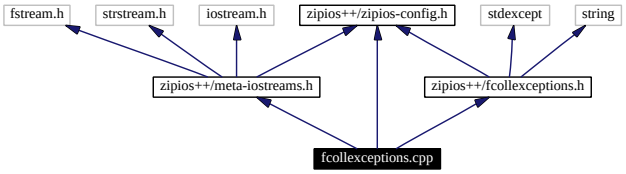
Definition in file fcoll_common.h.

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.

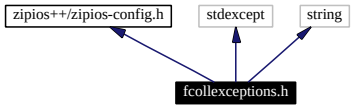
Definition in file fcollexceptions.cpp.

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.

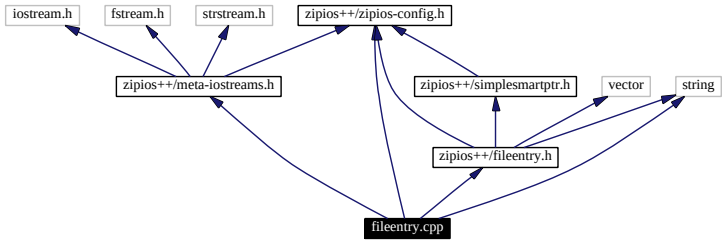
Definition in file fcollexceptions.h.

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.

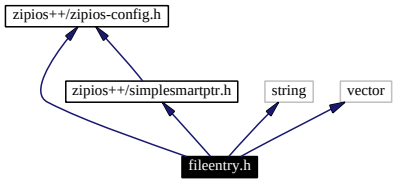
Definition in file fileentry.cpp.

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.

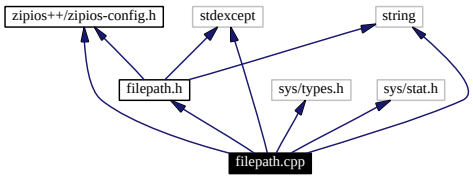
Definition in file fileentry.h.

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.

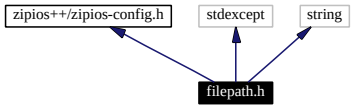
Definition in file filepath.cpp.

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.

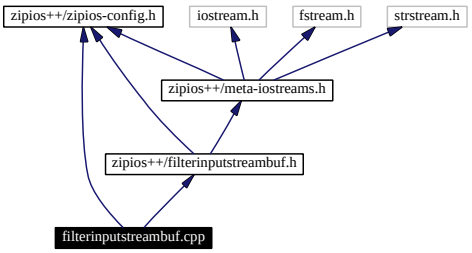
Definition in file filepath.h.

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.

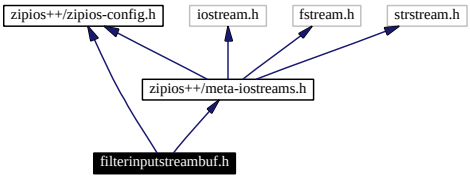
Definition in file filterinputstreambuf.cpp.

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.

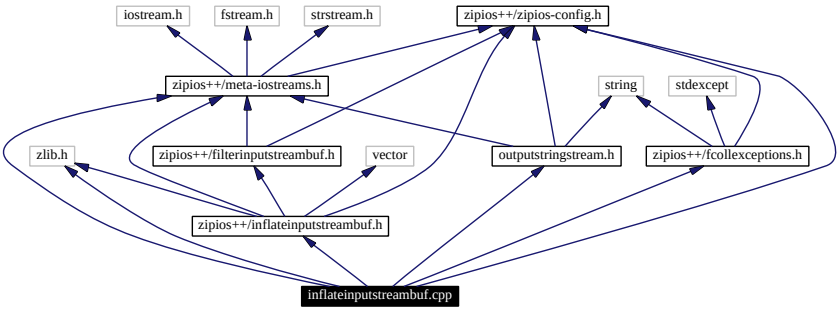
Definition in file filterinputstreambuf.h.

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.

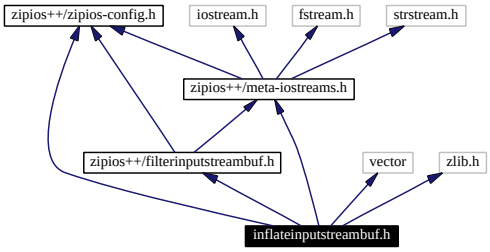
Definition in file inflateinputstreambuf.cpp.

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.

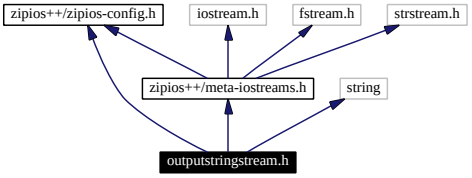
Definition in file inflateinputstreambuf.h.

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.

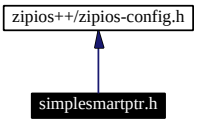
Definition in file outputstringstream.h.

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.

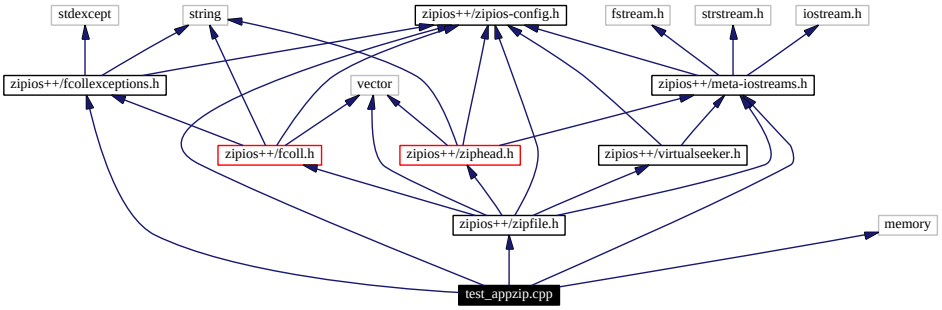
Definition in file simplesmartptr.h.

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++.

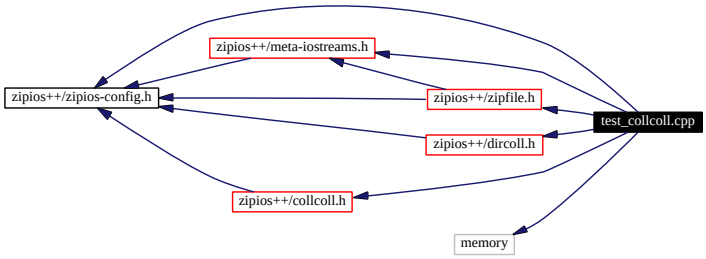
Definition in file test_appzip.cpp.

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++.

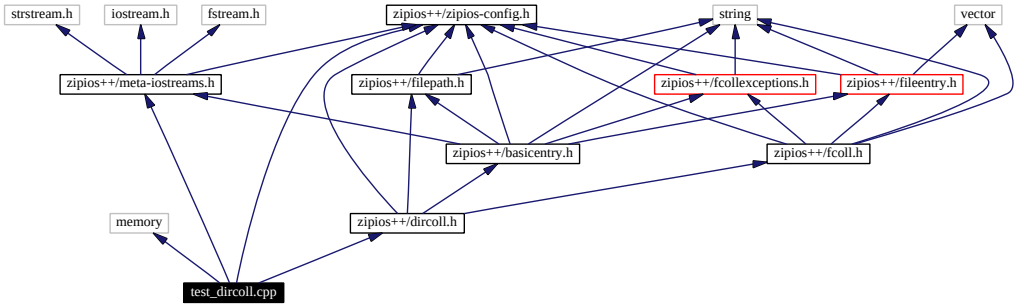
Definition in file test_collcoll.cpp.

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++.

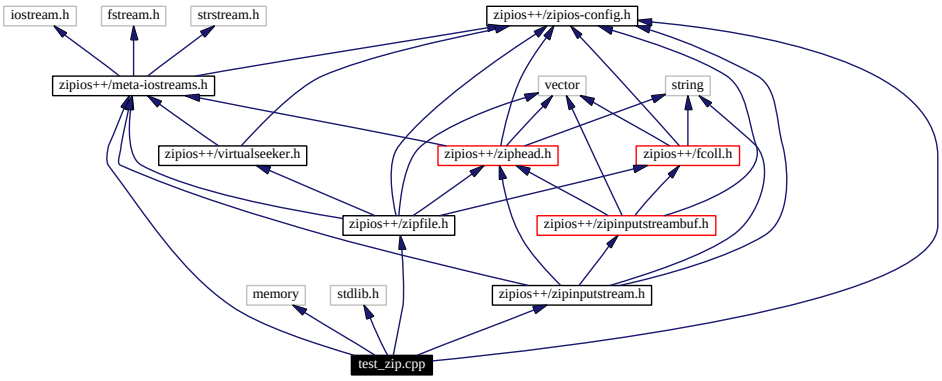
Definition in file test_dircoll.cpp.

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++.

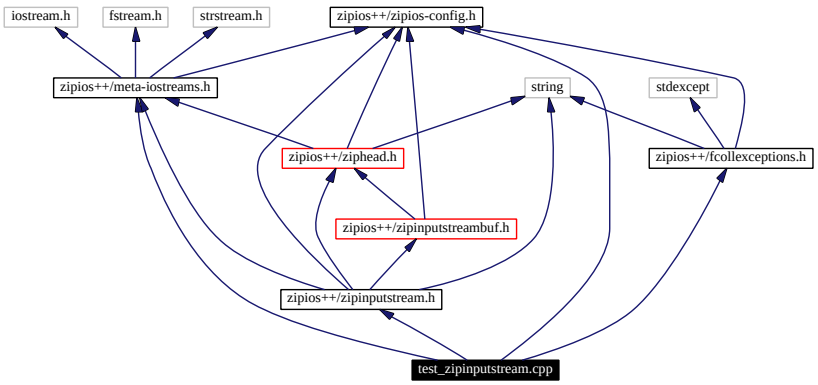
Definition in file test_zip.cpp.

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.

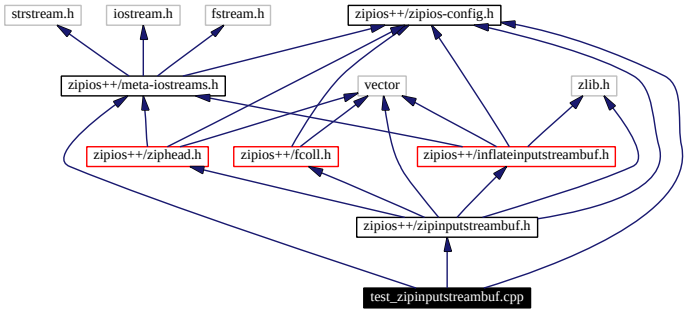
Definition in file test_zipinputstream.cpp.

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.

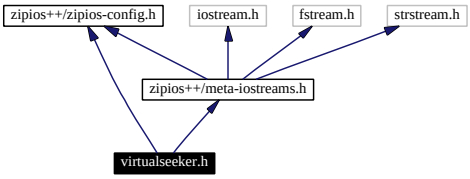
Definition in file test_zipinputstreambuf.cpp.

6.31 virtualseeker.h File Reference

Header file that defines VirtualSeeker.

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

Include dependency graph for virtualseeker.h:



Namespaces

- namespace **zipios**

6.31.1 Detailed Description

Header file that defines VirtualSeeker.

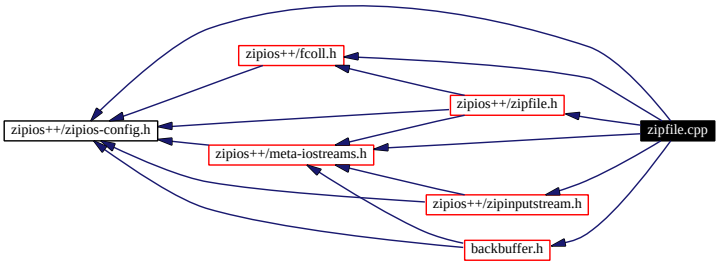
Definition in file virtualseeker.h.

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.

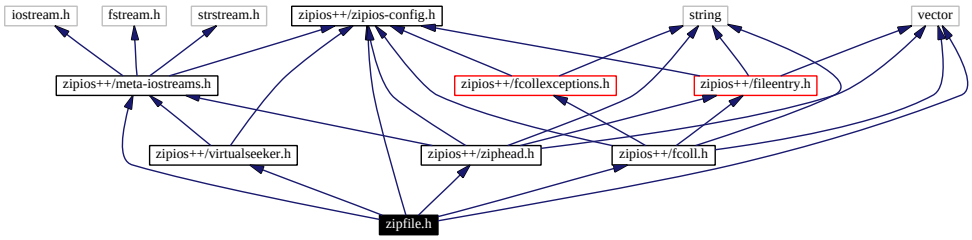
Definition in file zipfile.cpp.

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++/virtualseeker.h"
```

Include dependency graph for zipfile.h:



Namespaces

- namespace **zipios**

6.33.1 Detailed Description

Header file that defines ZipFile.

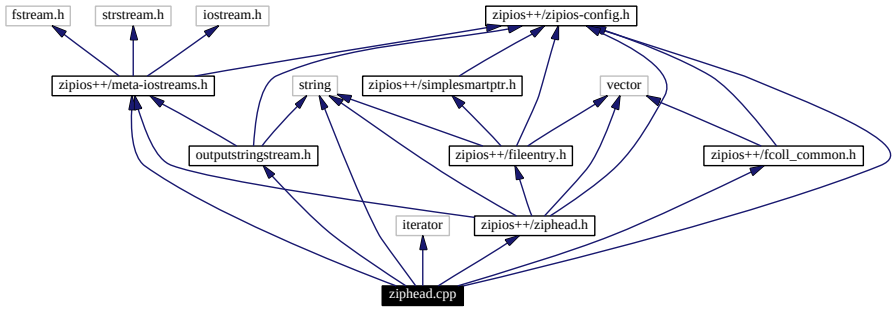
Definition in file zipfile.h.

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.

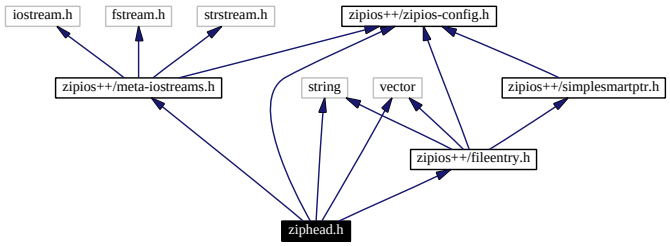
Definition in file ziphead.cpp.

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.

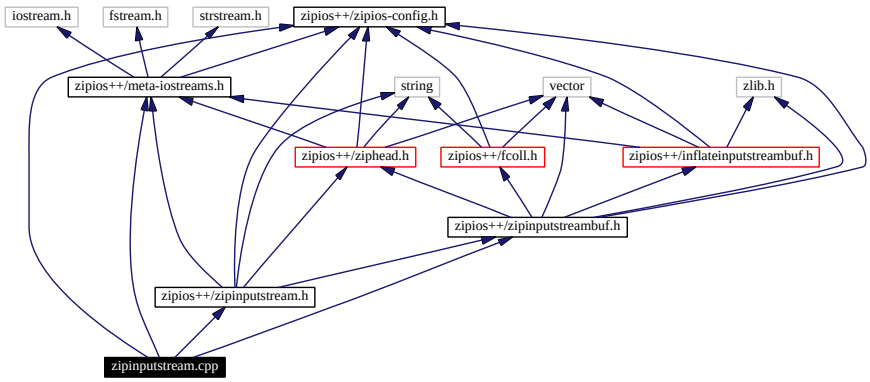
Definition in file ziphead.h.

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.

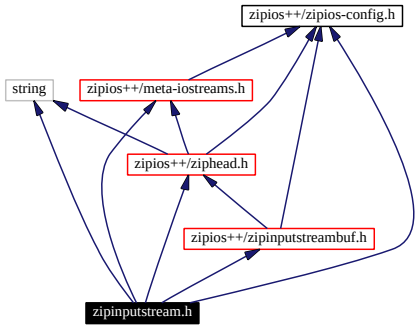
Definition in file zipinputstream.cpp.

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.

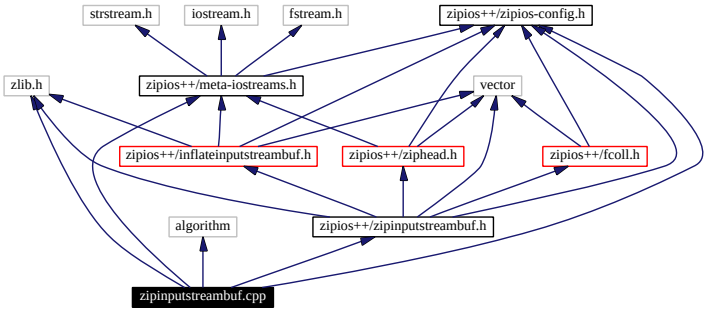
Definition in file zipinputstream.h.

6.38 zipinputstreambuf.cpp File Reference

Implementation of ZipInputStreambuf.

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

Include dependency graph for zipinputstreambuf.cpp:



Namespaces

- namespace **zipios**

6.38.1 Detailed Description

Implementation of ZipInputStreambuf.

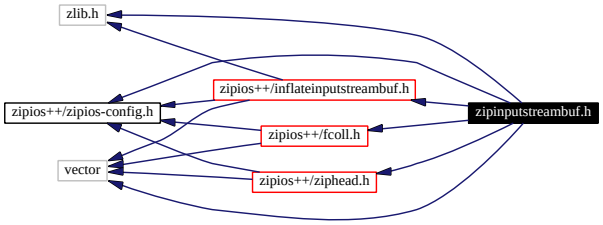
Definition in file zipinputstreambuf.cpp.

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