

---

**libkiwix**

**libzim-team**

**Dec 01, 2021**



**CONTENTS:**

<b>1</b>	<b>Libkiwix programming</b>	<b>1</b>
1.1	Introduction . . . . .	1
<b>2</b>	<b>Reference API</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
2.2	File Hierarchy . . . . .	3
2.3	Full API . . . . .	3
	<b>Index</b>	<b>49</b>



## **LIBKIWIX PROGRAMMING**

### **1.1 Introduction**

libkiwix is written in C++. To use the library, you need the include files of libkiwix have to link against libzim.

Errors are handled with exceptions. When something goes wrong, libzim throws an error, which is always derived from `std::exception`.

All classes are defined in the namespace `kiwix`.

libkiwix is a set of tools to manage zim files and provide some common fonctionnality. While libkiwix has some wrappers around libzim classes, they are deprecated and will be removed in the future.



## REFERENCE API

### 2.1 Class Hierarchy

### 2.2 File Hierarchy

### 2.3 Full API

#### 2.3.1 Namespaces

##### Namespace kiwix

###### Contents

- *Classes*
- *Enums*
- *Functions*
- *Typedefs*

###### Classes

- *Struct DownloadedFile*
- *Struct LibraryBase::Entry*
- *Class AriaError*
- *Class Book*
- *Class Book::Illustration*
- *Class Bookmark*
- *Class Download*
- *Class Downloader*
- *Class Entry*
- *Class Filter*

- *Class HumanReadableNameMapper*
- *Class IdNameMapper*
- *Class KiwixServe*
- *Class Library*
- *Class LibraryBase*
- *Class LibraryManipulator*
- *Class LibXMLDumper*
- *Class Manager*
- *Class NameMapper*
- *Class NoEntry*
- *Class OPDSDumper*
- *Class Reader*
- *Class Result*
- *Class Searcher*
- *Class SearchRenderer*
- *Class Server*
- *Class SuggestionItem*
- *Class UpdatableNameMapper*

## **Enums**

- *Enum supportedListMode*
- *Enum supportedListSortBy*

## **Functions**

- *Function kiwix::appendToDirectory*
- *Function kiwix::computeAbsolutePath*
- *Function kiwix::computeRelativePath*
- *Function kiwix::converta2toa3*
- *Function kiwix::fileExists*
- *Function kiwix::getCurrentDirectory*
- *Function kiwix::getDataDirectory*
- *Function kiwix::getExecutablePath*
- *Function kiwix::getFileContent*
- *Function kiwix::getLastPathElement*
- *Function kiwix::getMimeTypeForFile*



- *Function* `kiwix::getSearchUrl`
- *Function* `kiwix::isRelativePath`
- *Function* `kiwix::removeLastPathElement`
- *Function* `kiwix::sleep`
- *Function* `kiwix::split`

## Typedefs

- *Typedef* `kiwix::offset_type`
- *Typedef* `kiwix::size_type`
- *Typedef* `kiwix::SuggestionsList_t`

## Namespace pugi

## Namespace std

## Namespace zim

## 2.3.2 Classes and Structs

### Struct DownloadedFile

- Defined in `file_include_downloader.h`

### Struct Documentation

**struct** `kiwix::DownloadedFile`

#### Public Functions

**inline** `DownloadedFile()`

#### Public Members

`bool` **success**

`std::string` **path**

## Struct `LibraryBase::Entry`

- Defined in `file_include_library.h`

## Nested Relationships

This struct is a nested type of *Class `LibraryBase`*.

## Inheritance Relationships

### Base Type

- `public kiwix::Book` (*Class `Book`*)

## Struct Documentation

```
struct kiwix::LibraryBase::Entry : public kiwix::Book
```

### Public Members

*LibraryRevision* `lastUpdatedRevision` = 0

## Class `AriaError`

- Defined in `file_include_downloader.h`

## Inheritance Relationships

### Base Type

- `public runtime_error`

## Class Documentation

```
class kiwix::AriaError : public runtime_error
```

### Public Functions

```
inline AriaError (const std::string &message)
```

## Class Book

- Defined in file\_include\_book.h

## Nested Relationships

### Nested Types

- *Class Book::Illustration*

## Inheritance Relationships

### Derived Type

- `public kiwix::LibraryBase::Entry (Struct LibraryBase::Entry)`

## Class Documentation

### **class** kiwix::Book

A class to store information about a book (a zim file)

Subclassed by *kiwix::LibraryBase::Entry*

### Public Types

**typedef** std::vector<std::shared\_ptr<const *Illustration*>> **Illustrations**

### Public Functions

**Book** ()

**~Book** ()

bool **update** (const *Book* &other)

void **update** (const *Reader* &reader)

void **update** (const zim::Archive &archive)

void **updateFromXml** (const pugi::xml\_node &node, const std::string &baseDir)

void **updateFromOpds** (const pugi::xml\_node &node, const std::string &urlHost)

std::string **getHumanReadableIdFromPath** () const

**inline** bool **readOnly** () const

**inline** const std::string &**getId** () const

**inline** const std::string &**getPath** () const

**inline** bool **isPathValid** () const

**inline** const std::string &**getTitle** () const

**inline** const std::string &**getDescription** () const

```
inline const std::string &getLanguage () const
inline const std::string &getCreator () const
inline const std::string &getPublisher () const
inline const std::string &getDate () const
inline const std::string &getUrl () const
inline const std::string &getName () const
std::string getCategory () const
inline const std::string &getTags () const
std::string getTagStr (const std::string &tagName) const
bool getTagBool (const std::string &tagName) const
inline const std::string &getFlavour () const
inline const std::string &getOrigId () const
inline const uint64_t &getArticleCount () const
inline const uint64_t &getMediaCount () const
inline const uint64_t &getSize () const
const std::string &getFavicon () const
const std::string &getFaviconUrl () const
const std::string &getFaviconMimeType () const
Illustrations getIllustrations () const
inline const std::string &getDownloadId () const
inline void setReadOnly (bool readOnly)
inline void setId (const std::string &id)
void setPath (const std::string &path)
inline void setPathValid (bool valid)
inline void setTitle (const std::string &title)
inline void setDescription (const std::string &description)
inline void setLanguage (const std::string &language)
inline void setCreator (const std::string &creator)
inline void setPublisher (const std::string &publisher)
inline void setDate (const std::string &date)
inline void setUrl (const std::string &url)
inline void setName (const std::string &name)
inline void setFlavour (const std::string &flavour)
inline void setTags (const std::string &tags)
inline void setOrigId (const std::string &origId)
inline void setArticleCount (uint64_t articleCount)
```

```
inline void setMediaCount (uint64_t mediaCount)
inline void setSize (uint64_t size)
inline void setDownloadId (const std::string &downloadId)
```

### Protected Attributes

```
std::string m_id
std::string m_downloadId
std::string m_path
bool m_pathValid = false
std::string m_title
std::string m_description
std::string m_category
std::string m_language
std::string m_creator
std::string m_publisher
std::string m_date
std::string m_url
std::string m_name
std::string m_flavour
std::string m_tags
std::string m_origId
uint64_t m_articleCount = 0
uint64_t m_mediaCount = 0
bool m_readOnly = false
uint64_t m_size = 0
Illustrations m_illustrations
```

### Protected Static Attributes

```
static const Illustration missingDefaultIllustration
class Illustration
```

### Public Functions

```
const std::string &getData () const
```

### Public Members

```
uint16_t width = 48  
uint16_t height = 48  
std::string mimeType  
std::string url
```

### Class Book::Illustration

- Defined in file\_include\_book.h

### Nested Relationships

This class is a nested type of *Class Book*.

### Class Documentation

```
class kiwix::Book::Illustration
```

### Public Functions

```
const std::string &getData () const
```

### Public Members

```
uint16_t width = 48  
uint16_t height = 48  
std::string mimeType  
std::string url
```

### Class Bookmark

- Defined in file\_include\_bookmark.h

## Class Documentation

### **class** `kiwix::Bookmark`

A class to store information about a bookmark (an article in a book)

#### Public Functions

```
Bookmark ()  
~Bookmark ()  
void updateFromXml (const pugi::xml_node &node)  
inline const std::string &getBookId () const  
inline const std::string &getBookTitle () const  
inline const std::string &getUrl () const  
inline const std::string &getTitle () const  
inline const std::string &getLanguage () const  
inline const std::string &getDate () const  
inline void setBookId (const std::string &bookId)  
inline void setBookTitle (const std::string &bookTitle)  
inline void setUrl (const std::string &url)  
inline void setTitle (const std::string &title)  
inline void setLanguage (const std::string &language)  
inline void setDate (const std::string &date)
```

#### Protected Attributes

```
std::string m_bookId  
std::string m_bookTitle  
std::string m_url  
std::string m_title  
std::string m_language  
std::string m_date
```

## Class Download

- Defined in file\_include\_downloader.h

## Class Documentation

**class** kiwix::Download

### Public Types

**enum** StatusResult

*Values:*

**enumerator** K\_ACTIVE

**enumerator** K\_WAITING

**enumerator** K\_PAUSED

**enumerator** K\_ERROR

**enumerator** K\_COMPLETE

**enumerator** K\_REMOVED

**enumerator** K\_UNKNOWN

### Public Functions

**inline** Download()

**inline** Download(std::shared\_ptr<Aria2> p\_aria, std::string did)

void **updateStatus**(bool *follow* = false)

void **pauseDownload**()

void **resumeDownload**()

void **cancelDownload**()

**inline** StatusResult **getStatus**()

**inline** std::string **getDid**()

**inline** std::string **getFollowedBy**()

**inline** uint64\_t **getTotalLength**()

**inline** uint64\_t **getCompletedLength**()

**inline** uint64\_t **getDownloadSpeed**()

**inline** uint64\_t **getVerifiedLength**()

**inline** std::string **getPath**()

**inline** std::vector<std::string> &**getUris**()



### Protected Attributes

```
std::shared_ptr<Aria2> mp_aria
StatusResult m_status
std::string m_did = ""
std::string m_followedBy = ""
uint64_t m_totalLength
uint64_t m_completedLength
uint64_t m_downloadSpeed
uint64_t m_verifiedLength
std::vector<std::string> m_uris
std::string m_path
```

### Class Downloader

- Defined in file\_include\_downloader.h

### Class Documentation

```
class kiwix::Downloader
    A tool to download things.
```

### Public Functions

```
Downloader ()
virtual ~Downloader ()
void close ()
Download *startDownload (const std::string &uri, const std::vector<std::pair<std::string,
                                std::string>> &options = { })
Download *getDownload (const std::string &did)
inline size_t getNbDownload ()
std::vector<std::string> getDownloadIds ()
```

### Class Entry

- Defined in file\_include\_entry.h

## Class Documentation

### **class** `kiwix::Entry`

A entry represent an.. entry in a zim file.

### Public Functions

#### **Entry** (`zim::Entry entry`)

Construct an entry making reference to an zim article.

**Parameters** `article` – a `zim::Article` object

#### **virtual ~Entry** () = default

#### **inline** `std::string getPath () const`

Get the path of the entry.

The path is the “key” of an entry.

**Returns** the path of the entry.

#### **inline** `std::string getTitle () const`

Get the title of the entry.

**Returns** the title of the entry.

#### **inline** `std::string getContent () const`

Get the content of the entry.

The string is a copy of the content. If you don’t want to do a copy, use `get_blob`.

**Returns** the content of the entry.

#### **inline** `zim::Blob getBlob (offset_type offset = 0) const`

Get the blob of the entry.

A blob make reference to the content without copying it.

**Parameters** `offset` – The starting offset of the blob.

**Returns** the blob of the entry.

#### **inline** `zim::Blob getBlob (offset_type offset, size_type size) const`

Get the blob of the entry.

A blob make reference to the content without copying it.

#### **Parameters**

- **offset** – The starting offset of the blob.
- **size** – The size of the blob.

**Returns** the blob of the entry.

#### **inline** `zim::Item::DirectAccessInfo getDirectAccessInfo () const`

Get the info for direct access to the content of the entry.

Some entry (ie binary ones) have their content plain stored in the zim file. Knowing the offset where the content is stored an user can directly read the content in the zim file bypassing the `libkiwix/libzim`.

**Returns** A pair specifying where to read the content. The string is the real file to read (may be different that .zim file if zim is cut). The offset is the offset to read in the file. Return `<"",0>` if is not possible to read directly.

*size\_type* **getSize () const**

Get the size of the entry.

**Returns** the size of the entry.

std::string **getMimeType () const**

Get the mime\_type of the entry.

**Returns** the mime\_type of the entry.

bool **isRedirect () const**

Get if the entry is a redirect entry.

**Returns** True if the entry is a redirect.

bool **isLinkTarget () const**

Get if the entry is a link target entry.

**Returns** True if the entry is a link target.

bool **isDeleted () const**

Get if the entry is a deleted entry.

**Returns** True if the entry is a deleted entry.

*Entry* **getRedirectEntry () const**

Get the entry pointed by this entry.

**Throws** *NoEntry* – if the entry is not a redirected entry.

**Returns** the entry pointed.

*Entry* **getFinalEntry () const**

Get the final entry pointed by this entry.

Follow the redirection until a “not redirecting” entry is found. If the entry is not a redirected entry, return the entry itself.

**Returns** the final entry.

**inline const** zim::Entry &**getZimEntry () const**

Get the zim entry wrapped by this (kiwix) entry

**Returns** the zim entry

## Class Filter

- Defined in file\_include\_library.h

## Class Documentation

```
class kiwix::Filter
```

## Public Types

```
using Tags = std::vector<std::string>
```

## Public Functions

```
Filter()
```

```
~Filter() = default
```

```
Filter &local (bool accept)
```

Set the filter to check local.

A local book is a book with a path. If accept is true, only local book are accepted. If accept is false, only non local book are accepted.

```
Filter &remote (bool accept)
```

Set the filter to check remote.

A remote book is a book with a url. If accept is true, only remote book are accepted. If accept is false, only non remote book are accepted.

```
Filter &valid (bool accept)
```

Set the filter to check validity.

A valid book is a book with a path pointing to a existing zim file. If accept is true, only valid book are accepted. If accept is false, only non valid book are accepted.

```
Filter &acceptTags (const Tags &tags)
```

Set the filter to only accept book with corresponding tag.

```
Filter &rejectTags (const Tags &tags)
```

```
Filter &category (std::string category)
```

```
Filter &lang (std::string lang)
```

```
Filter &publisher (std::string publisher)
```

```
Filter &creator (std::string creator)
```

```
Filter &maxSize (size_t size)
```

```
Filter &query (std::string query, bool partial = true)
```

```
Filter &name (std::string name)
```

```
bool hasQuery () const
```

```
inline const std::string &getQuery () const
```

```
inline bool queryIsPartial () const
```

```
bool hasName () const
```

```
inline const std::string &getName () const
```

```
bool hasCategory () const
```

```
inline const std::string &getCategory () const
```

```
bool hasLang () const
```

```
inline const std::string &getLang () const
```

```
bool hasPublisher () const
```

```
inline const std::string &getPublisher () const
bool hasCreator () const
inline const std::string &getCreator () const
inline const Tags &getAcceptTags () const
inline const Tags &getRejectTags () const
```

## Class HumanReadableNameMapper

- Defined in file\_include\_name\_mapper.h

## Inheritance Relationships

### Base Type

- public kiwix::NameMapper (*Class NameMapper*)

## Class Documentation

```
class kiwix::HumanReadableNameMapper : public kiwix::NameMapper
```

### Public Functions

```
HumanReadableNameMapper (kiwix::Library &library, bool withAlias)
virtual ~HumanReadableNameMapper () = default
virtual std::string getNameForId (const std::string &id) const
virtual std::string getIdForName (const std::string &name) const
```

## Class IdNameMapper

- Defined in file\_include\_name\_mapper.h

## Inheritance Relationships

### Base Type

- public kiwix::NameMapper (*Class NameMapper*)

## Class Documentation

```
class kiwix::IdNameMapper : public kiwix::NameMapper
```

### Public Functions

```
inline virtual std::string getNameForId (const std::string &id) const  
inline virtual std::string getIdForName (const std::string &name) const
```

## Class KiwixServe

- Defined in file\_include\_kiwixserve.h

## Class Documentation

```
class kiwix::KiwixServe
```

### Public Functions

```
KiwixServe (const std::string &libraryPath, int port = 8181)  
~KiwixServe ()  
void run ()  
void shutDown ()  
bool isRunning ()  
inline int getPort ()  
int setPort (int port)
```

## Class Library

- Defined in file\_include\_library.h

## Inheritance Relationships

### Base Type

- private kiwix::LibraryBase (*Class LibraryBase*)

## Class Documentation

**class** `kiwix::Library` : **private** `kiwix::LibraryBase`  
 A *Library* store several books.

### Public Types

```
typedef LibraryRevision Revision
typedef std::vector<std::string> BookIdCollection
typedef std::map<std::string, int> AttributeCounts
```

### Public Functions

**Library** ()

**~Library** ()

**Library** (const *Library*&) = delete  
*Library* is not a copiable object. However it can be moved.

**Library** (*Library*&&)

void **operator=** (const *Library*&) = delete

*Library* &**operator=** (*Library*&&)

bool **addBook** (const *Book* &*book*)  
 Add a book to the library.

If a book already exist in the library with the same id, update the existing book instead of adding a new one.

**Parameters** **book** – The book to add.

**Returns** True if the book has been added. False if a book has been updated.

**inline** bool **addOrUpdateBook** (const *Book* &*book*)  
 A self-explanatory alias for *addBook()*

void **addBookmark** (const *Bookmark* &*bookmark*)  
 Add a bookmark to the library.

**Parameters** **bookmark** – the book to add.

bool **removeBookmark** (const std::string &*zimId*, const std::string &*url*)  
 Remove a bookmarkk

**Parameters**

- **zimId** – The zimId of the bookmark.
- **url** – The url of the bookmark.

**Returns** True if the bookmark has been removed.

const *Book* &**getBookById** (const std::string &*id*) const

const *Book* &**getBookByPath** (const std::string &*path*) const

*Book* **getBookByIdThreadSafe** (const std::string &*id*) const  
 std::shared\_ptr<*Reader*> **getReaderById** (const std::string &*id*)

`std::shared_ptr<zim::Archive> getArchiveById (const std::string &id)`

`bool removeBookById (const std::string &id)`

Remove a book from the library.

**Parameters** `id` – the id of the book to remove.

**Returns** True if the book were in the library and has been removed.

`bool writeToFile (const std::string &path) const`

Write the library to a file.

**Parameters** `path` – the path of the file to write to.

**Returns** True if the library has been correctly saved.

`bool writeBookmarksToFile (const std::string &path) const`

Write the library bookmarks to a file.

**Parameters** `path` – the path of the file to write to.

**Returns** True if the library has been correctly saved.

`unsigned int getBookCount (const bool localBooks, const bool remoteBooks) const`

Get the number of book in the library.

**Parameters**

- **localBooks** – If we must count local books (books with a path).
- **remoteBooks** – If we must count remote books (books with an url)

**Returns** The number of books.

`std::vector<std::string> getBooksLanguages () const`

Get all languages of the books in the library.

**Returns** A list of languages.

*AttributeCounts* `getBooksLanguagesWithCounts () const`

Get all languages of the books in the library with counts.

**Returns** A list of languages with the count of books in each language.

`std::vector<std::string> getBooksCategories () const`

Get all categories of the books in the library.

**Returns** A list of categories.

`std::vector<std::string> getBooksCreators () const`

Get all book creators of the books in the library.

**Returns** A list of book creators.

`std::vector<std::string> getBooksPublishers () const`

Get all book publishers of the books in the library.

**Returns** A list of book publishers.

`const std::vector<kiwix::Bookmark> getBookmarks (bool onlyValidBookmarks = true) const`

Get all bookmarks.

**Returns** A list of bookmarks

*BookIdCollection* `getBooksIds () const`

Get all book ids of the books in the library.

**Returns** A list of book ids.



*BookIdCollection* **filter** (**const** std::string &search) **const**  
*Filter* the library and generate a new one with the keep elements.

This is equivalent to `listBookIds (ALL, UNSORTED, search)`.

**Parameters** **search** – List only books with search in the title or description.

**Returns** The list of bookIds corresponding to the query.

*BookIdCollection* **filter** (**const** *Filter* &filter) **const**  
*Filter* the library and return the id of the keep elements.

**Parameters** **filter** – The filter to use.

**Returns** The list of bookIds corresponding to the filter.

void **sort** (*BookIdCollection* &bookIds, *supportedListSortBy* sortBy, bool ascending) **const**  
Sort (in place) bookIds using the given comparator.

**Parameters**

- **bookIds** – the list of book Ids to sort
- **comparator** – how to sort the books

**Returns** The sorted list of books

*BookIdCollection* **listBooksIds** (int *supportedListMode* = ALL, *supportedListSortBy* sortBy = UNSORTED, **const** std::string &search = "", **const** std::string &language = "", **const** std::string &creator = "", **const** std::string &publisher = "", **const** std::vector<std::string> &tags = {}, size\_t maxSize = 0) **const**

List books in the library.

**Parameters**

- **mode** – The mode of listing :
  - LOCAL : list only local books (with a path).
  - REMOTE : list only remote books (with an url).
  - VALID : list only valid books (without a path or with a path pointing to a valid zim file).
  - NOLOCAL : list only books without valid path.
  - NOREMOTE : list only books without url.
  - NOVALID : list only books not valid.
  - ALL : Do not do any filter (LOCAL or REMOTE)
  - Flags can be combined.
- **sortBy** – Attribute to sort by the book list.
- **search** – List only books with search in the title, description.
- **language** – List only books in this language.
- **creator** – List only books of this creator.
- **publisher** – List only books of this publisher.
- **maxSize** – Do not list book bigger than maxSize. Set to 0 to cancel this filter.

**Returns** The list of bookIds corresponding to the query.

LibraryRevision **getRevision** () **const**

Return the current revision of the library.

The revision of the library is updated (incremented by one) only by the [addBook\(\)](#) operation.

**Returns** Current revision of the library.

uint32\_t **removeBooksNotUpdatedSince** (LibraryRevision *rev*)

Remove books that have not been updated since the specified revision.

**Parameters** *rev* – the library revision to use

**Returns** Count of books that were removed by this operation.

## Friends

**friend class** OPDSDumper

**friend class** libXMLDumper

## Class LibraryBase

- Defined in file\_include\_library.h

## Nested Relationships

### Nested Types

- *Struct LibraryBase::Entry*

## Inheritance Relationships

### Derived Type

- `private kiwix::Library` (*Class Library*)

## Class Documentation

**class** kiwix::LibraryBase

This class is not part of the libkiwix API. Its only purpose is to simplify the implementation of the *Library*'s move operations and avoid bugs should new data members be added to *Library*.

Subclassed by *kiwix::Library*

## Protected Types

```
typedef uint64_t LibraryRevision
```

## Protected Functions

```
LibraryBase()
```

```
~LibraryBase()
```

```
LibraryBase(LibraryBase&&)
```

```
LibraryBase &operator=(LibraryBase&&)
```

## Protected Attributes

```
LibraryRevision m_revision
```

```
std::map<std::string, Entry> m_books
```

```
std::map<std::string, std::shared_ptr<Reader>> m_readers
```

```
std::map<std::string, std::shared_ptr<zim::Archive>> m_archives
```

```
std::vector<kiwix::Bookmark> m_bookmarks
```

```
std::unique_ptr<BookDB> m_bookDB
```

```
struct Entry : public kiwix::Book
```

## Public Members

```
LibraryRevision lastUpdatedRevision = 0
```

## Class LibraryManipulator

- Defined in file\_include\_manager.h

## Class Documentation

```
class kiwix::LibraryManipulator
```

## Public Functions

```
explicit LibraryManipulator(Library *library)
```

```
virtual ~LibraryManipulator()
```

```
inline Library &getLibrary() const
```

```
bool addBookToLibrary(const Book &book)
```

```
void addBookmarkToLibrary(const Bookmark &bookmark)
```

```
uint32_t removeBooksNotUpdatedSince(Library::Revision rev)
```

## Protected Functions

```
virtual void bookWasAddedToLibrary (const Book &book)
virtual void bookmarkWasAddedToLibrary (const Bookmark &bookmark)
virtual void booksWereRemovedFromLibrary ()
```

## Class LibXMLDumper

- Defined in file\_include\_libxml\_dumper.h

## Class Documentation

```
class kiwix::LibXMLDumper
    A tool to dump a Library into a basic library.xml
```

## Public Functions

```
LibXMLDumper () = default
LibXMLDumper (const Library *library)
~LibXMLDumper ()

std::string dumpLibXMLContent (const std::vector<std::string> &bookIds)
    Dump the library.xml

    Parameters id – The id of the library.
    Returns The library.xml content.

std::string dumpLibXMLBookmark ()
    Dump the bookmark of the library.

    Returns The bookmark.xml content.

inline void setBaseDir (const std::string &baseDir)
    Set the base directory used.

    Parameters baseDir – the base directory to use.

inline void setLibrary (const Library *library)
    Set the library to dump.

    Parameters library – The library to dump.
```

## Protected Attributes

```
const kiwix::Library *library
std::string baseDir
```

## Class Manager

- Defined in file\_include\_manager.h

## Class Documentation

**class** `kiwix::Manager`  
 A tool to manage a *Library*.

### Public Types

**typedef** `std::vector<std::string>` **Paths**

### Public Functions

**explicit Manager** (*LibraryManipulator* \*manipulator)

**explicit Manager** (*Library* \*library)

**bool readFile** (**const** `std::string` &path, **bool** readOnly = true, **bool** trustLibrary = true)  
 Read a `library.xml` and add book in the file to the library.

#### Parameters

- **path** – The (utf8) path to the `library.xml`.
- **readOnly** – Set if the library path could be overwritten latter with updated content.
- **trustLibrary** – use book metadata coming from XML.

**Returns** True if file has been properly parsed.

**void reload** (**const** *Paths* &paths)  
 Sync the contents of the library with one or more `library.xml` files.

The metadata of the library files is trusted unconditionally. Any books not present in the input `library.xml` files are removed from the library.

**Parameters** **paths** – The (utf8) paths to the `library.xml` files.

**bool readXml** (**const** `std::string` &xml, **const** **bool** readOnly = true, **const** `std::string` &libraryPath = "", **bool** trustLibrary = true)  
 Load a library content store in the string.

#### Parameters

- **xml** – The content corresponding of the library xml
- **readOnly** – Set if the library path could be overwritten latter with updated content.
- **libraryPath** – The library path (used to resolve relative path)

**Returns** True if the content has been properly parsed.

**bool readOpds** (**const** `std::string` &content, **const** `std::string` &urlHost)  
 Load a library content stored in a OPDS stream.

#### Parameters

- **content** – The content of the OPDS stream.
- **readOnly** – Set if the library path could be overwritten later with updated content.

- **libraryPath** – The library path (used to resolve relative path)

**Returns** True if the content has been properly parsed.

bool **readBookmarkFile** (const std::string &path)  
Load a bookmark file.

**Parameters** **path** – The path of the file to read.

**Returns** True if the content has been properly parsed.

std::string **addBookFromPathAndGetId** (const std::string &pathToOpen, const std::string &pathToSave = "", const std::string &url = "", const bool checkMetaData = false)

Add a book to the library.

**Parameters**

- **pathToOpen** – The path to the zim file to add.
- **pathToSave** – The path to store in the library in place of pathToOpen.
- **url** – The url of the book to store in the library.
- **checkMetaData** – Tell if we check metadata before adding book to the library.

**Returns** The id of the book if the book has been added to the library. Else, an empty string.

bool **addBookFromPath** (const std::string &pathToOpen, const std::string &pathToSave = "", const std::string &url = "", const bool checkMetaData = false)

Add a book to the library.

**Parameters**

- **pathToOpen** – The path to the zim file to add.
- **pathToSave** – The path to store in the library in place of pathToOpen.
- **url** – The url of the book to store in the library.
- **checkMetaData** – Tell if we check metadata before adding book to the library.

**Returns** True if the book has been added to the library.

## Public Members

std::string **writableLibraryPath**

bool **m\_hasSearchResult** = false

uint64\_t **m\_totalBooks** = 0

uint64\_t **m\_startIndex** = 0

uint64\_t **m\_itemsPerPage** = 0

### Protected Functions

```
bool readBookFromPath (const std::string &path, Book *book)
bool parseXmlDom (const pugi::xml_document &doc, bool readOnly, const std::string &library-
                  Path, bool trustLibrary)
bool parseOpdsDom (const pugi::xml_document &doc, const std::string &urlHost)
```

### Protected Attributes

```
std::shared_ptr<kiwix::LibraryManipulator> manipulator
```

### Class NameMapper

- Defined in file\_include\_name\_mapper.h

### Inheritance Relationships

#### Derived Types

- public kiwix::HumanReadableNameMapper (*Class HumanReadableNameMapper*)
- public kiwix::IdNameMapper (*Class IdNameMapper*)
- public kiwix::UpdatableNameMapper (*Class UpdatableNameMapper*)

### Class Documentation

**class** kiwix::NameMapper

Subclassed by *kiwix::HumanReadableNameMapper*, *kiwix::IdNameMapper*, *kiwix::UpdatableNameMapper*

#### Public Functions

```
virtual ~NameMapper () = default
virtual std::string getNameForId (const std::string &id) const = 0
virtual std::string getIdForName (const std::string &name) const = 0
```

### Class NoEntry

- Defined in file\_include\_entry.h

## Inheritance Relationships

### Base Type

- `public exception`

### Class Documentation

```
class NoEntry : public exception
```

### Class OPDSDumper

- Defined in `file_include_opds_dumper.h`

### Class Documentation

```
class kiwix::OPDSDumper
    A tool to dump a Library into a opds stream.
```

#### Public Functions

```
OPDSDumper () = default
```

```
OPDSDumper (Library *library)
```

```
~OPDSDumper ()
```

```
std::string dumpOPDSFeed (const std::vector<std::string> &bookIds, const std::string &query,
                          const
```

Dump the OPDS feed.

##### Parameters

- **bookIds** – the ids of the books to include in the feed
- **query** – the query used to obtain the list of book ids

**Returns** The OPDS feed.

```
std::string dumpOPDSFeedV2 (const std::vector<std::string> &bookIds, const std::string &query,
                           bool partial) const
```

Dump the OPDS feed.

##### Parameters

- **bookIds** – the ids of the books to include in the feed
- **query** – the query used to obtain the list of book ids
- **partial** – whether the feed should include partial or complete entries

**Returns** The OPDS feed.

```
std::string dumpOPDSCompleteEntry (const std::string &bookId) const
```

Dump the OPDS complete entry document.

**Parameters** **bookId** – the id of the book



**Returns** The OPDS complete entry document.

`std::string categoriesOPDSFeed() const`  
Dump the categories OPDS feed.

**Returns** The OPDS feed.

`std::string languagesOPDSFeed() const`  
Dump the languages OPDS feed.

**Returns** The OPDS feed.

`inline void setLibraryId(const std::string &id)`  
Set the id of the library.

**Parameters** `id` – the id to use.

`inline void setRootLocation(const std::string &rootLocation)`  
Set the root location used when generating url.

**Parameters** `rootLocation` – the root location to use.

`void setOpenSearchInfo(int totalResult, int startIndex, int count)`  
Set some informations about the search results.

**Parameters**

- **totalResult** – the total number of results of the search.
- **startIndex** – the start index of the result.
- **count** – the number of result of the current set (or page).

### Protected Attributes

```
kiwix::Library *library
std::string libraryId
std::string rootLocation
int m_totalResults
int m_startIndex
int m_count
```

### Class Reader

- Defined in file\_include\_reader.h

### Class Documentation

```
class kiwix::Reader
```

## Public Functions

**explicit Reader (const string zimFilePath)**

Create a [Reader](#) to read a zim file specified by zimFilePath.

**Parameters zimFilePath** – The path to the zim file to read. The zim file can be splitted (.zimaa, .zimab, ...). In this case, the file path must still point to the unsplitted path as if the file were not splitted (.zim extension).

**explicit Reader (const std::shared\_ptr<zim::Archive> archive)**

Create a [Reader](#) to read a zim file given by the Archive.

**Parameters archive** – The shared pointer to the Archive object.

**explicit Reader (int fd)**

**Reader (int fd, zim::offset\_type offset, zim::size\_type size)**

**~Reader ()** = default

unsigned int **getArticleCount () const**

Get the number of “displayable” entries in the zim file.

**Returns** If the zim file has a /M/Counter metadata, return the number of entries with the ‘text/html’ MIMEtype specified in the metadata. Else return the number of entries in the ‘A’ namespace.

unsigned int **getMediaCount () const**

Get the number of media in the zim file.

**Returns** If the zim file has a /M/Counter metadata, return the number of entries with the ‘image/jpeg’, ‘image/gif’ and ‘image/png’ in the metadata. Else return the number of entries in the ‘I’ namespace.

unsigned int **getGlobalCount () const**

Get the number of all entries in the zim file.

**Returns** Return the number of all the entries, whatever their MIMEtype or their namespace.

string **getZimFilePath () const**

Get the path of the zim file.

**Returns** the path of the zim file as given in the constructor.

string **getId () const**

Get the Id of the zim file.

**Returns** The uuid stored in the zim file.

[Entry](#) **getRandomPage () const**

Get a random page.

**Returns** A random [Entry](#). The entry is picked from all entries in the ‘A’ namespace. The main entry is excluded from the potential results.

[Entry](#) **getMainPage () const**

Get the entry of the main page.

**Returns** [Entry](#) of the main page as specified in the zim file.

bool **getMetadata (const string &name, string &value) const**

Get the content of a metadata.

**Parameters**

- **name** – [in] The name of the metadata.
- **value** – [out] The value will be set to the content of the metadata.

**Returns** True if it was possible to get the content of the metadata.

string **getName () const**

Get the name of the zim file.

**Returns** The name of the zim file as specified in the zim metadata.

string **getTitle () const**

Get the title of the zim file.

**Returns** The title of zim file as specified in the zim metadata. If no title has been set, return a title computed from the file path.

string **getCreator () const**

Get the creator of the zim file.

**Returns** The creator of the zim file as specified in the zim metadata.

string **getPublisher () const**

Get the publisher of the zim file.

**Returns** The publisher of the zim file as specified in the zim metadata.

string **getDate () const**

Get the date of the zim file.

**Returns** The date of the zim file as specified in the zim metadata.

string **getDescription () const**

Get the description of the zim file.

**Returns** The description of the zim file as specified in the zim metadata. If no description has been set, return the subtitle.

string **getLongDescription () const**

Get the long description of the zim file.

**Returns** The long description of the zim file as specified in the zim metadata.

string **getLanguage () const**

Get the language of the zim file.

**Returns** The language of the zim file as specified in the zim metadata.

string **getLicense () const**

Get the license of the zim file.

**Returns** The license of the zim file as specified in the zim metadata.

string **getTags (bool *original* = false) const**

Get the tags of the zim file.

**Parameters** **original** – If true, return the original tags as specified in the zim metadata. Else, try to convert it to the new ‘normalized’ format.

**Returns** The tags of the zim file.

string **getTagStr (const std::string &tagName) const**

Get the value (as a string) of a specific tag.

According to <https://wiki.openzim.org/wiki/Tags>

**Throws** **std::out\_of\_range** – if the specified tag is not found.

**Returns** The value of the specified tag.

bool **getTagBool** (const std::string &tagName) const  
Get the boolean value of a specific tag.

According to <https://wiki.openzim.org/wiki/Tags>

**Throws** **std::out\_of\_range** – if the specified tag is not found. **std::domain\_error** if the value of the tag cannot be convert to bool.

**Returns** The boolean value of the specified tag.

string **getRelation** () const  
Get the relations of the zim file.

**Returns** The relation of the zim file as specified in the zim metadata.

string **getFlavour** () const  
Get the flavour of the zim file.

**Returns** The flavour of the zim file as specified in the zim metadata.

string **getSource** () const  
Get the source of the zim file.

**Returns** The source of the zim file as specified in the zim metadata.

string **getScraper** () const  
Get the scraper of the zim file.

**Returns** The scraper of the zim file as specified in the zim metadata.

bool **getFavicon** (string &content, string &mimeType) const  
Get the favicon of the zim file.

**Parameters**

- **content** – [out] The content of the favicon.
- **mimeType** – [out] The mimeType of the favicon.

**Returns** True if a favicon has been found.

*Entry* **getEntryFromPath** (const std::string &path) const  
Get an entry associated to an path.

**Parameters** **path** – The path of the entry.

**Throws** **NoEntry** – If no entry correspond to the path.

**Returns** The entry.

*Entry* **getEntryFromEncodedPath** (const std::string &path) const  
Get an entry associated to an url encoded path.

Equivalent to `getEntryFromPath (urlDecode (path) ) ;`

**Parameters** **path** – The url encoded path.

**Throws** **NoEntry** – If no entry correspond to the path.

**Returns** The entry.

*Entry* **getEntryFromTitle** (const std::string &title) const  
Get un entry associated to a title.

**Parameters** **title** – The title.

**Returns** The entry throw *NoEntry* If no entry correspond to the url.

bool **searchSuggestions** (**const** string &prefix, unsigned int suggestionsCount, **const** bool reset = true)

Search for entries with title starting with prefix (case sensitive).

Suggestions are stored in an internal vector and can be retrieved using getNextSuggestion method. This method is not thread safe and is deprecated. Use : bool searchSuggestions(const string& prefix, unsigned int suggestionsCount, SuggestionsList\_t& results);

#### Parameters

- **prefix** – The prefix to search.
- **suggestionsCount** – How many suggestions to search for.
- **reset** – If true, remove previous suggestions in the internal vector. If false, add suggestions to the internal vector (until internal vector size is suggestionCount (or no more suggestion))

**Returns** True if some suggestions have been added to the internal vector.

bool **searchSuggestions** (**const** string &prefix, unsigned int suggestionsCount, SuggestionsList\_t &results)

Search for entries with title starting with prefix (case sensitive).

Suggestions are added to the result vector.

#### Parameters

- **prefix** – The prefix to search.
- **suggestionsCount** – How many suggestions to search for.
- **result** – The vector where to store the suggestions.

**Returns** True if some suggestions have been added to the vector.

bool **searchSuggestionsSmart** (**const** string &prefix, unsigned int suggestionsCount)

Search for entries for the given prefix.

If the zim file has a internal fulltext index, the suggestions will be searched using it. Else the suggestions will be search using searchSuggestions while trying to be smart about case sensitivity (using getTitleVariants).

In any case, suggestions are stored in an internal vector and can be retrieved using getNextSuggestion method. The internal vector will be reset. This method is not thread safe and is deprecated. Use : bool searchSuggestionsSmart(const string& prefix, unsigned int suggestionCount, SuggestionsList\_t& results);

#### Parameters

- **prefix** – The prefix to search for.
- **suggestionsCount** – How many suggestions to search for.

bool **searchSuggestionsSmart** (**const** string &prefix, unsigned int suggestionsCount, SuggestionsList\_t &results)

Search for entries for the given prefix.

If the zim file has a internal fulltext index, the suggestions will be searched using it. Else the suggestions will be search using searchSuggestions while trying to be smart about case sensitivity (using getTitleVariants).

In any case, suggestions are stored in an internal vector and can be retrieved using getNextSuggestion method. The internal vector will be reset.

**Parameters**

- **prefix** – The prefix to search for.
- **suggestionsCount** – How many suggestions to search for.
- **results** – The vector where to store the suggestions

**Returns** True if some suggestions have been added to the results.

bool **pathExists** (const string &path) const  
Check if the path exists in the zim file.

**Parameters** **path** – the path to check.

**Returns** True if the path exists in the zim file.

bool **hasFulltextIndex** () const  
Check if the zim file has a embedded fulltext index.

**Returns** True if the zim file has a embedded fulltext index and is not split (else the fulltext is not accessible).

std::vector<std::string> **getTitleVariants** (const std::string &title) const  
Get potential case title variations for a title.

**Parameters** **title** – a title.

**Returns** the list of variations.

bool **getNextSuggestion** (string &title)  
Get the next suggestion title.

**Parameters** **title** – [out] the title of the suggestion.

**Returns** True if title has been set.

bool **getNextSuggestion** (string &title, string &url)  
Get the next suggestion title and url.

**Parameters**

- **title** – [out] the title of the suggestion.
- **url** – [out] the url of the suggestion.

**Returns** True if title and url have been set.

bool **canCheckIntegrity** () const  
Get if we can check zim file integrity (has a checksum).

**Returns** True if zim file have a checksum.

bool **isCorrupted** () const  
Check is zim file is corrupted.

**Returns** True if zim file is corrupted.

unsigned int **getFileSize** () const  
Return the total size of the zim file.

If zim file is split, return the sum of all parts' size.

**Returns** Size of the size file is KiB.

zim::Archive \***getZimArchive** () const  
Get the zim file handler.

**Returns** The libzim file handler.

### Protected Attributes

`std::shared_ptr<zim::Archive> zimArchive`

`std::string zimFilePath`

`SuggestionsList_t suggestions`

`SuggestionsList_t::iterator suggestionsOffset`

### Class Result

- Defined in `file_include_searcher.h`

### Class Documentation

```
class kiwix::Result
```

#### Public Functions

```
inline virtual ~Result ()
```

```
virtual std::string get_url () = 0
```

```
virtual std::string get_title () = 0
```

```
virtual int get_score () = 0
```

```
virtual std::string get_snippet () = 0
```

```
virtual std::string get_content () = 0
```

```
virtual int get_wordCount () = 0
```

```
virtual int get_size () = 0
```

```
virtual std::string get_zimId () = 0
```

### Class Searcher

- Defined in `file_include_searcher.h`

### Class Documentation

```
class kiwix::Searcher
```

The *Searcher* class is responsible to do different kind of search using the fulltext index.

## Public Functions

### **Searcher** ( )

The default constructor.

### **~Searcher** ( )

### bool **add\_reader** (*Reader* \*reader)

Add a reader (containing embedded fulltext index) to the search.

**Parameters** **reader** – The *Reader* for the zim containing the fulltext index.

**Returns** true if the reader has been added. false if the reader cannot be added (no embedded fulltext index present)

### *Reader* \***get\_reader** (int index)

### void **search** (const std::string &search, unsigned int resultStart, unsigned int maxResultCount, const bool verbose = false)

Start a search on the zim associated to the *Searcher*.

Search results should be retrived using the getNextResult method.

#### **Parameters**

- **search** – The search query.
- **resultStart** – the start offset of the search results (used for pagination).
- **maxResultCount** – Maximum results to get from start (used for pagination).
- **verbose** – print some info on stdout if true.

### void **geo\_search** (float latitude, float longitude, float distance, unsigned int resultStart, unsigned int maxResultCount, const bool verbose = false)

Start a geographique search. The search return result for entry in a disc of center latitude/longitude and radius distance.

Search results should be retrived using the getNextResult method.

#### **Parameters**

- **latitude** – The latitude of the center point.
- **longitude** – The longitude of the center point.
- **distance** – The radius of the disc.
- **resultStart** – the start offset of the search results (used for pagination).
- **maxResultCount** – Maximum number of results to get from start (used for pagination).
- **verbose** – print some info on stdout if true.

### void **suggestions** (std::string &search, const bool verbose = false)

Start a suggestion search. The search made depend of the “version” of the embedded index.

- If the index is newer enough and have a title namespace, the search is made in the titles only.
- Else the search is made on the whole article content. In any case, the search is made “partial” (as adding ‘\*’ at the end of the query)

#### **Parameters**

- **search** – The search query.
- **verbose** – print some info on stdout if true.



*Result* \*getNextResult ()

Get the next result of a started search. This is the method to use to loop over the search results.

void restart\_search ()

Restart the previous search. Next call to getNextResult will return the first result.

unsigned int getEstimatedResultCount ()

Get a estimation of the result count.

zim::SearchResultSet getSearchResultSet ()

Get a SearchResultSet object for current search

inline unsigned int getResultStart ()

inline unsigned int getMaxResultCount ()

### Protected Functions

std::string beautifyInteger (const unsigned int *number*)

void closeIndex ()

void searchInIndex (string &*search*, const unsigned int *resultStart*, const unsigned int *maxResultCount*, const bool *verbose* = false)

### Protected Attributes

std::vector<*Reader*\*> readers

std::unique\_ptr<SearcherInternal> internal

std::unique\_ptr<SuggestionInternal> suggestionInternal

std::string searchPattern

unsigned int estimatedResultCount

unsigned int resultStart

unsigned int maxResultCount

### Class SearchRenderer

- Defined in file\_include\_search\_renderer.h

### Class Documentation

**class** kiwix::SearchRenderer

The SearcherRenderer class is used to render a search result to a html page.

## Public Functions

**SearchRenderer** (*Searcher* \**searcher*, *NameMapper* \**mapper*)

The default constructor.

**Parameters** **humanReadableName** – The global zim’s humanReadableName. Used to generate pagination links.

**SearchRenderer** (zim::SearchResultSet *srs*, *NameMapper* \**mapper*, unsigned int *start*, unsigned int *estimatedResultCount*)

**~SearchRenderer** ()

void **setSearchPattern** (const std::string &*pattern*)

void **setSearchContent** (const std::string &*name*)

Set the search content id.

void **setProtocolPrefix** (const std::string &*prefix*)

Set protocol prefix.

void **setSearchProtocolPrefix** (const std::string &*prefix*)

Set search protocol prefix.

**inline** void **setPageLength** (unsigned int *pageLength*)

set result count per page

std::string **getHtml** ()

Generate the html page with the results of the search.

## Protected Functions

std::string **beautifyInteger** (const unsigned int *number*)

## Protected Attributes

zim::SearchResultSet **m\_srs**

*NameMapper* \***mp\_nameMapper**

std::string **searchContent**

std::string **searchPattern**

std::string **protocolPrefix**

std::string **searchProtocolPrefix**

unsigned int **pageLength**

unsigned int **estimatedResultCount**

unsigned int **resultStart**

## Class Server

- Defined in file\_include\_server.h

## Class Documentation

**class** `kiwix::Server`

### Public Functions

**Server** (*Library* \*library, *NameMapper* \*nameMapper = nullptr)

The default constructor.

**Parameters** `library` – The library to serve.

**virtual** `~Server` ()

`bool` **start** ()

Serve the content.

`void` **stop** ()

Stop the daemon.

`void` **setRoot** (`const` std::string &root)

**inline** `void` **setAddress** (`const` std::string &addr)

**inline** `void` **setPort** (int port)

**inline** `void` **setNbThreads** (int threads)

**inline** `void` **setVerbose** (bool verbose)

**inline** `void` **setIndexTemplateString** (`const` std::string &indexTemplateString)

**inline** `void` **setTaskbar** (bool withTaskbar, bool withLibraryButton)

**inline** `void` **setBlockExternalLinks** (bool blockExternalLinks)

### Protected Attributes

*Library* \***mp\_library**

*NameMapper* \***mp\_nameMapper**

std::string **m\_root** = ""

std::string **m\_addr** = ""

std::string **m\_indexTemplateString** = ""

int **m\_port** = 80

int **m\_nbThreads** = 1

bool **m\_verbose** = false

bool **m\_withTaskbar** = true

bool **m\_withLibraryButton** = true

bool **m\_blockExternalLinks** = false

```
std::unique_ptr<InternalServer> mp_server
```

## Class SuggestionItem

- Defined in file\_include\_reader.h

## Class Documentation

**class** `kiwix::SuggestionItem`

The *SuggestionItem* is a helper class that contains the info about a single suggestion item.

### Public Functions

```
inline explicit SuggestionItem(const std::string &title, const std::string &normalizedTitle, const std::string &path, const std::string &snippet = "")
inline const std::string &getTitle() const
inline const std::string &getNormalizedTitle() const
inline const std::string &getPath() const
inline const std::string &getSnippet() const
inline bool hasSnippet() const
```

## Class UpdatableNameMapper

- Defined in file\_include\_name\_mapper.h

## Inheritance Relationships

### Base Type

- `public kiwix::NameMapper` (*Class NameMapper*)

## Class Documentation

**class** `kiwix::UpdatableNameMapper` : `public kiwix::NameMapper`

### Public Functions

```
UpdatableNameMapper(Library &library, bool withAlias)
virtual std::string getNameForId(const std::string &id) const
virtual std::string getIdForName(const std::string &name) const
void update()
```

### 2.3.3 Enums

#### Enum supportedListMode

- Defined in file\_include\_library.h

#### Enum Documentation

**enum** kiwix::supportedListMode

*Values:*

```
enumerator ALL
enumerator LOCAL
enumerator REMOTE
enumerator NOLOCAL
enumerator NOREMOTE
enumerator VALID
enumerator NOVALID
```

#### Enum supportedListSortBy

- Defined in file\_include\_library.h

#### Enum Documentation

**enum** kiwix::supportedListSortBy

*Values:*

```
enumerator UNSORTED
enumerator TITLE
enumerator SIZE
enumerator DATE
enumerator CREATOR
enumerator PUBLISHER
```

### 2.3.4 Functions

#### Function kiwix::appendToDirectory

- Defined in file\_include\_tools.h

## Function Documentation

`std::string kiwix::appendToDirectory(const std::string &basePath, const std::string &relativePath)`

Append a path to another one.

This function is provided as a small helper. It is probably better to use native tools to manipulate paths.

### Parameters

- **basePath** – the base path.
- **relativePath** – a path to add to the base path, must be a relative path.

**Returns** The concatenation of the paths, using the right separator.

## Function `kiwix::computeAbsolutePath`

- Defined in `file_include_tools.h`

## Function Documentation

`std::string kiwix::computeAbsolutePath(const std::string &path, const std::string &relativePath)`  
Compute the absolute path of a relative path based on another one

Equivalent to `appendToDirectory` followed by a normalization of the path.

This function is provided as a small helper. It is probably better to use native tools to manipulate paths.

### Parameters

- **path** – the base path (if empty, current directory is taken).
- **relativePath** – the relative path.

**Returns** a absolute path.

## Function `kiwix::computeRelativePath`

- Defined in `file_include_tools.h`

## Function Documentation

`std::string kiwix::computeRelativePath(const std::string &path, const std::string &absolutePath)`  
Compute the relative path of a path relative to another one

This function is provided as a small helper. It is probably better to use native tools to manipulate paths.

### Parameters

- **path** – the base path.
- **absolutePath** – the absolute path to find the relative path for.

**Returns** a relative path (pointing to `absolutePath`, relative to `path`).

### Function `kiwix::converta2toa3`

- Defined in `file_include_tools.h`

#### Function Documentation

`std::string kiwix::converta2toa3 (const std::string &a2code)`

Convert language code from iso2 code to iso3

This function is provided as a small helper. It is probably better to use native tools to manipulate locales.

**Parameters** `a2code` – a iso2 code string.

**Throws** `std::out_of_range` – if iso2 code is not known.

**Returns** the corresponding iso3 code.

### Function `kiwix::fileExists`

- Defined in `file_include_tools.h`

#### Function Documentation

`bool kiwix::fileExists (const std::string &path)`

checks if file exists.

This function returns boolean stating if file exists or not.

**Parameters** `path` – The absolute path provided in string format.

**Returns** Boolean representing if file exists or not.

### Function `kiwix::getCurrentDirectory`

- Defined in `file_include_tools.h`

#### Function Documentation

`std::string kiwix::getCurrentDirectory ()`

Return the current directory.

**Returns** the current directory (utf8 encoded)

### Function `kiwix::getDataDirectory`

- Defined in `file_include_tools.h`

## Function Documentation

`std::string kiwix::getDataDirectory()`

Return the data directory.

The data directory is a directory where to put data (zim files, ...) It depends of the platform and it may be changed by user using environment variable.

The resolution order is :

- `KIWIX_DATA_DIR` env variable (if set).
- On Windows : `.$APPDATA/kiwix` if `$APPDATA` is set . `.$USERPROFILE/kiwix` if `.$USERPROFILE` is set
- Else : `.$XDG_DATA_HOME/kiwix` if `.$XDG_DATA_HOME` is set . `.$HOME/.local/share/kiwix` if `.$HOME` is set
- current directory

**Returns** the path of the data directory (utf8 encoded)

## Function `kiwix::getExecutablePath`

- Defined in `file_include_tools.h`

## Function Documentation

`std::string kiwix::getExecutablePath (bool realPathOnly = false)`

Return the path of the executable

Some application may be packaged in auto extractible archive (Appimage) and the real executable is different of the path of the archive. If `realPathOnly` is true, return the path of the real executable instead of the archive launched by the user.

**Parameters** `realPathOnly` – If we must return the real path of the executable.

**Returns** the path of the executable (utf8 encoded)

## Function `kiwix::getFileContent`

- Defined in `file_include_tools.h`

## Function Documentation

`std::string kiwix::getFileContent (const std::string &path)`

Extracts content from given file.

This function provides content of a file provided it's path.

**Parameters** `path` – The absolute path provided in string format.

**Returns** Content of corresponding file in string format.



### Function `kiwix::getLastPathElement`

- Defined in `file_include_tools.h`

#### Function Documentation

`std::string kiwix::getLastPathElement (const std::string &path)`

Get the last element of a path.

This function is provided as a small helper. It is probably better to use native tools to manipulate paths.

**Parameters** `path` – a path.

**Returns** The base name of the path or empty string if none (ending with a separator).

### Function `kiwix::getMimeTypeForFile`

- Defined in `file_include_tools.h`

#### Function Documentation

`std::string kiwix::getMimeTypeForFile (const std::string &filename)`

provides mimetype from filename.

This function provides mimetype from file-name.

**Parameters** `filename` – string containing filename.

**Returns** mimetype from filename in string format.

### Function `kiwix::getSearchUrl`

- Defined in `file_include_opds_catalog.h`

#### Function Documentation

`std::string kiwix::getSearchUrl (const Filter &f)`

### Function `kiwix::isRelativePath`

- Defined in `file_include_tools.h`

## Function Documentation

bool **kiwix::isRelativePath** (const std::string &path)

Tell if the path is a relative path.

This function is provided as a small helper. It is probably better to use native tools to manipulate paths.

**Parameters** **path** – A utf8 encoded path.

**Returns** true if the path is relative.

## Function **kiwix::removeLastPathElement**

- Defined in file\_include\_tools.h

## Function Documentation

std::string **kiwix::removeLastPathElement** (const std::string &path)

Remove the last element of a path.

This function is provided as a small helper. It is probably better to use native tools to manipulate paths.

**Parameters** **path** – a path.

**Returns** The parent directory (or empty string if none).

## Function **kiwix::sleep**

- Defined in file\_include\_tools.h

## Function Documentation

void **kiwix::sleep** (unsigned int *milliseconds*)

Sleep the current thread.

This function is provided as a small helper. It is probably better to use native tools.

**Parameters** **milliseconds** – The number of milliseconds to wait for.

## Function **kiwix::split**

- Defined in file\_include\_tools.h

## Function Documentation

std::vector<std::string> **kiwix::split** (const std::string &str, const std::string &delims, bool  
dropEmpty = true, bool keepDelim = false)

Split a string

This function is provided as a small helper. It is probably better to use native tools.

Assuming text = “foo;bar;baz,oups;”

```
split(text, ":", true, true) => ["foo", ":", ":", "bar", ":", "baz,oups", ":", ""] split(text, ":", true, false) => ["foo",  
"bar", "baz,oups"] (default) split(text, ":", false, true) => ["foo", ":", ":", ":", "bar", ":", "baz,oups", ":", ""]  
split(text, ":", false, false) => ["foo", ":", "bar", "baz,oups", ""]
```

#### Parameters

- **str** – The string to split.
- **delims** – A string of potential delimiters. Each character in the string can be a individual delimiters.
- **dropEmpty** – true if empty part must be dropped from the result.
- **keepDelim** – true if delimiter must be included from the result.

**Returns** a list of part (potentially containing delimiters)

## 2.3.5 Defines

### Define DEPRECATED

- Defined in file\_include\_common.h

### Define Documentation

**DEPRECATED**

### Define KIWIX\_LIBRARY\_VERSION

- Defined in file\_include\_library.h

### Define Documentation

**KIWIX\_LIBRARY\_VERSION**

## 2.3.6 Typedefs

### Typedef kiwix::offset\_type

- Defined in file\_include\_common.h

### Typedef Documentation

**typedef** zim::offset\_type kiwix::offset\_type

**Typedef kiwix::size\_type**

- Defined in file\_include\_common.h

**Typedef Documentation**

```
typedef zim::size_type kiwix::size_type
```

**Typedef kiwix::SuggestionsList\_t**

- Defined in file\_include\_reader.h

**Typedef Documentation**

```
using kiwix::SuggestionsList_t = std::vector<SuggestionItem>
```

The *Reader* class is the class who allow to get an entry content from a zim file.

## D

DEPRECATED (*C macro*), 47

## K

kiwix::appendToDirectory (*C++ function*), 42

kiwix::AriaError (*C++ class*), 6

kiwix::AriaError::AriaError (*C++ function*), 6

kiwix::Book (*C++ class*), 7

kiwix::Book::~~Book (*C++ function*), 7

kiwix::Book::Book (*C++ function*), 7

kiwix::Book::getArticleCount (*C++ function*), 8

kiwix::Book::getCategory (*C++ function*), 8

kiwix::Book::getCreator (*C++ function*), 8

kiwix::Book::getDate (*C++ function*), 8

kiwix::Book::getDescription (*C++ function*), 7

kiwix::Book::getDownloadId (*C++ function*), 8

kiwix::Book::getFavicon (*C++ function*), 8

kiwix::Book::getFaviconMimeType (*C++ function*), 8

kiwix::Book::getFaviconUrl (*C++ function*), 8

kiwix::Book::getFlavour (*C++ function*), 8

kiwix::Book::getHumanReadableIdFromPath (*C++ function*), 7

kiwix::Book::getId (*C++ function*), 7

kiwix::Book::getIllustrations (*C++ function*), 8

kiwix::Book::getLanguage (*C++ function*), 7

kiwix::Book::getMediaCount (*C++ function*), 8

kiwix::Book::getName (*C++ function*), 8

kiwix::Book::getOrigId (*C++ function*), 8

kiwix::Book::getPath (*C++ function*), 7

kiwix::Book::getPublisher (*C++ function*), 8

kiwix::Book::getSize (*C++ function*), 8

kiwix::Book::getTagBool (*C++ function*), 8

kiwix::Book::getTags (*C++ function*), 8

kiwix::Book::getTagStr (*C++ function*), 8

kiwix::Book::getTitle (*C++ function*), 7

kiwix::Book::getUrl (*C++ function*), 8

kiwix::Book::Illustration (*C++ class*), 9, 10

kiwix::Book::Illustration::getData (*C++ function*), 10

kiwix::Book::Illustration::height (*C++ member*), 10

kiwix::Book::Illustration::mimeType (*C++ member*), 10

kiwix::Book::Illustration::url (*C++ member*), 10

kiwix::Book::Illustration::width (*C++ member*), 10

kiwix::Book::Illustrations (*C++ type*), 7

kiwix::Book::isPathValid (*C++ function*), 7

kiwix::Book::m\_articleCount (*C++ member*), 9

kiwix::Book::m\_category (*C++ member*), 9

kiwix::Book::m\_creator (*C++ member*), 9

kiwix::Book::m\_date (*C++ member*), 9

kiwix::Book::m\_description (*C++ member*), 9

kiwix::Book::m\_downloadId (*C++ member*), 9

kiwix::Book::m\_flavour (*C++ member*), 9

kiwix::Book::m\_id (*C++ member*), 9

kiwix::Book::m\_illustrations (*C++ member*), 9

kiwix::Book::m\_language (*C++ member*), 9

kiwix::Book::m\_mediaCount (*C++ member*), 9

kiwix::Book::m\_name (*C++ member*), 9

kiwix::Book::m\_origId (*C++ member*), 9

kiwix::Book::m\_path (*C++ member*), 9

kiwix::Book::m\_pathValid (*C++ member*), 9

kiwix::Book::m\_publisher (*C++ member*), 9

kiwix::Book::m\_readOnly (*C++ member*), 9

kiwix::Book::m\_size (*C++ member*), 9

kiwix::Book::m\_tags (*C++ member*), 9

kiwix::Book::m\_title (*C++ member*), 9

kiwix::Book::m\_url (*C++ member*), 9

kiwix::Book::missingDefaultIllustration (*C++ member*), 9

kiwix::Book::readOnly (*C++ function*), 7

kiwix::Book::setArticleCount (*C++ function*), 8

kiwix::Book::setCreator (*C++ function*), 8

kiwix::Book::setDate (*C++ function*), 8

```

kiwix::Book::setDescription (C++ function), 8
kiwix::Book::setDownloadId (C++ function), 9
kiwix::Book::setFlavour (C++ function), 8
kiwix::Book::setId (C++ function), 8
kiwix::Book::setLanguage (C++ function), 8
kiwix::Book::setMediaCount (C++ function), 8
kiwix::Book::setName (C++ function), 8
kiwix::Book::setOrigId (C++ function), 8
kiwix::Book::setPath (C++ function), 8
kiwix::Book::setPathValid (C++ function), 8
kiwix::Book::setPublisher (C++ function), 8
kiwix::Book::setReadOnly (C++ function), 8
kiwix::Book::setSize (C++ function), 9
kiwix::Book::setTags (C++ function), 8
kiwix::Book::setTitle (C++ function), 8
kiwix::Book::setUrl (C++ function), 8
kiwix::Book::update (C++ function), 7
kiwix::Book::updateFromOpds (C++ function), 7
kiwix::Book::updateFromXml (C++ function), 7
kiwix::Bookmark (C++ class), 11
kiwix::Bookmark::~Bookmark (C++ function), 11
kiwix::Bookmark::Bookmark (C++ function), 11
kiwix::Bookmark::getBookId (C++ function), 11
kiwix::Bookmark::getBookTitle (C++ function), 11
kiwix::Bookmark::getDate (C++ function), 11
kiwix::Bookmark::getLanguage (C++ function), 11
kiwix::Bookmark::getTitle (C++ function), 11
kiwix::Bookmark::getUrl (C++ function), 11
kiwix::Bookmark::m_bookId (C++ member), 11
kiwix::Bookmark::m_bookTitle (C++ member), 11
kiwix::Bookmark::m_date (C++ member), 11
kiwix::Bookmark::m_language (C++ member), 11
kiwix::Bookmark::m_title (C++ member), 11
kiwix::Bookmark::m_url (C++ member), 11
kiwix::Bookmark::setBookId (C++ function), 11
kiwix::Bookmark::setBookTitle (C++ function), 11
kiwix::Bookmark::setDate (C++ function), 11
kiwix::Bookmark::setLanguage (C++ function), 11
kiwix::Bookmark::setTitle (C++ function), 11
kiwix::Bookmark::setUrl (C++ function), 11
kiwix::Bookmark::updateFromXml (C++ function), 11
kiwix::computeAbsolutePath (C++ function), 42
kiwix::computeRelativePath (C++ function), 42
kiwix::converta2toa3 (C++ function), 43
kiwix::Download (C++ class), 12
kiwix::Download::cancelDownload (C++ function), 12
kiwix::Download::Download (C++ function), 12
kiwix::Download::getCompletedLength (C++ function), 12
kiwix::Download::getDid (C++ function), 12
kiwix::Download::getDownloadSpeed (C++ function), 12
kiwix::Download::getFollowedBy (C++ function), 12
kiwix::Download::getPath (C++ function), 12
kiwix::Download::getStatus (C++ function), 12
kiwix::Download::getTotalLength (C++ function), 12
kiwix::Download::getUris (C++ function), 12
kiwix::Download::getVerifiedLength (C++ function), 12
kiwix::Download::m_completedLength (C++ member), 13
kiwix::Download::m_did (C++ member), 13
kiwix::Download::m_downloadSpeed (C++ member), 13
kiwix::Download::m_followedBy (C++ member), 13
kiwix::Download::m_path (C++ member), 13
kiwix::Download::m_status (C++ member), 13
kiwix::Download::m_totalLength (C++ member), 13
kiwix::Download::m_uris (C++ member), 13
kiwix::Download::m_verifiedLength (C++ member), 13
kiwix::Download::mp_aria (C++ member), 13
kiwix::Download::pauseDownload (C++ function), 12
kiwix::Download::resumeDownload (C++ function), 12
kiwix::Download::StatusResult (C++ enum), 12
kiwix::Download::StatusResult::K_ACTIVE (C++ enumerator), 12
kiwix::Download::StatusResult::K_COMPLETE (C++ enumerator), 12
kiwix::Download::StatusResult::K_ERROR (C++ enumerator), 12
kiwix::Download::StatusResult::K_PAUSED (C++ enumerator), 12
kiwix::Download::StatusResult::K_REMOVED

```

(C++ enumerator), 12  
 kiwix::Download::StatusResult::K\_UNKNOWNkiwix::Filter::getAcceptTags (C++ function), 17  
 (C++ enumerator), 12  
 kiwix::Download::StatusResult::K\_WAITINGkiwix::Filter::getCategory (C++ function), 16  
 (C++ enumerator), 12  
 kiwix::Download::updateStatus (C++ function), 12  
 kiwix::Filter::getCreator (C++ function), 17  
 kiwix::DownloadedFile (C++ struct), 5  
 kiwix::Filter::getLang (C++ function), 16  
 kiwix::DownloadedFile::DownloadedFile (C++ function), 5  
 kiwix::Filter::getName (C++ function), 16  
 kiwix::DownloadedFile::path (C++ member), 5  
 kiwix::Filter::getPublisher (C++ function), 16  
 kiwix::DownloadedFile::success (C++ member), 5  
 kiwix::Filter::getQuery (C++ function), 16  
 kiwix::Filter::getRejectTags (C++ function), 17  
 kiwix::Downloader (C++ class), 13  
 kiwix::Filter::hasCategory (C++ function), 16  
 kiwix::Downloader::~Downloader (C++ function), 13  
 kiwix::Filter::hasCreator (C++ function), 17  
 kiwix::Downloader::close (C++ function), 13  
 kiwix::Filter::hasLang (C++ function), 16  
 kiwix::Downloader::Downloader (C++ function), 13  
 kiwix::Filter::hasName (C++ function), 16  
 kiwix::Downloader::getDownload (C++ function), 13  
 kiwix::Filter::hasPublisher (C++ function), 16  
 kiwix::Downloader::getDownloadIds (C++ function), 13  
 kiwix::Filter::hasQuery (C++ function), 16  
 kiwix::Downloader::getNbDownload (C++ function), 13  
 kiwix::Filter::lang (C++ function), 16  
 kiwix::Downloader::startDownload (C++ function), 13  
 kiwix::Filter::local (C++ function), 16  
 kiwix::Entry (C++ class), 14  
 kiwix::Filter::maxSize (C++ function), 16  
 kiwix::Entry::~Entry (C++ function), 14  
 kiwix::Filter::name (C++ function), 16  
 kiwix::Entry::Entry (C++ function), 14  
 kiwix::Filter::publisher (C++ function), 16  
 kiwix::Entry::getBlob (C++ function), 14  
 kiwix::Filter::query (C++ function), 16  
 kiwix::Entry::getContent (C++ function), 14  
 kiwix::Filter::queryIsPartial (C++ function), 16  
 kiwix::Entry::getDirectAccessInfo (C++ function), 14  
 kiwix::Filter::rejectTags (C++ function), 16  
 kiwix::Entry::getFinalEntry (C++ function), 15  
 kiwix::Filter::remote (C++ function), 16  
 kiwix::Entry::getMimeType (C++ function), 15  
 kiwix::Filter::Tags (C++ type), 16  
 kiwix::Entry::getPath (C++ function), 14  
 kiwix::Filter::valid (C++ function), 16  
 kiwix::Entry::getRedirectEntry (C++ function), 15  
 kiwix::getCurrentDirectory (C++ function), 43  
 kiwix::Entry::getSize (C++ function), 15  
 kiwix::getDataDirectory (C++ function), 44  
 kiwix::Entry::getTitle (C++ function), 14  
 kiwix::getExecutablePath (C++ function), 44  
 kiwix::Entry::getZimEntry (C++ function), 15  
 kiwix::getFileContent (C++ function), 44  
 kiwix::Entry::isDeleted (C++ function), 15  
 kiwix::getLastPathElement (C++ function), 45  
 kiwix::Entry::isLinkTarget (C++ function), 15  
 kiwix::getMimeTypeForFile (C++ function), 45  
 kiwix::Entry::isRedirect (C++ function), 15  
 kiwix::getSearchUrl (C++ function), 45  
 kiwix::fileExists (C++ function), 43  
 kiwix::HumanReadableNameMapper (C++ class), 17  
 kiwix::Filter (C++ class), 15  
 kiwix::HumanReadableNameMapper::~HumanReadableNameMapper (C++ function), 17  
 kiwix::Filter::~Filter (C++ function), 16  
 kiwix::HumanReadableNameMapper::getIdForName (C++ function), 17  
 kiwix::Filter::acceptTags (C++ function), 16  
 kiwix::HumanReadableNameMapper::getNameForId (C++ function), 17  
 kiwix::Filter::category (C++ function), 16  
 kiwix::HumanReadableNameMapper::HumanReadableNameMapper (C++ function), 17  
 kiwix::Filter::creator (C++ function), 16  
 kiwix::IdNameMapper (C++ class), 18  
 kiwix::IdNameMapper::getIdForName (C++ function), 18

---

```

kiwix::IdNameMapper::getNameForId (C++
    function), 18
kiwix::isRelativePath (C++ function), 46
kiwix::KiwixServe (C++ class), 18
kiwix::KiwixServe::~~KiwixServe (C++ func-
    tion), 18
kiwix::KiwixServe::getPort (C++ function),
    18
kiwix::KiwixServe::isRunning (C++ func-
    tion), 18
kiwix::KiwixServe::KiwixServe (C++ func-
    tion), 18
kiwix::KiwixServe::run (C++ function), 18
kiwix::KiwixServe::setPort (C++ function),
    18
kiwix::KiwixServe::shutDown (C++ function),
    18
kiwix::Library (C++ class), 19
kiwix::Library::~~Library (C++ function), 19
kiwix::Library::addBook (C++ function), 19
kiwix::Library::addBookmark (C++ function),
    19
kiwix::Library::addOrUpdateBook (C++
    function), 19
kiwix::Library::AttributeCounts (C++
    type), 19
kiwix::Library::BookIdCollection (C++
    type), 19
kiwix::Library::filter (C++ function), 20, 21
kiwix::Library::getArchiveById (C++ func-
    tion), 19
kiwix::Library::getBookById (C++ function),
    19
kiwix::Library::getBookByIdThreadSafe
    (C++ function), 19
kiwix::Library::getBookByPath (C++ func-
    tion), 19
kiwix::Library::getBookCount (C++ func-
    tion), 20
kiwix::Library::getBookmarks (C++ func-
    tion), 20
kiwix::Library::getBooksCategories (C++
    function), 20
kiwix::Library::getBooksCreators (C++
    function), 20
kiwix::Library::getBooksIds (C++ function),
    20
kiwix::Library::getBooksLanguages (C++
    function), 20
kiwix::Library::getBooksLanguagesWithCounts
    (C++ function), 20
kiwix::Library::getBooksPublishers (C++
    function), 20
kiwix::Library::getReaderById (C++ func-
    tion), 19
kiwix::Library::getRevision (C++ function),
    21
kiwix::Library::Library (C++ function), 19
kiwix::Library::listBooksIds (C++ func-
    tion), 21
kiwix::Library::operator= (C++ function), 19
kiwix::Library::removeBookById (C++ func-
    tion), 20
kiwix::Library::removeBookmark (C++ func-
    tion), 19
kiwix::Library::removeBooksNotUpdatedSince
    (C++ function), 22
kiwix::Library::Revision (C++ type), 19
kiwix::Library::sort (C++ function), 21
kiwix::Library::writeBookmarksToFile
    (C++ function), 20
kiwix::Library::writeToFile (C++ function),
    20
kiwix::LibraryBase (C++ class), 22
kiwix::LibraryBase::~~LibraryBase (C++
    function), 23
kiwix::LibraryBase::Entry (C++ struct), 6, 23
kiwix::LibraryBase::Entry::lastUpdatedRevision
    (C++ member), 6, 23
kiwix::LibraryBase::LibraryBase (C++
    function), 23
kiwix::LibraryBase::LibraryRevision
    (C++ type), 23
kiwix::LibraryBase::m_archives (C++ mem-
    ber), 23
kiwix::LibraryBase::m_bookDB (C++ mem-
    ber), 23
kiwix::LibraryBase::m_bookmarks (C++
    member), 23
kiwix::LibraryBase::m_books (C++ member),
    23
kiwix::LibraryBase::m_readers (C++ mem-
    ber), 23
kiwix::LibraryBase::m_revision (C++ mem-
    ber), 23
kiwix::LibraryBase::operator= (C++ func-
    tion), 23
kiwix::LibraryManipulator (C++ class), 23
kiwix::LibraryManipulator::~~LibraryManipulator
    (C++ function), 23
kiwix::LibraryManipulator::addBookmarkToLibrary
    (C++ function), 23
kiwix::LibraryManipulator::addBookToLibrary
    (C++ function), 23
kiwix::LibraryManipulator::bookmarkWasAddedToLibra
    (C++ function), 24
kiwix::LibraryManipulator::booksWereRemovedFromLib
    (C++ function), 24

```



---

kiwix::LibraryManipulator::bookWasAddedToLibrary (C++ function), 24	kiwix::LibraryManager::writableLibraryPath (C++ member), 26
kiwix::LibraryManipulator::getLibrary (C++ function), 23	kiwix::NameMapper (C++ class), 27
kiwix::LibraryManipulator::LibraryManipulator (C++ function), 23	kiwix::NameMapper::~~NameMapper (C++ function), 27
kiwix::LibraryManipulator::removeBooksNotUpdated (C++ function), 23	kiwix::NameMapper::getIdForName (C++ function), 27
kiwix::LibXMLDumper (C++ class), 24	kiwix::NameMapper::getNameForId (C++ function), 27
kiwix::LibXMLDumper::~~LibXMLDumper (C++ function), 24	kiwix::NoEntry (C++ class), 28
kiwix::LibXMLDumper::baseDir (C++ member), 24	kiwix::offset_type (C++ type), 47
kiwix::LibXMLDumper::dumpLibXMLBookmark (C++ function), 24	kiwix::OPDSDumper (C++ class), 28
kiwix::LibXMLDumper::dumpLibXMLContent (C++ function), 24	kiwix::OPDSDumper::~~OPDSDumper (C++ function), 28
kiwix::LibXMLDumper::library (C++ member), 24	kiwix::OPDSDumper::categoriesOPDSFeed (C++ function), 29
kiwix::LibXMLDumper::LibXMLDumper (C++ function), 24	kiwix::OPDSDumper::dumpOPDSCompleteEntry (C++ function), 28
kiwix::LibXMLDumper::setBaseDir (C++ function), 24	kiwix::OPDSDumper::dumpOPDSFeed (C++ function), 28
kiwix::LibXMLDumper::setLibrary (C++ function), 24	kiwix::OPDSDumper::dumpOPDSFeedV2 (C++ function), 28
kiwix::Manager (C++ class), 25	kiwix::OPDSDumper::languagesOPDSFeed (C++ function), 29
kiwix::Manager::addBookFromPath (C++ function), 26	kiwix::OPDSDumper::library (C++ member), 29
kiwix::Manager::addBookFromPathAndGetId (C++ function), 26	kiwix::OPDSDumper::libraryId (C++ member), 29
kiwix::Manager::m_hasSearchResult (C++ member), 26	kiwix::OPDSDumper::m_count (C++ member), 29
kiwix::Manager::m_itemsPerPage (C++ member), 26	kiwix::OPDSDumper::m_startIndex (C++ member), 29
kiwix::Manager::m_startIndex (C++ member), 26	kiwix::OPDSDumper::m_totalResults (C++ member), 29
kiwix::Manager::m_totalBooks (C++ member), 26	kiwix::OPDSDumper::OPDSDumper (C++ function), 28
kiwix::Manager::Manager (C++ function), 25	kiwix::OPDSDumper::rootLocation (C++ member), 29
kiwix::Manager::manipulator (C++ member), 27	kiwix::OPDSDumper::setLibraryId (C++ function), 29
kiwix::Manager::parseOpdsDom (C++ function), 27	kiwix::OPDSDumper::setOpenSearchInfo (C++ function), 29
kiwix::Manager::parseXmlDom (C++ function), 27	kiwix::OPDSDumper::setRootLocation (C++ function), 29
kiwix::Manager::Paths (C++ type), 25	kiwix::Reader (C++ class), 29
kiwix::Manager::readBookFromPath (C++ function), 27	kiwix::Reader::~~Reader (C++ function), 30
kiwix::Manager::readBookmarkFile (C++ function), 26	kiwix::Reader::canCheckIntegrity (C++ function), 34
kiwix::Manager::readFile (C++ function), 25	kiwix::Reader::getArticleCount (C++ function), 30
kiwix::Manager::readOpds (C++ function), 25	kiwix::Reader::getCreator (C++ function), 31
kiwix::Manager::readXml (C++ function), 25	kiwix::Reader::getDate (C++ function), 31
kiwix::Manager::reload (C++ function), 25	kiwix::Reader::getDescription (C++ function), 31

kiwix::Reader::getEntryFromEncodedPath (C++ function), 32  
 kiwix::Reader::getEntryFromPath (C++ function), 32  
 kiwix::Reader::getEntryFromTitle (C++ function), 32  
 kiwix::Reader::getFavicon (C++ function), 32  
 kiwix::Reader::getFileSize (C++ function), 34  
 kiwix::Reader::getFlavour (C++ function), 32  
 kiwix::Reader::getGlobalCount (C++ function), 30  
 kiwix::Reader::getId (C++ function), 30  
 kiwix::Reader::getLanguage (C++ function), 31  
 kiwix::Reader::getLicense (C++ function), 31  
 kiwix::Reader::getLongDescription (C++ function), 31  
 kiwix::Reader::getMainPage (C++ function), 30  
 kiwix::Reader::getMediaCount (C++ function), 30  
 kiwix::Reader::getMetadata (C++ function), 30  
 kiwix::Reader::getName (C++ function), 31  
 kiwix::Reader::getNextSuggestion (C++ function), 34  
 kiwix::Reader::getPublisher (C++ function), 31  
 kiwix::Reader::getRandomPage (C++ function), 30  
 kiwix::Reader::getRelation (C++ function), 32  
 kiwix::Reader::getScraper (C++ function), 32  
 kiwix::Reader::getSource (C++ function), 32  
 kiwix::Reader::getTagBool (C++ function), 32  
 kiwix::Reader::getTags (C++ function), 31  
 kiwix::Reader::getTagStr (C++ function), 31  
 kiwix::Reader::getTitle (C++ function), 31  
 kiwix::Reader::getTitleVariants (C++ function), 34  
 kiwix::Reader::getZimArchive (C++ function), 34  
 kiwix::Reader::getZimFilePath (C++ function), 30  
 kiwix::Reader::hasFulltextIndex (C++ function), 34  
 kiwix::Reader::isCorrupted (C++ function), 34  
 kiwix::Reader::pathExists (C++ function), 34  
 kiwix::Reader::Reader (C++ function), 30  
 kiwix::Reader::searchSuggestions (C++ function), 33  
 kiwix::Reader::searchSuggestionsSmart (C++ function), 33  
 kiwix::Reader::suggestions (C++ member), 35  
 kiwix::Reader::suggestionsOffset (C++ member), 35  
 kiwix::Reader::zimArchive (C++ member), 35  
 kiwix::Reader::zimFilePath (C++ member), 35  
 kiwix::removeLastPathElement (C++ function), 46  
 kiwix::Result (C++ class), 35  
 kiwix::Result::~~Result (C++ function), 35  
 kiwix::Result::get\_content (C++ function), 35  
 kiwix::Result::get\_score (C++ function), 35  
 kiwix::Result::get\_size (C++ function), 35  
 kiwix::Result::get\_snippet (C++ function), 35  
 kiwix::Result::get\_title (C++ function), 35  
 kiwix::Result::get\_url (C++ function), 35  
 kiwix::Result::get\_wordCount (C++ function), 35  
 kiwix::Result::get\_zimId (C++ function), 35  
 kiwix::Searcher (C++ class), 35  
 kiwix::Searcher::~~Searcher (C++ function), 36  
 kiwix::Searcher::add\_reader (C++ function), 36  
 kiwix::Searcher::beautifyInteger (C++ function), 37  
 kiwix::Searcher::closeIndex (C++ function), 37  
 kiwix::Searcher::estimatedResultCount (C++ member), 37  
 kiwix::Searcher::geo\_search (C++ function), 36  
 kiwix::Searcher::get\_reader (C++ function), 36  
 kiwix::Searcher::getEstimatedResultCount (C++ function), 37  
 kiwix::Searcher::getMaxResultCount (C++ function), 37  
 kiwix::Searcher::getNextResult (C++ function), 36  
 kiwix::Searcher::getResultStart (C++ function), 37  
 kiwix::Searcher::getSearchResultSet (C++ function), 37  
 kiwix::Searcher::internal (C++ member), 37  
 kiwix::Searcher::maxResultCount (C++ member), 37  
 kiwix::Searcher::readers (C++ member), 37  
 kiwix::Searcher::restart\_search (C++ function), 37

---

```

kiwix::Searcher::resultStart (C++ member), 37
kiwix::Searcher::search (C++ function), 36
kiwix::Searcher::Searcher (C++ function), 36
kiwix::Searcher::searchInIndex (C++ function), 37
kiwix::Searcher::searchPattern (C++ member), 37
kiwix::Searcher::suggestionInternal (C++ member), 37
kiwix::Searcher::suggestions (C++ function), 36
kiwix::SearchRenderer (C++ class), 37
kiwix::SearchRenderer::~~SearchRenderer (C++ function), 38
kiwix::SearchRenderer::beautifyInteger (C++ function), 38
kiwix::SearchRenderer::estimatedResultCount (C++ member), 38
kiwix::SearchRenderer::getHtml (C++ function), 38
kiwix::SearchRenderer::m_srs (C++ member), 38
kiwix::SearchRenderer::mp_nameMapper (C++ member), 38
kiwix::SearchRenderer::pageLength (C++ member), 38
kiwix::SearchRenderer::protocolPrefix (C++ member), 38
kiwix::SearchRenderer::resultStart (C++ member), 38
kiwix::SearchRenderer::searchContent (C++ member), 38
kiwix::SearchRenderer::searchPattern (C++ member), 38
kiwix::SearchRenderer::searchProtocolPrefix (C++ member), 38
kiwix::SearchRenderer::SearchRenderer (C++ function), 38
kiwix::SearchRenderer::setPageLength (C++ function), 38
kiwix::SearchRenderer::setProtocolPrefix (C++ function), 38
kiwix::SearchRenderer::setSearchContent (C++ function), 38
kiwix::SearchRenderer::setSearchPattern (C++ function), 38
kiwix::SearchRenderer::setSearchProtocolPrefix (C++ function), 38
kiwix::Server (C++ class), 39
kiwix::Server::~~Server (C++ function), 39
kiwix::Server::m_addr (C++ member), 39
kiwix::Server::m_blockExternalLinks (C++ member), 39
kiwix::Server::m_indexTemplateString (C++ member), 39
kiwix::Server::m_nbThreads (C++ member), 39
kiwix::Server::m_port (C++ member), 39
kiwix::Server::m_root (C++ member), 39
kiwix::Server::m_verbose (C++ member), 39
kiwix::Server::m_withLibraryButton (C++ member), 39
kiwix::Server::m_withTaskbar (C++ member), 39
kiwix::Server::mp_library (C++ member), 39
kiwix::Server::mp_nameMapper (C++ member), 39
kiwix::Server::mp_server (C++ member), 39
kiwix::Server::Server (C++ function), 39
kiwix::Server::setAddress (C++ function), 39
kiwix::Server::setBlockExternalLinks (C++ function), 39
kiwix::Server::setIndexTemplateString (C++ function), 39
kiwix::Server::setNbThreads (C++ function), 39
kiwix::Server::setPort (C++ function), 39
kiwix::Server::setRoot (C++ function), 39
kiwix::Server::setTaskbar (C++ function), 39
kiwix::Server::setVerbose (C++ function), 39
kiwix::Server::start (C++ function), 39
kiwix::Server::stop (C++ function), 39
kiwix::size_type (C++ type), 48
kiwix::sleep (C++ function), 46
kiwix::split (C++ function), 46
kiwix::SuggestionItem (C++ class), 40
kiwix::SuggestionItem::getNormalizedTitle (C++ function), 40
kiwix::SuggestionItem::getPath (C++ function), 40
kiwix::SuggestionItem::getSnippet (C++ function), 40
kiwix::SuggestionItem::getTitle (C++ function), 40
kiwix::SuggestionItem::hasSnippet (C++ function), 40
kiwix::SuggestionItem::SuggestionItem (C++ function), 40
kiwix::SuggestionsList_t (C++ type), 48
kiwix::supportedListMode (C++ enum), 41
kiwix::supportedListMode::ALL (C++ enumerator), 41
kiwix::supportedListMode::LOCAL (C++ enumerator), 41
kiwix::supportedListMode::NOLOCAL (C++ enumerator), 41
kiwix::supportedListMode::NOREMOTE (C++

```

*enumerator*), [41](#)  
kiwix::supportedListMode::NOVALID (C++  
*enumerator*), [41](#)  
kiwix::supportedListMode::REMOTE (C++  
*enumerator*), [41](#)  
kiwix::supportedListMode::VALID (C++  
*enumerator*), [41](#)  
kiwix::supportedListSortBy (C++ *enum*), [41](#)  
kiwix::supportedListSortBy::CREATOR  
(C++ *enumerator*), [41](#)  
kiwix::supportedListSortBy::DATE (C++  
*enumerator*), [41](#)  
kiwix::supportedListSortBy::PUBLISHER  
(C++ *enumerator*), [41](#)  
kiwix::supportedListSortBy::SIZE (C++  
*enumerator*), [41](#)  
kiwix::supportedListSortBy::TITLE (C++  
*enumerator*), [41](#)  
kiwix::supportedListSortBy::UNSORTED  
(C++ *enumerator*), [41](#)  
kiwix::UpdatableNameMapper (C++ *class*), [40](#)  
kiwix::UpdatableNameMapper::getIdForName  
(C++ *function*), [40](#)  
kiwix::UpdatableNameMapper::getNameForId  
(C++ *function*), [40](#)  
kiwix::UpdatableNameMapper::UpdatableNameMapper  
(C++ *function*), [40](#)  
kiwix::UpdatableNameMapper::update (C++  
*function*), [40](#)  
KIWIX\_LIBRARY\_VERSION (C *macro*), [47](#)